# An Efficient Online Placement Scheme for Cloud Container Clusters

Ruiting Zhou, Zongpeng Li, and Chuan Wu, *Senior Member, IEEE*

***Abstract*— Containers represent an agile alternative to virtual machines (VMs), for providing cloud computing services. Containers are more flexible and lightweight, and can be easily instrumented. Enterprise users often create clusters of interconnected containers to provision complex services. Compared to traditional cloud services, key challenges in container cluster (CC) provisioning lie in the optimal placement of containers while considering inter-container traffic in a CC. The challenge further escalates, when CCs are provisioned in an online fashion. We propose an online algorithm to address the above challenges, aiming to maximize the aggregate value of all served clusters. We first study a one-shot CC placement problem. Leveraging techniques of exhaustive sampling and ST rounding, we design an efficient one-shot algorithm to determine the placement scheme of a given CC. We then propose a primal-dual online placement scheme that employs the one-shot algorithm as a building block to make decisions upon the arrival of each CC request. Through both theoretical analysis and trace-driven simulations, we verify that the online placement algorithm is computationally efficient and achieves a good competitive ratio.**

***Index Terms*— Cloud container clusters, online algorithms, compact exponential optimization.**

## I. Introduction

CLOUD computing has become a new computing paradigm that provides computing services with on-demand access to resources such as CPU/GPU, RAM and disk storage. Cloud resources used to be packed into different types of virtual machines (VMs) to serve cloud users. More recently, cloud containers offer a light-weight alternative to VMs. Unlike VMs, containers do not require a full, dedicated operating system to be installed within them. They are able to operate with the minimum amount of resources and start in microseconds [1], providing a streamlined, easy-to-deploy method of cloud resource provisioning. Example cloud container services toady include Google Container Engine [2], Amazon EC2 Container service (ECS) [3], Aliyun Container Service [4], and Azure Container Service [5].

Besides purchasing individual containers, cloud users often require a collection of containers and the network in between, to create a container cluster (CC) for building a reliable and scalable distributed system. Typical examples include geo-distributed machine learning (ML) systems, and service chains in a Network Function Virtualization (NFV) environment. Geo-distributed machine learning derives useful information from large geo-dispersed data collections without moving them to a central location. A common use case of geo-distributed ML is to train data continuously produced at different locations. For example, e-commerce sites, *e.g.,* Amazon and Taobao, recommend items that are of particular interest to users by learning user behavior from continuously collected click-through data all over the world [6], using ML techniques such as logistic regression. In a geo-distributed ML job, many concurrent workers (typically implemented on containers) reside in different geographic locations to train data sets in proximity [7]. In each training iteration, workers exchange locally computed parameter updates to obtain the global ML model. A service chain refers to the structure of a network service where a sequences of virtual network functions (VNFs) are linked [8]. Many chains are deployed over the WAN with VNFs located in different locations, to process network flows between geo-dispersed sources and destinations. For example, an enterprise may request a CC to deploy an access control service chain "Firewall→IDS→Load Balancer", where instances of firewall, IDS and Load Balancer are encapsulated into containers. Web service flows can traverse this service chain, sending packets from a source to a destination. Container clusters are emerging as the new norm of virtual clusters. Compared to traditional virtual clusters, container clusters, *e.g.,* Google Container Cluster [9], Amazon ECS Cluster [10] and Azure Container Service Cluster [11], provide better performance for applications and enhance the elasticity by fast deployment of additional work nodes.

This work targets a more realistic and general setup in the deployment of CCs. We investigate the online CC placement problem that dynamically assembles CC as per user request. We take the perspective of a cloud service provider, who hosts cloud computing resources in multiple *zones*, where a zone may correspond to one or multiple servers, or a data center. The computing resources in a region owned by Amazon, for

Manuscript received April 19, 2018; revised March 10, 2019; accepted March 13, 2019. Date of current version April 16, 2019. This work was supported in part by NSFC under Grant 61502504, Grant 61628209, and Grant 61502347, in part by the Nature Science Foundation of Hubei Province under Grant 2016CFA030, in part by the Technological Innovation Major Projects of Hubei Province under Grant 2017AAA125, in part by the Science and Technology Program of Wuhan City under Grant 2018010401011288, in part by RGC, Hong Kong, under Contract HKU 17204715, Contract 17225516, and Contract C7036-15G (CRF), and in part by Huawei HIRP under Grant HO2016050002BE. *(Corresponding author: Zongpeng Li.)*

R. Zhou is with the Key Laboratory of Aerospace Information Security and Trusted Computing, School of Cyber Science and Engineering, Ministry of Education, Wuhan University, Wuhan 430000, China (e-mail: ruitingzhou@whu.edu.cn).

Z. Li is with the School of Computer Science, Wuhan University, Wuhan 430000, China (e-mail: zongpeng@whu.edu.cn).

C. Wu is with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: cwu@cs.hku.hk).

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/JSAC.2019.2906745

instance, are divided to *Availability Zones* [12]. The cloud service provider deploys containers and assembles CC upon requests on the fly. Each CC request can come and go at any time, and its placement is determined on the spot. The deployment of a CC involves not only the placement of containers, *i.e.,* assigning each container to a zone with free capacity, but also routing inter-container traffic, *i.e.,* identifying zones with available bandwidth in between to send traffic between neighbour containers. Even in the offline setting with full information, such a deployment problem translates into an NP-hard combinational optimization problem. The challenge further escalates when we target a practical online placement scheme that makes on-spot decisions upon the arrival of each CC request.

We extend the existing literature on virtual cluster provisioning, and propose an efficient online placement scheme such that: i) CCs with different values arrive stochastically; each CC specifies its required containers and the traffic demand between neighbor containers; ii) the algorithm is computationally efficient and executes in polynomial time; iii) the aggregate value of deployed CCs is approximately maximized. Our detailed contributions are summarized below.

**First**, we formulate the offline optimization problem as an integer program (IP) with quadratic constraints that capture inter-container traffic flow. While polynomial in size, the quadratic IP is non-linear and admits no direct application of the classic primal-dual schema for algorithm design. We leverage the recent *compact-exponential* optimization framework [13] to encode each valid placement scheme in a variable, and reformulate the original IP into a *compact-exponential Integer Linear Program (ILP)*, which contains only conventional packing-type constraints, but at the cost of involving an exponential number of variables. Solving the compact exponential ILP directly is still infeasible in practice, when complete knowledge over the entire system lifespan is not available. We instead first relax the resource capacity constraints that impose inter-CC coupling, and focus on a one-shot problem to determine the optimal placement of the current CC. We then design an online algorithm framework that simultaneously works on the compact-exponential ILP and its dual LP, invoking the one-shot algorithm as a subroutine, towards computing efficient placement based on values of dual variables.

**Second**, we reformulate the one-shot CC placement problem into an integer quadratic program (IQP), to minimize the placement cost for a given CC. We first consider a simplified scenario of a single type of computational resource. The IQP has an objective function of degree 2, and is proven NP-hard to solve. We apply an *exhaustive sampling* technique [14] based on a random-sampling process to reduce its degree from 2 to 1, at the cost of losing some accuracy. The degree-reduced problem becomes a general assignment problem with extra constraints. We solve this problem to optimality, and apply the *ST rounding* technique [15] to round the fractional solution to an integral solution. More specifically, we construct a bipartite graph based on the fractional solution, and output the minimum-cost integer matching in this graph. Theoretical analysis shows that our algorithm achieves a small approximation ratio. We then further consider the general scenario, and propose a heuristic algorithm to provide good solutions with low computational complexity.

**Third**, we proceed to consider resource capacity constraints, and design an online algorithm framework that utilizes the one-shot algorithm to determine each CC's placement upon its arrival, without relying on future information. We apply the primal-dual technique to the compact-exponential ILP and its dual LP, and interpret dual variables as unit resource prices at different times. Upon receiving a CC request, given current resource prices, the one-shot algorithm computes an $\alpha$-approximate placement scheme with an estimated cost. We divide the estimated cost by $\alpha$ to obtain a lower bound of the optimal cost, and compare the CC's value with it. If the value is higher, the CC is deployed and dual variables are updated; otherwise the CC request is discarded. We conduct theoretical analysis on the competitive ratio and prove its upper bound. The effectiveness of our one-shot and online algorithms are evaluated through trace-driven simulation studies.

In the rest of the paper, we review related work in Sec. II and describe the system model in Sec. III. We study the one-shot CC placement problem in Sec. IV and propose an online placement algorithm on Sec. V. Simulation studies are presented in Sec. VI. Sec. VII concludes the paper.

## II. Related Work

Early studies of cloud computing have focused on VM provisioning, in both offline and online settings. Zhang *et al.* [16] propose a randomized algorithm based on a decomposition technique for dynamic cloud resource provisioning, achieving a small approximation ratio. Shi *et al.* [17] present the first online combinatorial auction in the cloud computing paradigm. Zheng and Shroff [18] design online multi-resource allocation algorithms to schedule cloud jobs with deadlines. Tan *et al.* [19] propose a elastic cloud resource provisioning algorithm under premise of guaranteeing performance. Bitton *et al.* [20] design a batch dispatching algorithm to process cloud jobs. The above literature focuses on the deployment of separate VMs, without considering inter-VM traffic in a virtual cluster.

Kubernetes [21] is an open source platform for individual user to deploy and manage container clusters on public clouds. Its default resource-provisioning mechanism adjusts the number of containers running for the application only based on CPU utilization. Chang *et al.* [22] further propose a generic platform to facilitate dynamic resource provisioning based on Kubernetes, taking consideration of multi-types of resources. Along the direction of global container resource allocation, Tao *et al.* [23] propose two approaches for mapping user preferences to concrete container configuration parameters. They also design a node selection algorithm for container placement. They study a simple offline scenario, without considering the inter-container traffic. Waibel *et al.* [24] provide a fine-granular resource scheduling algorithm for elastic processes based on containers, without considering inter-container traffic. One related problem is the VNF placement problem which usually targets different optimization

objectives. Agarwal *et al.* [25] present a latency minimization strategy to make joint VNF placement and CPU assignment decisions. Tang *et al.* [26] forecast the traffic amount while placing VNF instances to minimize the inter-rack traffic. Jia *et al.* [27] investigate dynamic placement of VNF service chains, for operational cost minimization.

Cloud container cluster provisioning belongs to the category of virtual cluster (VC) provisioning (*a.k.a.* virtual network embedding/mapping). Along this direction, Chowdhury *et al.* [28] propose virtual network embedding algorithms that efficiently map virtual nodes and virtual links onto the substrate network resources. Li *et al.* [29] address the VM placement problem, by considering both the traffic cost and the physical machine utilization cost. Yu *et al.* [30] study the survivable VC embedding problem, which jointly optimizes primary and backup embeddings of VCs, with the goal of minimizing VM consumption. Dai *et al.* [31] design algorithms for the minimum energy virtual cluster embedding problem. They provide no proven guarantee on the approximation ratio. Different from the above literature, our one-shot CC placement problem has a different optimization objective. We target total cost minimization with a natural IQP formulation, requiring different solution techniques. We rigorously prove that our proposed algorithm can achieve a small approximation ratio. In addition, the above literature considers only one-time/offline scenario, while we further propose an efficient online CC placement scheme to handle dynamically arriving requests for CCs.

Towards online VC deployment, Evan and Medina [32] study the online multi-commodity flow routing problem. They focus on link capacity constraints but ignore node capacity constraints. Grandl *et al.* [33] propose a multi-resource cluster scheduler that assigns tasks to machines. The communication demand between different tasks is not modelled by them. Shi *et al.* [34] investigate online mechanism design to place inter-connected VMs in a geo-distributed IaaS cloud, taking both computational resources and communication resources into consideration. Their subproblem for each job's placement is trivial, since they specify several VM placement schemes for each job, while our subproblem is an NP-hard problem that computes the best placement for each CC. Our work is also the first to design an online primal-dual algorithm for CC placement, with proven performance guarantee.

The compact-exponential optimization framework was first applied by Zhou *et al.* [13]. They consider the scheduling of computing jobs that require separate VMs, while this work focuses on the placement of correlated containers in the form of container clusters. This work further advances the compact-exponential framework to handle nonlinear constraints and NP-hard subproblems. Our subproblem, namely the one-shot CC placement problem, is a special case of the quadratic assignment problem (see [35] for a detailed survey). We design a rounding algorithm that combines exhaustive sampling [14] and ST rounding [15] techniques for effective solutions. The online primal-dual method is a known powerful algorithmic technique for many NP-hard problems, such as the knapsack problem and the general packing problem [36]. However, our online optimization problem does not fall into such known categories. We propose a primal-dual online framework to solve our problem, and provide a new price function to update dual variables, which is the key towards achieving a good competitive ratio.

## III. System Model

We consider a cloud service provider who owns a pool of resources residing in $S$ zones, where a *zone* may correspond to one or a cluster of servers, or a data center. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. There are $K$ types of computation computation resources, as exemplified by CPU, RAM and disk. Each zone $s \in [S]$ has $C_{ks}$ units of type-$k$ resource. Zones are interconnected by broadband links. Active optical cables (AOC) and unshielded twisted pair (UTP) cables are often used for short links that connect zones in the same data center, while multi-mode or single-mode fibers are used to connect zones which correspond to different data centers [37]. Let $\mathbb{E}$ be the set of links, and let $D_{s_1,s_2}$ denote the bandwidth capacity of link $(s_1, s_2) \in \mathbb{E}$ that connects zones $s_1$ and $s_2$.

Over a large time span $1, 2, \ldots, T$, $I$ CC requests arrive stochastically to the system. Multiple requests can arrive simultaneously, and would be ordered randomly. Request $i$ arrives at time $t_i$, requiring a CC from $t_i^-$ to $t_i^+$. Each CC consists of a set of tailor-made containers. Let $\mathcal{V}_i$ and $V_i$ denote the set of containers and the number of containers in request $i$'s CC, respectively. A container $v \in \mathcal{V}_i$ consumes $a_{vk}^i$ amount of type-$k$ resource, $\forall k \in [K]$. Let $\Delta_{v_1,v_2}^i$ denote the bandwidth consumption for flow transfer from $v_1$ to $v_2$ in request $i$'s CC, when $v_1$ and $v_2$ reside distinct zones. A value $b_i$ is obtained if request $i$'s CC is deployed. In summary, request $i$ can be expressed as: $\Phi_i = \{b_i, V_i, \mathcal{V}_i, \{a_{vk}^i\}_{v\in\mathcal{V}_i, k\in[K]}, \{\Delta_{v_1,v_2}^i\}_{v_1,v_2\in\mathcal{V}_i}\}$.

Upon each request's arrival, the service provider immediately determines whether to serve it, and if so, how to place its CC. Decision variables for request $i$ include: i) $x_i \in \{0, 1\}$, indicating whether request $i$ is accepted $(1)$ or not $(0)$. ii) $y_{vs}^i, \forall v \in \mathcal{V}_i, \forall s \in [S]$, encoding the placement scheme of request $i$'s CC, where $y_{vs}^i = 1$ if zone $s$ is selected to host container $v$ and 0 otherwise. The service provider in practice wishes to reserve resources for different CC requests, and limits a single CC to occupy at most $B_{ks}$ units of type-$k$ resource in zone $s$. Such resource consumption bound is also customary in the cloud resource allocation literature [38] [18]. Fig. 1 shows a placement scheme for request 1. Our objective is to maximize the total valuation obtained from all CCs, subject to resource capacity constraints. The optimization problem can be formulated into the following integer program (IP):

$$\text{maximize} \sum_{i\in[I]} b_i x_i \tag{1}$$

$$\text{subject to :} \ x_i = \sum_{s\in[S]} y_{vs}^i, \quad \forall i \in [I], \ \forall v \in \mathcal{V}_i, \tag{1a}$$

$$\sum_{\substack{i\in[I]:\\ t_i^- \le t \le t_i^+}} \sum_{v\in\mathcal{V}_i} a_{vk}^i y_{vs}^i \le C_{ks}, \quad \forall k \in [K], \forall s \in [S], \ \forall t \in [T], \tag{1b}$$

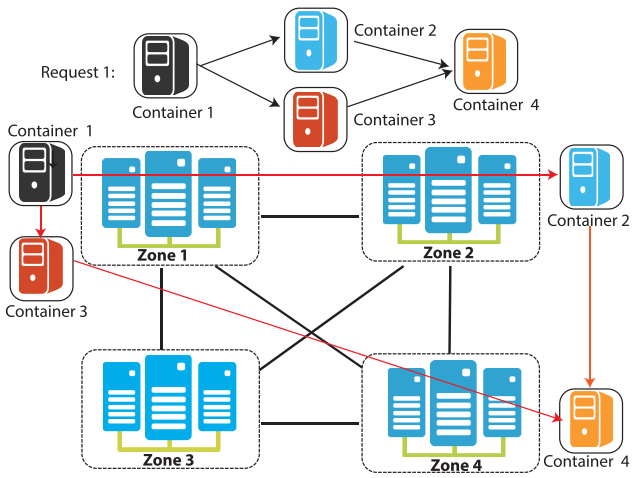



Fig. 1. Container cluster placement: an example.

$$\sum_{s\in[S]} y_{vs}^i \le 1, \quad \forall i \in [I],\ \forall v \in \mathcal{V}_i, \tag{1c}$$

$$\sum_{v\in\mathcal{V}_i} a_{vk}^i y_{vs}^i \le B_{ks}, \quad \forall i \in [I],\ \forall k \in [K],$$
$$\forall s \in [S],\ \forall t \in [t_i^-, t_i^+] \tag{1d}$$

$$\sum_{\substack{i\in[I]:\\ t_i^-\le t\le t_i^+}} \sum_{v_1,v_2\in\mathcal{V}_i} \Delta_{v_1,v_2}^i y_{v_1,s_1}^i y_{v_2,s_2}^i \le D_{s_1,s_2},$$
$$\forall (s_1,s_2) \in \mathbb{E},\ \forall t \in [T], \tag{1e}$$

$$x_i, y_{vs}^i \in \{0,1\}, \quad \forall i \in [I],\ \forall v \in \mathcal{V}_i,\ \forall s \in [S]. \tag{1f}$$

Constraint (1a) ensures that request $i$'s CC is deployed only when it is accepted, since request $i$'s container $v$ is placed to a zone $s$ only when $x_i = 1$. Constraint (1b) guarantees that at any time, allocated resources at a zone do not exceed its capacity. Constraint (1c) indicates that a container in a CC request resides in at most one zone. Constraint (1d) enforces the upper-bound of each CC's resource occupation at a zone $s$. Link capacity constraints are modelled by (1e).

Even in the offline setting, with complete knowledge given, the polynomial-sized IP (1) is NP-hard to solve. To verify, consider a special case of IP (1) where each CC consists of one container, $T = 1$ and $B_{ks} = D_{s_1,s_2} = +\infty$. Then the classic multidimensional knapsack problem, which is known to be NP-hard, is equivalent to the special case of IP (1). The challenge further escalates when we consider quadratic constraints (1e). To address these challenges, we resort to the compact-exponential technique [13], which can reformulate IP (1) into an equivalent ILP with packing structure, at the price of involving an exponential number of variables:

$$\text{maximize} \sum_{i\in[I]} \sum_{l\in\zeta_i} b_{il} x_{il} \tag{2}$$

$$\text{subject to :} \sum_{\substack{i\in[I]:\\ t_i^-\le t\le t_i^+}} \sum_{l\in\zeta_i} f_{m,t}^{il} x_{il} \le C_m,$$
$$\forall m \in \mathcal{M},\ \forall t \in [T], \tag{2a}$$

$$\sum_{l\in\zeta_i} x_{il} \le 1, \quad \forall i \in [I], \tag{2b}$$

$$x_{il} \in \{0,1\}, \quad \forall i \in [I],\ \forall l \in \zeta_i. \tag{2c}$$

In the above compact-exponential ILP, $\zeta_i$ is the set of feasible placement schemes for request $i$. A *feasible* scheme is a vector $l = \{y_{vs}^i\}$ that satisfies (1c) and (1d). Variable $x_{il} \in \{0,1\}$ indicates whether request $i$'s scheme $l$ is accepted (1) or not (0). We regard each computation resource at each zone and the bandwidth at each link as different resources. Consequently, the total number of resource types is $KS + |\mathbb{E}|$. Let $\mathcal{M}$ be the set of resource types and $C_m$ be the capacity of type-$m$ resource, $\forall m \in \mathcal{M}$. $f_{m,t}^{il}$ denotes the total type-$m$ resource consumption of request $i$'s scheme $l$ at time $t$. For example, if $m$ corresponds to type-$k$ resource at zone $s$, $f_{m,t}^{il} = \sum_{v\in\mathcal{V}_i} a_{vk}^i y_{vs}^i, \forall t \in [t_i^-, t_i^+]$. Constraint (2a) is equivalent to (1b) and (1e). Constraint (2ab) ensures that each CC is placed according to at most one scheme.

We relax $x_{il} \in \{0,1\}$ to $x_{il} \ge 0$, and introduce dual variables $p_{m,t}$ and $u_i$ to constraints (2a) and (2b). The dual of the relaxation of program (2) is:

$$\text{minimize} \sum_{t\in[T]} \sum_{m\in\mathcal{M}} C_m p_{m,t} + \sum_{i\in[I]} u_i \tag{3}$$

$$\text{subject to: } u_i \ge b_{il} - \sum_{m\in\mathcal{M}} \sum_{t\in[t_i^-,t_i^+]} f_{m,t}^{il} p_{m,t},$$
$$\forall i \in [I],\ \forall l \in \zeta_i, \tag{3a}$$

$$p_{m,t}, u_i \ge 0, \quad \forall i \in [I],\ \forall m \in \mathcal{M},\ \forall t \in [T]. \tag{3b}$$

IP (1) and ILP (2) have the same optimal objective value. To solve ILP (2), complete knowledge over the entire system lifespan is required. However, our algorithm needs to work in an online fashion, making on-spot decisions without relying on knowledge of future request arrivals. To this end, we leverage the primal-dual technique that determines the primal solution based on dual variables. We interpret dual variable $p_{m,t}$ as the unit price of type-$m$ resource at time $t$. Upon the arrival of a request, we compute its CC's placement scheme based on current resource prices. We first focus on a one-shot CC placement problem, which relaxes resource capacity constraints (2a) that impose temporal correlation in online decision making; we design an efficient algorithm to determine a CC's placement scheme with the goal of cost minimization. We then propose an online algorithm framework that employs the one-shot optimization as a building block to make on-spot decisions upon CC request arrivals.

We make two assumptions in this work. First, we assume that a CC's valuation is proportional to its resource consumption in each time slot $f_{m,t}^{il}$ (where $f_{m,t}^{il} > 0$): $1 \le \frac{b_i}{f_{m,t}^{il}} \le U, \forall i, l, m, t$, where $U$ is a constant. Second, we assume that the ratio between a scheme's resource consumption at each time slot and the resource capacity is bounded: $\frac{f_{m,t}^{il}}{C_m} \le \frac{1}{\log\lambda}, \forall i, l, m, t$, where $\lambda = 2(\alpha U + 1)$ and $\alpha$ represents the approximation ratio of the one-shot CC placement algorithm. This assumption implies that the resource demand of each container is small compared to the capacity of each zone.

TABLE I
SUMMARY OF NOTATION

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $I$ | # of CC requests | $[X]$ | integer set $\{1,\ldots,X\}$ | $S$ | # of zones | $\mathbb{E}$ | set of links |
| $b_i$ | the value of request $i$' CC | $K$ | # of types of computational resource | | | | |
| $T$ | # of time slots | $C_{ks}$ | capacity of type-$k$ resource at zone $s$ | | | | |
| $t_i^-(t_i^+)$ | start (end) time of request $i$ | $D_{s_1,s_2}$ | bandwidth capacity of link $(s_1,s_2)$ | | | | |
| $\mathcal{V}_i(V_i)$ | set(number) of containers in request $i$'s CC | | | | | | |
| $a_{vk}^i$ | amount of type-$k$ resource consumed by $v \in \mathcal{V}_i$ | | | | | | |
| $\Delta_{v_1,v_2}^i$ | traffic from $v_1$ to $v_2$ in request $i$'s CC | | | | | | |
| $B_{ks}$ | upper bound of type-$k$ resource consumption | | | | | | |
| $x_i$ | request $i$ is accepted (1) or not (0) | | | | | | |
| $y_{vs}^i$ | container $v$ is assigned to zone $s$ or not in $i$'s CC | | | | | | |
| $x_{il}$ | request $i$'s scheme $l$ is accepted (1) or not (0) | | | | | | |
| $f_{m,t}^{il}$ | demand of type-$m$ resource at $t$ by $i$'s scheme $l$ | | | | | | |
| $p_{m,t}$ | cost of unit type-$m$ resource at $t$ | | | | | | |
| $\lambda$ | $2(\alpha U+1)$, where $1 \le \frac{b_i}{f_{m,t}^{il}} \le U, \forall i,l,m,t$ | | | | | | |

Here $\lambda$ (related to $U$) is an important parameter and will be used in our online algorithm design. Notations are summarized in Table 1 for ease of reference.

## IV. Approximation Algorithm Design for Container Cluster Placement

In this section, we first formulate the one-shot CC placement problem in Sec. IV-A. A rounding algorithm and a heuristic algorithm are then designed and analyzed in Sec. IV-B and Sec. IV-C, respectively.

### A. Cost Minimization Problem

We include the same constraints (1c) and (1d) related to the current CC request $i$ from IP (1), and exclude resource capacity constraints (1b) and (1e). Given current resource prices (to be decided in Sec. V), we compute the computation cost $P_{vs}^i$, *i.e.,* the cost of placing container $v$ in zone $s$, and the communication cost $P_{v_1,v_2,s_1,s_2}^i$, *i.e.,* the cost of sending traffic from $v_1$ to $v_2$ through link $(s_1,s_2)$. The cost minimization problem has the following natural IQP formulation:

$$\text{minimize} \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} P_{vs}^i y_{vs}^i + \sum_{\substack{v_1,v_2\in\mathcal{V}_i,\\(s_1,s_2)\in\mathbb{E}}} P_{v_1,v_2,s_1,s_2}^i y_{v_1,s_1}^i y_{v_2,s_2}^i \tag{4}$$

$$\text{subject to:} \sum_{v\in\mathcal{V}_i} a_{vk}^i y_{vs}^i \le B_{ks}, \quad \forall k\in[K],\ \forall s\in[S], \tag{4a}$$

$$\sum_{s\in[S]} y_{vs}^i = 1, \quad \forall v\in\mathcal{V}_i, \tag{4b}$$

$$y_{vs}^i \in \{0,1\}, \quad \forall v\in\mathcal{V}_i,\ \forall s\in[S]. \tag{4c}$$

If we set $a_{vk}^i = B_{ks} = 1, \forall v,k,s$, IQP (4a) degrades to the minimum quadratic assignment problem [35], which has been proven NP-hard, and does not have any known constant-factor approximation algorithm in polynomial time (assuming P $\neq$ NP). When the number of zones is small ($S < 5$), we can compute the optimal solution by enumerating all options. However, $S$ can be a large number in practice. For example, the Amazon cloud infrastructure is currently comprised of 42 availability zones [12]. It is essential to compute a good solution efficiently, in a short time. To solve the IQP, we first simplify the model by considering a single type of computation resource, and present a rounding algorithm that approximately solves the problem with worst-case performance guarantee. We later propose a heuristic algorithm that can solve the general version, without theoretical performance guarantee.

### B. A Rounding Algorithm With Performance Guarantee

Given a sole type of computation resource, let $a_v^i$ and $B_s$ represent $a_{vk}^i$ and $B_{ks}$, respectively. The *degree* of an integer program is $d$ if its objective function is a degree-$d$ polynomial function. IQP (4) is a degree-2 integer program. We first apply *exhaustive sampling* [14] to reduce its degree from 2 to 1. We convert IQP (4) to an ILP, with some loss of accuracy. The ILP is a generalized assignment problem with side constraints. We solve its LP relaxation exactly, and then round the fractional solution to an approximate integral solution through another technique of *ST rounding* [15].

More specifically, let ${y_{vs}^i}^*$ be the optimal solution to IQP (4a), and let ${F_{v_1,s_1}^i}^* = \sum_{v_2\in\mathcal{V}_i}\sum_{s_2\in[S]} P_{v_1,v_2,s_1,s_2}^i {y_{v_2,s_2}^i}^*$, we can reformulate IQP (4a) to the following ILP:

$$\text{minimize} \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} (P_{vs}^i + {F_{vs}^i}^*) y_{vs}^i \tag{5}$$

$$\text{subject to :}\ \text{Constraints}(4a-4c)$$

$$\sum_{v_2\in\mathcal{V}_i}\sum_{s_2\in[S]} P_{v_1,v_2,s_1,s_2}^i y_{v_2,s_2}^i = {F_{v_1,s_1}^i}^*, \quad \forall v_1\in\mathcal{V}_i,\ \forall s_1\in[S]. \tag{5d}$$

However, it is difficult to obtain the exact value of ${F^i_{v_1,s_1}}^*$. We strive to compute reasonable guesses by exhaustively listing all placement schemes of a random sample. For ease of presentation, we normalize $P^i_{v_1,v_2,s_1,s_2}$ and $P^i_{vs}$, so that $\max\{\max_{v_1,v_2,s_1,s_2}\{P^i_{v_1,v_2,s_1,s_2}\}, \max_{v,s}\{P^i_{vs}\}\} = 1$. The sampling procedure is as follows:

i) We first pick a random sample $\mathcal{W}$ of $n = O(\log V_i/\epsilon^2)$ containers from $\mathcal{V}_i$. Let $\mathcal{W} = \{v_{j_1}, \ldots, v_{j_n}\}$. Exhaustively go through each of the $S^n$ ways of placing containers in $\mathcal{W}$. For each placement that satisfies constraints (4a) and (4b), we compute an estimate ${F^i_{v_1,s_1}}$ of ${F^i_{v_1,s_1}}^*$ by setting

$$F^i_{v_1,s_1} = \frac{V_i}{n} \sum_{v_2 \in \mathcal{W}} \sum_{s_2 \in [S]} P^i_{v_1,v_2,s_1,s_2} y^i_{v_2,s_2},$$

where $y^i_{v_2,s_2} = 1$ if container $v_2$ is allocated to zone $s_2$. Note that we try all possible placements of containers in $\mathcal{W}$. Therefore, the "correct" placement in which $y^i_{v_2,s_2} = {y^i_{v_2,s_2}}^*, \forall v_2 \in \mathcal{W}, s_2 \in [S]$ is ensured to be tried. We call the estimate corresponding to this assignment *the special estimate*, denoted as ${F^i_{v_1,s_1}}^s$. We will show that ${F^i_{v_1,s_1}}^s$ satisfies $|{F^i_{v_1,s_1}}^* - {F^i_{v_1,s_1}}^s| \le \epsilon V_i$ in Lemma 2.

ii) Next, for each estimate $F^i_{v_1,s_1}$ of ${F^i_{v_1,s_1}}^*$, we consider the following LP:

$$\text{minimize} \sum_{v \in \mathcal{V}_i} \sum_{s \in [S]} (P^i_{vs} + F^i_{vs}) y^i_{vs} \tag{6}$$

subject to: Constraints $(4a - 4b)$

$$\sum_{v_2 \in \mathcal{V}_i} \sum_{s_2 \in [S]} P^i_{v_1,v_2,s_1,s_2} y^i_{v_2,s_2} \le F^i_{v_1,s_1} + \epsilon V_i, \quad \forall v_1 \in \mathcal{V}_i,\ \forall s_1 \in [S], \tag{6c}$$

$$\sum_{v_2 \in \mathcal{V}_i} \sum_{s_2 \in [S]} P^i_{v_1,v_2,s_1,s_2} y^i_{v_2,s_2} \ge F^i_{v_1,s_1} - \epsilon V_i, ) \quad \forall v_1 \in \mathcal{V}_i,\ \forall s_1 \in [S], \tag{6d}$$

$$y^i_{vs} \ge 0, \quad \forall v \in \mathcal{V}_i,\ \forall s \in [S]. \tag{6e}$$

Let $\overline{y^i_{vs}}$ and $\overline{\eta}$ be the optimal solution and the optimal objective value of LP (6), respectively. We round the fractional solution $\overline{y^i_{vs}}$ to an approximate integral solution ${y^i_{vs}}'$ using an algorithm $A_{round}$. The approximate integral solution satisfies constraint (4b) and slightly violates (4a) by allowing each CC to occupy at most twice its resource usage bound at each zone.

iii) Because there are at most $S^n$ estimates, we solve LP (6) as above for each $F^i_{v_1,s_1}$. Among the solutions, we output the one whose corresponding placement minimizes the objective of IQP (4). We summarize our algorithm in $A_{sub1}$.

We next describe the details of $A_{round}$, which applies ST rounding [15] to round the fractional solution to an integral solution. In preparation, we first construct a weighted bipartite graph $\mathcal{B} = (V', S', E')$ based on $\overline{y^i_{vs}}$, as follows:

i) One side of the bipartite graph consists of container nodes: $V' = \{v_v : v = 1, \ldots, V_i\}$, and the other side includes zone nodes: $S' = \{s_{s\theta} : s = 1, \ldots, S, \theta = 1, \ldots, \Theta_s\}$, where $\Theta_s = \lceil \sum_{v \in \mathcal{V}_i} \overline{y^i_{vs}} \rceil$. We assign $\Theta_s$ nodes for zone $s$.

ii) For zone $s$, sort the containers placed to it by non-increasing resource demand $a^i_v$. For notation simplicity, we assume that $a^i_1 \ge a^i_2 \ldots, \ge a^i_{V_i}$.

**Algorithm 1** An Approximation Algorithm $A_{sub1}$

**Input**: $\Phi_i, \{P^i_{vs}\}, \{P^i_{v_1,v_2,s_1,s_2}\}, \{a^i_v\}, \epsilon$

1: Define $n = O(\log V_i/\epsilon^2)$;
2: Pick a random subset $\mathcal{W}$ of $n$ containers from $\mathcal{V}_i$; Let $\mathcal{W} = \{v_{j_1}, \ldots, v_{j_n}\}$;
3: **for** e **do**ach placement of containers in $\mathcal{W}$ **do**
4: Update the corresponding $y^i_{vs}$;
5: Define $F^i_{v_1,s_1} = \frac{V_i}{n} \sum_{v_2 \in \mathcal{W}} \sum_{s_2 \in [S]} P^i_{v_1,v_2,s_1,s_2} y^i_{v_2,s_2}$, $\forall v_1 \in \mathcal{V}_i, s_1 \in [S]$;
6: Solve LP (5) exactly. Let $\overline{y^i_{vs}}$ be the optimal solution;
7: $\{{y^i_{vs}}'\} = A_{round}(\{\overline{y^i_{vs}}\}, \{P^i_{vs} + F^i_{vs}\})$;
8: $\Lambda_j = \sum_{v \in \mathcal{V}_i} \sum_{s \in [S]} P^i_{vs} {y^i_{vs}}' + \sum_{v_1,v_2 \in \mathcal{V}_i} \sum_{(s_1,s_2) \in \mathbb{E}} P^i_{v_1,v_2,s_1,s_2} {y^i_{v_1,s_1}}' {y^i_{v_2,s_2}}'$;
9: **end for**
10: $cost_i = \min_j \Lambda_j$; Save the placement $\{y^i_{vs}\}$ which leads to the minimum $\Lambda_j$ to $l^*$;
11: **Return** $cost_i, l^*$.

**Algorithm 2** A Rounding Algorithm $A_{round}$

**Input**: $\{\overline{y^i_{vs}}\}, \{P^i_{vs} + F^i_{vs}\}$;

1: Build a bipartite graph $\mathcal{B} = (V', S', E')$ based on $\overline{y^i_{vs}}$;
2: Find the minimum-cost integer matching that exactly matches all container nodes in $\mathcal{B}$;
3: Update ${y^i_{vs}}' = 1$ if node $v$ is mapped to node $s$;
4: **Return** $\{{y^i_{vs}}'\}$.

iii) For zone $s$, consider nodes $s_{s\theta}, \theta = 1, \ldots, \Theta_s$ as bins of capacity 1, and consider $\overline{y^i_{vs}}, v \in \mathcal{V}_i$ as pieces of containers placed into these bins. With respect to the non-increasing order of $a^i_v$, we pack pieces into node $s_{s1}$ one by one, until placing piece $v$ results in bin overflow (exceeding 1). We then place a fraction of $v$ to saturate the capacity of node $s_{s1}$, and place the remaining fraction of $v$ to node $s_{s2}$. We carry on this process until all pieces are placed into bins.

iv) If there is a positive fraction of container $v$ packed into node $s_{s\theta}$, edge $(v_v, s_{s\theta})$ is added to $E'$. We define a vector $y'(v_v, s_{s\theta})$ on edge $(v_v, s_{s\theta})$, which equals the fraction of $v$ packed into $s_{s\theta}$. The cost of this edge, $w(v_v, s_{s\theta})$, is set to $P^i_{vs} + F^i_{vs}$.

Vector $y'$ is a fractional matching in bipartite graph $\mathcal{B}$ that exactly matches all container nodes. The cost of the fractional matching is $\overline{\eta}$. By matching theory [39], there exists an integral matching with cost at most $\overline{\eta}$ that exactly matches all container nodes. We continue to compute such an integral matching.

*1) Theoretical Analysis:*

*a) Feasibility and Running Time:*

*Theorem 1:* The integral solution $y^i_{vs}$ returned by $A_{sub1}$ satisfies constraint (4b) and slightly violates constraint (4a) by allowing $\sum_{v \in \mathcal{V}_i} a^i_v y^i_{vs} \le 2B_s, \forall s \in [S]$.

*Proof:* $A_{round}$ generates an integral matching that exactly matches all container nodes. As a result, constraint (4b) is satisfied. We next examine constraint (4a). For each zone node $s_{s\theta}$, let $a^{\max}_{s\theta}$ denote the maximum of $a^i_v$ corresponding

to edges $(v_v, s_{s\theta}), \forall v_v$, and let $a^{\min}_{s\theta}$ denote the corresponding minimum. We have $a^{\min}_{s\theta} \geq a^{\max}_{s,\theta+1}, \theta = 1, \ldots, \Theta_s - 1$. Then the total resource consumption of containers assigned to zone $s$ by any integral matching in $\mathcal{B}$ is at most $\sum_{\theta=1}^{\Theta_s} a^{\max}_{s\theta}$. Clearly, $a^{\max}_{s1} \leq B_s$.

$$\sum_{\theta=2}^{\Theta_s} a^{\max}_{s\theta} \leq \sum_{\theta=1}^{\Theta_s-1} a^{\min}_{s\theta} \leq \sum_{\theta=1}^{\Theta_s-1} \sum_{v:(v_v,s_{s\theta})\in E'} a^i_v y'(v_v, s_{s\theta})$$
$$\leq \sum_{\theta=1}^{\Theta_s} \sum_{v:(v_v,s_{s\theta})\in E'} a^i_v y'(v_v, s_{s\theta}) = \sum_{v\in\mathcal{V}_i} a^i_v \overline{y^i_{vs}} \leq B_s,$$

which proves the theorem. □

*Theorem 2:* $A_{sub1}$ is a polynomial time algorithm, with time complexity $O(S^{n+1}V_i^2 \log V_i)$.

*Proof:* Lines 1–2 take $O(n)$ steps to initialize set $\mathcal{W}$. The `for` loop in lines 3–9 iterates at most $S^n$ times. In each iteration, lines 4–5 can be accomplished in $O(SV_i)$ steps. The complexity of the ST rounding algorithm in lines 6–7 is $O(SV_i^2 \log V_i)$ [15]. Line 8 computes the objective value in $O(SV_i)$ steps. Thus, the complexity of the `for` loop is $O(S^{n+1}V_i^2 \log V_i)$. Lines 10–11 return the output in $O(S^n)$ time. In summary, the time complexity of $A_{sub2}$ is $O(S^{n+1}V_i^2 \log V_i)$, which is polynomial to the input size. □

*b) Approximation Ratio:* The *approximation ratio* is the upper-bound ratio of the objective value achieved by $A_{sub1}$ to the optimal objective value of IQP (4a).

*Lemma 1: (***Chernoff bound***, Lemma 24 in [40])* Let $X_1, \ldots, X_k$ be independent random variables such that $0 \leq X_i \leq 1$. Let $X = \sum_{i=1}^k X_i$ and $\mu = E(X)$, $Pr[|X - \mu|] \geq \sigma] \leq 2e^{-2\sigma^2/k}$.

*Lemma 2:* Assume $n = q \log V_i/\epsilon^2$, the special estimate ${F^i_{v_1,s_1}}^s$, with probability at least $1 - \frac{2}{V_i^{2q}}$, is in the range of $[{F^i_{v_1,s_1}}^* - \epsilon V_i, {F^i_{v_1,s_1}}^* + \epsilon V_i]$.

*Proof:* We define $n$ random variables $Y_1, \ldots, Y_n$ and let $Y_j = \sum_{s_2\in[S]} P_{v_1,v_2,s_1,s_2} {y^i_{v_j,s_2}}^*, \forall v_j \in \mathcal{W}$. Note that $0 \leq Y_j \leq 1, j = 1, \ldots, n$. Then $Y = \sum_{j=1}^n Y_j = \frac{{F^i_{v_1,s_1}}^s n}{V_i}$ and $E[Y] = \frac{n}{V_i}{F^i_{v_1,s_1}}^*$. By Lemma 1, we have

$$Pr[|{F^i_{v_1,s_1}}^s - {F^i_{v_1,s_1}}^*| \geq \epsilon V_i]$$
$$= Pr[\frac{n}{V_i}|{F^i_{v_1,s_1}}^s - {F^i_{v_1,s_1}}^*| \geq \epsilon n]$$
$$= Pr[|Y - E[Y]| \geq \epsilon n] \leq 2\, e^{-2\epsilon^2 n}$$
$$= 2\, e^{-2\epsilon^2\ q \log V_i/\epsilon^2} = \frac{2}{V_i^{2q}}.$$

Thus, $Pr[|{F^i_{v_1,s_1}}^s - {F^i_{v_1,s_1}}^*| \leq \epsilon V_i] \geq 1 - \frac{2}{V_i^{2q}}$. □

*Lemma 3:* Upon termination, $A_{sub1}$ outputs an integral solution $y^i_{vs}$ that satisfies $\Lambda(y^i_{vs}) \leq \Lambda^* + (1+\epsilon)V_i^2$, where $0 \leq \epsilon \leq 1$, $\Lambda(y^i_{vs})$ is the objective value of IQP (4) produced by $y^i_{is}$ and $\Lambda^*$ is the optimal objective value of IQP (4).

*Proof:* Recall that ${y^i_{vs}}^*$ is the optimal solution to IQP (4). Lemma 2 implies that with high probability, ${y^i_{vs}}^*$ is a feasible solution to LP (6) when we input the special estimate ${F^i_{v_1,s_1}}^s$. Because $\overline{y^i_{vs}}$ is the optimal solution to LP (6), the integral solution ${y^i_{vs}}'$, which is rounded from $\overline{y^i_{vs}}$, satisfies:

$$\sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} (P^i_{vs} + {F^i_{vs}}^s){y^i_{vs}}'$$
$$= \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} (P^i_{vs} + {F^i_{vs}}^s)\overline{y^i_{vs}} \leq \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} (P^i_{vs} + {F^i_{vs}}^s){y^i_{vs}}^*$$
$$\leq \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} P^i_{vs}{y^i_{vs}}^* + \sum_{v_1\in\mathcal{V}_i}\sum_{s_1\in[S]} ({F^i_{v_1,s_1}}^* + \epsilon V_i){y^i_{v_1,s_1}}^*$$
$$= \Lambda^* + \epsilon V_i(\sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} {y^i_{vs}}^*) = \Lambda^* + \epsilon V_i^2.$$

Because $0 \leq P^i_{v_1,v_2,s_1,s_2} \leq 1$, we can obtain $\sum_{v_2\in\mathcal{V}_i}\sum_{s_2\in[S]} P^i_{v_1,v_2,s_1,s_2}{y^i_{v_2,s_2}}' \leq {F^i_{vs}}^s + V_i$. As a result, $\Lambda({y^i_{vs}}')$, the objective value of IQP (4) achieved by ${y^i_{vs}}'$, is at most $\sum_{v\in\mathcal{V}_i}\sum_{s\in[S]}(P^i_{vs} + {F^i_{vs}}^s){y^i_{vs}}' + V_i^2$.

Among different rounded integral solutions, the output $y^i_{is}$ minimizes the objective value of IQP (4), thus,

$$\Lambda(y^i_{vs}) \leq \Lambda({y^i_{vs}}') \leq \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} (P^i_{vs} + {F^i_{vs}}^s){y^i_{vs}}' + V_i^2$$
$$\leq \Lambda^* + \epsilon V_i^2 + V_i^2 = \Lambda^* + (1+\epsilon)V_i^2.$$ □

*Theorem 3:* The approximation ratio of $A_{sub1}$ is $\alpha$, where $\alpha = 1 + (1+\epsilon)cV_i$ and $c = \max_{v_1,v_2,s_1,s_2}\{P^i_{v_1,s_1}/P^i_{v_2,s_2}\}$.

*Proof:* The optimal objective value of IQP (4) is $\Lambda^* \geq \sum_{v\in\mathcal{V}_i}\sum_{s\in[S]} P^i_{vs}{y^i_{vs}}^* \geq \frac{V_i}{c}$ since $P^i_{vs} \geq \frac{1}{c}, \forall v, s$. According to Lemma 3, we have

$$\frac{\Lambda(y^i_{vs})}{\Lambda^*} \leq 1 + \frac{(1+\epsilon)V_i^2}{\Lambda^*} \leq 1 + (1+\epsilon)cV_i.$$ □

### C. A Heuristic Algorithm

We next introduce a heuristic algorithm that starts with an empty solution, and iteratively selects a container for greedy assignment to an available zone. At the $v$-th iteration, we simply choose the $v$-th container from set $\mathcal{V}_i$. Let $\Delta(y^i_{vs})$ denote the increment of the objective value of IQP (4) due to the assignment of $y^i_{vs} = 1$. To select a zone for container $v$, we first compute a candidate set $\mathbb{S}$ that includes all available zones. If the amount of allocated type-$k$ resource in zone $s$ has exceeded the bound $B_{ks}$, $s$ is removed from $\mathbb{S}$. We continue to compute the value of $\Delta(y^i_{vs})$, for $s \in \mathbb{S}$, and choose $s^*$ so that $\Delta(y^i_{vs^*}) = \min_{s\in Set}\Delta(y^i_{vs})$. We assign container $v$ to zone $s^*$. The time complexity of this algorithm is $O(VS)$ since we execute $V_i$ iterations and evaluate $S$ zones during each iteration. We summarize this algorithm in $A_{sub2}$. While we do not prove a performance guarantee, trace-driven simulations in Sec. VI show that $A_{sub2}$ can produce a 2-approximate solution in all cases tested.

## V. Online Algorithm Design

Leveraging the cost minimization algorithms from Sec. IV as a building block, we next present our online algorithm in Sec. V-A. Upon the arrival of each request, the algorithm determines immediately whether to accept it, and if so, how to place its CC. Theoretical analysis is conducted in Sec. V-B.

**Algorithm 3** A Greedy Algorithm $A_{sub2}$

**Input**: **Input**: $\Phi_i, \{P^i_{vs}\}, \{P^i_{v_1,v_2,s_1,s_2}\}, \{a^i_{vk}\}$
1: **for all** $v \in \mathcal{V}_i$ **do**
2: $\mathbb{S} = [S]$;
3: **for all** $s \in [S]$ **do**
4: **if** $z^i_{ks} + a^i_{vk} > B_{ks}, \forall k \in [K]$ **then**
5: $\mathbb{S} = \mathbb{S} \backslash s$;
6: **end if**
7: **end for**
8: **for all** $s \in \mathbb{S}$ **do**
9: Compute the increase $\Delta(y^i_{vs})$ of the objective function value of IQP (4a) due the assignment of $y^i_{vs} = 1$;
10: **end for**
11: $s^* = \arg\min_{s \in \mathbb{S}}\{\Delta(y^i_{vs})\}$;
12: $y^i_{vs^*} = 1; z_{ks^*} = z^i_{ks^*} + a^i_{vk}, \forall k \in [K]$;
13: **end for**
14: $cost_i = \sum_{v \in \mathcal{V}_i} \sum_{s \in [S]} P^i_{vs} y^i_{vs} + \sum_{v_1,v_2 \in \mathcal{V}_i} \sum_{(s_1,s_2) \in \mathbb{E}} P^i_{v_1,v_2,s_1,s_2} y^i_{v_1,s_1} y^i_{v_2,s_2}$; Save $\{y^i_{vs}\}$ to $l^*$;
15: **Return** $cost_i, l^*$.

### A. Online Algorithm Framework

Our main idea for the online algorithm design is as follows. We resort to the help of the classic primal-dual technique, and apply it to the compact-exponential ILP (2) and its dual (3). If request $i$'s placement scheme $l$ is accepted, then let $x_{il} = 1$, and update the corresponding variables $x_i$ and $y^i_{vs}$ in IP (1) according to $l$. Upon arrival of a request $i$, a set of primal variables $x_{il}, \forall l \in \zeta_i$ and the associated dual constraints $(b_{il} - \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f^{il}_{m,t} p_{m,t}, \forall l \in \zeta_i)$ appear. Complementary slackness requires that $x_{il}$ remains zero unless constraint (3a) is tight for scheme $l$. Next, let's examine the right hand side (RHS) of constraint (3a). If we interpret dual variable $p_{m,t}$ as the price per unit of type-$m$ resource at time $t$, then $\sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f^{il}_{m,t} p_{m,t}$ is the placement cost of request $i$'s CC according to scheme $l$. The RHS of (3a) can be viewed as request $i$'s utility with scheme $l$. If we interpret dual variable $u_i$ as request $i$'s utility, $u_i$ can be assigned to the maximum of 0 and the RHS of (3a), *i.e.,*

$$u_i = \max\big\{0, \max_{l \in \zeta_i}\{b_{il} - \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f^{il}_{m,t} p_{m,t}\}\big\}. \tag{7}$$

Accordingly, if $u_i > 0$, we accept request $i$; otherwise, we reject request $i$. The challenge lies in finding scheme $l$ that maximizes the RHS of (3a), which is equivalent to finding the scheme that minimizes the placement cost of request $i$'s CC. Given $p_{m,t}$, it is easy to compute the computation cost $P^i_{vs}$ and communication cost $P^i_{v_1,v_2,s_1,s_2}$. Then the problem becomes the one-shot CC placement problem we studied in Sec. IV, and can be formulated into an IQP (4). Assume that $A_{sub}$ is an $\alpha$-approximation algorithm for IQP (4). It returns a scheme $l^*$ with cost $cost_i$. Then, we have $\frac{cost_i}{\alpha} \le \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f^{il}_{m,t} p_{m,t}, \forall l \in \zeta_i$. Let $u_i = \max\big\{0, b_i - \frac{cost_i}{\alpha}\big\}$, satisfying constraints (3a) for any $l \in \zeta_i$. If $u_i > 0$, request $i$ is accepted and its CC is placed according to scheme $l^*$; if $u_i \le 0$, request $i$ is rejected.

**Algorithm 4** A Primal-Dual Online Framework $A_{online}$

**Input**: $\{\Phi_i\}, \{C_m\}, \{a^i_{vk}\}, U, \alpha$
1: Initialize $\lambda = 2(\alpha U + 1)$;
2: **Upon the arrival of the $i$th CC request**
3: Compute $\{P^i_{vs}\}$ and $\{P^i_{v_1,v_2,s_1,s_2}\}$ based on $\{p_{m,t}\}$;
4: $\big(cost_i, l^*\big) = A_{sub}\big(\Phi_i, \{P^i_{vs}\}, \{P^i_{v_1,v_2,s_1,s_2}\}, \{a^i_{vk}\}\big)$;
5: **if** $b_i - \frac{cost_i}{\alpha} > 0$ **then**
6: $u_i = b_i - \frac{cost_i}{\alpha}; x_i = 1; x_{il^*} = 1$;
7: Update $y^i_{vs}$ and $f^{il^*}_{m,t}$ according to option $l^*$.
8: Accept request $i$ and allocate its CC according to $y^i_{vs}$;
9: **for** $t \in [t_i^-, t_i^+]$ **do**
10: $z_{m,t} = z_{m,t} + f^{il^*}_{m,t}, \forall m \in \mathcal{M}$;
11: $p_{m,t} = \lambda^{\frac{z_{m,t}}{C_m}} - 1, \forall m \in \mathcal{M}$;
12: **end for**
13: **else**
14: Reject request $i$.
15: **end if**

We next discuss the update of dual variable $p_{m,t}$. Recall that $p_{m,t}$ represents the unit price of type-$m$ resource at $t$. We define a new variable $z_{m,t}$ as the amount of type-$m$ resource consumed at $t$, and let $p_{m,t}$ be a function of $z_{m,t}$, as follows:

$$p_{m,t}(z_{m,t}) = \lambda^{\frac{z_{m,t}}{C_m}} - 1, \forall m \in \mathcal{M}, \forall t \in [T], \tag{8}$$

where $\lambda = 2(\alpha U + 1)$. $p_{m,t}$ starts at zero and increases exponentially with the increase of resource consumption. $p_{m,t}$ is close to zero when resources are abundant, allowing CCs to consume resources freely. It grows quickly to a carefully designed large value $\lambda$ when $z_{m,t}$ is close to the capacity $C_m$, so that the service provider will barely allocate any type-$m$ resource to a CC, unless its valuation is sufficiently high.

$A_{online}$ in Algorithm 4 is our online algorithm, which calls $A_{sub}$ for each CC request to determine its placement scheme. Note that $A_{online}$ can call either of the algorithms designed in the previous section ($A_{sub1}$ and $A_{sub2}$), or any alternative that solves IQP (4), as its sub-routine. By default, all variables are set to zero. Line 1 initializes the value of $\lambda$, to prepare for the update of the dual variable $p_{m,t}$. Upon the arrival of a request $i$, we first compute the computation and communication costs when assigning containers to different zones in line 3. $A_{sub}$ is executed for each request to compute a placement scheme $l^*$ and the corresponding cost $cost_i$. If request $i$ can obtain a positive utility in some scheme, it is accepted, and the corresponding primal and dual variables are updated (lines 6–8). We then increase the usage of resources and raise resource prices accordingly (lines 9–12).

### B. Theoretical Analysis

Next we analyze properties of $A_{online}$, based on the assumption that $A_{sub}$ can compute an $\alpha$-approximate solution to IQP (4) in polynomial time.

*1) Feasibility and Running Time:*

*Lemma 4:* $A_{online}$ computes a feasible solution to IP (2) and one for ILP (2a), respectively.

*Proof:* We first examine ILP (2). Constraint (2b) is satisfied because line 6 in $A_{online}$ guarantees that only one option can be accepted. Next, we prove that the capacity constraint (2a) is never violated. Otherwise, let request $i$ be the first accepted request that violates the capacity constraint of type-$m$ resource at time $t$ with option $l^*$. The amount of allocated type-$m$ resource before request $i$ arrives is: $z_{m,t} > C_m - f_{m,t}^{il^*}$. Under the assumption that $\frac{f_{m,t}^{il^*}}{C_m} \leq \frac{1}{\log \lambda}$, the price of type-$m$ resource at $t$ for request $i$ is:

$$p_{m,t} \geq \lambda^{1-\frac{f_{m,t}^{il^*}}{C_m}} - 1 \geq \lambda^{1-\frac{1}{\log \lambda}} - 1 \geq \frac{\lambda}{2} - 1 = \alpha U.$$

Thus, by $U \geq \frac{b_i}{f_m^{il^*}(t)}$, we can obtain

$$cost_i \geq p_{m,t} f_{m,t}^{il^*} \geq \alpha U f_m^{il^*} \geq \alpha b_i.$$

We further have $b_i - \frac{cost_i}{\alpha} \leq 0$, which contradicts the assumption that request $i$ is accepted with $b_i - \frac{cost_i}{\alpha} > 0$. Thus, constraint (2a) holds.

We next investigate IP (1). Constraints (1c), (1d) and (1f) are satisfied by algorithm $A_{sub}$. In addition, the correspondance relation between IP (1) and ILP (2) guarantees constraints (1a), (1b), and (1e) hold. □

*Lemma 5:* $A_{online}$ outputs a feasible solution to LP (3).

*Proof:* Let $OPT_i$ be the optimal objective value of the subproblem in (4) for request $i$. $OPT_i$ equals $\min_{l \in \zeta_i} \{\sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f_{m,t}^{il} p_{m,t}\}$. Because $A_{sub}$ is an $\alpha$-approximation algorithm that generates an objective value $cost_i$, we have $\frac{cost_i}{\alpha} \leq OPT_i$. If $b_i - \frac{cost_i}{\alpha} > 0$, $A_{online}$ updates dual variable $u_i$ in line 6, we obtain

$$u_i = b_i - \frac{cost_i}{\alpha} \geq b_i - OPT_i \geq b_i - \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f_{m,t}^{il} p_{m,t}, \quad \forall l \in \zeta_i.$$

Otherwise, $u_i = 0 \geq b_i - OPT_i$. Therefore, constraint (3a) holds for each request $i$ and the lemma follows. □

*Theorem 4:* $A_{online}$ generates feasible solutions for IP (1), ILP (2) and LP (3) in polynomial time.

*Proof:* Line 1 takes one step to compute the value of $\lambda$. Upon the arrival of request $i$, line 3 takes $O((SV_i)^2)$ steps to initialize the cost vector. $A_{sub}$ in line 4 runs in polynomial time to compute placement cost. Within the body of the `if` statement, lines 6–8 update primal variables in $O(V_i S + (t_i^+ - t_i^- + 1)|\mathcal{M}|)$ steps. The complexity of the `for` loop in lines 9–12 is $O((t_i^+ - t_i^- + 1)|\mathcal{M}|)$. Therefore, the running time of $A_{online}$ is polynomial. Combining Lemma 4 and Lemma 5, we finish the proof. □

*2) Competitive Ratio:* We next analyze the competitive ratio of $A_{online}$. The *competitive ratio* is the upper-bound ratio of optimal objective of IP (1) to the objective value achieved by $A_{online}$. We first prove the primal-dual analysis framework in Lemma 6, which guides the analysis of the competitive ratio. We next define the *Resource-Price Relationship* for $A_{online}$ and the differential version of it, respectively. We prove that if the Resource-Price Relationship holds for a given $\beta$, $A_{online}$ satisfies the inequality in Lemma 6. We then present the value of $\beta$ in Lemma 8 and prove that $A_{online}$ is $\alpha\beta$-competitive in Theorem 5.

Let $OPT_1$ and $OPT_2$ be the optimal objective values of IP (1) and ILP (2), respectively. We have $OPT_1 = OPT_2$. Let $P_i$ and $D_i$ denote the objective value of primal ILP (2) and that of dual LP (3) returned by $A_{online}$ after processing request $i$. Let $P_0$ and $D_0$ be the initial values. $A_{online}$ guarantees $P_0 = D_0 = 0$. Let $P_I$ and $D_I$ be the final primal and dual objective values achieved by $A_{online}$.

*Lemma 6:* If there exist two constants $\alpha \geq 1$ and $\beta \geq 1$ such that $P_i - P_{i-1} \geq \frac{1}{\alpha\beta}(D_i - D_{i-1}), \forall i \in [I]$, then the competitive ratio of $A_{online}$ is $\alpha\beta$.

*Proof:* Summing up the inequalities for each request $i$, we have

$$P_I = \sum_i (P_i - P_{i-1}) \geq \frac{1}{\alpha\beta} \sum_i (D_i - D_{i-1}) = \frac{1}{\alpha\beta} D_I.$$

According to weak duality [36], $D_I \geq OPT_2$, hence, $P_I \geq \frac{1}{\alpha\beta} OPT_2 = \frac{1}{\alpha\beta} OPT_1$. The competitive ratio of $A_{online}$ is $\alpha\beta$. □

We next define a *Resource-Price Relationship* and prove that if it holds for a given $\beta$, then the primal and dual objective values achieved by $A_{online}$ satisfy the inequality in Lemma 6. Let $p_{m,t}^i$ denote the price of type-$m$ resource at time $t$ after handling request $i$. $z_{m,t}^i$ represents the amount of consumed type-$m$ resource at time $t$ after processing request $i$.

*Definition 1:* The Resource-Price Relationship for $A_{online}$ with $\beta \geq 1$ is: $p_{m,t}^{i-1}(z_{m,t}^i - z_{m,t}^{i-1}) \geq \frac{1}{\beta} C_m (p_{m,t}^i - p_{m,t}^{i-1})$, $\forall i \in [I], \forall m \in \mathcal{M}, \forall t \in [t_i^-, t_i^+]$.

*Lemma 7:* If the Resource-Price Relationship holds for a given $\beta \geq 1$, then $A_{online}$ guarantees that $P_i - P_{i-1} \geq \frac{1}{\alpha\beta}(D_i - D_{i-1}), \forall i \in [I]$.

*Proof:* If request $i$ is rejected, then $P_i - P_{i-1} = D_i - D_{i-1} = 0$. Otherwise, we assume that request $i$ is accepted and placed according to option $l$. The increment of the primal objective value is: $P_i - P_{i-1} = b_i$. Note that $A_{online}$ assigns $u_i$ to $b_i - \frac{1}{\alpha} \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} f_{m,t}^{il} p_{m,t}$ when request $i$ with option $l$ is accepted. Therefore,

$$b_i = u_i + \frac{1}{\alpha} \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} p_{m,t}^{i-1} (z_{m,t}^i - z_{m,t}^{i-1}).$$

The increase of the dual objective value is:

$$D_i - D_{i-1} = u_i + \sum_{m \in \mathcal{M}} \sum_{t \in [t_i^-, t_i^+]} C_m (p_{m,t}^i - p_{m,t}^{i-1}).$$

By summing up the Resource-Price Relationship over all $m \in \mathcal{M}$ and $t \in [t_i^-, t_i^+]$, we can obtain:

$$P_i - P_{i-1} \geq u_i + \frac{1}{\alpha\beta}(D_i - D_{i-1} - u_i).$$

Since $u_i \geq 0$ and $\alpha\beta \geq 1$, we have $P_i - P_{i-1} \geq \frac{1}{\alpha\beta}(D_i - D_{i-1})$. □

In order to compute the value of $\beta$, we make the following mild assumption and define the differential version of the Resource-Price Relationship based on it.

*Assumption 1:* The job demand is much smaller than the resource's capacity, *i.e.,* $f_{m,t}^{il} \ll C_m$.

In the real world, a job's demand is usually smaller than a type of resource's capacity in a large data center. We make this

assumption mainly to facilitate our theoretical analysis, such that techniques from calculus (differentiation) can be used. We don't consider extreme cases which are rare in practice. For example, if a high-valued bid demanding almost all the resource is rejected, because a small fraction of the resource is used by other users, then the worst-case competitive ratio can be infinitely large. It is also worth noting that, similar assumptions are made in relevant literature of online resource allocation [43]–[45]. We also relax this assumption completely in our simulation studies.

Under Assumption 1, $z_{m,t}^i - z_{m,t}^{i-1}$ can be expressed as $dz_{m,t}$. The derivative of the Resource-Price Relationship under the above assumption is:

*Definition 2:* The Differential Resource-Price Relationship for $A_{online}$ with $\beta \geq 1$ is: $p_{m,t}dz_{m,t} \geq \frac{C_m}{\beta}dp_{m,t}, \forall m \in \mathcal{M}, \forall t \in [t_i^-, t_i^+]$.

*Lemma 8:* $\beta = \ln \lambda$ and the price function defined in (8) satisfy the Differential Resource-Price Relationship.

*Proof:* The derivative of the price function is: $dp_{m,t} = \lambda^{\frac{z_{m,t}}{C_m}} \frac{\ln \lambda}{C_m} dz_{m,t}$. The Differential Resource-Price Relationship is:

$$(\lambda^{\frac{z_{m,t}}{C_m}} - 1)dz_{m,t} \geq \frac{C_m}{\beta}\lambda^{\frac{z_{m,t}}{C_m}} \frac{1}{C_m} \ln \lambda dz_{m,t}$$

$$\Rightarrow \beta \geq \ln \lambda \frac{\lambda^{\frac{z_{m,t}}{C_m}}}{\lambda^{\frac{z_{m,t}}{C_m}} - 1} \geq \ln \lambda.$$

Therefore this lemma holds for $\beta = \ln \lambda$. □

*Theorem 5:* The online auction $A_{online}$ in Alg. 4 is $\alpha\beta$-competitive, where $\beta = \ln \lambda$ and $\alpha$ is the approximate ratio of $A_{sub}$.

*Proof:* Under Assumption 1, $dz_{m,t} = z_{m,t}^i - z_{m,t}^{i-1}$ is much smaller than the capacity of type-$m$ resource ($C_m$), we have $dp_{m,t} = p'_m dz_{m,t} = p_{m,t}^i - p_{m,t}^{i-1}$. As a result, we can conclude that the Resource-Price Relationship holds for $\beta = \ln \lambda$. Then, combining Lemma 6 and Lemma 7, we finish the proof. □

## VI. Performance Evaluation

We evaluate the performance of our one-shot algorithms $A_{sub1}$, $A_{sub2}$ and online algorithm $A_{online}$ through trace-driven simulation studies. We first introduce the simulation setup for evaluation of the two one-shot algorithms. The default number of zones is set to 9 according to the number of Google data centers in the United States [41]. We exploit Google Cluster Data version 1 [42], and configure each CC according to each job's information in the trace. We assume that each CC contains 2–8 containers and consumes two types of computational resource, since the trace data only includes resource demands for CPU and RAM. The resource consumption $a_{vk}^i$ is set according to the resource demand of each subtask in the trace [42]. The traffic volume between containers $\Delta_{v_1,v_2}^i$ is randomly generated within a range of $[0, 10]$. The cost $P_{vs}^i$ and $P_{v_1,v_2,s_1,s_2}^i$ are randomly drawn from $[0, 1]$. The default value of $B_{ks}$ is 10. For the online setup, we assume each time slot is 5 minutes and the system spans 100 slots by default. Each request's start and end times are set based on each job's timestamp in the trace. The resource capacities, $C_{ks}$ and $D_{s_1,s_2}$ are randomly generated within $[50, 100]$. The request value $b_i$ is randomly chosen from an interval determined by $U$, whose default value is 50. We repeat each set of simulations 20 times, and use the average result to plot the corresponding figure.

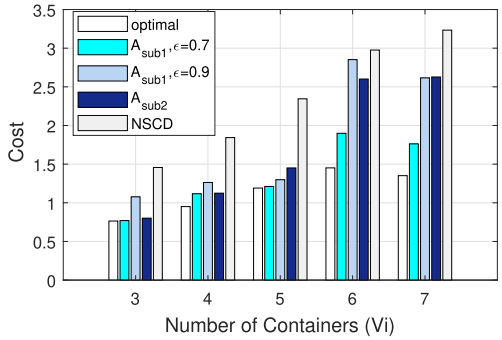



Fig. 2. Cost of $A_{sub1}$, $A_{sub2}$ and NSCD [23] under different values of $V_i$.

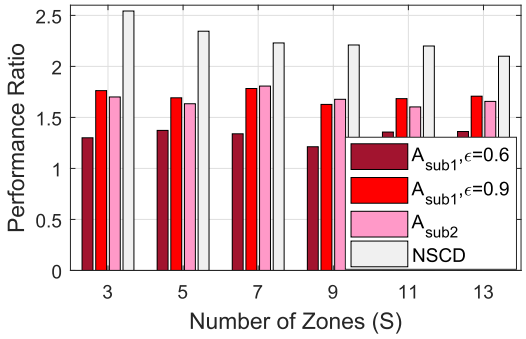



Fig. 3. Performance ratio of $A_{sub1}$, $A_{sub2}$ and NSCD [23] under different values of $S$.

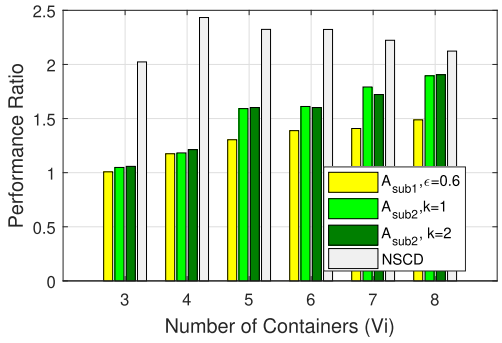



Fig. 4. Performance ratio of $A_{sub1}$, $A_{sub2}$ and NSCD [23] under different values of $V_i$ and $K$.

### A. Performance of $A_{sub1}$ and $A_{sub2}$

*1) Cost and Performance Ratio:* Fig. 2 compares the total cost produced by $A_{sub1}$ and $A_{sub2}$ with the optimal cost under different numbers of containers. We can observe that the gap between the cost of $A_{sub1}$ and the optimal cost becomes larger when the number of containers increases, and gets smaller when the value of $\epsilon$ decreases, which is in line with the analysis in Lemma 3. In addition, $A_{sub1}$ achieves a lower cost than $A_{sub2}$ when we input a smaller $\epsilon$.

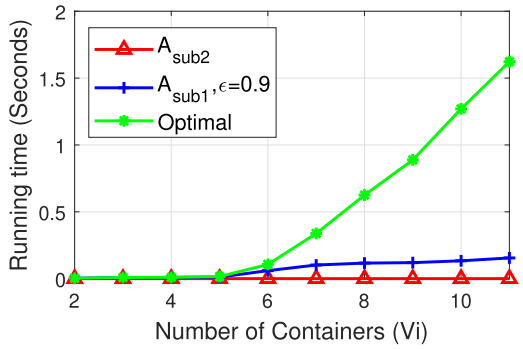



Fig. 5. The average running time of $A_{sub1}$ and $A_{sub2}$ with different $V_i$.

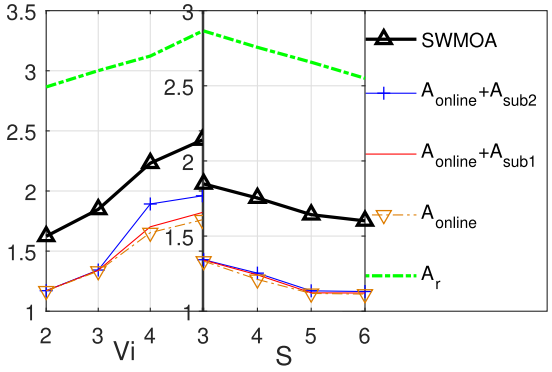



Fig. 6. Performance ratio of $A_{online}$, $A_r$ and SWMOA in [34] under different $V$ and $S$.

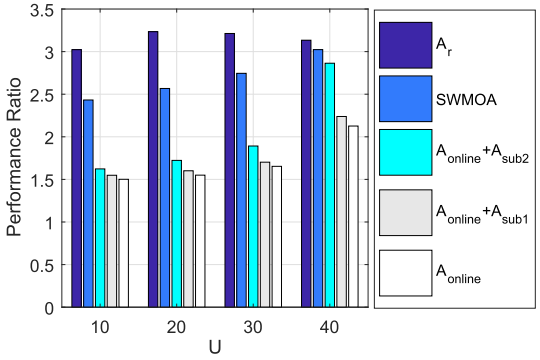



Fig. 7. Performance ratio of $A_{online}$, $A_r$ and SWMOA in [34] with different $U$.

We next examine the *performance ratio*, measured by the ratio of the objective value of IQP (4) generated by our algorithms to the optimal objective value of IQP (4). We fix the number of containers to 5, and plot the performance ratios of $A_{sub1}$ and $A_{sub2}$ in Fig. 3. It can be observed that both $A_{sub1}$ and $A_{sub2}$ perform well with a low performance ratio ($< 2$). The value of $S$ has little impact on the performance of $A_{sub1}$ while the value of $\epsilon$ is related to the ratio, echoing Theorem 3. $A_{sub1}$ outperforms $A_{sub2}$ when $\epsilon$ is relatively small (0.6). We further modify the number of containers and plot the ratios in Fig. 4. The ratio increases with the growth of the number of containers, validating the analysis in Theorem 3 that the value of $V_i$ determines the approximate ratio. Moreover, when there is more than one type of computational resource, $A_{sub2}$ also works well and the ratio is smaller than 2. We implement Tao *et al.*'s one-time algorithm, NSCD [23], which deploys the container cluster to the cheapest zone, for comparison with our one-shot Algorithms $A_{sub1}$ and $A_{sub2}$. In Fig. 2, Fig. 3 and Fig. 4, we can observe that our one-shot algorithms have a better performance than NSCD.

*2) Time Complexity:* We apply the tic and toc functions in MATLAB to measure the execution time of the main program of $A_{sub1}$ and $A_{sub2}$ without counting the initialization stage. We run 20 tests on a laptop computer (Intel Core i7-6700HQ/16GB RAM) and present the average result in Fig. 5. We implement the optimal one-shot algorithm by listing all possible placement schemes. We can observe that both $A_{sub1}$ and $A_{sub2}$ run much faster than the optimal algorithm. The running time grows with an increasing $V_i$, and the observed values are below 0.5 seconds.

### B. Performance of $A_{online}$

*1) Performance Ratio and Objective Value:* We first examine the performance ratio of $A_{online}$, when we call the sub-algorithm that exactly solves IQP (4) (labeled by $A_{online}$), $A_{sub1}$ (labeled by $A_{online} + A_{sub1}$) and $A_{sub2}$ (labeled by $A_{online}+A_{sub2}$), in $A_{online}$. The *performance ratio* of $A_{online}$ is the ratio of the optimal objective value of IP (1) to the objective value of IP (1) generated by $A_{online}$. Based on the observation from the above subsection, we set $\alpha = 2$ for both $A_{sub1}$ and $A_{sub2}$. We fix the number of containers in each CC to 3 and the number of CC requests to 100, but vary the number of zones. The results are plotted in the right of Fig. 6. We observe that the ratio drops sharply with the increase of $S$, but remains steady when $S \geq 5$. This is because more zones bring more placement options. As a result, each CC request has a high probability of being accepted by $A_{online}$, leading to a better performance. When $S$ is large enough, the ratio is dominated by other parameters, *e.g.*, $V_i$ and $U$. The left of Fig. 6 illustrates that a lower ratio comes with a smaller number of CCs. Comparing Fig. 3 and Fig. 4 with Fig. 6, we can conclude that our online algorithm framework incurs only a small loss in performance ratio. In Fig. 7 and Fig. 8, we examine the impact of two parameters: $U$ and $I$. Again, $A_{online}$ with the optimal sub-algorithm has the best performance. The observed ratios are better than the theoretical worst-case bound and remain at a low level. Fig. 7 shows that the performance ratio is larger for a larger value of $U$. The theoretical competitive ratio proven in Theorem 5 implies this result. The ratio fluctuates with the number of requests in Fig. 8, which indicates that the value of $I$ does not have a major influence on the ratio.

We further compare our online algorithm with two related algorithms: i) Shi *et al.*'s online algorithm [34], SWMOA, which also makes decisions based on the current resource prices. Their price function depends on the number of resources and the number of time slots; ii) an online algorithm, $A_r$, which only considers current available resources, *i.e.*, accepts the CC request if there exists one possible placement option (under capacity limit). We have implemented their algorithms and evaluated them using the same trace data. In Fig. 6, we can observe that our online algorithm can achieves a much lower performance ratio than SWMO and $A_r$, under different values of $V_i$ and $S$. In Fig. 7 and Fig. 8, we vary the value of another two parameters, $U$ and $I$, and still

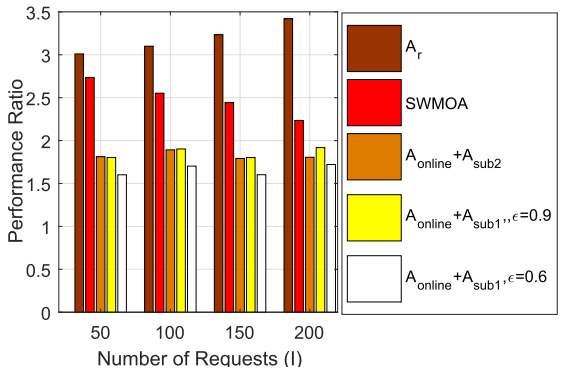



Fig. 8. Performance ratio of $A_{online}$, $A_r$ and SWMOA in [34] with different $I$.

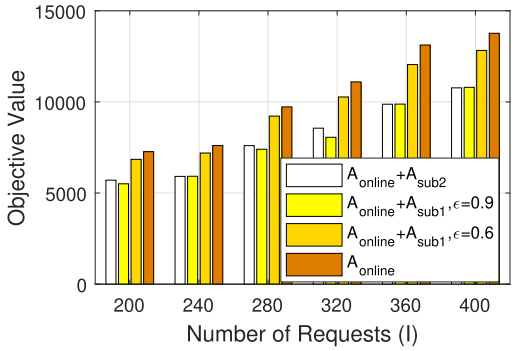



Fig. 9. Objective Value achieved by $A_{online}$.

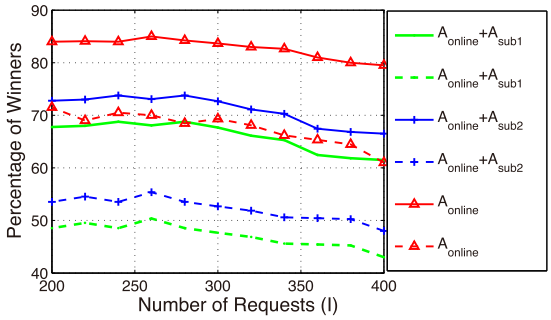



Fig. 10. Percentage of winners under different $I$.

obtain the same observation that our online algorithm always outperforms SWMOA and $A_r$.

We next investigate the objective value of ILP (2a) achieved by $A_{online}$. Fig. 9 reflects that there is an upward trend with a larger number of requests. The underlying reason is that $A_{online}$ can select more high-value requests from a large set of participants. Similar to the observation in Fig. 7, the objective value achieved by $A_{online} + A_{sub1}$ with a small $\epsilon$ is higher than that of $A_{online} + A_{sub2}$.

*2) Winner Satisfaction and Time Complexity: User satisfaction*, which is measured by the percentage of winners, is shown in Fig. 10. The three solid lines represent the percentages of winners when $C_{ks} = 100$, and the three dot lines are for the case of $C_{ks} = 50$. We can see that the percentage of winners drops when a high number of CCs wait for deployment. The reason can be explained as follows: The number of winners remains relatively steady when the resource capacity is fixed.

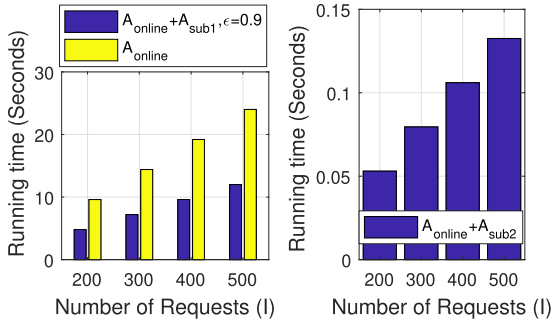



Fig. 11. The average running time of $A_{online}$.

Therefore, only a small percentage of CC requests can be selected from a large set. The resource capacity influences the number of winners. Thus, a higher percentage of winners comes with a large capacity. We next fix the number of containers in each CC to 5. Fig. 11 shows the average running time of our online algorithms with varying number of requests. Again, the shortest running time is observed when we call $A_{sub2}$ in our online algorithm. $A_{online}$ with $A_{sub1}$ has a slightly longer running time, followed by $A_{online}$ with the optimal one-shot algorithm.

## VII. Conclusion

This work presented an efficient online algorithm for placing container clusters in cloud zones, taking container deployment and the demand of inter-container traffic into consideration. Our online placement scheme consists of a one-shot algorithm that determines the placement scheme for the current CC and an online algorithm framework that decomposes the online decision making into on-spot decisions based on resource prices. We leverage exhaustive sampling and ST rounding techniques to compute quality solutions to the one-shot problem, and further exploit compact-exponential and the online primal-dual techniques for guaranteeing a good competitive ratio. Our online algorithm achieves computational and economical efficiencies.

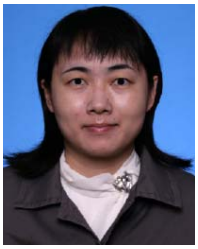
## References

[1] ZDNet. *Containers: Fundam. to Cloud's Evolution*. Accessed: Feb. 10, 2018. [Online]. Available: https://goo.gl/PPWmxe
[2] Google. *Container Engine*. Accessed: Feb. 10, 2018. [Online]. Available: https://cloud.google.com/container-engine/
[3] Amazon. *Amazon EC2 Container Service*. Accessed: Feb. 10, 2018. [Online]. Available: https://aws.amazon.com/ecs/
[4] Aliyun. *Container Service*. Accessed: Feb. 10, 2018. [Online]. Available: https://goo.gl/CnLfBQ
[5] Microsoft. *Azure Container Service*. Accessed: Feb. 10, 2018. [Online]. Available: https://goo.gl/Uh9Vh5
[6] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*. New York, NY, USA: Springer, 2007, pp. 325–341.
[7] K. Hsieh *et al.*, "Gaia: Geo-distributed machine learning approaching lan speeds," in *Proc. NSDI*, 2017, pp. 629–647..
[8] S. Gu, Z. Li, C. Wu, and C. Huang, "An efficient auction mechanism for service chains in the NFV market," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 458-498.
[9] Google. (2018) *Container Clusters*. [Online]. Available: https://goo.gl/7Jt8tC
[10] Amazon. (2016). *Amazon ECS Clusters*. [Online]. Available: https://goo.gl/3pbXwB
[11] Microsoft. (2015). *Azure Container Service Cluster*. [Online]. Available: https://goo.gl/URvRNg
[12] Amazon. (2016). *Regions Availability Zones*. [Online]. Available: https://goo.gl/tpSGRy

[13] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An efficient cloud market mechanism for computing jobs with soft deadlines," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 793–805, Apr. 2017.
[14] S. Arora, D. Karger, and M. Karpinski, "Polynomial time approximation schemes for dense instances of NP-hard problems," in *Proc. ACM STOC*, Feb. 1995, pp. 193–210.
[15] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, no. 1, pp. 461–474, Feb. 1993.
[16] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. IEEE INFOCOM*, 2014.
[17] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," in *Proc. ACM SIGMETRICS*, May 2014, pp. 478–520.
[18] Z. Zheng and N. B. Shroff, "Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[19] Y. Tan, F. Wu, Q. Wu, and X. Liao, "Resource stealing: A resource multiplexing method for mix workloads in cloud system," *J. Supercomput.*, vol. 75, no. 1, pp. 33–49, Jan. 2019.
[20] S. Bitton, Y. Emek, and S. Kutten, "Efficient dispatching of job batches in emerging clouds," in *Proc. IEEE INFOCOM*, Feb. 2018, pp. 56–66.
[21] Kubernetes. (2013). *Production-Grade Container Orchestration*. [Online]. Available: https://kubernetes.io
[22] C.-C. Chang, S.-R. Yang, E.-H. Yeh, P. Lin, and J.-Y. Jeng, "A kubernetes-based monitoring platform for dynamic cloud resource provisioning," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.
[23] Y. Tao, X. Wang, and X. Xu, "Containerized resource provisioning framework for multimedia big data application," *Multimedia Tools Appl.*, vol. 77, no. 9, pp. 11439–11457, May 2018.
[24] P. Waibel, A. Yeshchenko, S. Schulte, and J. Mendling, "Optimized container-based process execution in the cloud," in *OTM*. New York, NY, USA: Springer, 2018.
[25] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE INFOCOM*, May 2018, pp. 258–269.
[26] H. Tang, D. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 530–543, 2019.
[27] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.
[28] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1–9.
[29] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proc. IEEE INFOCOM*, May 2014, pp. 1–15.
[30] R. Yu, G. Xue, X. Zhang, and D. Li, "Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers," in *Proc. IEEE INFOCOM*, May 2017, pp. 258–263.
[31] X. Dai, Y. Wang, J. M. Wang, and B. Bensaou, "Energy efficient virtual cluster embedding in public data centers," in *Proc. IEEE GLOBECOM*, May 2015, pp. 1–12.
[32] G. Even and M. Medina, "Online multi-commodity flow with high demands," in *Proc. Int. Workshop Approximation Online Algorithms*, Jun. 2012, pp. 236–246.
[33] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," in *Proc. ACM SIGCOMM*, Feb. 2014, pp. 455–466.
[34] W. Shi, C. Wu, and Z. Li, "An online mechanism for dynamic virtual cluster provisioning in geo-distributed clouds," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–4.
[35] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 176, no. 2, pp. 657–690, 2007.
[36] N. Buchbinder and J. S. Naor, "The design of competitive online algorithms via a primal–dual approach," *Found. Trends® in Theoretical Computer Science*, vol. 3, nos. 2–3, pp. 93–263, 2009.
[37] (2016). *Guide to Fiber Optimazation Premise Cabling*. [Online]. Available: https://goo.gl/Yd8UsW
[38] B. Lucier, I. Menache, J. S. Naor, and J. Yaniv, "Efficient online scheduling for deadline-sensitive jobs," in *Proc. ACM SPAA*, Jul. 2013, pp. 305–314.
[39] L. Lovász and M. D. Plummer, *Matching theory*. New York, NY, USA: American Mathematical Society, 2009, p. 367.
[40] S. Arora, A. Frieze, and H. Kaplan, "A new rounding procedure for the assignment problem with applications to dense graph arrangement problems," in *Proc. IEEE FOCS*, May 1996, pp. 1–5.
[41] Wikipedia. (2015) *Google Data Centers*. [Online]. Available: https://goo.gl/oKYNri
[42] (2013). *Google Cluster Data, TraceVersion1*. [Online]. Available: https://goo.gl/A9qhru
[43] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau, "Online auctions in IaaS clouds: welfare and profit maximization with server costs," in *Proc. of ACM SIGMETRICS*, 2015.
[44] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," *Operations Res.*, vol. 62, no. 4, pp. 876–890, 2014.
[45] P. Jaillet and X. Lu. (2012). "Near-optimal online algorithms for dynamic resource allocation problems." [Online]. Available: https://arXiv:1208.2596

**Ruiting Zhou** received the M.S. degree in telecommunications from Hong Kong University of Science and Technology, Hong Kong, in 2008, the M.S. degree in computer science from the University of Calgary, Canada, in 2012, and the Ph.D. degree from the Department of Computer Science, University of Calgary, Canada, in 2018. She has been an Associate Professor with the School of Cyber Science and Engineering, Wuhan University, since 2018. She has published research papers in top-tier computer science conferences and journals, including the IEEE INFOCOM, the IEEE/ACM TON, the IEEE JSAC, the IEEE TMC. Her research interests include cloud computing, machine learning, and mobile network optimization. She has received the NSERC Canada Graduate Scholarship, the Alberta Innovates Technology Futures (AITF) Doctoral Scholarship, and the Queen Elizabeth ll Graduate Scholarship, from 2015 to 2018. She serves as a reviewer for international conferences such as the IEEE/ACM IWQOS. She also serves as a reviewer for journals such us the IEEE JSAC, the IEEE TMC, the IEEE TCC, the IEEE TWC, and the IEEE TRANSACTIONS ON SMART GRID.

**Zongpeng Li** received the B.E. degree in computer science from Tsinghua University in 1999 and the Ph.D. degree from the University of Toronto in 2005. He has been with the University of Calgary and then Wuhan University. He has co-authored papers that received Best Paper Award at the following conferences: PAM 2008, HotPOST 2012, and ACM e-Energy 2016. His research interests are in computer networks and cloud computing. He was named as an Edward S. Rogers Senior Scholar in 2004, and he has received the Alberta Ingenuity New Faculty Award in 2007. He was nominated as the Alfred P. Sloan Research Fellow in 2007. He has received the Department Excellence Award from the Department of Computer Science, University of Calgary, the Outstanding Young Computer Science Researcher Prize from the Canadian Association of Computer Science, and the Research Excellence Award from the Faculty of Science, University of Calgary.



**Chuan Wu** (SM'05) received the B.Eng. and M.Eng. degrees from the Department of Computer Science and Technology, Tsinghua University, China, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2008. Since 2008, she has been with the Department of Computer Science, The University of Hong Kong, where she is currently an Associate Professor and the Associate Head of curriculum and development matters. Her current research interests are in the areas of cloud computing, distributed machine learning/big data analytics systems, network function virtualization, and data center networking. She is a member of ACM. She has served as the Chair for the Interest Group on multimedia services and applications over emerging networks (MEN) of the IEEE Multimedia Communication Technical Committee (MMTC) from 2012 to 2014. She has also served as a TPC members and reviewers for various international conferences and journals. She was a co-recipient of the Best Paper Award from HotPOST 2012 and ACM e-Energy 2016. She is an Associate Editor of the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, and *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*.