This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

Providing Service Continuity in Clouds under Power Outage

Weiwei Wu, *Member, IEEE,* Jianping Wang, *Member, IEEE,* Kejie Lu, *Member, IEEE,* Wen Qi, Feng Shan, *Member, IEEE,* Junzhou Luo, *Member, IEEE*

***Abstract*—In cloud computing, it is crucial to maintain service continuity, while power outage is one of the most common and serious threats. To improve the resilience of cloud against power outage, a service provider usually deploys emergency energy supply (e.g., UPSs and generators) in a data center. When a power outage at a data center happens, the cloud service provider needs to make the operation decision on which subset of VMs to keep running and which servers to host such VMs to minimize its loss (or maximize its profit) using the emergency energy supply while the selected VMs are running in the affected data center until they are finished, migrated to other data centers, or normal power supply of the affected data center has been restored. No prior research has theoretically studied such a cloud service continuity problem under power outage. In this paper, we tackle this challenge and investigate the cloud service continuity problem. Specifically, we consider that a profit is associated with maintaining the continuity of a service, denoted as *service continuity profit*. Based on that we first formulate an optimization problem that aims to maximize the total profit subject to energy constrains. After showing the hardness of the problem, we focus on the design of approximation algorithms for solving the problem, where we consider two practical cases. In the first one with sufficient number of servers for re-provisioning, we develop a constant approximation algorithm of which the worst-case performance approaches the optimal solution within a constant factor (≈ 4.5-6.4). In the second one, we consider the general case with limited number of servers, and we develop an approximation algorithm with an approximation ratio of around 5.7-8. By combining these two algorithms together, we can achieve both good worst-case performance and average performance. Simulation results demonstrate the efficiency in terms of maximizing the service continuity profit of the proposed algorithms.**

***Index Terms*—Service continuity, power outage, cloud recovery, VM consolidation, energy-efficient scheduling, approximation algorithm, profit maximization.**

I. Introduction

Despite the salient features of cloud computing, cloud services may be interrupted due to various issues. As reported in [1], [2], data centers across the world suffer frequently from the outages and costs $600,000 per incident on average. Among the root causes of those outages, the top one is power outage [1]. For example, Amazon Web Services suffered an approximately 4-hour power outage in its North Virginia data center from 8:50 p.m. to 1:09 a.m. on June 14, 2012, causing major disruption to numerous web companies that rely on Amazon's cloud service, including Instagram, Netflix and Pinterest [3]. In 2015, a cloud data center in Hong Kong that is part of Alibaba Ltd., the biggest e-commence company of China, also suffered a 14-hour disruption due to power outage from 9:37 a.m. to 11:39 p.m. on June 21 [4].

Clearly, service interruption has significant negative impacts on both the cloud provider and customers, especially for the critical services sensitive to interruption. To improve the resilience of cloud services against power outage, there are two main approaches. First, a common practice in the cloud industry is to install emergency energy supply, such as UPSs and generators. Second, since a cloud service provider usually owns distributed data centers, it can migrate some services to other data centers that are not affected by power outage, while migration time varies among different services as it takes time to launch a Virtual Machine (VM) image, migrate storage and establish network connection. Based on these strategies, to maintain the continuity of a service, the VM supporting the service needs to keep operating locally until the minimum time point, named deadline, among the time points at which the service is finished, the VM supporting the service is successfully migrated to the new data center, or the normal power supply of the affected data center is back, which can be considered as the *continuity requirement*.

Although the approaches above are viable, we note that there is a lack of a theoretical study in the literature that addresses how to optimally exploit emergency energy and external data centers during power outage. In this paper, we tackle this challenging issue and investigate a *cloud service continuity* (CSC) problem under power shortage. Specifically, we consider that each service has a specific continuity requirement, and we define a *service continuity value/profit* associated with each VM/service if its continuity requirement is satisfied. Since the amount of emergency energy supply is limited, the CSC problem is an optimization problem whose objective is to maximize the total service continuity profit by identifying (VM selection) and re-provisioning a subset of existing VMs to *physical machines* (PMs) in the local data center, subject to the resource requirements of VMs, the resource capacity of PMs, as well as the limitation of emergency energy supply.

Now we use an example shown in Fig. 1 to illustrate that inefficient utilization of emergency energy may result

W. Wu, J. Luo and F. Shan are with School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu, P.R.China (Emails: {weiweiwu, jluo, shanfeng}@seu.edu.cn).

J. Wang and W. Qi are with Department of Computer Science, City University of Hong Kong, Hong Kong (Email: jianwang@cityu.edu.hk, qi.wen@my.cityu.edu.hk).

K. Lu is with the Department of Electrical and Computer Engineering, University of Puerto Rico at Mayaguez, PR, USA, and with the School of Computer Engineering, Shanghai University of Electric Power, Shanghai, China, (Email: kejie.lu@upr.edu).

The work is supported in part by Hong Kong Research Grant Council under GRF 120612 and CRF C7036-15G.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

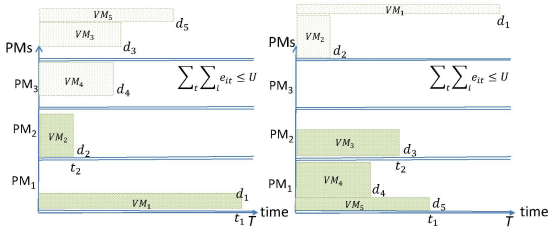


Fig. 1. Two exemplary schedules for providing service continuity. Each rectangle represents a VM with service continuity requirement, of which the height is its demand of resource and the width is the time period of continuity requirement required for running VM locally before the service is finished or transferred. The light-colored ones fail to keep their service continuity due to the lack of emergency energy.

in high loss of service continuity profit. In this example, when power outage strikes, there are five VMs requiring their service continuity to be satisfied with the support of an amount U of emergency energy under power outage. Each VM is represented as a rectangle where the height is its demand of resource (*e.g.,* CPUs) and the width is the time period of continuity requirement, during which it needs to keep running until the service hosted by the VM is finished or transferred. Suppose simply that there are three PMs in total, each with an identical capacity of resource. With limited amount of emergency energy, the first schedule chooses/attempts to maintain the service continuity of VM_1 and VM_2 by consolidating them to the first two PMs, each with one VM consolidated, where e_{it} is the power consumption of PM_i at time t. Suppose that the profit for keeping each VM's service continuity is approximately the size of its area. Then, another schedule that chooses VMs with similar width (VM_3, VM_4, VM_5) and attempts to fully utilize the resource/capacity of PMs, say, the second schedule, would gain more overall profit of service continuity under the same budget constraint of emergency energy. Compared with the second schedule, the first one spends too much power on maintaining the continuity of VM_1 and keeping PM_1 active, but fails to efficiently utilize the resource as well as the power over time. Therefore, there is a need for designing efficient scheduling strategy to maintain service continuity with maximum profit under power outage.

In the literature, such a CSC problem has not been investigated before but there are some related studies. One of the most relevant work is VM consolidation [5], [6], with which VMs can be re-provisioned and idle PMs can be shut down to reduce the power consumption. Although VM consolidation has been studied extensively in the past, most of the studies focus on minimizing the operational cost of a single data center (*e.g.* [7]–[10]) or geo-distributed data centers (*e.g.*, [11]–[13]). Moreover, all these prior work have assumed unlimited power supply which can support all VMs indefinitely, thus have no need to study VM selection strategy. This is different from CSC problem that exploits the limited emergency energy and needs to design both VM selection strategy and VM consolidation strategy under power shortage so as to maximize the total service continuity profit. Besides VM consolidation, there are some other related work concerning about cloud recovery, but few works have theoretically addressed the scheduling problem on providing service continuity during power outage. A full review can be referred to in Section II. To summarize, we note that existing studies cannot be applied to solve our CSC problem which has both different objectives and constraints.

To fully utilize the limited emergency energy, ideally if there are sufficient number of PMs for re-provisioning, we shall select and re-provision VMs to servers according to their service continuity requirements, i.e., time period during which they require to keep running in the affected data center. By doing this, VMs on a server could finish their services around the same time and then that server can be shut down. If the number of PMs for re-provisioning is limited, VMs with various service continuity requirements may have to be packed together, which will prolong the service time of such servers under power shortage. In this paper, we consider both aforementioned scenarios and develop algorithms with constant approximation ratios to maximize the service continuity profit.

The contributions of this paper are summarized as follows.

- This paper studies the service continuity problem for cloud data centers under power outage. This is the first work to theoretically study scheduling algorithms to exploit the emergency energy and maximize the profit of keeping service continuity in cloud data centers under power shortage.
- When the number of PMs for re-provisioning is sufficiently large, we propose a constant approximation algorithm that achieves a profit within around 4.5-6.4 times of the optimal solution. We first propose a procedure to get a bundle of VMs with high total profit, and then apply it iteratively to re-provision each bundle of VMs to a PM so as to gain high aggregated profit of service continuity.
- When the number of PMs for re-provisioning is limited, we propose a constant approximation algorithm (with an approximation ratio $\approx$ 5.7-8). We introduce a novel fractional version of the problem with well-organized optimal structures and transform its optimal solution to be a feasible solution of CSC problem with small loss of approximation.
- Finally, the two algorithms above are extended and combined to be a service continuity strategy that can achieve a good average performance, as well as a theoretical worst-case bounded performance. Simulation results verify that the average performance of the proposed strategy is close to the optimal profit that is achievable for the optimal solution.

The organization of the paper is as follows. Section II reviews the related work. Section III formulates the problem and provides the overview of our solutions. Section IV develops an approximation algorithm for the situation that the number of PMs for re-provisioning is sufficiently large. Section V studies the general case with the limitation on the number of PMs. Numerical results are presented in Section VI. We conclude the paper in Section VII.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

II. Related work

Much research effort has been devoted in developing energy-efficient scheduling algorithms in data centers [14]. We just review the ones most related to the problem/technique considered in this paper.

One of the techniques most related to the one adopted in this paper is VM consolidation [5], [6], with which VMs can be re-provisioned and idle PMs can be shut down to the efficiency of energy usage. Although VM consolidation has been studied extensively in the past, most of the studies focus on minimizing the operational cost of a single data center (*e.g.* [7]–[10]) or geo-distributed data centers (*e.g.*, [11]–[13]). Rich research works focus on optimizing the data center operation such as energy efficiency and quality of service (QoS). [15], [16] reduce the power consumption by reducing the number of idle servers. The works in [17], [18] leverage energy storage devices to avoid high electricity usage when the electricity price increases. Xu [19] proposes an idea of using partial execution to reduce the peak power demand and energy cost of data centers. [7], [20], [21] consider parameterized optimization objectives to minimize the energy cost and response time. [22] proposes to make temperature-aware workload management for geo-distributed data centers. [23], [24] consider geo-graphical electricity price diversity to reduce the operation cost of data centers. More related works can be referred to in survey papers [5], [6].

The service continuity problem studied in this paper needs to exploit the emergency energy and provide service continuity for a subset of VMs, that is carefully selected, to maximize the total profit of service continuity with the support of limited emergency energy. We note that a VM consolidation technique can be resorted to enhance the efficiency of emergency energy usage. However, all existing works in the literature of VM consolidation introduced above have assumed unlimited power supply, which can support all VMs indefinitely, and aim at minimizing the operation cost with the support of infinite energy supply. They cannot be applied to our scenario that exploits the limited emergency energy and needs to design both VM selection strategy and VM consolidation strategy under power short age, so as to maximize the total profit of keeping service continuity.

Prior to our work, only a few works have discussed the problem on cloud recovery or providing service continuity. Wood et al. [25] discuss about the possibility of providing the disaster recovery as a service in the cloud. [26] provides a solution to the recovery of IT services in the event of a disaster with the help of the cloud. Develder et al. [27] and Habib et al. [28] exploit the backup scheme to provide the resilience or protection ability in case of link or server failure. Few prior works have theoretically addressed the restoration scheme or the problem on how to schedule the VMs and provide service continuity with limited emergency energy under power outage.

In summary, we note that existing studies cannot be applied to solve the service continuity problem under consideration. To the best knowledge of the authors, this paper is the first work to theoretically study the scheduling problem on providing service continuity with maximum profit in cloud data center under power shortage.

III. Problem formulation and overview of our solution

In this section, we formulate the cloud service continuity problem and provide the overview of our solutions.

A. System model

Suppose the power supply outside the cloud data center is interrupted at time 0 due to a disaster, starting from which the only available power is the emergency energy (such as UPSs, power generator) with limited energy, and the outside power supply is expected to be restored at time T.

When the outside power supply is cut off, the cloud service provider can redirect some services to other data centers that are not affected by power outage. Since cloud services are realized by Virtual Machines (VMs), to maintain the continuity of a service, the VM supporting the service needs to keep operating locally until the time point, named deadline, at which the service is finished or the VM supporting the service is successfully launched in the new data center and ready for redirection. Let $J = \{1, 2, ..., n\}$ be the set of VMs that is running in the cloud data center before power outage occurs. To facilitate the scenario and capture the fundamental challenge in maintaining service continuity under power shortage, we measure the resource demand of the VMs by the request of CPUs or vCPUs (virtual CPUs) since the CPU usage usually takes up a significant share of the total power needed. Each VM j is assumed to have a *size/demand* s_j, requesting s_j units of CPUs or vCPUs. When power outage strikes, the VM supporting the service needs to keep operating locally until the minimum time point, named deadline, among the time points at which the service is finished, the VM supporting the service is successfully migrated to the new data center not affected by the power outage or the normal power supply is back. Let d_j be the deadline of VM j, which specifies a period $[0, d_j)$ of *continuity requirement* that is used to keep the VM operating and maintaining the service continuity. Note that such a continuity requirement can include both the time required for running VMs and the time caused by migrating the VMs (either to another PM or to the remote data center).

Thus, to satisfy its continuity requirement, VM j needs s_j units of resource (resource demand) in time interval $[0, d_j)$ for keeping operating. Assume that it contributes a value/profit p_j if its service continuity requirement is satisfied. Here, profit p_j will be called *service continuity profit* of VM j and deadline d_j will be called the *length* of VM j. We assume naturally that no new VM demands will be accepted during the period $[0, T]$ of the power outage, thus all VMs are available at time 0.

Let $I = \{1, 2, ..., m\}$ be the set of *physical machines (PMs)* in the cloud data center. Each PM is assumed to be identical, each with the same capacity V of resource.

B. Problem formulation

Due to the lack of emergency energy during power outage, not all VMs can keep operating until their deadlines of continuity requirements, thus the service continuity requirements

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

of some VMs may be violated. It is necessary to design both a VM selection strategy and a VM consolidation strategy so as to efficiently utilize the emergency energy and maximize the total service continuity profit of the VMs for the cloud provider.

A re-provision/consolidation strategy would consolidate the VMs to the PMs and shut down idle machines to efficiently utilize the energy. Define x_{ij} as an indicator to indicate if VM j is allocated/re-provisioned to PM i. That is, $x_{ij} = 1$ if VM j is executed on PM i and $x_{ij} = 0$ otherwise. Thus, we have

$$x_{ij} = \{0, 1\}, \quad \forall 1 \leq i \leq m, \forall 1 \leq j \leq n. \tag{1}$$

Due to the lack of emergency energy and the constraints of resource capacities, some VMs have to be interrupted before their deadlines, thus fail to keep the service continuity. That is, it is possible that $\sum_{1 \leq i \leq m} x_{ij} = 0$ for some VM j. In this case we say that they fail to be allocated/consolidated. Thus, before designing a consolidation strategy under power shortage, it is critical to determine a VM selection strategy: determine which set of VMs should be chosen and consolidated (with $\sum_{1 \leq i \leq m} x_{ij} = 1$).

After determining the VM selection, an efficient consolidation strategy under power shortage is required to consolidate the selected VMs by efficiently utilizing the emergency energy. Here, each VM j is executed/consolidated on at most one PM.

$$\sum_{1 \leq i \leq m} x_{ij} \leq 1, \quad \forall 1 \leq j \leq n. \tag{2}$$

Eq. (1) and (2) are called the *assignment constraints*. The number of PMs activated for re-provisioning/consolidation should be at most m, which is called the *PM constraint*.

Because of the resource constraints of PMs, the total demands of VMs consolidated on PM i must not exceed V in total. That is,

$$\sum_{1 \leq j \leq n} s_j x_{ij} \leq V, \quad \forall 1 \leq i \leq m. \tag{3}$$

This is called the *resource capacity constraint*. We say that VM j *fits* PM i if it can be allocated to PM i without exceeding the resource capacity.

Before modeling the power constraint, we first introduce the power consumption model adopted in this paper. Let $u_{it} \in [0, 1]$ be the CPU utilization rate of machine i at time t,

$$u_{it} = \sum_{j: t \in [0, d_j)} s_j x_{ij} / V, \quad \forall t, \forall i. \tag{4}$$

We say PM i is *active* at time t if $u_{it} > 0$, thus it uses one active/power-on PM time unit. As measured in prior works [15], [16], the power consumption of a PM approximately follows the function below,

$$e_{it} = \begin{cases} (1 - \alpha) u_{it}^{\mu} + \alpha, & \text{if } u_{it} > 0 \\ 0, & \text{if } u_{it} = 0 \end{cases} \tag{5}$$

where $\mu \geq 1$ and the peak power with full CPU utilization is normalized to be 1; the constant α is the idle power, which is typically ranging in $[0.5, 0.7]$ (and barely below than 0.5) as noted in practical measurements [15], [29], [30].

We measure the migration cost as the migration time used for migrating a VM to a new PM, since this term affects the energy overhead of migration as well as the migration latency, as measured in [31]. Note that instead of explicitly formulating the migration time of a VM, we have incorporated it into the time period $[0, d_j)$ of continuity requirement of each VM, as introduced in the definition of continuity requirement.

We assume that there are U units of emergency energy when power outage strikes. Equivalently, it is able to support a PM with full CPU utilization to execute U time units after normalization. A schedule should satisfy the *power constraint* that the total power consumed should be at most U,

$$\sum_{1 \leq i \leq m, 1 \leq t \leq T} e_{it} \leq U. \tag{6}$$

Obviously, PM i must be powered on until all assigned VMs finish their execution. Let t_i be the length/number of power-on time of PM i, then it should be no less than the required running time of any VM consolidated on the current PM. We thus have

$$t_i \geq d_j x_{ij}, \quad \forall 1 \leq j \leq n, \forall 1 \leq i \leq m. \tag{7}$$

For VM j, if its service continuity requirement is satisfied (with $\sum_{1 \leq i \leq m} x_{ij} = 1$), then the cloud service provider gains a *value/profit* p_j from VM j; otherwise, it gains a profit 0. If a VM is consolidated in the service continuity schedule, then we say the schedule *executes/starts* the VM, interchangeably.

Our objective is to maximize the aggregated profit of VMs whose service continuity requirement is satisfied, i.e.,

$$\sum_{1 \leq j \leq n} \left(p_j \sum_{1 \leq i \leq m} x_{ij} \right). \tag{8}$$

Therefore, the cloud service continuity problem under consideration is summarized in the following definition (ILP formulation).

Definition 1 (CSC problem)**.** *The Cloud Service Continuity (CSC) problem is to determine the allocation $\{x_{ij} | i \in I, j \in J\}$ to maximize the service continuity profit (8) with the satisfaction of the assignment constraint (1) and (2), the resource capacity constraint (3), the power constraint (6) and the constraint (7) and the PM constraint.*

We investigate the CSC problem from the approximation point of view. We say that an algorithm is γ-approximation if it always achieves a profit within $\frac{1}{\gamma}$ times that of the optimal solution for any input/instance I. That is,

$$\frac{ALG(I)}{OPT(I)} \geq \frac{1}{\gamma}, \ \forall I. \tag{9}$$

where $ALG(I)$ and $OPT(I)$ are respectively the service continuity profit achieved by the algorithm and the optimal solution.

C. Overview of our solutions

In CSC problem, we need to select and re-provision/consolidate the VMs to exploit the emergency energy and maximize the service continuity profit of VMs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

whose continuity requirements are satisfied using limited emergency energy.

The CSC problem (even consolidating with unlimited number of PMs) can be easily proved to be NP-hard by reducing the classical NP-hard knapsack problem to an instance of it. Therefore, in this paper, we focus on the design of approximation algorithms.

The high level idea of the design is as follows. We observe that the idle power α is a large constant compared with the peak power and a PM using U active PM time units spends at most U units of peak power (since it is normalized to be 1). Thus, algorithms that uses at most U active PM time units guarantee to satisfy the power constraints.

We first consider the situation that the number of PMs available for re-provisioning is sufficiently large in Section IV. This allows us to activate a new PM if necessary. We develop an algorithm BRP (Bundle Re-provision) to maximize the service continuity profit of the provider. BRP ensures that each bundle of VMs assigned to one PM is with high profit per unit of power-on time and is proved to be $\frac{2e}{(1-\delta)(e-1)\alpha}$-approximation compared to the optimal solution.

Next, in Section V, we consider the general case that the number of PMs available for re-provisioning has a limit and develop a final algorithm, called CSC-SCHEDULE, by combining two efficient algorithms introduced below to achieve a good average performance as well as a worst-case bounded performance.

With the restriction on the number of PMs for re-provision, the PMs should be carefully activated to ensure that each activated PM gains high total profit by allocating the VMs. One intuitive idea to deal with the new constraint is to extend BRP by greedily activating the PMs until all m PMs are used up. This revised algorithm is called BRP*. In general, BRP* has a good average performance. However, this may cause the problem that the activated PM gains low total profit when all chosen VMs have short lengths, although the profit per power-on time unit is high. As a matter of fact, this may make the worst-case performance be arbitrarily bad.

Thus, we combine BRP* with another algorithm, called FRP* (Fractional Re-provision), which will be proved constant approximation in terms of worst-case performance. In the design of FRP*, we first introduce a fraction version of CSC problem with nice optimal structures and develop a novel algorithm to find its optimal solution, and then we transform this solution back to be a feasible solution of the original problem with a loss of small approximation ratio. By returning the better one between BRP* and FRP*, we have the final schedule CSC-SCHEDULE.

IV. ALGORITHM DESIGN WITH SUFFICIENT NUMBER OF PMS

In this section, we consider CSC problem in clouds under a natural condition that the number of PMs available for re-provisioning is sufficiently large. Since a PM using U active PM time units guarantees to spent at most U peak power, we introduce a *time-constrained CSC problem* first where the original power constraint is replaced by the *time-constraint* that the total number of power-on time units is not allowed to exceed U. We will devise an algorithm for the time-constrained CSC problem, which tries to fully utilize the CPU resource at each unit of power-on time and meanwhile guarantees to satisfy the power constraints in the original CSC problem. Then, we will prove that the proposed algorithm is close to the optimal profit in time-constrained CSC problem, which will be further proved to approach the optimal solution with the original power constraints.

Here, a sufficient number of PMs does not mean that it is free to activate as many as wanted, due to the lack of emergency power. Thus, the VMs need to be carefully selected and consolidated to PMs to gain the maximum profit.

One intuitive and promising idea is to design a greedy algorithm that allocates the VMs in the order of decreasing profit-per-deadline and adopts first-fit strategy to allocate the VM to the first active PM and activate one new PM if no active PMs are available. However, a simple example can be easily found to show that such a greedy algorithm has unbounded approximation ratio, since it does not take the resource capacities of PMs and profit diversity of VMs into consideration, which may lead to the waste of resource of a PM as well as the waste of power or power-on time in the long run.

Our basic idea for solving the CSC problem is to consider a key sub-problem to guarantee the efficient usage of each power-on time unit in activating PMs and then adopt it as a building block to activate new PMs. The key sub-problem under consideration is, given a budget of U units of power-on time, maximizing the profit of VMs chosen to be executed in one single PM.

The arrangement of this section is as follows. We first develop a basic procedure to solve the key sub-problem above. Then, we propose an algorithm for the CSC problem by iteratively calling the basic procedure. Finally, we analyze the the proposed algorithm and prove its constant approximation.

A. A basic procedure

In this subsection, given t power-on time units, we examine the key sub-problem on how to select VMs from J to run on a single PM with capacity V during $[0, t)$, so that the total profit is maximized. Define $J^{\leq t} \subseteq J$ to be the set of VMs whose deadlines satisfy $d_j \leq t$. Obviously, only VMs with deadlines not greater than t can start on such a PM. Thus, we only need to consider the set $J^{\leq t}$.

We use a basic procedure to solve the problem above. This procedure will find a subset of VMs from a given set $\hat{J}$ such that the total profit is maximized with the constraint that the total resource demands of the selected VMs is at most V. Given a set $\hat{J}$ of VMs and a capacity V, let procedure PROFITABLEBUNDLE($V, \hat{J}$) return the desired subset and *profit*$(V, \hat{J})$ be the maximum profit.

Note that such a sub-problem can be reduced to be the transitional knapsack problem, thus it admits a dynamic programming algorithm. Let $p(j, v)$ be the maximum profit achievable using only the first j VMs in $\hat{J}$ with a capacity v. The profit $p(j, v)$ is either achieved by $p(j-1, v)$ when the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

j-th VM is not selected or by $p(j-1, v-s_j)+p_j$ when the j-th VM is selected. Thus, the recursion function is

$$p(j,v) = \max\{p(j-1,v), \\ p(j-1,v-s_j)+p_j\}. \quad (10)$$

Based on the recursion function above, we can implement PROFITABLEBUNDLE($V, \hat{J}$) by computing $p(|\hat{J}|, V)$. The time complexity is $O(nV)$ and pseudo-polynomial due to its dependency on value V. If needed, we can further remove the dependency on value V of the time complexity by introducing a pre-set small error factor δ, a value/constant that can be arbitrarily close to 0, so as to obtain a total profit arbitrarily close to $p(|\hat{J}|, V)$ within a factor of $(1-\delta)$ (namely, a fully polynomial-time approximation scheme, FPTAS), using $O(n^3\frac{1}{\delta})$ running time given any error bound δ. The detailed implementation is similar to the FPTAS for the well-known knapsack problem [32], thus is omitted here.

To solve the key sub-problem, we only need to invoke procedure PROFITABLEBUNDLE($V, J^{\leq t}$), where $J^{\leq t}$ is the set of VMs whose deadlines satisfy $d_j \leq t$.

B. Algorithm design

Based on the solution above, we are now ready to solve the CSC problem with sufficient number of PMs for re-provisioning.

The idea of the design is to activate one single PM to run a bundle of VMs (with the largest possible total profit per unit of power-on time) first at each iteration. And then, we iteratively activate a new PM while satisfying the power-on time-constraints as well as the power constraints. The intuition behind such a general idea is to efficiently utilize the resource capacity and achieve high profit when consuming each power-on time unit.

In detail, the first bundle of VMs is computed as follows. With the help of algorithm PROFITABLEBUNDLE, we compute the maximum profit *profit*$(V, \hat{J}^{\leq t})$ given any duration $[0, t)$. The first bundle is the set of VMs that achieves the maximum value $\frac{\textit{profit}(V,\hat{J}^{\leq t})}{t}$ among all possible integer $t \in [0, \min\{T, U\}]$. Then, the algorithm iteratively finds the rest of the bundles.

The detailed design of the algorithm is presented in BRP. Let A_k be the subset of VMs allocated in iteration k and $S_k = \cup_{1\leq i\leq k} A_i$. In iteration k, the algorithm calls the procedure PROFITABLEBUNDLE($V, (J\backslash S_{k-1})^{\leq t}$) to test every time $1 \leq t \leq \min\{T, U\}$ to find a value $\hat{t} = \arg\max_t \frac{\textit{profit}(V,(J\backslash S_{k-1})^{\leq t})}{t}$ and set A_k to be the VMs that achieve *profit*$(V, (J\backslash S_{k-1})^{\leq \hat{t}})$. It activates the k-th PM to allocate VMs in A_k and gets profit *profit*$(V, (J\backslash S_{k-1})^{\leq \hat{t}})$ with the cost of an amount E_k of power consumption. The algorithm iteratively finds the next set S_{k+1} and terminates either when all VMs are chosen or the total energy consumption exceeds the capacity U after adding the set S_{k+1}. Should this case occur, the algorithm returns the set with higher profit between S_k and $\bar{S}$, where $\bar{S}$ is the maximum profit achievable for one single PM with the input of VMs in J and U time units.

In terms of the time complexity, computing S_k would call the procedure PROFITABLEBUNDLE() at most $\min\{T, U\} \cdot \min\{m, U\}$ times, while computing $\bar{S}$ would call the procedure PROFITABLEBUNDLE() at most $\min\{T, U\}$ times, where typically we have $U > T$ and $U > m$ when power outage happens. Thus, the running time of BRP is $O(mTnV)$ when setting $\delta = 0$.

Algorithm 1 BRP(V, J, m)

```
Set A_0 = S_0 = ∅
Set k = 1, t̂ = 0
while J\S_{k-1} ≠ ∅ and U > 0 do
    Compute t̂ = arg max_{1≤t≤min{T,U}} profit(V,(J\S_{k-1})^{≤t}) / t
    if U − t̂ > 0 then
        A_k = ProfitableBundle(V, (J\S_{k-1})^{≤t̂})
        Activate the k-th PM to run the VMs in A_k.
        S_k = S_{k-1} ∪ A_k.
        Compute the amount of power, say E_k, consumed on
        on the k-th PM.
        k = k + 1, U = U − E_k.
    end if
end while
Compute t̄ = arg max_{1≤t≤{T,U}} profit(V, J^{≤t})
Let S̄ = ProfitableBundle(V, J^{≤t̄}).
if the profit in S_k is larger than that of S̄ then
    return the profit in S_k.
else
    return the profit in S̄.
end if
```

C. Approximation ratio analysis

Now we analyze the performance of the Algorithm BRP. Recall the definition of the time-constrained CSC problem and denote by OPT^c the subset of VMs scheduled by the optimal solution of the time-constrained CSC problem using at most U units of power-on time, and let $p(OPT^c)$ be the total profit of those VMs. Denote by $p(OPT)$ the optimal profit of the optimal solution in the original CSC problem. We first prove that the algorithm achieves within $\frac{2e}{(1-\delta)(e-1)}$ times of the optimal profit in time-constrained CSC problem where e is the natural constant and δ is the constant error factor introduced to bound the performance on the total profit.

Suppose that the algorithm finishes in p iterations and the algorithm returns set S_p at termination. Let A_k be the VMs returned in the k-th iteration, and $p(A_k)$ be the total profit of VMs in A_k, where $1 \leq k \leq p$.

We first prepare a basic property for the greedy rule of the algorithm. The following lemma states the fact that each bundle of VMs selected in the iterations is profitable.

Lemma 1. $p(S_k) - p(S_{k-1}) \geq (1-\delta)t_k \cdot \frac{p(OPT^c\backslash S_{k-1})}{U}$ *for all* $1 \leq k \leq p$.

Proof. Let t_k be the largest deadline of VMs in A_k. Let $opt(t, J)$ be the maximum achievable profit to run the VMs in J by activating only one PM with t time units. For the first

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

iteration, assuming t' is the value that achieves $opt(t', J)\frac{1}{t'} = \max_t opt(t, J)\frac{1}{t}$, we have $p(A_1)\frac{1}{t_1} = \max_t \textit{profit}(t, V, J)\frac{1}{t} = \textit{profit}(t', V, J)\frac{1}{t'} \geq (1-\delta)opt(t', J)\frac{1}{t'}$ where the last inequality holds by the fact that PROFITABLEBUNDLE returns a subset with at least a profit $\textit{profit}(t', V, J) \geq (1-\delta)opt(t', J)$. Moreover, $opt(t', J)\frac{1}{t'} = \max_t opt(t, J)\frac{1}{t} \geq p(OPT^c)\frac{1}{U}$ since $opt(t', J)\frac{1}{t'}$ is the maximum profit per power-on time that is achievable, and activating any PM in the optimal solution OPT^c of time-constrained CSC problem achieves at most the profit $opt(t', J)\frac{1}{t'}$ per unit time. Therefore, combining these inequalities, we have $p(A_1)\frac{1}{t_1} \geq (1-\delta)opt(t', J)\frac{1}{t'} \geq (1-\delta)p(OPT^c)\frac{1}{U}$.

Let t' be the value that achieves $opt(t', J\backslash S_{k-1})\frac{1}{t'} = \max_t opt(t, J\backslash S_{k-1})\frac{1}{t}$. Then, in later iteration k, we have $(p(S_k) - p(S_{k-1}))\frac{1}{t_k} = \max_t \textit{profit}(t, V, J\backslash S_{k-1})\frac{1}{t} = \textit{profit}(t', V, J\backslash S_{k-1})\frac{1}{t'} \geq (1-\delta)opt(t', J\backslash S_{k-1})\frac{1}{t'}$ where the first inequality holds because of the greedy rule in the k-th iteration applied in the algorithm and the last inequality is correct because the procedure PROFITABLEBUNDLE returns a subset with at least a profit $\textit{profit}(t', V, J\backslash S_{k-1}) \geq (1-\delta)opt(t', J\backslash S_{k-1})$. Moreover, $opt(t', J\backslash S_{k-1})\frac{1}{t'} = \max_t opt(t, J\backslash S_{k-1})\frac{1}{t} \geq p(OPT^c\backslash S_{k-1})\frac{1}{U}$, since $opt(t', J\backslash S_{k-1})\frac{1}{t'}$ is the maximum profit per cost that is achievable among the VMs $J\backslash S_{k-1}$ and activating any PM in OPT^c achieves at most the profit $opt(t', J\backslash S_{k-1})\frac{1}{t'}$ per unit time. Combining these inequalities, we have $(p(S_k) - p(S_{k-1}))\frac{1}{t_k} \geq (1-\delta)opt(t', J\backslash S_{k-1})\frac{1}{t'} \geq (1-\delta)p(OPT^c, J\backslash S_{k-1})\frac{1}{U}$. This completes the proof. □

Based on the relation between $p(S_k)$ and $p(S_{k-1})$ in every two adjacent iterations in Lemma 1, we can establish the relation between $p(S_k)$ and $p(OPT^c)$ in the following lemma.

Lemma 2. *$p(S_k) \geq (1-\delta)(1-\Pi_{i=1}^{k}(1-\frac{t_i}{U}))p(OPT^c)$ for all $1 \leq k \leq p$.*

Proof. Since $\frac{p(A_1)}{t_1} \geq (1-\delta)(\frac{p(OPT^c)}{U})$, it holds for the base case $k=1$, i.e., $p(A_1) \geq (1-\delta)\frac{t_1}{U}p(OPT^c)$. We prove the lemma by induction with the induction hypothesis $p(S_{k-1}) \geq (1-\delta)(1-\Pi_{i=1}^{k-1}(1-\frac{t_i}{U}))p(OPT^c)$. With the hypothesis, we have

$$\begin{aligned}
p(S_k) &\geq p(S_{k-1}) + (1-\delta)t_k \cdot \tfrac{p(OPT^c\backslash S_{k-1})}{U} \\
&\geq p(S_{k-1}) + (1-\delta)\cdot\tfrac{t_k}{U}(p(OPT^c) - p(S_{k-1})) \\
&\geq (1-\delta)(1-\tfrac{t_k}{U})p(S_{k-1}) + (1-\delta)t_k\cdot\tfrac{p(OPT^c)}{U} \\
&\geq (1-\delta)(1-\tfrac{t_k}{U})(1-\Pi_{i=1}^{k-1}(1-\tfrac{t_i}{U}))p(OPT^c) \\
&\quad +(1-\delta)t_k\cdot\tfrac{p(OPT^c)}{U} \\
&= -(1-\delta)\Pi_{i=1}^{k}(1-\tfrac{t_i}{U})p(OPT^c) \\
&\quad +(1-\delta)(1-\tfrac{t_k}{U})p(OPT^c) + (1-\delta)t_k\cdot\tfrac{p(OPT^c)}{U} \\
&= -(1-\delta)\Pi_{i=1}^{k}(1-\tfrac{t_i}{U})p(OPT^c) + (1-\delta)p(OPT^c) \\
&= (1-\delta)(1-\Pi_{i=1}^{k}(1-\tfrac{t_i}{U}))p(OPT^c)
\end{aligned}$$

where the first inequality holds by Lemma 1, the second inequality follows by $p(OPT^c\backslash S_{k-1}) \geq p(OPT^c) - p(S_{k-1})$, the fourth inequality holds by the induction hypothesis and the last three equalities follow by merging the items. □

Based on Lemma 1 and Lemma 2, we derive the worst-case performance bound of the algorithm compared with the optimal profit in time-constrained CSC problem.

Lemma 3. *Algorithm* BRP *achieves within $\frac{2e}{(1-\delta)(e-1)}$ times of the optimal profit in time-constrained CSC problem with at most U units of power-on time.*

Proof. Assume that A_{p+1} is the last set which makes t_{p+1} exceed the power-on time constraints with $\sum_{i=1}^{p+1} t_i \geq U$. We have

$$\begin{aligned}
&p(S_p) + p(A_{p+1}) \\
&\geq (1-\delta)(1-\Pi_{i=1}^{p+1}(1-\tfrac{t_i}{U}))p(OPT^c) \\
&\geq (1-\delta)(1-\Pi_{i=1}^{p+1}(1-\tfrac{t_i}{\sum_{i=1}^{p+1}t_i}))p(OPT^c) \\
&\geq (1-\delta)(1-(1-\tfrac{1}{p+1})^{p+1})p(OPT^c) \\
&\geq (1-\delta)(1-\tfrac{1}{e})p(OPT^c)
\end{aligned}$$

where the first inequality follows by Lemma 2, the second inequality holds by the fact $\sum_{i=1}^{p+1} t_i \geq U$, the last two inequalities hold by the fact that the minimum value of $\Pi_{i=1}^{p+1}(1-\frac{t_i}{\sum_{i=1}^{p+1}t_i})$ is achieved with $t_1 = t_2 = ... = t_{p+1}$ and approaches $\frac{1}{e}$ with large $p+1$. Accordingly, $\max\{p(S_p), p(A_{p+1})\} \geq \frac{p(S_p)+p(A_{p+1})}{2} \geq \frac{(1-\delta)(e-1)}{2e}p(OPT^c)$. Therefore, the algorithm returns $\max\{p(S_p), p(\bar{S})\} \geq \max\{p(S_p), p(A_{p+1})\} \geq \frac{(1-\delta)(e-1)}{2e}p(OPT^c)$, thus is within $\frac{2e}{(1-\delta)(e-1)}$ times of the optimal solution in time-constrained CSC problem with at most U units of power-on time. □

Finally, we prove the feasibility and constant approximation of the proposed algorithm in the original CSC problem, as concluded in the following theorem.

Theorem 1. *Algorithm* BRP *achieves $\frac{2e}{(1-\delta)(e-1)\alpha}$-approximation for CSC problem, i.e. approximately within* 4.5-6.4 *times of the optimal solution with typical idle power $\alpha \in [0.5, 0.7]$.*

Proof. Algorithm BRP schedules the VM execution with at most U units of power-on time, thus uses at most U units of (peak) power. Hence, it is feasible for the original CSC problem. Furthermore, it returns a profit that is at least $\frac{(1-\delta)(e-1)}{2e}p(OPT^c)$ where $p(OPT^c)$ is the optimal profit in time-constrained CSC problem with at most U units of power-on time. Let $p(OPT^{\hat{c}})$ be the optimal profit of time-constrained CSC problem with at most $\frac{U}{\alpha}$ units of power-on time. Obviously, $p(OPT^c) \geq \alpha p(OPT^{\hat{c}})$ since the number of power-on time units in $p(OPT^c)$ is α times that of $p(OPT^{\hat{c}})$. Moreover, it is easy to see that any feasible solution of the original CSC problem can utilize at most $\frac{U}{\alpha}$ units of power-on time according to the definition of idle power, thus is a feasible solution of the time-constrained CSC problem with $\frac{U}{\alpha}$ units of power-on time. Thus, we have $p(OPT^{\hat{c}}) \geq p(OPT)$. Therefore, Algorithm BRP returns a profit that is at least $\frac{(1-\delta)(e-1)\alpha}{2e}p(OPT)$ and hence is $\frac{2e}{(1-\delta)(e-1)\alpha}$-approximation for the original CSC problem. □

V. ALGORITHM DESIGN FOR GENERAL CSC PROBLEM

In this section, we study the general case in which there exists a limitation on the number of PMs for re-provisioning. We try to develop a schedule by combining two algorithms to achieve a good average performance as well as constant approximation in terms of worst-case performance.

According to the overview of our idea described in Section III, we can simply extend BRP to get the algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

BRP* for the general case, by iteratively activating new PMs until no more PMs are available. In general, this algorithm may perform well in average case, but a simple instance can be easily found to show that it has unbounded worst-case performance. Developing an algorithm with worst-case bounded performance for the general case is quite difficult. With the limit on the available PMs, packing VMs with similar deadlines to a PM, as BRP* does, may just choose VMs with short VMs and use up the PMs easily while achieving only a low total profit on each PM.

To address this challenge, we go another way around. Instead of directly addressing the original problem, we introduce a novel fractional version of the problem, called Fractional Time-constrained CSC (FCSC) problem, which has nice structural properties and can be solved optimally. Thus, we first develop a non-trivial algorithm to optimally solve it, and then transform the fractional solution back to be the (integral) solution of the original problem. Based on the optimality of the fractional solution returned for FCSC, we will ensure that the transformed solution would achieve a total profit close to the fractional solution, thereby achieving a low loss in approximation ratio.

The arrangement of this section is as follows. We first introduce the FCSC problem. Next, we develop an optimal algorithm for the fractional CSC problem. Then, we transform the solution of the fractional CSC problem back to be a feasible solution of the original problem and prove its constant approximation. Last, we combine the two algorithms developed to get the final service continuity strategy.

A. A fractional time-constrained CSC problem

In this subsection, we still define the time-constrained CSC problem to be the CSC problem with the power constraint replaced by the time-constraint that the total number of power-on time units used is at most U, and further introduce a fractional time-constrained CSC problem (FCSC). We will prove that the optimal solution of the fractional time-constrained CSC problem can be computed and transformed to be a feasible solution of the CSC problem later.

Recall that a feasible solution of time-constrained CSC problem integrally selects the VMs and re-provisions/consolidates them to the PMs with the satisfaction of resource capacity constraints and power-on time constraints. We can treat each VM as two dimensional rectangle, with one dimension to be the size/height and the other dimension to be the length/width. Re-provisioning the selected VMs to a PM with capacity V is equivalent to placing the rectangles into a large rectangle with height V.

We first define the *virtual PMs* before introducing the fractional CSC problem. Take each PM as a virtual PM such that a VM can request just a part (but not all) of its demand from a virtual PM. Further consider that all m virtual PMs form a large virtual PM with capacity mV. With such an assumption, the assignment constraint is relaxed and a VM is allowed to be allocated to two virtual PMs. For example, when VM j is allocated to two virtual PMs 1 and 2 respectively with demand s_j^1 and s_j^2, then virtual PMs 1 and 2 respectively need to provide s_j^1 and s_j^2 units of resource (and thus keep operating) during the period $[0, d_j)$. Thus, we assume that, when hosting the VMs, the virtual PM needs to keep operating until the largest deadline of the VMs allocated even if it just serves partial resource demand of the VM. The total execution time of the virtual PMs still should not violate the power-on time constraints.

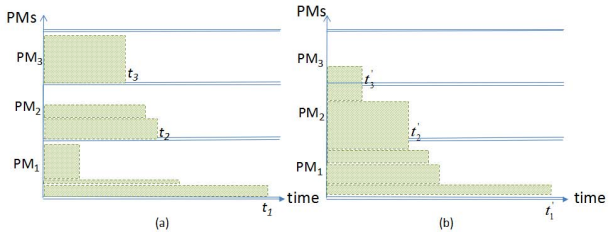


Fig. 2. An example showing how to transform the optimal solution of time-constrained CSC problem to be a feasible solution of FCSC problem without using more resources or power-on time.

With the assumptions of virtual PMs above, the *fractional time-constrained CSC problem* (FCSC problem) is defined as follows. It integrally selects a set of VMs and allocates them to the virtual PMs under both the resource capacity constraint on virtual PMs and the time-constraint that the total number of power-on time units used is at most U. The objective is to maximize the profit of VMs selected and allocated. Note that for FCSC problem, any set of VMs with total size not exceeding mV can be placed into the virtual PMs, without violating the resource capacity constraints since the VMs are allowed to be allocated to two virtual PMs. However, the power-on time-constraints on the virtual PMs may be violated.

An important property for the FCSC problem introduced above is that its optimal solution is at least that of the time-constrained CSC problem, thus can provide an upper bound for the original CSC problem. Its proof also indicates the existence of a novel algorithm for computing the optimal solution of the FCSC problem.

To prove that the optimal profit of FCSC problem is at least that of the time-constrained CSC problem, it is sufficient to transform the optimal solution of the time-constrained CSC problem to be a feasible solution of the FCSC problem. That is, we show that all the VMs selected in the optimal solution of time-constrained CSC problem can be allocated to virtual PMs and get a feasible solution for FCSC problem. Assume that J_{opt} is the set of VMs selected in the optimal solution of time-constrained CSC problem. We allocate the VMs in J_{opt} to the virtual PMs in the order of non-increasing length as follows. We allocate the first VM with the largest length to the first virtual PM, and then greedily allocate the next VM and activate a new virtual PM only when the resources on the current virtual PM are uses up. Fig. 2 demonstrates an example showing the transformation.

The resulting allocation can be proved to be a feasible solution for FCSC problem with the assumption that a VM is allowed to be allocated to two virtual PMs, which is concluded in the following lemma.

Lemma 4. *The optimal profit for FCSC problem is at least*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

that of the time-constrained CSC problem.

Proof. Let J_{opt} be the optimal solution of CSC problem. Note that the transformation takes J_{opt} as a input. The VMs in J_{opt} are sorted in the order of non-increasing length and allocated one by one to the virtual PMs. The proof is simply based on the observation that such an allocation always activates a new virtual PM when the resources on activated ones are fully occupied and the new virtual PM is activated using the minimum length of power-on time. Such a transformed allocation is obviously a feasible solution for FCSC problem and does not utilize more resources or power-on time than J_{opt} of CSC problem. This completes the proof. □

Consider the large virtual PM with mV units of resource that is formed by all virtual PMs to serve the VMs. Take as if each unit of the resource is taken from a space with addresses in $[1, mV]$. If the unit of resource in address $c \in [1, mV]$ is allocated to VM j, we say VM j *occupies* address c.

The transformation rule defined for proving Lemma 4 implies the following critical property of the optimal solution for FCSC problem, which will help us to design its optimal algorithm. That is, it makes the activated virtual PMs, except the last one, use up their resource. Thus, equivalently, we can say that VMs in J_{opt} *occupy* all the addresses in $[1, s_{opt}]$ or addresses in $[1, s_{opt}]$ are *fully occupied* by VMs in J_{opt}. In general, there is an immediate lemma for the optimal solution of FCSC problem following by Lemma 4.

Lemma 5. *Assuming that J_{fopt} is the set of VMs chosen by the optimal solution of FCSC problem, then there exists an optimal allocation for FCSC problem where VMs in J_{fopt} occupy all the addresses in $[1, s_{fopt}]$ with $s_{fopt} = \sum_{j \in J_{fopt}} s_j$, and moreover, a VM with larger length occupies the lower address.*

B. Optimal algorithm for FCSC problem

Based on the definition of FCSC problem and its properties introduced above, we develop a novel algorithm to optimally solve the FCSC problem. To find the optimal solution for FCSC problem, we still have to deal with the power-on time-constraints and PM constraint when VMs are allocated to virtual PMs.

The key idea is to develop a dynamic programming algorithm by seeking a recursion function (or recursive sub-problem) for FCSC problem. One possible idea, which is intuitive but may fail, is that we can try to find the maximum achievable profit using the addresses in $[1, c]$ and find the largest one among all possible c with $1 \leq c \leq \min\{mV, s_{sum}\}$ to find the optimal profit. With such an intuition, the method to compute the optimal profit for allocating n VMs is to find the larger value between the profit found in two recursive sub-problems, the maximum profit for allocating the first $n-1$ VMs in addresses $[1, c]$ when the n-th VM is not allocated and the maximum profit for allocating the first $n-1$ VMs in addresses $[1, c - s_n]$ when the n-th VM is allocated. However, such an intuition fails in dealing with the power-on time constraints when designing the recursion function. This is because, if VM j is allocated to a virtual PM that is not activated before allocation, then it needs d_j more power-on time; if VM j is allocated to a virtual PM that is already activated, then it may need no more power-on time. The problem above lies in the following fact: a recursion function for dynamic programming function needs to compute the function in a bottom-up manner; but when comparing these two sub-problems, we have no information on whether the PM before allocation is activated or not.

We note that the above intuition fails because when we consider the allocation for the current VM, we are lacking of the structure information for the recursive sub-problem before allocating the current VM.

Thus, we try to make more structural information for the problem. Note that there are multiple optimal allocations for FCSC problem that can achieve the same optimal profit. Recall that Lemma 5 implies that there exists an optimal solution for FCSC problem where the selected VMs fully occupy the addresses starting from address 1 and ending at address s_{fopt}, and furthermore, a VM with larger length is allocated first to lower address.

We utilize such a key property to design the recursion function. With such a property, we just try to find the optimal allocation which fully occupies the addresses in $[1, s_{fopt}]$ but ignore all other optimal allocations, and further resort the VMs so that $s_1 \geq s_2 \geq ... \geq s_n$. Such an idea restricts our search space, but allows us to assume that the virtual PM, whose resources are partially used but not used up before allocating the current VM, is already activated.

Thus, given an address space $[1, c]$, our objective is reduced to computing the maximum profit by selecting a subset J_c of VMs in the first k VMs so that the total size $\sum_{j \in J_c, j \leq k} s_j = c$ (which implies that all the VMs in J_c fully occupy the addresses in $[1, c]$ and meanwhile the last selected VM is ending at address c), and meanwhile, the virtual PMs for allocating such VMs use at most t units of power-on time. Let function $p(k, c, t)$ return such a maximum profit.

The definition above helps make the recursive sub-problem well-structured, which is able to lead to a novel recursive function below to compute $p(k, c, t)$. A tricky part of the proof for deriving this recursion function is how to identify the case that the addresses in $[1, c]$ are not possible to be fully occupied by integrally selecting from the first k VMs.

Lemma 6. *The recursion function to compute $p(k, c, t)$ is as follows,*

$$p(k,c,t) = \begin{cases} \max \begin{cases} p(k-1,c,t) \\ p(k-1,c-s_k,t), \\ \qquad \textit{if } c - s_k > V \cdot (\lceil \frac{c}{V} \rceil - 1) \\ p(k-1,c-s_k,t-d_k), \\ \qquad \textit{if } 0 \leq c - s_k \leq V \cdot (\lceil \frac{c}{V} \rceil - 1) \end{cases} \\ 0, \quad \textit{if } c = 0 \textit{ and } t \geq 0 \\ -\infty, \quad \textit{if } j = 0 \textit{ and } c > 0 \\ -\infty, \quad \textit{if } c - s_k < 0 \\ -\infty, \quad \textit{if } t < d_k \end{cases} \tag{11}$$

Proof. Recall the definition of $p(k, c, t)$. If $p(k, c, t)$ is the optimal profit of FCSC problem, then this implies that $c = s_{jopt}$ and the chosen VMs fully occupy the addresses in $[1, c]$ and the lower address is occupied by the VM with larger length.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

According to the definition of the recursion function, if $p(k,c,t)$ is the optimal solution, then this implies that $c = s_{opt}$ and all addresses in $[1, c-s_k]$ are occupied before allocating VM k. Thus, we can simply assume that all addresses in $[1, c-s_k]$ are already occupied and the first $\lceil \frac{c-s_k}{V} \rceil$ virtual PMs that contribute at least one resource in addresses $[1, c-s_k]$ is already activated.

With such a structural information, we know VM k should occupy addresses in $[c-s_k+1, c]$ that is belonging to the $\lceil \frac{c}{V} \rceil$-th virtual PM. If $\lceil \frac{c}{V} \rceil - \lceil \frac{c-s_k}{V} \rceil = 0$ or equivalently $c - s_k > V \cdot (\lceil \frac{c}{V} \rceil - 1)$, then the $\lceil \frac{c}{V} \rceil$-th virtual PM already hosts at least one VM among the first $k-1$ VMs. This implies that it is already activated for execution with larger than d_k time units before allocating VM k. Thus, no more power-on time need to be used to activate the $\lceil \frac{c}{V} \rceil$-th virtual PM when allocating VM k. If $\lceil \frac{c}{V} \rceil - \lceil \frac{c-s_k}{V} \rceil = 1$, then the $\lceil \frac{c}{V} \rceil$-th virtual PM does not host any VM before allocating VM k and d_k more units of power-on time should be used to activate it. Consequently, when VM k is selected, we can set $p(k,c,t) = p(k-1, c-s_k, t) + p_k$ if $c - s_k > V \cdot (\lceil \frac{c}{V} \rceil - 1)$ and $p(k,c,t) = p(k-1, c-s_k, t-d_k) + p_k$ if $0 \le c - s_k \le V \cdot (\lceil \frac{c}{V} \rceil - 1)$. If VM k is not selected, then we have $p(k,c,t) = p(k-1,c,t)$. Obviously, we need to set $p(\cdot, 0, t) = 0$ if $c = 0$ and $t \ge 0$ for the initialization. The analysis above has specified the main idea of the recursion function, which relies on the assumption that the address in $[1, c]$ are fully occupied.

However, one more critical issue has not been addressed, which is also a tricky part in the design. Since the dynamic programming algorithm needs to compute a general $p(k,c,t)$ in a bottom-up manner, we need to tackle the case that the addresses $[1, c]$ are not possible to be fully occupied by selecting from the first k VMs.

To identify this situation, we set $p(k,c,t) = -\infty$ when using the first k VMs is impossible to fully utilize the resources in $[1, c]$. Thus, we need to identify the condition with which $p(k,c,t)$ is set to be $-\infty$. In general, the last selected VM will occupied the address c according to the definition of $p(k,c,t)$. We discuss all possible cases of the recursion function. If $t < d_k$, then this implies any VM among the first k VMs has a length larger than t, thus it is impossible to fully utilize the addresses in $[1, c]$ when $c \neq 0$ and hence $p(k,c,t) = -\infty$. If $c - s_k < 0$ then k should not be allocated and $p(k,c,t) = p(k-1,c,t)$. If $j = 0$ and $c > 0$, then obviously $p(k,c,t) = -\infty$. This completes the proof in deriving the recursive function. □

Recall that we want to find the optimal allocation that the selected VMs fully occupy the addresses in $[1, c]$. Accordingly, the defined function $p(k,c,t)$ returns a positive value only when the addresses in $[1, c]$ are fully occupied by the selected VMs.

Finally, to find the optimal profit for FCSC problem, denoted by $OPT(FCSC)$, we need to enumerate $c \in [1, \min\{mV, s_{sum}\}]$ to find exactly the value $c = s_{fopt}$. Thus, the optimal profit for FCSC problem can be computed by,

$$OPT(FCSC) = max_{1 \le c \le \min\{mV, s_{sum}\}} p(n,c,U). \quad (12)$$

Algorithm 2 FRACTIONALCSC(J, U, m)

for $c = 1$ **to** $\min\{mV, s_{sum}\}$ **do**
  Set $p(0, \cdot, \cdot) = -\infty$.
  Set $p(\cdot, 0, t) = 0$ for all $t \ge 0$.
  for $k = 1$ **to** n **do**
    for $cc' = 1$ **to** c **do**
      for $t = 1$ **to** U **do**
        if $t < d_k$ or $cc' < 0$ **then**
          $p(k, cc', t) = -\infty$
        else
          $temp1 = p(k-1, cc', t)$
          if $cc' - s_k > V \cdot (\lceil \frac{cc'}{V} \rceil - 1)$ **then**
            $temp2 = p(k-1, cc' - s_k, t) + p_k$
          else if $0 \le cc' - s_k \le V \cdot (\lceil \frac{cc'}{V} \rceil - 1)$ **then**
            $temp2 = p(k-1, cc' - s_k, t - d_k) + p_k$
          else
            $p(k, cc', t) = -\infty$
          end if
          $p(k, cc', t) = \max\{temp1, temp2, temp3\}$
        end if
      end for
    end for
  end for
  $retP = \max\{p(n, c, U)\}$
  Find the set of selected VMs S in the recursion function with a backward search.
end for
return $(retP, S)$

More details can be found in Algorithm FRACTIONALCSC which implements the dynamic programming method stated above. The following theorem concludes the optimality of FRACTIONALCSC.

Theorem 2. *Algorithm* FRACTIONALCSC *optimally solves FCSC problem in* $O(m^2nUV^2)$ *steps.*

Proof. The optimality of FRACTIONALCSC follows directly from the recursion function we derived.

To calculate the recursion function $p(k,c,t)$, we need to enumerate n possible values of the first parameter, at most mV values of the second parameter and U possible values of the last one. In the final step, to find the value $\max\{p(n,c,U)\}$, another mV possible values should be enumerated. Therefore, the time complexity is $O(m^2nUV^2)$. □

C. Transforming from fractional solution to integral solution

Now we transform the optimal solution of FCSC problem to be a feasible solution of the CSC problem without losing much performance. Recall that in FCSC problem, it allows the VMs to be divisible when they are assigned to PMs. Therefore, we need to transform the fractional solution to be the (integral) solution of CSC problem.

The main difficulty lies in the fact that the transformation may result in the utilization of more than m PMs and more than U units of power-on time. The transformation idea is based on the fact that the optimal allocation for FCSC problem

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

fully occupies the addresses in $[1, s_{fopt}]$, as stated in Lemma 4, which can be used to control the usage in the number of PMs that is needed for transformation.

Thus, we use m PMs and U units of power-on time to recover the fraction solutions that uses $\lfloor \frac{m}{2} \rfloor$ PMs and $\lfloor \frac{U}{2} \rfloor$ power-on time. For the ease of discussion, we only discuss the case that $\frac{U}{2}, \frac{m}{2}$ are integers since the flooring operation does not affect the constant approximation of the algorithm.

Let $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ be the optimal solution of FCSC problem with the input of $\frac{U}{2}$ power-on time and $\frac{m}{2}$ PMs. According to Lemma 5, all VMs fractionally assigned to PMs in $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ are allocated to at most two PMs. Thus, we can assign the PMs in $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ one by one in the order of non-increasing length and activate at most twice of the PMs to integrally assign all these VMs. More details can be found in Algorithm FRP*.

In terms of the time complexity of Algorithm FRP*, computing the optimal solution $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ would cost $O(m^2nUV^2)$ time when calling Algorithm FRACTIONALCSC, while allocating the VMs chosen by FRACTIONALCSC would cost at most $O(nm)$ time. Thus, the running time of Algorithm FRP* is $O(m^2nUV^2)$. Fig. 3 demonstrates an exemplary solution returned by Algorithm FRP*.

According to Lemma 4, the algorithm for FCSC problem finds the optimal allocation that fully occupies addresses in $[1, s_{fopt}]$ and a VM with larger length occupies lower address. Thus, activating twice the number of PMs and power-on time used in FRACTIONALCSC($J, \frac{U}{2}, \frac{m}{2}$) is enough to get an integral solution, which satisfies the PM constraints and power-on time constraints in the time-constrained CSC problem. Hence, it further satisfies the power constraint of the original CSC problem since using U units of power-on time implies that the total power used is at most U units of (peak) power. Therefore, the algorithm outputs a feasible solution for CSC problem.

Algorithm 3 FRP*(V, J, m)

```
1: (p, S)= FractionalCSC(J, U/2, m/2)
2: sort VMs in S by their length in a non-decreasing order
3: for each ĵ in S do
4:    if VM ĵ is fractionally selected then
5:       let VM j be the VM with full demand corresponding
         to ĵ.
6:    else
7:       j = ĵ
8:    end if
9:    if VM j can be put into the current PM then
10:      put VM j into the current PM
11:   else
12:      activate a new PM and put VM j into the new PM
13:   end if
14:   profit = profit + p_j
15: end for
16: return profit
```

Based on the the discussion above, we further prove the constant approximation of Algorithm FRP* by applying the upper bound established in Lemma 4.

Theorem 3. *Algorithm* FRP* *is* $\frac{4}{\alpha}$*-approximation for CSC problem, i.e. approximately within* 5.7-8 *times of the optimal solution with typical idle power* $\alpha \in [0.5, 0.7]$.

Proof. The profit achieved in Algorithm FRP* is no less than that of $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ since all VMs selected in $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ are executed. That is, FRP*$(J, U, m) \geq OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$. Obviously, we have $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2}) \geq \frac{1}{4}OPT(FCSC, J, U, m)$ because the input VMs are the same and the input parameters are scaled by a factor of 2. Moreover, assuming that $OPT(CSC^c, J, U, m)$ is the optimal solution of the time-constrained CSC problem, then $OPT(FCSC, J, U, m) \geq OPT(CSC^c, J, U, m)$ according to Lemma 4. Therefore, it is true that FRP*$(J, U, m) \geq OPT(FCSC, J, \frac{U}{2}, \frac{m}{2}) \geq \frac{1}{4}OPT(FCSC, J, U, m) \geq \frac{1}{4}OPT(CSC^c, J, U, m)$. Let $OPT(CSC)$ be the optimal profit of CSC problem. Applying similar proof in Theorem 1, we can further derive that $OPT(CSC^c, J, U, m) \geq \alpha OPT(CSC)$. Thus, FRP*$(J, U, m) \geq \frac{\alpha}{4}OPT(CSC)$ and FRP* is $\frac{4}{\alpha}$-approximation. □

D. Combining the results

As what is shown above, Algorithm FRP* has a constant bound in terms of worst-case performance. In order to further enhance its practical significance, we combine it with BRP* that has a good average performance in general, so as to achieve both good average performance and theoretically worst-case bounded performance. The method is to simply run these two algorithms once and return the one that achieves higher profit. Algorithm CSC-SCHEDULE presents the final algorithm.

Algorithm 4 CSC-SCHEDULE(V, J, m)

```
1: profit_1 = BRP*(V, J, m)
2: profit_2 = FRP*(V, J, m)
3: return max{profit_1, profit_2}
```

Obviously, CSC-SCHEDULE has the same constant approximation ratio as FRP*, thus its worst-case performance is well-bounded.

VI. SIMULATION RESULTS

The theoretical analysis has verified the worst-case performance bounds of the algorithm proposed in this paper. In this section, we perform simulations for the algorithm to further validate its average performances over the achievable profit.

No prior works have addressed the service continuity problem studied in this paper, thus we compare our algorithm with the natural greedy schedule mentioned in Section IV. Furthermore, we compare the performance of our algorithm with the optimal solution. Since computing the optimal solution for the CSC problem is NP-hard, we compare the performance of CSC-SCHEDULE with the upper bound of the optimal solution, denoted as *OPT_UB*, which is obtained by relaxing the ILP formulation to be LP formulation with $x_{ij} \in [0, 1]$. Since no real-trace has provided a dataset similar to that of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

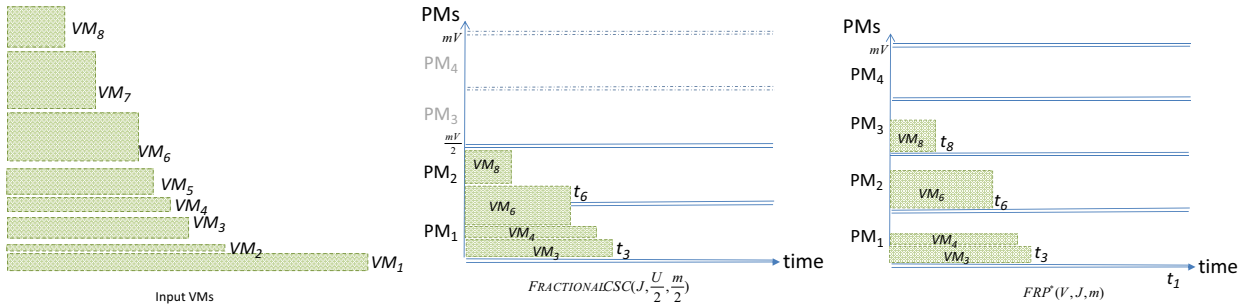


Fig. 3. An example showing the results returned by Algorithm FRP* where U is set with $U = 2(d_3 + d_6)$ and the profit of each VM/rectangle is equal to the size of its area. (a) All VMs sorted by length, as the input of the dynamical programming algorithm FRACTIONALCSC. (b) A solution returned by Algorithm FRACTIONALCSC(J, $\frac{U}{2}$, $\frac{m}{2}$) using $\frac{U}{2}$ units of power-on time and $\frac{m}{2}$ PMs, which is the optimal solution of the fractional time-constrained CSC problem, $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$. (c) Solution returned by FRP*(V, J, m), which is a feasible solution of CSC problem transformed from $OPT(FCSC, J, \frac{U}{2}, \frac{m}{2})$ using at most U units of power-on time and m PMs.

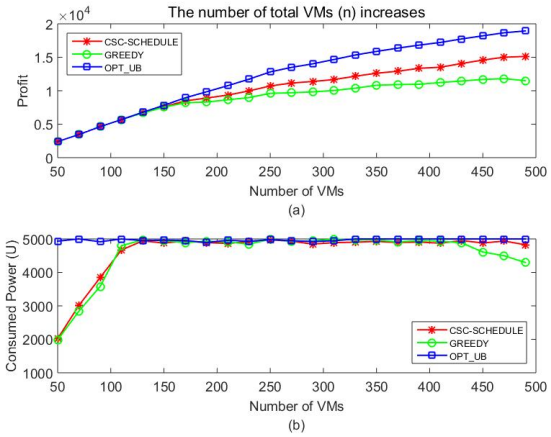


Fig. 4. (a) Profit achieved when the number of VMs varies. (b) Amount of energy consumed when the number of VMs varies.

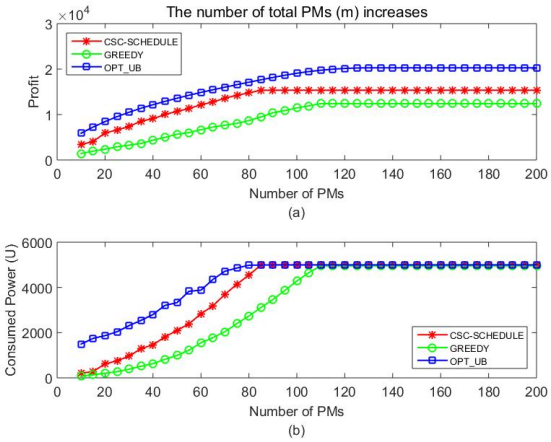


Fig. 5. (a) Profit achieved when the number of PMs varies. (b) Amount of energy consumed when the number of PMs varies.

the continuity maintenance problem with VM requirements considered in this paper, we consider random input of VMs in our simulations.

In the simulation, we set the power-restored time $T = 100$(min). We assume that there are $m = 100$ homogeneous PMs, each with 32 CPU cores. Each PM with full utilization would consume one (normalized) unit of energy per minute. We set the power budget to be $U = 5000$. The deadline of each VM is assumed to be a random number in $[1, T]$. The demand of each VM (number of CPU cores requested) is assumed to be an integer randomly ranging in $[1, 32]$. The profit of the VMs is uniformly generated from $[1, 100]$. The parameters of the power consumption function are set with $\alpha = 0.5, \mu = 1$.

1) Impact of the number of VMs: Fig. 4(a) shows the change in the profit achieved by different schedules when the number of VMs increases in the range of $[50, 500]$ with a step of 20. The curves are generated by increasing the number of VMs, where each point on the curves of the results is generated in a single run. Generally, the profit of the algorithms increases with the number of VMs. The profit achieved by our algorithm CSC-SCHEDULE is much higher (up to 25%) than the greedy schedule does. Moreover, we can see from the figure that the profit achieved approaches that of *OPT_UB* (within 80%), thus is even closer to the optimal solution.

Fig. 4(b) illustrates the amount of power used by different schedules when the number of VMs varies. At the beginning, the power used by CSC-SCHEDULE (and Greedy) increases with the rise of the number of VMs and then reaches the limit $U = 5000$ with $n \geq 150$, where U becomes the bottleneck, while *OPT_UB* almost always uses up the energy since it can produce fractional (but maybe infeasible) solutions. We can see from Figure 4(a) and 4(b) that the profit achieved by the three schedules are nearly the same before the available power becomes the bottleneck, and after that, the profit achieved differs much for the three schedules.

2) Impact of the number of PMs: Fig. 5(a) shows the profit achieved by the three schedules when the number of PMs varies. In this scenario, the number of PMs $m \in [10, 200]$, the number of VMs $n = 500$, and the amount of power-on time $U = 5000$. The profit obtained by CSC-SCHEDULE is about

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2728795, IEEE Transactions on Services Computing

BRP*	FRP*	CSC-SCHEDULE	Greedy
$O(mnVT)$	$O(m^2nV^2U)$	$O(mnV \cdot \max\{T, mVU\})$	$O(n \cdot \max\{\log n, m\})$

TABLE I
TIME COMPLEXITIES OF THE ALGORITHMS

20%-40% higher than that of the greedy schedule. Moreover, it achieves a profit close to (within 75%) that of *OPT_UB*, thus even approaches that of the optimal solution. The profit of the algorithms increases with the number of PMs and becomes stable when $m \geq 110$.

Fig. 5(b) shows the amount of power consumed by different schedules when the number of PMs varies. The trends of lines in this figure are similar. The amount of power used increases when $m \leq 100$ and becomes stable after that because all schedules almost use up the energy. We can see from Fig. 5(a) and 5(b) that the profit achieved by CSC-SCHEDULE and Greedy becomes stable when the available power is used up.

The simulations above have demonstrated the good average performance on maximizing the profit of our proposed algorithm. In Table I, we further compare the time complexity of the algorithms. Although the greedy algorithm runs fastest (with $O(n \cdot \max\{\log n, m\})$ time) due to its greedy nature, it achieves 25% less profit in providing service continuity than CSC-SCHEDULE does, as demonstrated in the simulation results above. The running time of CSC-SCHEDULE depends on the maximum one between BRP* and FRP*. Algorithm BRP* runs in $O(mnVT)$ time, while Algorithm FRP* runs in $O(m^2nV^2U)$ time. In our simulation, we found that around 98% of the solutions returned by Algorithm BRP* have higher profit that of FRP*. Thus, in practice, we can just return the solution of BRP* as the final solution of CSC-SCHEDULE to significantly reduce its running time to be $O(mnVT)$ without losing much profit.

Combining with the theoretical bound derived for the worst-case performance, these together verify the efficiency of our proposed algorithm.

VII. FUTURE WORKS AND CONCLUDING REMARKS

In this paper, we introduce and theoretically study the scheduling problem to maintain cloud service continuity with maximum profit under power shortage. We develop efficient scheduling algorithms with theoretical guarantees/small approximation ratios to maximize the profit of VMs of which the service continuity requirements are satisfied.

This work has conducted a theoretical study on how to provide service continuity under power outage. Our theoretical study can shed some light on the service continuity strategy design under power shortage. In our preliminary study, we focus on the resource of CPUs and vCPUs considering that the CPU usage takes up significant share of power needed. In future works, it is worth extending the study to the case with multiple types of resources. Although the basic algorithmic idea introduced in this paper can be extended to be adaptive to the multi-resource scenario, however, it is quite challenging to develop algorithms with theoretical performance guarantee since there exist multiple capacity constraints with respect to the multiple types of resources. Thus, we would like to leave it as an open problem and study it in future work.

REFERENCES

[1] Q. P. S. Team, "Average cost of data center outages: $627,418 per incident," in *http://www.qpsolutions.net/2015/03/average-cost-of-data-center-outages-627418-per-incident/*. Quality Power Solutions Ltd, March 18, 2015.

[2] Data center knowledge, in *http://www.datacenterknowledge.com/archives/category/manage/uptime/*. Penton Ltd, website.

[3] G. Smith, "Amazon power outage exposes risks of cloud computing," http://www.huffingtonpost.com/2012/07/02/amazon-power-outage-cloud-computing_n_1642700.html, 2012.

[4] Y. Qu, G. Yang, and X. Liu, "Aliyun data center in hk suffers 14-hour disruption," http://english.caixin.com/2015-06-24/100822037.html, 2015.

[5] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 445–468, 2013.

[6] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, 2014.

[7] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *TON*, vol. 21, no. 5, pp. 1378–1391, 2013.

[8] B. Guenter, N. Jain, and C. Williams, "Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning," in *INFOCOM'11*, 2011, pp. 1332–1340.

[9] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *CLOUD '09*, 2009, pp. 17–24.

[10] M. A. Adnan, R. Sugihara, and R. K. Gupta, "Energy efficient geographical load balancing via dynamic deferral of workload," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 188–195.

[11] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4, 2009, pp. 123–134.

[12] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser, "Renewable and cooling aware workload management for sustainable data centers," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, 2012, pp. 175–186.

[13] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *ICDCS'12*, 2012, pp. 22–31.

[14] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 445–468, 2013.

[15] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ACM Sigplan Notices*, vol. 44, no. 3, 2009, pp. 205–216.

[16] C.-C. Lin, P. Liu, and J.-J. Wu, "Energy-efficient virtual machine provision algorithms for cloud systems," in *UCC'11*, 2011, pp. 81–88.

[17] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *SIGMETRICS*, 2011.

[18] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. K. Fathy, "Energy storage in datacenters: What, where and how much?" in *SIGMETRICS*, 2012.

[19] H. Xu and B. Li, "Reducing electricity demand charge for data centers with partial execution," in *international conference on Future energy systems*, 2014.

[20] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *SIGMETRICS11*, 2011, pp. 233–244.

[21] H. Lim, A. Kansal, and J. Liu, "Power budgeting for virtualized data centers," in *USENIX ATC11*, 2011, p. 59.

[22] H. Xu, C. Feng, and B. Li, "Temperature aware workload management in geo-distributed datacenters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 373–374, 2013.

[23] Z. Xu and W. Liang, "Minimizing the operational cost of data centers via geographical electricity price diversity," in *Cloud Computing*, 2013.

[24] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 123–134.

[25] T. Wood, E. Cecchet, K. Ramakrishnan, P. Shenoy, J. Van Der Merwe, and A. Venkataramani, "Disaster recovery as a cloud service: Economic benefits & deployment challenges," in *2nd USENIX workshop on hot topics in cloud computing*, 2010, pp. 1–7.

[26] M. Klems, S. Tai, L. Shwartz, and G. Grabarnik, "Automating the delivery of it service continuity management through cloud service orchestration," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, 2010, pp. 65–72.

[27] C. Develder, J. Buysse, B. Dhoedt, and B. Jaumard, "Joint dimensioning of server and network infrastructure for resilient optical grids/clouds," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 5, pp. 1591–1606, 2014.

[28] M. F. Habib, M. Tornatore, M. De Leenheer, F. Dikbiyik, and B. Mukherjee, "A disaster-resilient multi-content optical datacenter network architecture," in *Transparent Optical Networks (ICTON), 2011 13th International Conference on*. IEEE, 2011, pp. 1–4.

[29] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services." in *NSDI*, vol. 8, 2008, pp. 337–350.

[30] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[31] W. Dargie, "Estimation of the cost of vm migration," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2014, pp. 1–8.

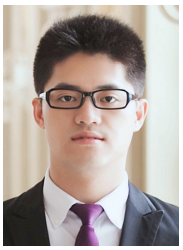
[32] E. L. Lawler, "Fast approximation algorithms for knapsack problems," *Mathematics of Operations Research*, vol. 4, no. 4, pp. 339–356, 1979.

Weiwei Wu is an associate professor in Southeast University, P.R. China. He received his BSc degree in South China University of Technology and the PhD degree from City University of Hong Kong (CityU, Dept. of Computer Science) and University of Science and Technology of China (USTC) in 2011, and went to Nanyang Technological University (NTU, Mathematical Division, Singapore) for post-doctorial research in 2012. His research interests include optimizations and algorithm analysis, wireless communications, crowdsourcing, cloud computing, reinforcement learning, game theory and network economics.

Jianping Wang is an associate professor in the Department of Computer Science at City University of Hong Kong. She received the B.S. and the M.S. degrees in computer science from Nankai University, Tianjin, China in 1996 and 1999, respectively, and the Ph.D. degree in computer science from the University of Texas at Dallas in 2003. Jianping's research interests include dependable networking, optical networks, cloud computing, service oriented networking and data center networks.

Kejie Lu received the BSc and MSc degrees in Telecommunications Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1994 and 1997, respectively. He received the PhD degree in Electrical Engineering from the University of Texas at Dallas in 2003. In 2004 and 2005, he was a Postdoctoral Research Associate in the Department of Electrical and Computer Engineering, University of Florida. In July 2005, he joined the Department of Electrical and Computer Engineering, University of Puerto Rico at Mayaguez, where he is currently an Associate Professor. His research interests include architecture and protocols design for computer and communication networks, performance analysis, network security, and wireless communications

Wen Qi received his B.E. degree from the Department of Automation, Nankai University in 2009, and the M.S. degree in computer science from the City University of Hong Kong in 2013, and Ph.D degree in 2016 in the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing, networking, and security.



Feng Shan received his Ph.D. degree in Computer Science from Southeast University, China in 2015. He is currently an Assistant Professor at School of Computer Science and Engineering, Southeast University. He was a Visiting Scholar at the School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO, USA, from 2010 to 2012. His research interests are in the areas of energy harvesting, wireless power transfer, algorithm design and analysis.

Junzhou Luo received the BS degree in applied mathematics and the MS and PhD degrees in computer network, all from Southeast University, China, in 1982, 1992, and 2000, respectively. He is a full professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and he is a member of the ACM and chair of ACM SIGCOMM China. His research interests are next generation network architecture, network security, cloud computing, and wireless LAN.