Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds

Haitao Yuan, *Student Member, IEEE*, Jing Bi, *Member, IEEE*, Wei Tan, *Senior Member, IEEE*, and Bo Hu Li

***Abstract*—As cloud computing is becoming growingly popular, consumers' tasks around the world arrive in cloud data centers. A private cloud provider aims to achieve profit maximization by intelligently scheduling tasks while guaranteeing the service delay bound of delay-tolerant tasks. However, the aperiodicity of arrival tasks brings a challenging problem of how to dynamically schedule all arrival tasks given the fact that the capacity of a private cloud provider is limited. Previous works usually provide an admission control to intelligently refuse some of arrival tasks. Nevertheless, this will decrease the throughput of a private cloud, and cause revenue loss. This paper studies the problem of how to maximize the profit of a private cloud in hybrid clouds while guaranteeing the service delay bound of delay-tolerant tasks. We propose a profit maximization algorithm (PMA) to discover the temporal variation of prices in hybrid clouds. The temporal task scheduling provided by PMA can dynamically schedule all arrival tasks to execute in private and public clouds. The sub problem in each iteration of PMA is solved by the proposed hybrid heuristic optimization algorithm, simulated annealing particle swarm optimization (SAPSO). Besides, SAPSO is compared with existing baseline algorithms. Extensive simulation experiments demonstrate that the proposed method can greatly increase the throughput and the profit of a private cloud while guaranteeing the service delay bound.**

***Note to Practitioners*—This paper aims to solve the problem of task scheduling for a private cloud in hybrid clouds. The aperiodicity of arrival tasks brings a challenge of maximizing the profit of a private cloud provider while guaranteeing the service delay bound of delay-tolerant tasks. Existing methods usually provide an admission control to refuse some of arrival tasks that exceed the capacity of a private cloud. This paper first proposes an architecture of temporal task scheduling in hybrid clouds. Based on the architecture, a PMA algorithm is proposed to provide temporal task scheduling which can maximize the profit of a private cloud by intelligently dispatching arrival tasks to execute in private and public clouds within the service delay bound. Preliminary simulation experiments show that the proposed temporal task scheduling is feasible but it has not yet been implemented in real hybrid clouds. In future research, we will extend this work to incorporate multiple different tasks that require heterogeneous resources including bandwidth and storage.**

***Index Terms*—Heuristic algorithm, hybrid clouds, profit maximization, service delay, task scheduling.**

Manuscript received January 24, 2015; revised May 25, 2015; accepted January 30, 2016. Date of publication February 23, 2016; date of current version January 04, 2017. This paper was recommended for publication by Associate Editor J. Gao and Editor M. P. Fanti upon evaluation of the reviewers' comments. *(Corresponding author: Jing Bi).*

H. Yuan and B. H. Li are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuanhaitao@buaa.edu.cn; bohuli@buaa.edu.cn).

J. Bi is with the School of Software Engineering, Beijing University of Technology, and Beijing Engineering Research Center for IoT Software and Systems, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

W. Tan is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: wtan@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2016.2526781

I. Introduction

CLOUD computing can efficiently provide on-demand computing resources over the network to consumers worldwide [1]. Typically, computing resources in cloud data centers are dynamically delivered to consumers using a pay-as-you-go pricing model [2]. In addition, the economy of scale brought by cloud computing attracts an increasing number of companies to deploy their applications in cloud data centers. As a typical part of cloud, Infrastructure as a Service (IaaS) provides the foundation for applications [3]. Typical IaaS providers such as Rackspace and Amazon EC2 [4] provide services to consumers based on a pay-per-use model. An IaaS provider manages its own limited resources. Therefore, similar to the definition in [5], from the perspective of an IaaS provider, private cloud in this paper denotes a resource-constrained IaaS provider that may outsource some of its tasks to execute in external public clouds when it cannot deliver promised quality-of-service (QoS) with its resources. A private cloud provider aims to provide services to consumers' tasks in the most cost-effective way while guaranteeing the specified QoS. Therefore, profit maximization is a critically important goal for a private cloud provider [6].

The uncertainty and aperiodicity of arrival tasks makes it difficult to predict the future arrival tasks, and brings a major challenge to operators of a private cloud. Therefore, it is possible that a private cloud provider can not satisfy all arrival tasks with its limited resources if the arrival tasks are massive. The existing works usually provide an admission control mechanism to refuse some of arrival tasks that exceed the capacity of a private cloud [7]. Nevertheless, this will decrease the throughput of a private cloud, and inevitably cause revenue loss to the private cloud provider. However, the mechanism of hybrid clouds enables a private cloud provider to make use of public clouds where resources are delivered in the form of virtual machines (VMs) when resources of a private cloud is fully occupied [8].

Delay-tolerant tasks usually have a strict service delay bound to meet. Moreover, cloud providers such as Amazon EC2 provide resources in the form of VMs to paying consumers. In the real life, the execution prices of each VM type offered by public cloud providers vary with time [9]. Besides, the power prices in a private cloud also express temporal diversity [7]. Therefore, this presents an opportunity to maximize the profit of a private cloud by a temporal task scheduling while guaranteeing the strict service delay bound of all tasks. The temporal task scheduling can intelligently dispatch arrival tasks to execute in private and public clouds within the service delay bound.

There are several differences between the proposed temporal task scheduling in hybrid clouds and conventional areas including manufacturing, transportation, etc. Therefore, we clearly summarize the differences as follows.

1) The virtualization of cloud enables resources to be encapsulated as VMs that can be delivered on demand. Besides, services in a cloud can be dynamically configured and delivered. Therefore, a private cloud can keep scalable and schedule tasks using VMs provided by other public clouds even if resources in a private cloud is fully occupied. However, the available resources in conventional scheduling are usually limited and fixed.
2) The economy of scale enabled by the cloud makes a private cloud provider focus on maximizing its profit by providing a pay-as-you-go pricing model. However, scheduling in conventional areas usually cannot support a pay-as-you-go model because it is difficult to deliver essential resources (e.g., manufacturing tools) on demand.
3) The prices in hybrid clouds vary during the service delay bound. The temporal diversity in prices presents an opportunity to propose a temporal task scheduling that maximizes the profit of a private cloud provider in hybrid clouds. However, compared to applications in a cloud, the scheduling period in conventional areas is much longer. Therefore, the service delay bound of arrival tasks for cloud applications cannot be guaranteed with conventional scheduling.

In this paper, we study the profit maximization problem for a private cloud provider in hybrid clouds. The execution prices of VMs in public clouds, and the power prices in the private cloud exhibit temporal diversity. We formulate the profit maximization problem for a private cloud provider and propose PMA to solve it. Then, we adopt public real-life workload to evaluate the proposed method. Extensive simulations have shown that the proposed method outperforms existing task scheduling methods in terms of throughput and profit.

The main contribution of this paper is described as follows. First, this work focuses on delay-tolerant tasks and strictly guarantees the service delay bound of all tasks. Second, we propose an architecture of temporal task scheduling in hybrid clouds where a private cloud can outsource some of its arrival tasks that exceed the capacity of a private cloud to public clouds provided that the service delay bound of all task is strictly guaranteed. Third, this work proposes a PMA algorithm to maximize the profit of a private cloud provider by intelligently scheduling arrival tasks to execute in private and public clouds.

The rest of this paper is organized as follows. Section II discusses the related work in the literature. Section III presents the architecture of temporal task scheduling in hybrid clouds. Section IV presents the system model and problem formulation. The details of the proposed solution algorithms are given in Section V. Extensive simulation experiments are conducted to evaluate the effectiveness of the proposed method in Section VI. Finally, Section VII concludes this paper.

II. Related Work

This section presents an overview of papers related to this research topic and compares the proposed temporal task scheduling with existing works.

Resource allocation is a basic problem in cloud data centers [10]–[12]. The objective of resource allocation is to reasonably provision limited resources in cloud data centers to process consumers' arrival tasks with the constraint that the performance requirement of arrival tasks must be ensured. So far, there have been a growing number of recent studies to investigate the problem of resource allocation in cloud data centers [13]–[15]. Authors in [13] proposed a lightweight simulation system to model real-time resource allocation in cloud data centers. Authors in [14] presented a method to optimize data center resource and to support green computing according to application demands. Authors in [15] studied the effect of future workload information on dynamic provisioning of resources, and presented a decentralized algorithm to dynamically provision resources. However, all above works focus on resource allocation without the consideration of profit maximization of a private cloud.

Task scheduling in data centers has attracted much attention as the demand for cloud applications increases [5], [7], [16]–[18]. Authors in [16] presented an approach that can dynamically allocate machines in cloud data centers to minimize the total energy consumption by considering the heterogeneity of both workloads and machines. Authors in [17] proposed an algorithm to utilize idle time of allocated resources to replicate tasks. Therefore, the performance variation of resources on workflows with soft deadlines can be mitigated. Authors in [18] proposed three different algorithms to provide energy-aware scheduling of tasks. The algorithms were tested and compared with other existing scheduling algorithms. However, all above works focus on task scheduling without consideration of the temporal variation of prices within the service delay bound in private and public clouds. Authors in [5] proposed an approach to schedule tasks to maximize the profit of a private cloud while guaranteeing the corresponding delay bound. This scheduling problem was formulated and solved by a heuristic algorithm. However, this work in [5] does not consider the price variation in private and public clouds during the delay bound. Authors in [7] proposed a two-stage design to minimize energy cost of data centers and dynamically schedule tasks to execute in data centers. However, this work in [7] chooses to refuse excessive tasks that exceed the capacity of a private cloud. In contrast, the temporal task scheduling in our work aims to maximize the profit of a private cloud provider by considering the temporal diversity in prices, and intelligently scheduling all arrival tasks to execute in private and public clouds.

A few existing works investigated the profit maximization problem by adopting queueing models to evaluate the service delay performance [19]–[21]. Authors in [19] studied the configuration of a multiserver system for profit maximization for a cloud. An M/M/m queueing model is adopted to analytically formulate a multiserver system in their method. The expected profit of a cloud in each time slot is calculated. Authors in [20]

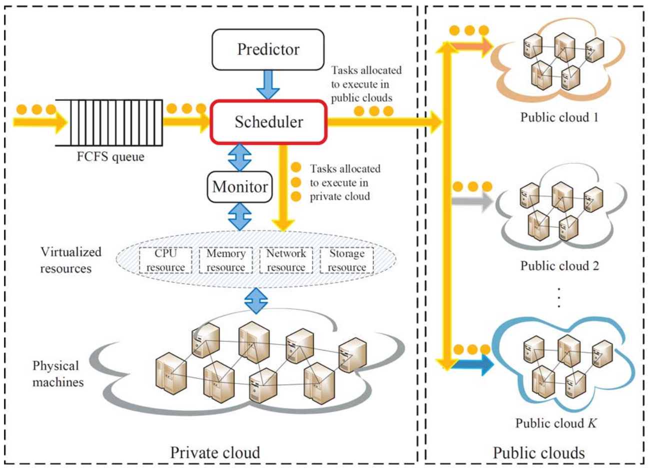


Fig. 1. Architecture of temporal task scheduling in hybrid clouds.

proposed a hybrid queueing model to specify the number of virtual machines for a multitier application in a cloud. Then, this paper formulates a nonlinear constrained problem and develops a heuristic algorithm to maximize the profit of a cloud provider. These works usually select the metric of the expected service delay. Then, based on the expected service delay, a utility function is proposed to model the expected profit. Authors in [21] derived a mathematical model to calculate the profit in large data centers. Based on this model, a profit maximization method with and without behind-the-meter renewable generators is developed and assessed. However, these works can only guarantee the expect delay bound of consumers' tasks. The long tail in service delay of tasks means that some of tasks may experience longer service delay [22]. Different from above works, the proposed temporal task scheduling can guarantee the service delay bound of all arrival tasks.

III. Architecture of Temporal Task Scheduling in Hybrid Clouds

The architecture of the temporal task scheduling in hybrid clouds is shown in Fig. 1. The architecture consists of two major parts which are private and public clouds, respectively. In the private cloud, a huge number of computers are pooled to provide virtualized resources including CPU, memory, network and storage. *Monitor* component monitors the virtualized resource pool, and reports the latest information to the *Scheduler* component. The arrival tasks from consumers are enqueued into a FCFS queue. All the information about the FCFS queue (e.g., the number of arrival tasks) is sent to the *Scheduler* component. In addition, the *Predictor* component can execute typical prediction algorithms to predict future information about private and public clouds based on the corresponding historical data. In this paper, we mainly consider the *Scheduler* component that executes the proposed temporal task scheduling to maximize the profit of a private cloud while guaranteeing the service delay bound of all tasks. Based on the information collected from the *Monitor* component, the *Predictor* component and the FCFS queue, the *Scheduler* component executes the temporal task scheduling and determines the number of scheduled tasks that execute in private and public clouds at each time slot, respectively.

IV. Modeling and Formulation

In this section, we present the formulation of temporal task scheduling that maximizes the profit of a private cloud provider in hybrid cloud. In this paper, similar to the work in [21], the private cloud system is modeled as a discrete-time system that evolves in a sequence of equal-length time slots.

Our work focuses on the delay-tolerant tasks. Each task may be massive-scale data analysis, scientific computing, large-scale image processing, etc. With the massive-scale deployment of commodity computers in cloud data centers, more and more cloud data centers support many-core computing and incline to parallelize the applications. Thus, each task can be decomposed to functionally equivalent small tasks that can be mapped into multiple computers and completed in a shorter time [23]. Besides, there are many existing works that can predict the future information according to past data [24]. Therefore, to make the formulation clear, we explicitly make several assumptions organized as follows.

1) It is assumed that the number of arrival tasks in time slot t, λ_t, is known.
2) We assume that the *Predictor* component can report the information including the future charging prices and the future power prices of a private cloud, the future execution prices of VMs in public clouds, and the future running time of tasks to the *Scheduler* component.
3) Similar to the work [7], we assume that each task can be replaced by a number of parallelized subtasks which are small enough to execute. Therefore, each task can finish its execution within one time slot.
4) It is assumed that the capacities of public clouds are unlimited to process tasks. Therefore, each task scheduled to execute in public clouds can finish its execution in one time slot.
5) We assume that time and cost related to data and network is negligible.

In addition, all tasks have a strict service delay bound, B, by when it must be completed, i.e., tasks that arrive in time slot t can be scheduled from time slot t to $t + B$. However, all tasks arriving in time slot t must have been scheduled by the time slot $t+B$. The parameters and decision variables of the problem are summarized in Tables I and II, respectively.

In any time slot t, arrival tasks of consumers are enqueued into an FCFS queue. Therefore, let Δ_t denote the cumulation of arrival tasks during the period of t time slots. In addition, let D_t denote the cumulation of tasks scheduled during the period of t time slots. Therefore, Δ_t and D_t can be calculated as follows:

$$\Delta_t = \sum_{i=1}^{t} \lambda_i, D_t = \sum_{i=1}^{t} \left(d_i + \sum_{k=1}^{K} y_i^k d_i^k \right). \tag{1}$$

Our final objective is to maximize the profit of the private cloud that is determined by its revenue and cost. Let $Profit$ denote the profit of the private cloud, which is the objective function in the profit maximization problem marked with **P1** . As is shown in (2), $Profit$ is determined by the gained revenue of the private cloud, $Revenue$, and the corresponding cost of the private cloud, $Cost$

$$Profit = Revenue - Cost. \tag{2}$$

TABLE I
PROBLEM PARAMETERS

Notation	Definition
B	Service delay bound of all tasks
Δ_t	Cumulation of the arrival tasks during the period of t time slots, i.e., $\sum_{i=1}^{t}\lambda_i$
D_t	Cumulation of tasks scheduled during the period of t time slots, i.e., $\sum_{i=1}^{t} d_i$
λ_t	Number of arrival tasks in time slot t
K	Number of public clouds
k	Public cloud k
r_t	Average runtime of each task in time slot t
$\tilde{r}_{t+m}$	Estimated average runtime of each task in time slot t+m
p_t^{char}	Charging price of one unit of tasks in time slot t
$\tilde{p}_{t+m}^{char}$	Estimated charging price of one unit of tasks in time slot t+m
p_t^{power}	Power price in a private cloud in time slot t
$\tilde{p}_{t+m}^{power}$	Estimated power price in a private cloud in time slot t+m
c_t^k	Execution price of VMs in public cloud k in time slot t
$\tilde{c}_{t+m}^k$	Estimated execution price of VMs in public cloud k in time slot t+m
φ	Amount of power consumed to execute one unit of tasks
CPU_t	Average number of CPUs required by each arrival task executing in a private cloud in time slot t
mem_t	Average size of memory required by each arrival task executing in a private cloud in time slot t
CPU_{Cap}	Capacity of CPUs in a private cloud
mem_{Cap}	Capacity of memory in a private cloud

TABLE II
DECISION VARIABLES

Notation	Definition
d_t	Number of tasks scheduled to execute in time slot t in a private cloud
$\tilde{d}_{t+m}$	Estimated number of tasks scheduled to execute in time slot t+m(1≤m≤B) in a private cloud
y_t^k	Binary decision variable, such that y_t^k=1 if tasks are allocated to public cloud k in time slot t; otherwise, y_t^k=0
$\tilde{y}_{t+m}^k$	Binary decision variable, such that $\tilde{y}_{t+m}^k$=1 if tasks are allocated to public cloud k in time slot t+m; otherwise, $\tilde{y}_{t+m}^k$=0
d_t^k	Number of tasks scheduled to execute in public cloud k in time slot t
$\tilde{d}_{t+m}^k$	Estimated number of tasks scheduled to execute in public cloud k in time slot t+m(1≤m≤B)

As is shown in (3), $Revenue$ is obtained by calculating the revenue brought by the scheduled tasks from time slot t to time slot $t + B$. At time slot t and $t + m(1 \le m \le B)$, arrival tasks allocated to execute in private and public clouds contribute revenue to the private cloud according to the corresponding charging price of the private cloud at that time slot

$$Revenue = (p_t^{char}(r_t d_t + \sum_{k=1}^{K}(y_t^k r_t d_t^k))) + (\sum_{m=1}^{B}(\tilde{p}_{t+m}^{char} (\tilde{r}_{t+m}\tilde{d}_{t+m} + \sum_{k=1}^{K}(\tilde{y}_{t+m}^k \tilde{r}_{t+m}\tilde{d}_{t+m}^k)))). \quad (3)$$

Considering management and profit, some applications are only managed by the private cloud provider that directly provides services to consumers' tasks and charges consumers based on services they use. Let p_t^{char} denote the charging price of one unit of tasks in time slot t. Consumers pay money for the execution of their tasks to the private cloud provider based on the specified p_t^{char}. However, resources in private cloud are limited, and therefore it may make use of VMs provided by public clouds and execute excessive tasks. Therefore, private cloud needs to pay the execution cost of VMs in public clouds. Let $c_t^k(1 \le k \le K)$ denote the execution price of VMs in public cloud k in time slot t. Usually, the private cloud provider first negotiates with public clouds and specifies the execution prices of VMs in public clouds. Then, based on the execution prices of VMs in public clouds, the private cloud provider determines the charging prices of one unit of tasks. This paper focuses on the profit-driven private cloud. Therefore, p_t^{char} is greater than $c_t^k(1 \le k \le K)$; otherwise, the private cloud provider will lose money by scheduling tasks to execute in public clouds.

As is shown in (4), $Cost$ is obtained by calculating the cost of the private cloud including the cost of energy consumed by the execution of tasks allocated to the private cloud, and the execution cost of VMs in public clouds brought by tasks scheduled to execute in public clouds from time slot t to time slot $t + B$

$$Cost = (p_t^{power}\varphi d_t + \sum_{k=1}^{K}(y_t^k c_t^k r_t d_t^k)) + (\sum_{m=1}^{B}(\tilde{p}_{t+m}^{power}\varphi \tilde{d}_{t+m} + \sum_{k=1}^{K}(\tilde{y}_{t+m}^k \tilde{c}_{t+m}^k \tilde{r}_{t+m}\tilde{d}_{t+m}^k))). \quad (4)$$

Therefore, based on (2)–(4), the final profit maximization problem that is denoted as **P1** can be summarized as follows:

$$\underset{d_t,\tilde{d}_{t+m},d_t^k,\tilde{d}_{t+m}^k,y_t^k,\tilde{y}_{t+m}^k}{Max} \quad Profit = \{Revenue - Cost\}$$

subject to

$$d_t CPU_t \le CPU_{Cap} \quad (5)$$
$$d_t mem_t \le mem_{Cap} \quad (6)$$
$$\tilde{d}_{t+m} CPU_{t+m} \le CPU_{Cap}, 1 \le m \le B \quad (7)$$
$$\tilde{d}_{t+m} mem_{t+m} \le mem_{Cap}, 1 \le m \le B \quad (8)$$
$$\sum_{k=1}^{K} y_t^k \le 1 \quad (9)$$
$$\sum_{k=1}^{K} \tilde{y}_{t+m}^k \le 1, 1 \le m \le B \quad (10)$$
$$\Delta_{t-B-1} + \lambda_{t-B} \le D_{t-1} + \left(d_t + \sum_{k=1}^{K} y_t^k d_t^k\right) \quad (11)$$
$$\Delta_{t-B-1} + \sum_{u=t-B}^{t-B+m} \lambda_u \le D_{t-1} + \left(d_t + \sum_{k=1}^{K} y_t^k d_t^k\right) + \sum_{u=t+1}^{t+m}\left(\tilde{d}_u + \sum_{k=1}^{K}\tilde{y}_u^k \tilde{d}_u^k\right), 1 \le m \le B \quad (12)$$
$$\Delta_{t-B-1} + \sum_{u=t-B}^{t} \lambda_u = D_{t-1} + \left(d_t + \sum_{k=1}^{K} y_t^k d_t^k\right) + \sum_{u=t+1}^{t+B}\left(\tilde{d}_u + \sum_{k=1}^{K}\tilde{y}_u^k \tilde{d}_u^k\right) \quad (13)$$
$$y_t^k, \tilde{y}_{t+m}^k \in \{0,1\}, 1 \le m \le B \quad (14)$$
$$d_t \ge 0, \tilde{d}_{t+m} \ge 0, d_t^k \ge 0, \tilde{d}_{t+m}^k \ge 0, 1 \le m \le B. \quad (15)$$

Constraints (5)–(8) show that the total CPU and memory demand of the arrival tasks in time slot t and t+$m(1 \le m \le B)$

cannot exceed the CPU and memory capacity of the private cloud, respectively. Note that CPU_t and mem_t in the constraints (5)–(8) denote the average number of CPUs and the average size of memory required by each arrival task executing in the private cloud in time slot t, respectively. Besides, we can obtain $CPU_t(mem_t)$ by dividing the total required CPUs (memory) of all arrival tasks by the number of all arrival tasks in time slot t. Therefore, CPU_t and mem_t are calculated once a time slot. Constraints (9) and (10) mean that in each time slot t and $t+m(1 \leq m \leq B)$, if there are tasks that are scheduled to execute in public clouds, these tasks can only be allocated to one public cloud, i.e., $\sum_{k=1}^{K} y_t^k = 1$ or $\sum_{k=1}^{K} \tilde{y}_{t+m}^k = 1$. Constraint (11) means that by time slot t, all the tasks that arrive in time slot $t-B$ or earlier must have been scheduled to execute in private and public clouds. Moreover, constraint (12) denotes that by time slot $t+m$, all the tasks that arrive in time slot $t+m-B$ or earlier must have been scheduled to execute in private and public clouds. Constraint (13) denotes that by time slot $t+B$, the expected cumulative scheduled tasks must be equal to the cumulative arrival tasks in time slot t. Constraints (14) and (15) denote the valid ranges of decision variables.

We adopt the method of penalty function to convert the constrained problem **P1** into the unconstrained one **P2**. Let $augProfit$ and $Penalty$ denote the value of new augmented objective function, and the value of penalty function, respectively. Parameter σ is a relatively large positive number that highlights the impact of $Penalty$ on $augProfit$. Therefore, problem **P2** is shown as follows:

$$\underset{d_t,\tilde{d}_{t+m},d_t^k,\tilde{d}_{t+m}^k,y_t^k,\tilde{y}_{t+m}^k}{Min} \quad \{augProfit = -Profit + \sigma \cdot Penalty\}.$$

In problem **P2**, each penalty corresponding to each equality or inequality constraint is added to the original objective function, $Profit$. In the above equation, $Penalty$ can be calculated using (16)

$$Penalty = \sum_{i=1}^{p}(\max\{0, -g_i(\mathbf{x})\})^{\alpha} + \sum_{j=1}^{q} |h_j(\mathbf{x})|^{\beta}. \quad (16)$$

In (16), $\mathbf{x}$ denotes the vector of decision variables consisting of $d_t, \tilde{d}_{t+m}, d_t^k, \tilde{d}_{t+m}^k, y_t^k, \tilde{y}_{t+m}^k$. Besides, α and β are two constant positive parameters. There are p inequality constraints and q equality constraints in problem **P1**. Let $g_i(\mathbf{x}) \geq 0(1 \leq i \leq p)$ denote the inequality constraint i in problem **P1**. Therefore, the penalty of the inequality constraint i is $(\max\{0, -g_i(\mathbf{x})\})^{\alpha}$. Let $h_j(\mathbf{x}) = 0(1 \leq j \leq q)$ denote the equality constraint j in problem **P1**. Therefore, the penalty of the equality constraint j is $|h_j(\mathbf{x})|^{\beta}$. For example, constraint (5) can be transformed to $CPU_{Cap} - d_t CPU_t \geq 0$. Therefore, the corresponding penalty is $(\max\{0, -(CPU_{Cap} - d_t CPU_t)\})^{\alpha}$. In this way, the constrained problem, **P1** is converted into the unconstrained one **P2** that can be solved using the solution algorithms described in Section V.

Before the end of this section, we shall make further discussion about our model and the underlying assumptions. We formulate the problem as a relatively complex one that includes many equality and inequality constraints. Therefore, the complexity of mathematical modeling makes it difficult and time-consuming to find the optimal scheduling strategy that satisfies all constraints. However, the complex formulation provides more accurate modeling of scheduling for delay-tolerant tasks in hybrid cloud. In addition, we assume that the future information is known. However, in real cloud, the prediction of information (workload, price, etc.) usually needs relatively a long time to obtain the prediction model by training based on a large amount of historical data. Our work ignores the time of information prediction. Therefore, the essential data maybe unavailable when the proposed solution algorithms start to execute. This may bring unexpected failure that cannot guarantee the service delay bound of all tasks. However, the wide deployment of high-performance servers [25], and the recent improvement of prediction algorithms in the era of big data [26] may make the prediction time negligible in the near future.

V. Solution Algorithms

Section IV describes the optimization problem in each time slot. This section presents the PMA algorithm described in Algorithm 1. In problem **P2**, y_t^k and $\tilde{y}_{t+m}^k(1 \leq m \leq B)$ are discrete integer variables, while $d_t, \tilde{d}_{t+m}, d_t^k, \tilde{d}_{t+m}^k(1 \leq m \leq B)$ are continuous variables. Problem **P2** involves both discrete and continuous variables. Therefore, problem **P2** is a typical mixed integer linear programming (MILP) problem that is difficult to solve [27]. The MILP problems can be solved by many existing algorithms, e.g., brand and bound. However, these algorithms do not work well in large scale optimization problems. Besides, these algorithms usually rely on the structure of a specific problem and convert the original problem into another particular one that can be solved more easily. In addition, these algorithms often find a global optimum at the expense of long execution time according to the complexity of a specific problem.

Algorithm 1: Profit Maximization Algorithm (PMA)

Input: $p_t^{char}, r_t, \tilde{p}_{t+m}^{char}, \tilde{r}_{t+m}, p_t^{power}, \varphi, c_t^k, \tilde{p}_{t+m}^{power}, \tilde{c}_{t+m}^k, CPU_t, CPU_{Cap}, mem_t, mem_{Cap}, \lambda_t, 1 \leq m \leq B$

Output: $d_t, \tilde{d}_{t+m}, d_t^k, \tilde{d}_{t+m}^k, y_t^k, \tilde{y}_{t+m}^k, 1 \leq m \leq B$

```
1:  for $t \leftarrow 0$ to $B$ do
2:      $\lambda_t \leftarrow 0$
3:  end for
4:  $\Delta_0 \leftarrow 0$
5:  $D_B \leftarrow 0$
6:  $t \leftarrow B+1$ {The index of real workload in array $\lambda$ is from $B+1$ to $B+Length$. $Length$ denotes the total number of time slots}
7:  while $t \leq Length$ do
8:      Randomly initialize $d_t, \tilde{d}_{t+m}, d_t^k, \tilde{d}_{t+m}^k, y_t^k, \tilde{y}_{t+m}^k$
9:      $[d_t, d_t^k, y_t^k] \leftarrow$ SAPSO($t, p_t^{char}, \tilde{p}_{t+m}^{char}, r_t, \tilde{r}_{t+m}, d_t, \tilde{d}_{t+m}, \varphi, p_t^{power}, \tilde{y}_{t+m}^k, y_t^k, \tilde{p}_{t+m}^{power}, c_t^k, \tilde{c}_{t+m}^k, d_t^k, \tilde{d}_{t+m}^k, CPU_t, CPU_{Cap}, mem_t, mem_{Cap}, \Delta_t, D_t$) {solve P2 to obtain $d_t$, $d_t^k$ and $y_t^k$ in time slot t}
10:     $\Delta_{t-B} \leftarrow \Delta_{t-B-1} + \lambda_{t-B}$
11:     $D_t \leftarrow D_{t-1} + d_t$
12:     for $k \leftarrow 1$ to $K$ do
13:         $D_t \leftarrow D_t + d_t^k * y_t^k$
14:     end for
15:     $t \leftarrow t+1$
16: end while
```

Typical metaheuristics do not require any auxiliary information about the mathematical structure of optimization problems. Besides, the robustness and easy implementation of typical metaheuristics make them commonly adopted to solve complex optimization MILP problems. Therefore, we can adopt typical metaheuristics, e.g., simulated annealing (SA) algorithm [28] and particle swarm optimization (PSO) algorithm [29] to solve problem **P2**. However, typical heuristic algorithms including the SA algorithm and the PSO algorithm have respective strengths and weaknesses.

The SA algorithm is a popular metaheuristic that can be easily implemented to solve continuous and discrete optimization problems. The SA algorithm is versatile and robust since it does not rely on any restrictive properties of the optimization problem. Therefore, it can deal with reasonably complex models with linear and nonlinear constraints. In addition, one key feature of the SA algorithm that it provides a method to escape from a local optimum by allowing moves that worsen the value of objective function in the hope of achieving a global optimum. It has been proved that the SA algorithm can approach global optimality by carefully controlling the cooling rate of the temperature. However, the main drawback of the SA algorithm is its slow speed at which it converges especially when the search space is large [30].

The PSO algorithm is a population-based optimization algorithm, and has many advantages including simple implementation and quick convergence. Nevertheless, the PSO algorithm is easy to trap into a local optimum. It has been shown that though the computational time of the PSO algorithm is shorter than that of other existing metaheuristics, the precision of its final solution to the large-sized complex optimization problems is relatively poor [31]. Therefore, the PSO algorithm is not robust to solve problems with different constraints.

Based on the typical heuristic algorithms, we propose the hybrid heuristic optimization algorithm, simulated annealing particle swarm optimization (SAPSO) to solve problem **P2**. The proposed SAPSO algorithm combines the strengths of both algorithms. In the SAPSO algorithm, the velocity of each particle is dynamically changed according to its own and other particles' positions in the search space. Different from the PSO algorithm, each particle in the SAPSO algorithm updates its position based on the Metropolis acceptance criterion [30] in the SA algorithm. In each iteration, the objective function values of the current solution and a newly generated one are compared. Improved solutions are directly accepted, while some of inferior solutions are also accepted in the hope of escaping from a local optimum and obtaining a global optimum. Each particle determines the way to allocate tasks in hybrid cloud in current time slot. The final purpose of the SAPSO algorithm is to produce an optimal particle that can maximize the profit of the private cloud. Line 9 in Algorithm 1 is used to solve problem **P2** using the proposed SAPSO algorithm to obtain d_t, d_t^k, y_t^k in time slot t.

The SAPSO algorithm is described in Algorithms 2 and 3. Algorithm 2 shows the first part of the SAPSO algorithm, while Algorithm 3 shows the second part of the SAPSO algorithm. Note that the ***augProfit*** function in Algorithms 2 and 3 calculates the value of augmented objective function described in problem **P2**. However, the implementation detail of the ***augProfit*** function is omitted for the clarity of this paper. Besides, the ***transform*** function whose implementation detail is also omitted transforms the position of the final globally optimal particle into decision variables, d_t, d_t^k and y_t^k.

Algorithm 2: SAPSO Algorithm-Part 1

```
Input: $t$, $p_t^{char}$, $\tilde{p}_{t+m}^{char}$, $r_t$, $\tilde{r}_{t+m}$, $d_t$, $\tilde{d}_{t+m}$, $\varphi$, $p_t^{power}$, $\tilde{y}_{t+m}^k$, $y_t^k$, $\tilde{p}_{t+m}^{power}$, $c_t^k$, $\tilde{c}_{t+m}^k$, $d_t^k$, $\tilde{d}_{t+m}^k$, $CPU_t$, $CPU_{Cap}$, $mem_t$, $mem_{Cap}$, $\Delta_t$, $D_t$
Output: $d_t$, $d_t^k$, $y_t^k$
 1: Initialize the number of iterations, $numIters$
 2: Initialize the number of particles, $numParticles$
 3: Initialize the upper and lower bound of inertia weight, $w_{\max}$ and $w_{\min}$
 4: Initialize the acceleration coefficients, $c_1$ and $c_2$
 5: Initialize the velocity limit of each particle, $v_{\max}$
 6: Initialize the temperature cooling rate, $cr$
 7: Initialize an initial temperature, $temperature$
 8: $dN \leftarrow (2*K+1)*(B+1)$
 9: $Position \leftarrow zeros(numParticles, dN+1)$
10: for $i \leftarrow 1$ to $numParticles$ do
11:     $temp \leftarrow augProfit\ (t, Position(i, 1:dN), \Delta_{t-B}, D_t)$
12:     $Position(i, dN+1) \leftarrow temp$
13: end for
14: $Position_{local} \leftarrow Position$
15: $Position_{global} \leftarrow Position(1, :)$
16: for $i \leftarrow 1$ to $numParticles$ do
17:     if $Position_{global}(1, dN+1) < Position(i+1, dN+1)$ then
18:         $Position_{global} \leftarrow Position(i+1, :)$
19:     end if
20: end for
21: $Velocity \leftarrow -v_{\max} + 2*v_{\max}*rand(numParticles, dN)$
22: $oldPosition \leftarrow zeros(1, dN+1)$
23: $oldVelocity \leftarrow zeros(1, dN)$
24: $iterIndex \leftarrow 1$
25: $w \leftarrow w_{\max} - (w_{\max} - w_{\min})/numIters * iterIndex$
```

Algorithm 3: SAPSO Algorithm-Part 2

```
26: while $iterIndex \leq numIters$ do
27:     for $i \leftarrow 1$ to $numParticles$ do
28:         $oldVelocity(1, :) \leftarrow Velocity(i, :)$
29:         $Velocity(i, :) \leftarrow w*Velocity(i, :) + c_1 * Velocity \leftarrow -v_{\max} + 2*v_{\max}*rand(numParticles, dN)$ $rand()*(Position_{local}(i, 1:dN) - Position(i, 1:dN)) + c_2*rand()*(Position_{global}(1, 1:dN) - Position(i, 1:dN))$
30:         for $j \leftarrow 1$ to $dN$ do
31:             if $Velocity(i, j) < -v_{\max}$ then
32:                 $Velocity(i, j) \leftarrow -v_{\max}$
33:             end if
```

```
34:          if $Velocity(i,j) > v_{\max}$ then
35:             $Velocity(i,j) \leftarrow v_{\max}$
36:          end if
37:       end for
38:       $oldPosition(1,:) \leftarrow Position(i,:)$
39:       $oldaugProfit \leftarrow oldPosition(1, dN+1)$
40:       $Position(i, 1{:}dN) \leftarrow Position(i, 1{:}dN) + Velocity(i,:)$
41:       $newaugProfit \leftarrow$ augProfit $(t, Position(i, 1:dN), 1\Delta_{t-B}, D_t)$
42:       $Position(i, dN+1) \leftarrow newaugProfit$
43:       $\delta_{augProfit} \leftarrow newaugProfit - oldaugProfit$
44:       if $\delta_{augProfit} > 0$ then
45:          if $\exp(-\delta_{augProfit}/temperature) < rand()$ then
46:             $Velocity(i,:) \leftarrow oldVelocity(1,:)$
47:             $Position(i,:) \leftarrow oldPosition(1,:)$
48:          end if
49:       end if
50:       if $Position(i, dN+1) < Position_{local}(i, dN+1)$ then
51:          $Position_{local}(i,:) \leftarrow Position(i,:)$
52:       end if
53:       if $Position(i, dN+1) < Position_{global}(1, dN+1)$ then
54:          $Position_{global}(1,:) \leftarrow Position(i,:)$
55:       end if
56:    end for
57:    $w \leftarrow w_{\max} - (w_{\max} - w_{\min})/numIters * iterIndex$
58:    $iterIndex \leftarrow iterIndex + 1$
59:    $temperature \leftarrow temperature * cr$
60: end while
61: $[d_t, d_t^k, y_t^k] \leftarrow$ transform $(Position_{global}(1, 1:dN))$
62: return $d_t, d_t^k, y_t^k$
```

VI. Performance Evaluation

A. Experimental Setting

In this section, we adopt public real-world workload traces in Google compute systems[1] to evaluate the efficiency of the proposed PMA algorithm. The dataset in Fig. 2 shows the CPU and memory requirements of different tasks in Google production cluster for 370 min in May 2011. These traces contain 1,352,804,107 tasks during the period. Fig. 2 illustrates four types of tasks, i.e., task type 1, 2, 3, and 4. In this experiment, we use tasks of type 1 to evaluate the performance of the proposed PMA algorithm. In addition, $CPU_t(mem_t)$ is calculated by dividing the total required CPUs (memory) of all arrival tasks by the number of all arrival tasks in time slot t.

The parameter setting is shown as follows.

[1]https://code.google.com/p/googleclusterdata/

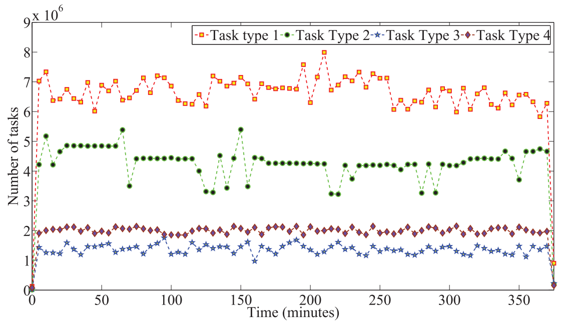


Fig. 2. Arrival tasks of different types in Google's workload traces.

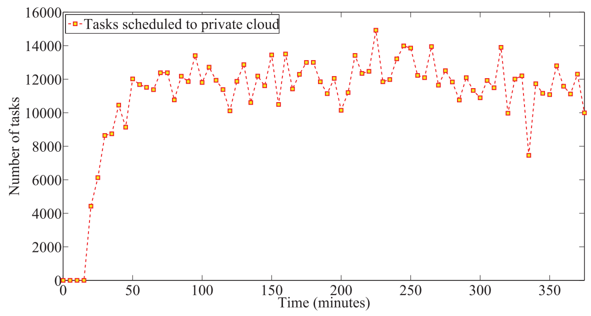


Fig. 3. Tasks scheduled to private cloud.

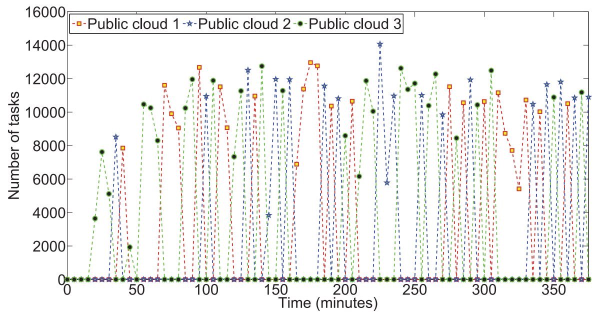


Fig. 4. Tasks scheduled to public clouds.

1) Private cloud: r_t(H), p_t^{char}(\$/H), and p_t^{power}(\$/MWH) are sampled from the uniform distribution over intervals (0,5/60), (0.48,0.96), and (10,20), respectively. Besides, $\varphi = 0.00005$(MWH), $CPU_{Cap} = 2048$, $mem_{Cap} = 1024$(GB).
2) Public clouds: c_t^1(\$/H), c_t^2(\$/H), and c_t^3(\$/H) are sampled from the uniform distribution over intervals (0.17,0.34), (0.175,0.35), and (0.18,0.36), respectively.
3) SAPSO: $numIters = 1000$. $numParticles = 100$. $w_{\max} = 0.95$. $w_{\min} = 0.4$. $c_1 = 0.5$. $c_2 = 0.5$. $v_{\max} == 500$. $cr = 0.975$. $temperature = 10^8$.

B. Experimental Results

Figs. 3 and 4 illustrate the tasks scheduled to private and public clouds, respectively. It is assumed that the capacities of public clouds are unlimited. Therefore, Fig. 4 shows that only one public cloud receives tasks while the number of the tasks scheduled to the other two public clouds are zero at each time slot.

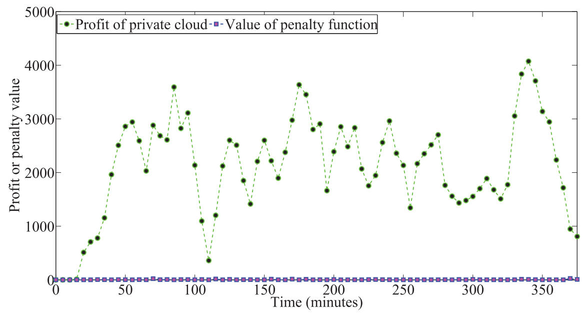


Fig. 5. Profit and penalty in problem **P1** in each time slot.

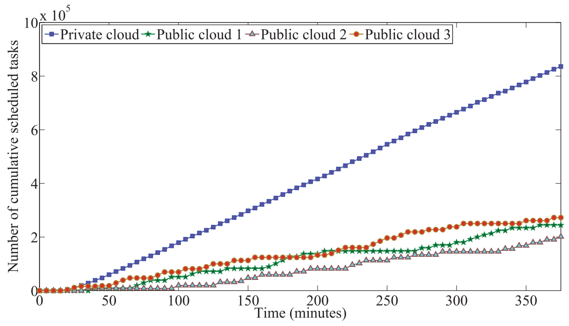


Fig. 7. Cumulative tasks in private and three public clouds.

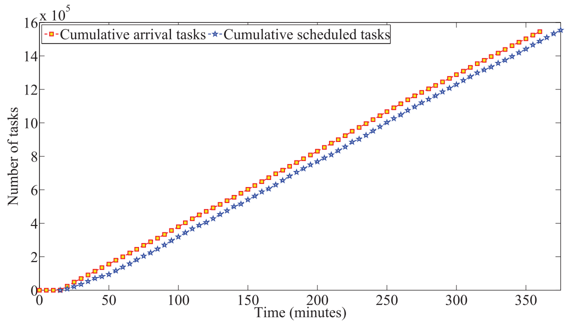


Fig. 6. Cumulative arrival tasks and cumulative scheduled tasks.

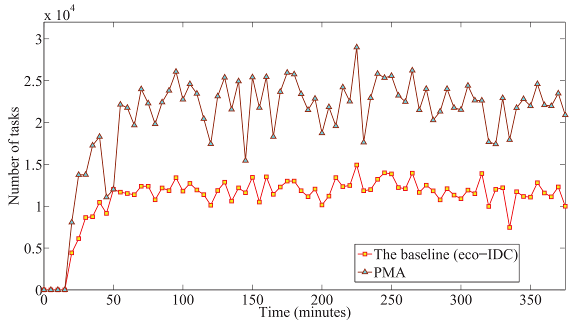


Fig. 8. Comparison of throughout with the work [7].

Fig. 5 illustrates the profit of the private cloud and the value of penalty function in each time slot. The result shows that the penalty value of the solution specified by the proposed PMA algorithm is nearly zero. Therefore, the PMA algorithm can find the efficient solution which can bring the profit for the private cloud provider in each time slot.

Fig. 6 illustrates the cumulative arrival tasks and the cumulative scheduled tasks, respectively. In this experiment, the service delay bound B is set to three time slots, i.e., 15 min. Fig. 6 shows that all the arrival tasks can be scheduled within the service delay bound B. This experiment shows that the temporal task scheduling provided by the proposed PMA algorithm can guarantee the service delay bound of all the arrival tasks.

Fig. 7 illustrates the cumulative tasks in the private and three public clouds. It is obviously observed that the amount of scheduled tasks in the private cloud is much larger than any public cloud. Note that the private cloud provider schedules the arrival tasks between itself and public clouds in the most cost-effective way. The private cloud provider needs to pay the execution cost of VMs in public clouds if there are tasks scheduled to execute in public clouds. Fig. 7 shows that the private cloud provider inclines to schedule tasks to itself. However, the private cloud provider may also schedule some tasks to execute in public clouds in two cases. The first case is that the private cloud provider cannot meet the delay bound of all arrival tasks with its limited resources. The second case is that the execution prices of VMs in public clouds is low enough in a specific time slot. In any time slot, the selected tasks that execute in public clouds are allocated to the public cloud with the lowest execution price. Therefore, the allocated tasks in three public clouds in any time slot reflect the variation of execution prices of VMs in three public clouds.

Fig. 8 illustrates the throughout comparison of the proposed PMA algorithm and the baseline [7]. The work in [7] adopts a two-stage design to admit and schedule workload. However, this work refuses some of the arrival tasks when the amount of arrival tasks exceeds the capacity of the private cloud. In contrast, the temporal task scheduling in our method can intelligently schedule the arrival tasks to execute in private and public clouds. Therefore, as is shown in Fig. 8, compared to this baseline, the throughput of the PMA algorithm can be increased by 87.5% on average. In addition, the throughput difference between this baseline and the PMA algorithm varies from 14066 to 1931. The result in Fig. 8 shows that the proposed PMA algorithm can greatly improve the throughput of the private cloud.

Fig. 9 illustrates the profit comparison between the proposed PMA algorithm and this baseline [7]. The refused tasks in this baseline bring profit loss for the private cloud provider. However, in our PMA algorithm, the private cloud provider intelligently schedules all the arrival tasks to execute in private and public clouds. Therefore, the refused tasks in [7] are also scheduled to execute in private and public clouds using the PMA algorithm. Though these tasks cause additional cost to the private cloud provider, they also bring corresponding revenue to the private cloud provider. As is shown in Fig. 9, compared to this baseline, the profit of the PMA algorithm can be increased by 23.44% on average. In addition, the profit difference between this baseline and the PMA algorithm varies from 1143.13 to 11.07. The result shows that the proposed PMA algorithm can increase the profit of the private cloud in comparison to this baseline.

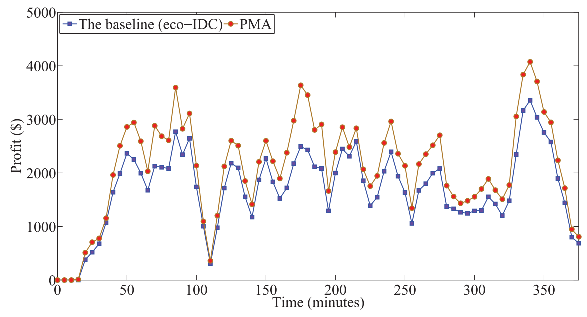


Fig. 9. Comparison of profit with the work [7].

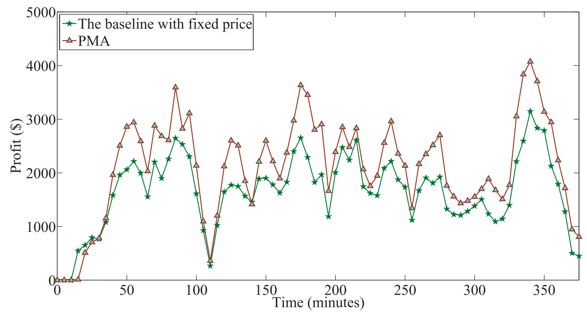


Fig. 10. Profit of the PMA and the baseline [5].

Fig. 10 shows the profit comparison between the proposed PMA and this baseline [5]. This baseline assumes that the prices of private and public clouds are fixed during the service delay bound, B. Therefore, the scheduling method proposed in this baseline does not consider the temporal diversity in prices during the service delay bound. To evaluate the effectiveness of the proposed PMA, the prices in this baseline are fixed and set to the corresponding average values of varying prices in the PMA. Parameters in this baseline are set as follows. $p_t^{char} = 0.72(\$/\mathrm{H})$. $p_t^{power} = 15(\$/\mathrm{MWH})$. $c_t^1 = 0.51(\$/\mathrm{H})$. $c_t^2 = 0.525(\$/\mathrm{H})$. $c_t^3{=}0.54(\$/\mathrm{H})$. Fig. 10 illustrates that the profit of the PMA is larger than that of this baseline in about 94.7% of time. The profit difference between the PMA and this baseline varies from 1241.6 to 13 in about 94.7% of time. Compared to this baseline, the profit of the PMA can be increased by 29.73% on average. This result shows that the PMA performs better than this baseline in terms of the profit. Therefore, the temporal task scheduling provided by the PMA can bring more profit than this baseline with fixed prices during the service delay bound.

In order to evaluate the effectiveness of the proposed temporal task scheduling, we provide the profit comparison between our method and the baseline algorithm which does not queue tasks [7]. Here, the baseline algorithm with no queueing tasks denotes the operation that does not queue the arrival tasks and begins to execute the arrival tasks immediately when they arrive. Let Algo-NQ denote the baseline algorithm with no queueing tasks. Fig. 11 shows the comparison of profit with and without queueing tasks. Our method intelligently queues tasks and schedules them at suitable later time while guaranteeing the corresponding service delay bound by considering the prices variation in hybrid cloud. Fig. 11 shows that the profit of the PMA algorithm is larger than that of the Algo-NQ in

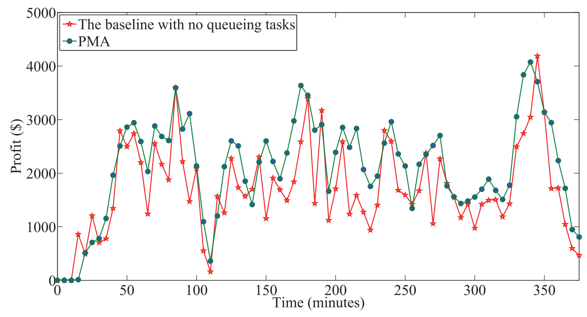


Fig. 11. Comparison of profit with and without queueing tasks.

about 82.8% of time. The profit difference between the PMA algorithm and the Algo-NQ varies from 1639.65 to 8.53 in the about 82.8% of time. Compared to the Algo-NQ, the profit of the PMA algorithm can be increased by 42.37% on average. The reason for this result is that with the consideration of the prices variation, the PMA algorithm can provide the temporal task scheduling while guaranteeing the corresponding service delay bound of all tasks.

C. Methods Compared

To prove the effectiveness of the SAPSO algorithm, we compare it with other two baselines including the SA and PSO algorithms. The reason for choosing these baselines is described as follows.

1) Though the SA algorithm requires relatively a long time to find a global optimum, it has been demonstrated that the SA algorithm can converge to a global optimum by carefully specifying the cooling rate of the temperature. Therefore, the comparison between the SAPSO and SA algorithms can demonstrate the precision of the final solution determined by the SAPSO algorithm.
2) Though the precision of the solution determined by the PSO algorithm is relatively poor, the convergence speed of the PSO algorithm is quick. Therefore, the comparison between the SAPSO and PSO algorithms can show the convergence speed of the SAPSO algorithm.

Fig. 12 illustrates the average evolutionary curves of the SAPSO algorithm and the baselines including the PSO and SA algorithms in the tenth time slot (45–50 min). It is obviously shown that the PSO algorithm converges to its final solution in the smallest number of iterations in comparison to the SA and the SAPSO algorithms. However, the final solution determined by the PSO algorithm is the worst because it brings the smallest profit and the largest penalty. The SA algorithm is the slowest and needs 850 iterations to obtain its final solution. The profit specified by the SA algorithm is 20.6 times larger than that of the PSO algorithm but still smaller than that of the SAPSO algorithm. Finally, the SAPSO algorithm only needs 68 iterations to find its final solution which can generates the profit of 2718$ for the private cloud. Therefore, the SAPSO algorithm can increase the profit by 462.5$ in much smaller number of iterations in comparison to the SA algorithm.

Fig. 13 illustrates the profit of the SAPSO algorithm and the baselines including the PSO and SA algorithms. It is obviously observed that the profit of the PSO algorithm is the lowest

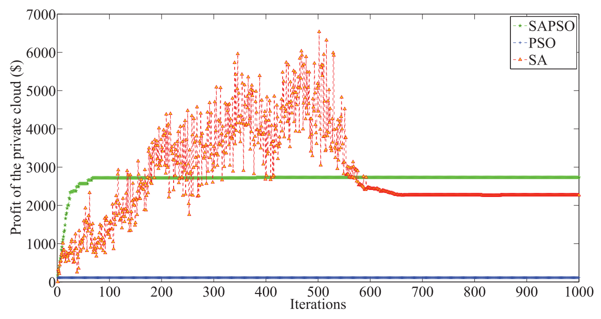


Fig. 12. Average evolutionary curves for the tenth time slot.

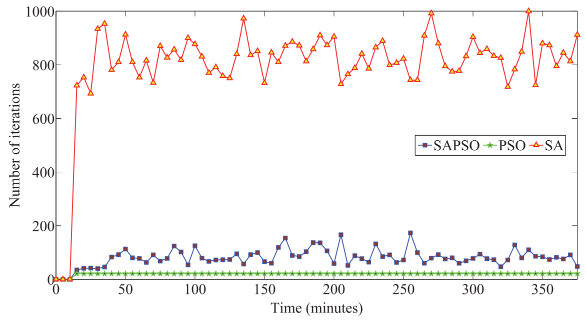


Fig. 14. Iterations of SAPSO, PSO, and SA algorithms.

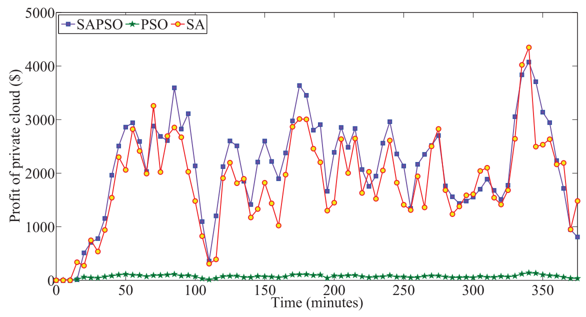


Fig. 13. Profit of SAPSO, PSO, and SA algorithms.

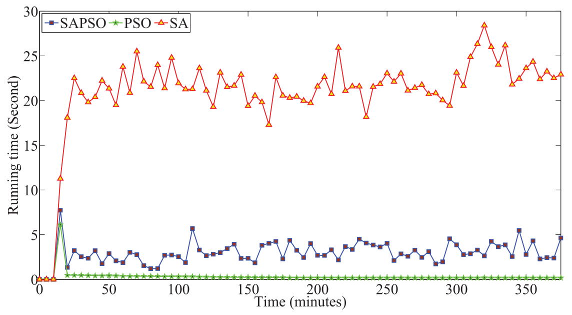


Fig. 15. Running time of SAPSO, PSO, and SA algorithms.

among three algorithms. The result shows that the performance of the PSO algorithm is the worst. The reason is that compared to SAPSO and SA algorithms, PSO can only find the locally optimal solution which corresponds to the lowest profit among three algorithms. The profit of SAPSO and SA algorithms are larger than that of PSO. Therefore, the performance of SAPSO and SA algorithms surpasses that of PSO. However, compared to the PSO algorithm, the profit of the SAPSO algorithm is larger than that of the SA algorithm in about 80% of time. The profit difference between the SAPSO algorithm and the SA algorithm varies from 1212.14$ to 15.7486$ in the about 80% of time. Compared to the SA algorithm, the profit of the SAPSO algorithm can be increased by 29.113% on average.

Fig. 14 illustrates the comparison of the iterations of the SAPSO algorithm and the baselines including the PSO and SA algorithms. It is clearly observed that the number of iterations of the SA algorithm is much larger than that of the PSO and SAPSO algorithms. The largest and the average number of iterations of the SA algorithm in all time slots are 1000 and 787, respectively. The average number of iterations of the SA algorithm is 9.8 times larger than that of the SAPSO algorithm, and 39 times larger than that of the PSO algorithm. This result shows that though the SA algorithm can find the close-to-optimal solution, it slowly converges to the final solution. Compared to the SA algorithm, the number of iterations of the SAPSO algorithm is much smaller. However, the SAPSO algorithm can find better solution to our problem in fewer iterations in comparison to the SA algorithm. Finally, the number of iterations of the PSO algorithm is the lowest, however, this result is caused by the quick convergence to a locally optimal solution which brings larger penalty and lower profit in comparison with the SA and the SAPSO algorithms.

Similarly, Fig. 15 illustrates the running time comparison of the SAPSO algorithm and the baselines including the PSO and SA algorithms. The result in Fig. 15 is consistent with the result in Fig. 14. The running time of the SA algorithm is the largest among three algorithms, however, the slow convergence of the SA algorithm can still only find the locally optimal solution. The average running time of the SA algorithm is 20.96 s, which is about 7 times larger than that of the SAPSO algorithm, 2.96 s, and about 69.8 times larger than that of the PSO algorithm, 0.3 s. The running time of the PSO algorithm is the lowest, however, this result is largely caused by the quick convergence to a locally optimal solution. The running time of the SAPSO algorithm is much smaller than that of the SA algorithm, and slightly larger than that of the PSO algorithm. However, the final solution of the SAPSO algorithm brings the largest profit and the lowest penalty in comparison to the SA and the PSO algorithms.

We also compare the CPU and memory utilization of SA, PSO, and SAPSO algorithms to make the following observations. Figs. 16 and 17 show the comparison of CPU and memory utilization of three algorithms, respectively. The CPU (memory) utilization of the PSO algorithm is the lowest among three algorithms. This reason is that the PSO algorithm quickly converges to a locally optimal solution. In addition, the SA algorithm also cannot find the globally optimal solution to our problem. The specified solution by the SA algorithm brings large penalty for our problem. Therefore, as is shown in Figs. 16 and 17, the CPU (memory) utilization of the SA algorithm is oscillating with time. However, the SAPSO algorithm can increase the diversity of solutions and provide global search capability. Therefore, as is shown in Figs. 16 and 17, the CPU (memory) utilization of

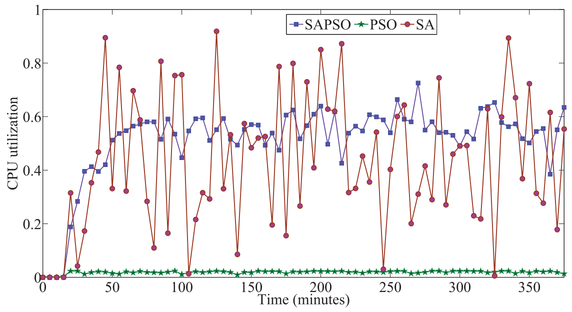


Fig. 16. CPU utilization of SAPSO, PSO, and SA algorithms.

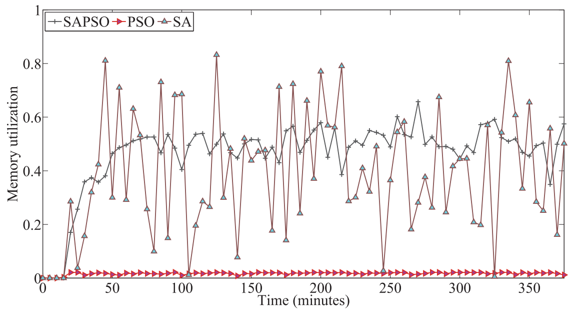


Fig. 17. Memory utilization of SAPSO, PSO, and SA algorithms.

the SAPSO algorithm is stable and varies from 40% to 75% in most of time slots.

VII. Conclusion

The economy of scale offered by cloud computing has attracted an increasing number of corporations to deploy their applications in cloud data centers. The uncertainty of the arrival tasks brings a big challenge for a private cloud to schedule all the arrival tasks while guaranteeing the service delay bound. In this paper, we study the profit maximization problem of a private cloud provider by utilizing the temporal variation of prices in hybrid cloud. To solve the problem, this paper proposes a profit maximization algorithm (PMA) to provide the temporal task scheduling which can dynamically schedule all the arrival tasks to execute in private and public clouds. Each iteration in the PMA algorithm is tackled by the proposed hybrid heuristic optimization algorithm, simulated annealing particle swarm optimization (SAPSO). Experimental results demonstrate that the proposed approach can greatly increase the profit and throughput of a private cloud while meeting the service delay bound. In the future, we would like to investigate the optimality gap between our current close-to-optimal solution and the theoretically optimal solution. In addition, we would like to implement a realistic cloud and evaluate our proposed scheduling method. Besides, we also would like to extend our work to consider scheduling of cloud storage tasks.

References

[1] L. Wu, S. K. Garg, S. Versteeg, and R. Buyya, “SLA-based resource provisioning for hosted software-as-a-service applications in cloud computing environments,” *IEEE Trans. Services Comput.*, vol. 7, no. 3, pp. 465–485, Jul. 2014.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Revenue maximization with optimal capacity control in infrastructure as a service cloud markets,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 261–274, Jul. 2015.

[3] D. Bruneo, “A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, Mar. 2014.

[4] K. Hwang, X. Bai, Y. Shi, M. Li, W. G. Chen, and Y. Wu, “Is the same instance type created equal? Exploiting heterogeneity of public clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, Jan. 2016.

[5] X. Zuo, G. Zhang, and W. Tan, “Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud,” *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 564–573, Apr. 2014.

[6] J. Mei, K. Li, A. Ouyang, and K. Li, “A profit maximization scheme with guaranteed quality of service in cloud computing,” *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3064–3078, Nov. 2015.

[7] J. Luo, L. Rao, and X. Liu, “Temporal load balancing with service delay guarantees for data center energy cost optimization,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 775–784, Mar. 2014.

[8] M. Menzel, R. Ranjan, L. Wang, S. U. Khan, and J. Chen, “CloudGenius: A hybrid decision support method for automating the migration of web application clusters to public clouds,” *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1336–1348, May 2015.

[9] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, “Exploring fine-grained resource rental planning in cloud computing,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 304–317, Jul. 2015.

[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. Autom. Sci. Eng.*, doi: 10.1109/TASE.2015.2427234, to be published.

[11] 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 Trans. Autom. Sci. Eng.*, doi: 10.1109/TASE.2015.2503325, to be published.

[12] H. Yuan, J. Bi, B. H. Li, and W. Tan, “Cost-aware request routing in multi-geography cloud data centres using software-defined networking,” *Enterprise Inform. Syst.*, 2015, doi: 10.1080/17517575.2015.1048833, to be published.

[13] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, “A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 153–161, Jan. 2015.

[14] W. Song, Z. Xiao, Q. Chen, and H. Luo, “Adaptive resource provisioning for the cloud using online bin packing,” *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2647–2660, Nov. 2014.

[15] T. Lu, M. Chen, and L. Andrew, “Simple and effective dynamic provisioning for power-proportional data centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1161–1171, Jun. 2013.

[16] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, “Dynamic heterogeneity-aware resource provisioning in the cloud,” *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan. 2014.

[17] R. Calheiros and R. Buyya, “Meeting deadlines of scientific workflows in public clouds with tasks replication,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1787–1796, Jul. 2014.

[18] P. Agrawal and S. Rao, “Energy-aware scheduling of distributed systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1163–1175, Oct. 2014.

[19] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, “Optimal multiserver configuration for profit maximization in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[20] J. Bi, H. Yuan, M. Tie, and W. Tan, “SLA-based optimisation of virtualised resource for multi-tier web applications in cloud data centres,” *Enterprise Inform. Syst.*, vol. 9, no. 7, pp. 743–767, Nov. 2015.

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

[22] R. Ranjan, R. Buyya, P. Leitner, A. Haller, and S. Tai, “A note on software tools and techniques for monitoring and prediction of cloud services,” *Software: Practice and Experience*, vol. 44, no. 7, pp. 771–775, May 2014.

[23] J. Diaz, C. Munoz-Caro, and A. Nino, “A survey of parallel programming models and tools in the multi and many-core era,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1369–1386, Aug. 2012.

[24] A. Riccardi, F. Fernandez-Navarro, and S. Carloni, “Cost-sensitive AdaBoost algorithm for ordinal regression based on extreme learning machine,” *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1898–1909, Oct. 2014.

[25] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, “Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 1131–1146, Jan. 2013.

[26] S. Aman, Y. Simmhan, and V. Prasanna, “Holistic measures for evaluating prediction models in smart grids,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 475–488, Feb. 2015.

[27] K. Deng, Y. Sun, S. Li, Y. Lu, J. Brouwer, P. G. Mehta, M. Zhou, and A. Chakraborty, “Model predictive control of central chiller plant with thermal energy storage via dynamic programming and mixed-integer linear programming,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 565–579, Apr. 2015.

[28] S. Lyden and M. E. Haque, “A simulated annealing global maximum power point tracking approach for PV modules under partial shading conditions,” *IEEE Trans. Power Electron.*, vol. 31, no. 6, pp. 4171–4181, Jun. 2016.

[29] R. Zou, V. Kalivarapu, E. Winer, J. Oliver, and S. Bhattacharya, “Particle swarm optimization-based source seeking,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 865–875, Jul. 2015.

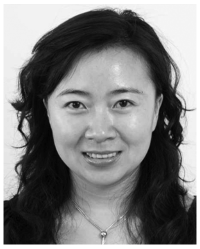
[30] S. Huda, J. Yearwood, and R. Togneri, “Hybrid metaheuristic approaches to the expectation maximization for estimation of the hidden markov model for signal modeling,” *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1962–1977, Oct. 2014.

[31] N. J. Cheung, X. M. Ding, and H. B. Shen, “OptiFel: A convergent heterogeneous particle swarm optimization algorithm for Takagi-Sugeno fuzzy modeling,” *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 919–933, Aug. 2014.

Haitao Yuan (S’15) received the B.S. and M.S. degrees from Northeastern University, Shenyang, China, in 2010 and 2012, respectively. He is currently working toward the Ph.D. degree at the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

He was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, in 2015. His research interests include cloud computing, resource allocation, and energy efficiency.

Mr. Yuan was the recipient of the 2011 Google Excellence Scholarship.



Jing Bi (M’13) received the Ph.D. degree from Northeastern University, Shenyang, China, in 2011.

She is currently an Assistant Professor with the School of Software Engineering, Beijing University of Technology, and the Beijing Engineering Research Center for IoT Software and Systems, Beijing, China. From 2013 to 2015, she was a Postdoctoral Researcher at the Department of Automation, Tsinghua University, China. From 2009 to 2010, she participated in research on cloud computing at the IBM China Research Laboratory. Her research interests include service computing, cloud computing, large-scale data analytics and resource optimization.

Dr. Bi was the recipient of the 2009 IBM Ph.D. Fellowship Award.

Wei Tan (M’12–SM’13) received the Ph.D. degree in automation from Tsinghua University, Beijing, China, in 2008.

He is currently a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. From 2008 to 2010, he was a Researcher with the Computation Institute, University of Chicago, Chicago, IL, USA, and Argonne National Laboratory, Lemont, IL, USA. His research interests include NoSQL, cloud computing, scientific workflows, and Petri nets.

Dr. Tan is a member of the Association for Computing Machinery. He is an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

Bo Hu Li graduated from Tsinghua University, Beijing, China, in 1961.

He was a Visiting Scholar with the University of Michigan, Ann Arbor, MI, USA, from 1980 to 1982. His research interests include cloud manufacturing, distributed simulation, high-performance computing, and cloud computing.

Mr. Li is currently a Fellow of the Chinese Academy of Engineering. He was the recipient of the Lifetime Achievement Award from the Society for Modeling and Simulation International (SCS).