Achieving Cost Optimization for Tenant Task Placement in Geo-Distributed Clouds

Luyao Luo, *Graduate Student Member, IEEE*, Gongming Zhao, *Member, IEEE*, Hongli Xu, *Member, IEEE*, Zhuolong Yu, and Liguang Xie, *Senior Member, IEEE*

***Abstract*— Cloud infrastructure has gradually displayed a tendency of geographical distribution in order to provide anywhere, anytime connectivity to tenants all over the world. The tenant task placement in geo-distributed clouds comes with three critical and coupled factors: *regional diversity in electricity prices*, *access delay for tenants*, and *traffic demand among tasks*. However, existing works disregard either the regional difference in electricity prices or the tenant requirements in geo-distributed clouds, resulting in increased operating costs or low user QoS. To bridge the gap, we design a cost optimization framework for tenant task placement in geo-distributed clouds, called TanGo. However, it is non-trivial to achieve an optimization framework while meeting all the tenant requirements. To this end, we first formulate the electricity cost minimization for task placement problem as a constrained mixed-integer non-linear programming problem. We then propose a near-optimal algorithm with a tight approximation ratio $(1 - 1/e)$ using an effective submodular-based method. Results of in-depth simulations based on real-world datasets show the effectiveness of our algorithm as well as the overall 10%-30% reduction in electricity expenses compared to commonly-adopted alternatives.**

***Index Terms*— Geo-distributed cloud, task placement, cost effectiveness, multi region, regionless.**

I. Introduction

DEPLOYING enterprise user applications (*e.g.*, Netflix [2], Disney+ [3]) to a shared and multi-region cloud infrastructure has become a new norm to meet application requirements including latency (*e.g.*, 100-150 ms for video streaming) and data sovereignty regulation (*e.g.*, European Union GDPR [4]). This geo-distributed deployment model requires cloud providers (*e.g.*, AWS [5], Microsoft Azure [6], and Google Cloud [7]) to build a multi-region and massive infrastructure in a global scale, setting up tens of data centers across the continents, connecting them in a global backbone network with purpose-built high bandwidth fibers or rent bandwidth from ISPs. Since cloud applications are mainly composed of a number of tasks, this paper focuses on task placement in a geo-distributed cloud. As illustrated in Fig. 1, cloud provider's data centers are spread to multiple regions, offering global coverage and geographical selections to cloud users, aka tenants, for task placement.

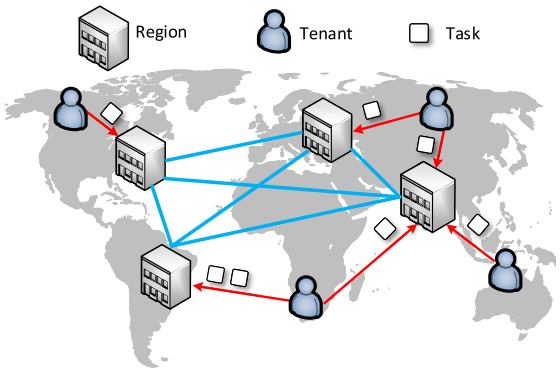


Fig. 1. An example of global tenants placing tasks based on a geographically distributed cloud.

Building hyper-scale cloud data centers near a populated area is neither eco-friendly nor economy efficient due to wasted energy during electricity transmission (*e.g.*, low-cost power supply from the US Midwest to the East Coast, or from China West to China East). The electricity consumed by clouds, correlated to the surging number of servers and the intensive workloads hosted on each server, has been rising in a rapid pace and accounting for 60%-70% of overall operating costs [8]. Therefore, existing cloud deployment model is far from ideal, calling for an innovative set of design on a more cost-effective and eco-friendly cloud deployment. As an effort towards this ambitious goal, our paper aims to answer a meaningful yet unexplored question: Is it possible to schedule and place tenant tasks in a cross-region manner so that overall electricity cost is reduced significantly while meeting desired tenant/application requirements?

Although implicating a positive impact on our environment, it is a non-trivial mission to achieve efficient task placement in a multi-region multi-tenant cloud due to three coupled factors,

Manuscript received 21 January 2023; revised 15 June 2023 and 17 August 2023; accepted 14 September 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor D. Han. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62102392 and Grant 62372426, in part by the National Science Foundation of Jiangsu Province under Grant BK20210121, in part by the Hefei Municipal Natural Science Foundation under Grant 2022013, in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences under Grant 2023481, and in part by the Fundamental Research Funds for the Central Universities. Some preliminary results of this paper were published in the Proceedings of IEEE INFOCOM 2023 [DOI: 10.1109/INFOCOM53939.2023.10229004]. *(Corresponding author: Gongming Zhao.)*

Luyao Luo, Gongming Zhao, and Hongli Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China, and also with the Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu 215123, China (e-mail: lly00@mail.ustc.edu.cn; gmzhao@ustc.edu.cn; xuhongli@ustc.edu.cn).

Zhuolong Yu is with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: zhuolong@cs.jhu.edu).

Liguang Xie is with Virginia Tech, Blacksburg, VA 24061 USA (e-mail: xie@vt.edu).

Digital Object Identifier 10.1109/TNET.2023.3319434

TABLE I
COMPARISON OF ADVANTAGES AND DISADVANTAGES OF EXISTING WORKS

Cost-effective solutions	Cloud Features		Tenant Requirements	
	Price Diversity	Bandwidth Limitation	Access Delay	Traffic Demand
Intra-region (*e.g.*, [9], [10], [11], [12])	✗	✗	✗	✗
Inter-region Power-aware (*e.g.*, [13], [14], [15], [16], [17])	✗	✓	*partial*	✓
Inter-region Price-aware (*e.g.*, [18], [19], [20])	✓	✗	*partial*	✗
TanGo	✓	✓	✓	✓

namely, regional diversity in electricity prices, access delay for tenants, and traffic demand among tasks.

- *Regional diversity in electricity prices*: Relying on the source of power (*e.g.*, hydroelectric, wind, or natural gas) and transmission distance from power plants, the unit price of electricity may vary significantly for data centers located in distinct regions. For example, the annual average day-ahead on peak pricing is $32.57/MWh in the US Northwest compared to $62.71/MWh in New York [21].
- *Access delay for tenants* refers to round-trip access latency between clients and tenant applications/services deployed in the cloud. The desired latency is determined by types of tenant tasks (*e.g.*, time-sensitive versus time-insensitive), and the actual latency varies substantially based on the geographic distance between clients and cloud regions [22], [23], [24].
- *Traffic demand among tasks* refers to the required delay/bandwidth among multiple tasks of the same tenant. The desired delay/bandwidth also depends on the type of tenant tasks, *e.g.*, new computing paradigms like MapReduce [25] and distributed machine learning [26] require high bandwidth among tasks. If deployed in distinct locations (*e.g.*, one in the US East and the other in the US South), a high volume of traffic among tasks may contradict with the limited bandwidth capacity among data centers [16].

Given diversities of tenant demands and regional electricity prices, it requires a fresh look from the community to seek for a practical, efficient and all-in-one task placement solution. To our best knowledge, existing works on tenant task placement often disregard the regional difference of clouds or the tenant requirements for tasks [13], [14], [15], [16], [17], [18], [19], [20], resulting in increased operating costs or low user quality-of-service. For example, some task placement solutions have been proposed to minimize the total electricity cost by leveraging the electricity price difference among multiple regions [18], [19], [20]. However, these works ignore various traffic demands between tasks. When an application imposes a heavy requirement on inter-task communication (*e.g.*, distributed training), it is preferred to co-locate tasks in the same region, otherwise task placement can be more flexible. Overlooking the traffic demands may result in a higher operating cost.

To conquer these challenges, this paper presents an optimization framework, named *TanGo*, wherein tenants can specify their various demands over tasks and cloud providers can place tenant tasks in a cost-effective manner. The core of TanGo is a near-optimal task placement algorithm that could minimize the total cost while satisfying all the demands and constraints. In summary, we make the following contributions:

1) We propose TanGo, a cost optimization framework that includes a mathematical model of the geo-distributed task placement problem. TanGo minimizes the overall electricity cost while meeting all diverse demands.
2) We formulate the task placement problem as a mixed-integer non-linear optimization problem and give a submodular-based solution. We prove that our algorithm is close to optimal and bounded by a tight approximation factor of $(1 - 1/e)$ and then extend to scenarios with duplicated task placement.
3) We conduct extensive experiments based on real-world regional topology, electricity pricing map, and tenant datasets including Alibaba Cluster Trace [27] and Google Cluster Trace [28]. The results show that TanGo can reduce the electricity cost by up to 10%-30% compared with existing solutions.

The rest of this paper is organized as follows. Section II discusses the limitations of the related studies and explains our motivation. Section III gives the formulation of the electricity cost minimization for task placement problem and proposes a polynomial-time submodular-based algorithm. In Section IV, the effectiveness of the proposed algorithm is evaluated using two different task datasets and real-world electricity prices. Section V gives some related works for this paper, and Section VI concludes the paper.

II. BACKGROUND AND MOTIVATION

In this section, we first discuss existing efforts on reducing costs in a geo-distributed cloud and then show their limitations through a motivation example. Finally, we give the overview and workflow of TanGo.

A. Current Cost-Effective Solutions and Limitations

We summarize the advantages and disadvantages of existing works on reducing the electricity consumption/cost as in Table I. According to our research, there are primarily three categories of cost-effective options as follows.

Intra-region: In a single region, some studies consider reducing the resource consumption by scheduling traffic (*e.g.*, TrafficShaper [10]) or workload (*e.g.*, Workloadcompactor [11]). Some other studies consider adopting

eco-friendly energies [9], [12] to reduce the overall cost inside a data center or a region. However, in practice, with the desire for low-cost computing and always-on connectivity of tenants from all over the world, cloud providers often cannot restrict the task distribution on their servers to a particular data center due to computing power constraints and access delay demands.

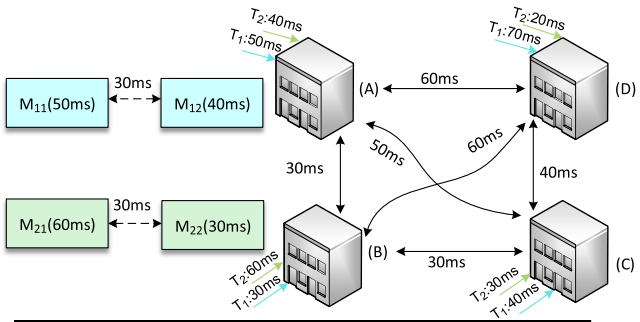
Inter-region power-aware: Some previous works [13], [14], [15], [16], [17] make efforts to reduce power consumption through multi-region task placement in a geo-distributed cloud while offering low access delay to end users [14] or reducing inter-DC traffic volume [16], [17]. However, the objective of cloud providers like Azure [6], AWS [5], and others is to lower overall operating cost, which is based on not only how much power they use, but also the price of the power source, which varies greatly in different regions. These works often ignore the regional differences in resource prices, which may result in increased operating costs.

Inter-region price-aware: Motivated by the geographical diversity in electricity prices, some works [18], [19], [20] focus on the problem of reducing the electricity cost of data centers by redirecting user requests to different data centers with regional electricity price diversity consideration. For example, the authors in [19] study the task placement problem over geo-distributed data centers while guaranteeing the user quality-of-service (*i.e.*, access delay). However, most of these price-aware works consider tasks to be run independently and fail to capture the traffic relation between tasks, thus are not appropriate for the current large-scale distributed cloud system.

In practice, there are mainly two strategies of task placement in current public clouds [29]. One is *Lowest-Delay-First (LDF)*, where tenants select the region with the lowest access delay to place tasks. The other one is *Lowest-Cost-First (LCF)* where cloud providers choose the region with the lowest electricity price that meets tenants' access delay demand. For both strategies, tasks from a tenant tend to be placed in the same region so as to meet various traffic demands between tasks. Furthermore, in the evaluation section, we show the superiority of our approach compared with these two strategies even when we modified the two strategies to allow for cross-region placement while considering the traffic demands.

B. A Motivation Example

Fig. 2 shows a simple scenario of task placement in a geo-distributed cloud. In this scenario, there are four regions (A, B, C, D) and two tenants (T_1 and T_2). Tenant T_1 needs to place two tasks (M_{11} and M_{12}) in these regions while tenant T_2 has two other tasks (M_{21} and M_{22}) to be placed. Fig. 2 also shows the access delay demand for each task and traffic demand among tasks required by each tenant, as well as the delay between tenants and regions. For instance, the access delay between region A and tenants T_1, T_2 is 60ms, 40ms, respectively, and the access delay demand of tenant T_1 for tasks M_{11}, M_{12} is 50ms, 40ms, respectively. Furthermore, we assume that the electricity consumption of each task is 1MWh for simplicity, and each region can accommodate up



Strategy	Region($/MWh)				Total Cost($)
	A(30)	B(50)	C(40)	D(60)	
Lowest-Delay-First	–	M_{11}, M_{12}	–	M_{21}, M_{22}	220
Lowest-Cost-First	–	–	M_{11}, M_{12}	M_{21}, M_{22}	200
TanGo	M_{11}	M_{12}, M_{21}	M_{22}	–	170

Fig. 2. An example of three strategies for placing tenant tasks in different regions. *Top left:* the access delay demand for each task and the delay demand among tasks. *Top right:* the delay between tenants and regions, and the delay among regions. *Bottom:* task placement decisions of each strategy and the overall cost.

to two tasks. The electricity price in regions A, B, C, and D is 30, 50, 40, and 60 ($/MWh), respectively, to distinguish the regional difference in electricity price. We show the placement decisions of LCF and LDF along with our solutions as follows.

- *Lowest-Delay-First Strategy:* If we follow this strategy, then two tasks of tenant T_1 should be placed in region B since it has the lowest access delay (30ms) to tenant T_1. The same is true for tenant T_2 to place tasks M_{21} and M_{22} in region D for the lowest access delay (20ms). As a result, the total cost of these tasks is \$220 ($\$50 \times 2 + \$60 \times 2$).
- *Lowest-Cost-First Strategy:* Since tenant T_1 requires a 40ms access delay for task M_{12}, it places tasks M_{11} and M_{12} in region C with the lowest price. As for tenant T_2, it has to select region D due to the access delay demand for task M_{22}. Then the total cost is \$200 for this strategy.
- *TanGo:* If we place task M_{11} in region A, M_{12} and M_{21} in region B, M_{22} in region C, the total cost becomes \$170 while satisfying all the demands.

This example demonstrates that under the premise of meeting tenant requirements, distributing tasks of a tenant to different regions can cut more costs than other methods. In fact, cross-region task placement is becoming a popular topic. For instance, the East Data West Compute Project has been launched in China to maximize resource utilization by transferring the massive data volume generated in China East to the region in China West with abundant computing power. Motivated by it, this paper proposes a cost optimization framework, called TanGo, which aims to reduce the electricity cost of task placement in geo-distributed clouds while still satisfying all the tenant requirements.

C. Overview of TanGo

As depicted in Fig. 3, TanGo is composed of two fundamental parts: the control plane and the data plane. Specifically, the control plane consists of two components: the state collector and the placement optimizer. The state collector

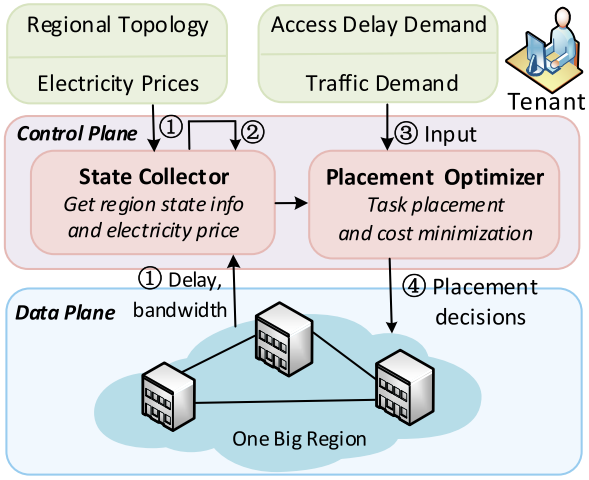


Fig. 3. Overview and workflow of TanGo. TanGo is mainly composed of two parts: the control plane and the data plane. Specifically, the control plane consists of two components: the state collector and the placement optimizer. The data plane consists of data centers located in different regions.

TABLE II
IMPORTANT NOTATIONS

Notations	Semantics
$\mathcal{R}$	the set of cloud regions
$\mathcal{T}$	the set of cloud tenants
$\mathcal{I}$	the set of tasks
$\mathcal{I}^t$	the set of tasks of tenant t
c_k	unit electricity price in region k
τ_k^t	delay between regions k and tenant t
$\tau_{k,k'}$	delay between regions k and k'
$b_{k,k'}$	bandwidth capacity between regions k and k'
r_k	computing power provided by region k
$\tau_{k,k'}$	delay between regions k and k'
τ_i^t	access delay demand for task I_i^t
$\tau_{i,i'}^t$	delay demand between tasks I_i^t and $I_{i'}^t$
$b_{i,i'}^t$	bandwidth demand between tasks I_i^t and $I_{i'}^t$
r_i^t	computing power required by tasks I_i^t
$x_{i,k}^t$	whether task I_i^t will be placed in region k or not

is responsible for collecting regional state information (*e.g.*, regional topology and electricity price). The cloud providers could utilize the detailed information along with the tenant requirements to make optimal placement decisions by the placement optimizer. Moreover, the data plane consists of data centers located in different regions. TanGo provides tenants a "One Big Region" abstraction of the data plane, wherein the tenant can specify their various requirements over all or a subset of tasks without specifying the placement locations.

Fig. 2 briefly describes the workflow of TanGo. ① To make optimal placement decisions, the state collector first collects the regional information, including the regional topology, current electricity prices and the delay/bandwidth between any pair of regions. ② Considering some information may vary over time, the state collector works on a periodical basis. For example, in regions with wholesale power markets, the state collector updates electricity prices hourly or every 15 minutes [30]. ③ Once TanGo obtains tenant inputs, the placement optimizer outputs the mapping of tasks to regions. ④ Based on the output, TanGo places tasks in the clouds. We describe how to achieve cost minimization in detail in Section III.

III. Problem Formulation and Algorithm Design

In this section, we give a detailed modeling of the electricity cost minimization problem for the geo-distributed task placement (CTP). Then we propose an effective submodular-based algorithm to achieve the near-optimal solutions with a $(1-1/e)$ approximation guarantee. The important notations are listed in Table II.

A. Problem Formulation

Network Model. A typical geo-distributed cloud is segregated into different regions from a global standpoint, providing tenants with geographical selections for task placement. Specifically, we use $\mathcal{R} = \{k_1, \ldots, k_K\}$ to represent the set of regions, where $K = |R|$ is the number of regions. The set of tenants in the cloud is denoted as $\mathcal{T}=\{t_1, \ldots, t_{|\mathcal{T}|}\}$. As tenants deploy different kinds of tasks in clouds, we use $\mathcal{I}$ to denote the set of tasks and $\mathcal{I}^t = \{I_1^t, \ldots, I_{|\mathcal{I}^t|}^t\}$ to denote the set of tasks of tenant t.

Inputs and Outputs. As stated in Section II-C, we obtain inputs from two aspects. 1) First, the cloud provider collects regional information by state collector periodically. This information consists of the electricity price in each region and the delay/bandwidth among regions. We did not explicitly model the specific routing and switching mechanisms for the network inter-connectivity between regions, which have been extensively explored by previous studies [16], [31], [32]. Instead, in our current setting, we assume that the data centers are connected via dedicated links that are either self-built or rented from infrastructure providers, with known bandwidth and latency constraints. This simplification allowed us to focus on the optimization of task placement within regions, which is the main focus of this paper. We use $\tau_{k,k'}$ and $b_{k,k'}$ to represent the delay and bandwidth between regions k and k', respectively. The delay and bandwidth among regions can be measured by the state-of-the-art techniques [33], [34]. We also use r_k to denote the computing power provided by region k, which can be measured by the number of CPU cores. 2) Second, each tenant specifies its requirements on tasks, including the access delay demand and traffic demand among tasks. We take these requirements as inputs of our algorithm. Specifically, constant $\tau_{i,t}$ represents the access delay demand between tenant t and task I_i^t. $\tau_{i,i'}^t$ denotes the delay demand between tasks I_i^t and $I_{i'}^t$ (*e.g.*, 100 ms for online conferencing [35]). $b_{i,i'}^t$ denotes the bandwidth demands between tasks I_i^t and $I_{i'}^t$. The key step of CTP is to determine in which regions a tenant's tasks will be placed. We use binary variable $x_{i,k}^t$ to denote whether the task I_i^t will be placed in region k or not.

Constraints. A cost optimization framework for tenant task placement should satisfy the following constraints:

1) *Task Placement Constraint:* Task I_i^t from tenant t should be placed in one and only one region. That is, $\sum_k x_{i,k}^t = 1, \forall I_i^t \in \mathcal{I}^t, t \in \mathcal{T}$.
2) *Access Delay Constraint:* Since tenants require different access delays to their tasks, each task can only be placed in regions close enough to the tenant. It follows $x_{i,k}^t \tau_k^t \leq \tau_i^t, \forall I_i^t \in \mathcal{I}^t, t \in \mathcal{T}, k \in \mathcal{R}$.
3) *Traffic Delay Constraint:* The communication delay between any pair of tasks from a tenant should not exceed the traffic delay demand posed by the tenant. It means $x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \leq \tau_{i,i'}^t, \forall I_i^t, I_{i'}^t \in \mathcal{I}^t, t \in \mathcal{T}, k, k' \in \mathcal{R}$.
4) *Region Bandwidth Constraint:* The total traffic between any pair of regions (k, k') should not exceed the bandwidth capacity constraints $b_{k,k'}$, *i.e.*, $\sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \leq b_{k,k'}, \forall k, k' \in \mathcal{R}$.
5) *Computing Power Constraint:* The placement of a task occupies the computing power of the corresponding region. The computing power capacity constraint of each region k should be satisfied. That is, $\sum_t \sum_i x_{i,k}^t r_i^t \leq r_k, \forall k \in \mathcal{R}$.

Objective. Our objective is to minimize the total electricity cost on the premise of meeting tenant requirements, regional computing power and bandwidth limitations. We give the following problem formulation:

$$\min \sum_k E_k$$

$$S.t \begin{cases} \sum_k x_{i,k}^t = 1, & \forall i, t \\ x_{i,k}^t \tau_k^t \leq \tau_i^t, & \forall i, t, k \\ x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \leq \tau_{i,i'}^t, & \forall (i, i'), t, (k, k') \\ \sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \leq b_{k,k'} & \forall (k, k') \\ \sum_t \sum_i x_{i,k}^t r_i^t \leq r_k, & \forall k \\ E_k = \sum_i \sum_t x_{i,k}^t r_i^t c_k, & \forall k \\ x_{i,k}^t \in \{0, 1\} & \forall i, t, k \end{cases} \quad (1)$$

The first set of equations indicates the task placement constraint. The second to the fifth sets of inequalities denotes the access delay constraint, traffic delay constraint, region bandwidth constraint, and computing power constraint, respectively. The sixth set of equations calculates the total electricity cost E_k of each region k. Our objective is to minimize the total electricity cost of all regions, that is, $\sum_k E_k$.

Example. To demonstrate the practical implications of delay and bandwidth constraints in real-world applications, we consider an actual application example. For instance, within one web application, there are different types of tasks, including web server tasks, application server tasks, and database tasks. Communication between these tasks is crucial for ensuring the smooth functioning of the application. Users accessing the web application will typically interact with the web server, which handles incoming requests and provides the necessary responses. Therefore, this task requires low access latency to ensure a seamless user experience. When a user needs to access specific data, such as retrieving information or submitting forms, the web server communicates with the application server to process the request. The application server, in turn, interacts with the database server to retrieve or update the required data. Consequently, there are bandwidth and delay constraints between the application server and the database server to ensure efficient data retrieval and feedback to the user.

Algorithm 1 Searching for Available Region Set of Each Task

```
Step 1: Initialization withe the access delay demand
Initialize each available set A(I_i^t) to the empty set
for I_i^t ∈ I do
    for k ∈ R do
        if τ_k^t ≤ τ_i^t then
            A(I_i^T) ← A(I_i^T) + k
        end if
    end for
end for
Step 2: Iterative Update
for t ∈ T do
    Set Flag ← 1
    while Flag == 1 do
        Set Flag ← 0
        for I_i^t ∈ I^t do
            for k ∈ A(I_i^t) do
                if τ_{k,k'} ≥ τ_{i,i'}^t ∀k' ∈ A(I_{i'}^t), ∃i' ∈ I^t − I_i^t then
                    A(I_i^t) ← A(I_{i'}^t) − k
                    Flag ← 1
                end if
            end for
        end for
    end while
    Output the available set of each task in I^t
end for
```

Theorem 1: The CTP problem is NP-hard.

Proof: We prove the NP-hardness by showing that the Multiple Knapsack Problem (MKP) [36] is a special case of CTP. In fact, if we dismiss all the delay and bandwidth constraints (*i.e.*, access delay constraint, traffic delay constraint, and the region bandwidth constraint), our CTP problem turns to be a Multiple Knapsack Problem, where each region k can be viewed as a knapsack with capacity r_k and each task I_i^t can be viewed as an item with weight r_i^t. In this case, the goal of this problem is to place tasks in different regions to maximize the final revenue while satisfying the capacity constraints of all the regions. Since the Multiple Knapsack Problem is a special case of our problem, we can conclude that the CTP problem is NP-hard. □

The optimization formulation of task placement with various constraints in Eq. (1) results in a complex mixed integer non-linear programming problem that is computationally hard. Despite using a state-of-the-art LP solver (*e.g.*, Gurobi [37]), it still needs an order of hours to solve even for relatively small scales (*e.g.*, 1000 tasks) [38]. Thus, how to design an efficient algorithm for CTP is challenging.

B. Algorithm Design

1) Preliminaries: In general, we need to place the tasks in $K = |\mathcal{R}|$ regions while meeting the tenant requirements, *i.e.*, the access delay and traffic demand among tasks. After placing tasks in K regions, these tasks is certainly divided into K sets, denoted as $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_K\}$, where tasks in $\mathcal{I}_k$ are placed in region k. Then the total electricity cost of all tasks is expressed as $\sum_{k\in K} C(\mathcal{I}_k)$, where $C(\mathcal{I}_k) = \sum_{I_i^t \in \mathcal{I}_k} c_k r_i^t$, c_k is the unit electricity price in region k and r_i^t is the computing power required by task I_i^t. We know that the total electricity cost of all tasks will not exceed $C_{max} = \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{I}^t} c_{max} r_i^t$, where $c_{max} = \max_k c_k$ is the maximum unit electricity price among these regions. That means the minimization problem in Eq. (1) can be converted into the following equivalent maximization problem in Eq. (2), where $C_{max} - \sum_{k\in K} C(\mathcal{I}_k)$ means the total cost reduction.

$$\max C_{max} - \sum_{k\in K} C(\mathcal{I}_k)$$
$$S.t \begin{cases} \sum_k x_{i,k}^t = 1, & \forall i,t \\ x_{i,k}^t \tau_k^t \le \tau_i^t, & \forall i,t,k \\ x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \le \tau_{i,i'}^t, & \forall (i,i'),t,(k,k') \\ \sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \le b_{k,k'} & \forall (k,k') \\ \sum_t \sum_i x_{i,k}^t r_i^t \le r_k, & \forall k \\ x_{i,k}^t \in \{0,1\} & \forall i,t,k \end{cases} \tag{2}$$

Obviously, the optimal solution to Eq. (2) is also the optimal solution to Eq. (1). The problem formulation in Eq. (2) is similar to a clustering problem, where we need to divide the task set $\mathcal{I}$ into K clusters so as to maximize the electricity cost reduction of all tasks. However, the selectable regions for each task are restricted due to the various demands for access delay with the tenant and traffic with other tasks. We call these regions as *available regions* for tasks and give the definition of the available region set as follows:

Definition 1: Given a task I_i^t from tenant t, a subset $A(I_i^t)$ of $\mathcal{R}$ is defined as available region set for task I_i^t if $\tau_k^t \le \tau_i^t$ and $\tau_{k,k'} \le \tau_{i,i'}^t$ for all $k \in A(\mathcal{I}_i^t)$ and $I_{i'}^t$ in k' for all $k' \in \mathcal{R} - A(\mathcal{I}_i^t)$.

To start, we use the access delay demand to ascertain whether the region is available or not. After determining the placement location for a task, we update the available set $A(I_i^t)$ of each task I_i^t based on the traffic demand among tasks iteratively as shown in Alg.1. Since there are at most $|\mathcal{I}|$ tasks for each tenant, the iterative update procedure in step 2 runs at most $|\mathcal{I}|$ times. Thus, the time complexity of Alg. 1 is $O(K|\mathcal{I}|^2)$.

Submodular function. Our algorithm is based on efficient computations of a submodular set function H, which defines the maximum cost reduction by dividing the tasks into several sets. Without loss of generality, we consider that the unit price of the regions is sorted in ascending order. That is, $c_1 \le c_2 \le \cdots \le c_K$. We then give the definition of the submodular set function H as follows.

Definition 2: Given the set Φ, which contains disjoint subsets of $\mathcal{I}$, the reduction of cost achieved by dividing the tasks according to Φ is defined as:

$$H(\Phi) = C_{max} - \sum_{\Phi_n\in\Phi} \sum_{I_i^t\in\Phi_n} c_{k_n} r_i^t \tag{3}$$

where k_n is the region with the lowest electricity price that can accommodate all tasks in Φ_n. It is determined in Alg. 2.

Next, we give the definition of submodularity and prove that the function H is submodular in Section III-B.3.

Definition 3: (Submodularity [39]): Given a finite set E, a real-valued function z on the set of subsets of E is called *submodular* if $z(S\cup\{e\}) - z(S) \le z(S'\cup\{e\}) - z(S')$ for all $S' \subseteq S \subseteq E$ and $e \in E - S$.

To maintain the computing power and bandwidth constraints of the region k, we only focus on the task set $B \subset \mathcal{I}$ without breaking the constraints. That is,

$$\begin{cases} \sum_{I_i^t\in B} r_i^t \le r_k \\ \sum_{I_i^t\in B} \sum_{I_{i'}^t\in k'} b_{i,i'}^t \le b_{k,k'}, \forall k' \in \mathcal{R} \end{cases} \tag{4}$$

We call the task sets satisfying Eq. (4) as *feasible task sets* for region k. The feasible sets can be explored efficiently by simply performing a depth-first search [40] on tasks to which region k is available through the available region sets. During each iteration of the depth-first search, we gradually expand the candidate feasible task set by adding untraversed tasks and simultaneously update the leftover computing power and bandwidth between other regions.

2) Algorithm Description: Given these insights, we propose the submodular-based algorithm (SM-CTP) for the CTP problem in detail, which is formally described in Alg. 2. SM-CTP consists of three steps. In the first step, the algorithm computes the available region set for each task, and feasible task sets for each region in advance (Line 2), and starts with an empty set Φ (Line 3). In the second step (Lines 5-14), it loops through the possible feasible task set S for each region to find the maximum function value $\max_S H(\Phi\cup\{S\})$ (Lines 5-11). At the end of each iteration, we add the feasible task set S with the maximum submodular function value into Φ (Line 12). After that, we update the available region set for each task and the feasible task sets for each region based on the updated available region sets (Lines 13-14). The algorithm performs $K-1$ iterations until we obtain K sets of tasks in Φ. In the third step (Lines 16-18), we obtain the mapping relationship between tasks and regions (*i.e.*, $x_{i,k}^t$).

3) Performance Analysis: We analyze the approximation performance of our proposed algorithm based on the following lemmas.

Lemma 2: Given the set U as the power set of $\mathcal{I}$, the function H defined in Eq. (3) is submodular on U.

Proof: Without loss of generality, we consider an arbitrary set $\Phi \subseteq U$ and an arbitrary set $M \subseteq \mathcal{I}$. Assume that M does not intersect with other sets in Φ, *i.e.*, $M \cap S = \emptyset, \forall S \in \Phi$. Then, we have

$$H(\Phi\cup\{M\}) - H(\Phi) = \sum_{I_i^t\in M} (c_{max} - c_{k_m}) r_i^t \tag{5}$$

Algorithm 2 SM-CTP: Submodular-Based Algorithm for CTP

Step 1: Initialization
Compute the available region set for each task, and the set of feasible task sets B_k for each region k
$\Phi \leftarrow \emptyset$
Step 2: Iterative Selection
while $|\Phi| \leq K-1$ **do**
  Set $tmp \leftarrow 0, opt \leftarrow 0$
  for $k \in \mathcal{R}$ **do**
    for $S \in B_k - \Phi$ **do**
      $tmp \leftarrow H(\Phi \cup \{S\})$
      if $tmp > opt$ **then**
        $opt \leftarrow tmp$, $S^* \leftarrow S$
      end if
    end for
  end for
  $\Phi \leftarrow \Phi + \{S^*\}$
  Update available region sets with Alg. 1
  Update the feasible task sets based on the updated available region sets
end while
$\Phi \leftarrow \Phi + \{\mathcal{I} - \bigcup_{S\in\Phi} S\}$
Step 3: Assignment of tasks and regions
for $S \in \Phi$ **do**
  Set $x_{i,t}^k = 1$ if $I_i^t \in S$ if S is taken out from B_k
end for

where k_m is the region with the lowest electricity price that can accommodate all tasks in M after placing tasks in Φ. That is, k_m is an available region for all tasks in M, and M is a feasible task set for region k_m. Given an arbitrary subset $\Phi' \subseteq \Phi$, it also follows

$$H(\Phi' \cup \{M\}) - H(\Phi') = \sum\nolimits_{I_i^t \in M} (c_{max} - c_{k'_m}) r_i^t \quad (6)$$

where k'_m is the region with the lowest electricity price that can accommodate all tasks in M after placing tasks in Φ'.

Note that two situations may happen: 1) The task sets in $\Phi - \Phi'$ do not affect the placement results of tasks in M. That is to say, after placing the tasks in $\Phi - \Phi'$, region k'_m is still able to accommodate all tasks in M. In this situation, tasks in M will be placed in the same region k'_m, *i.e.*, $c_{k_m} = c_{k'_m}$. 2) If region k'_m cannot accommodate all tasks in M after placing the tasks inside $\Phi - \Phi'$, tasks in M should be placed in another available region k_m. In this situation, we know that $c_{k_m} > c_{k'_m}$. As a result, in any situation, we have

$$c_{k_m} \geq c_{k'_m} \quad (7)$$

From Eq. (7), we can get:

$$\sum\nolimits_{I_i^t \in M} (c_{max} - c_{k_m}) r_i^t \leq \sum\nolimits_{I_i^t \in M} (c_{max} - c_{k'_m}) r_i^t \quad (8)$$

Combining Eqs. (5), (6) and (8), we know that:

$$H(\Phi \cup \{M\}) - H(\Phi) \leq H(\Phi' \cup \{M\}) - H(\Phi') \quad (9)$$

According to Definition 3, we show that the set function H is submodular. □

Lemma 3: For a real-valued submodular and non-decreasing function $z(S)$ on U, the optimization problem $max_{S\subseteq U}\{z(S) : |S| \leq K, z(S)\ is\ submodular\}$ can reach a $(1-1/e)$ approximation factor if the algorithm performs greedily [39].

Theorem 4: Our SM-CTP algorithm achieves a $(1-1/e)$ approximation factor for the maximization problem as formulated in Eq. (2).

Proof: We have proved that the function H is submodular in Lemma 2. Besides, for any set Φ containing subsets of $\mathcal{I}$ and $M \subseteq \mathcal{I}$ with $M \cap S = \emptyset, \forall S \in \Phi$, it follows:

$$H(\Phi \cup \{M\}) - H(\Phi) \geq 0 \quad (10)$$

since c_{max} is the maximum unit electricity price among regions. The equal sign is held only in the case where k_m is the region with the highest price. Thus, the function H is non-decreasing. By applying Lemma 3, we prove that our proposed algorithm can reach a $(1-1/e)$ approximation factor for the CTP problem in Eq. (2). □

In fact, Nemhauser [41] has proved that any algorithm evaluating the submodular function at a polynomial number of sets will not be able to obtain an approximation guarantee better than $(1-1/e)$, unless $NP = P$. Thus, we have:

Theorem 5: The maximization problem in Eq. (2) does not admit a polynomial-time algorithm with approximation ratio $1-1/e+\epsilon$ unless $NP = P$, where ϵ is an arbitrary positive constant.

We should note that the number of feasible sets for each region may be exponential. However, the work [42] has shown that a polynomial number of feasible sets are enough for performance optimization. To achieve the trade-off optimization between algorithm complexity and network performance, we only construct the polynomial number (with input the quadratic number of tasks $|\mathcal{I}|^2$) of feasible sets for each region. Under this condition, the function H is calculated $O(K|\mathcal{I}|^2)$ times in each iteration and the algorithm runs in $K-1$ iterations. As a result, the time complexity of SM-CTP is $O(K^2|\mathcal{I}|^2)$.

C. Duplicated Placement

The CTP problem considers the situations where each task only needs to be placed once in a cloud. The one-location-only placement strategy, however, does not always entirely satisfy the requirements of tenants. In some cases (*e.g.*, CDN [43]), tenants may have different geographical access requirements from different locations. For example, as shown in Fig. 4, the tenant has an access delay demand of less than 30ms for this task at two different locations (locations 1 and 2). However, no matter which region we place the task in, we fail to satisfy the access delay demands of both locations simultaneously. One way to solve this problem is to duplicate the task in two copies in regions A and C separately. In this way, the tenants can access the task from region A at location 1 and from region C at location 2 while satisfying all the access delay demands.

Under these cases, the SM-CTP algorithm may fail because the searching procedure for the available region set of each task could return an empty set if there exists no region that can meet all the delay demand constraints. Motivated by the duplicated deployment strategy [44], [45], we adopt this strategy to enhance our algorithm to cope with different

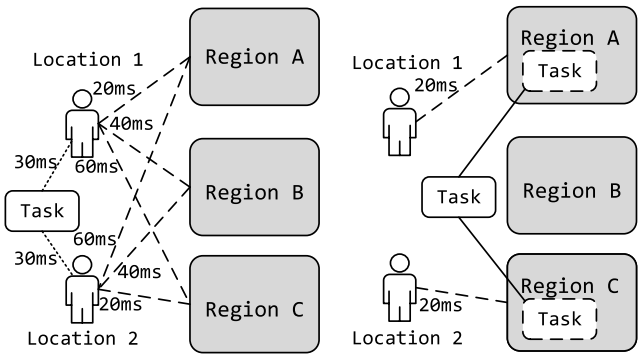


Fig. 4. An example of duplicated deployment strategies for replicating tenant tasks in different regions. In this example, we assume that the tenant has access delay demands at two different locations. *Left:* the access delay demand for a task and the delay between tenant and regions. *Right:* The task is replicated into two copies and placed in region A and region C separately to meet all the access delay demands.

situations. In general, we try to solve the duplicated placement problem as follows:

$$\min \sum_k E_k$$
$$S.t \begin{cases} \sum_k x_{i,k}^t \geq 1, & \forall i,t \\ 0 < x_{i,k}^t \tau_k^{t,n} \leq \tau_i^{t,n}, & \forall i,t,n,\exists k \\ 0 < x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \leq \tau_{i,i'}^t, & \forall (i,i'),t,\exists (k,k') \\ \sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \leq b_{k,k'} & \forall (k,k') \\ \sum_t \sum_i x_{i,k}^t r_i^t \leq r_k, & \forall k \\ E_k = \sum_i \sum_t x_{i,k}^t r_i^t c_k, & \forall k \\ x_{i,k}^t \in \{0,1\} & \forall i,t,k \end{cases} \tag{11}$$

The constant $\tau_i^{t,n}$ denotes the n-th access delay demand of tenant t at location n for task I_i^t. The first set of inequalities indicates the task placement constraint. That is, task I_i^t from tenant t should be placed in at least one region. The second set of inequalities denotes the access delay constraint, where $x_{i,k}^t \tau_k^{t,n}$ represents the access delay from location n to region k if we place task I_i^t in the region k. That is to say, there exists at least one region where task I_i^t is placed to meet all the access delay demand. The third set of inequalities expresses the traffic delay constraint. That is, there exists at least one pair of regions where task I_i^t and $I_{i'}^t$ be placed that can satisfy each traffic delay demand. The fourth to sixth sets of inequalities remains the same as in Eq. (1).

On the one hand, adding more task replicas makes it easier to meet the requirements of tenants. On the other hand, placing more task replicas means a significant increase in power consumption. Thus, our objective here is also to minimize the total electricity cost of all regions, that is, deploying as few task replicas as to cover all the tenants' requirements. To this end, we give an iterative algorithm to gradually add duplicate tasks to the task set $\mathcal{I}$ if Alg. 1 fails to find a non-empty available set for any task, as shown in Alg. 3.

Algorithm 3 The Overall Procedure of Duplicated Placement A

1: **Step 1: Initialization**
2: Initialize each available set $A(I_i^t)$ by Alg. 1
3: **Step 2: Adding duplicated tasks**
4: **for** $I_i^t \in \mathcal{I}$ **do**
5: **if** $A(I_i^t)$ is $\emptyset$ **then**
6: Put a duplicated task $I_{i'}^t$ into task set $\mathcal{I}$
7: **if** $\tau_k^{t,n} \leq \tau_i^{t,n}$ for each each access delay demand $\tau_i^{t,n}$ **then**
8: $A(I_i^t) \leftarrow A(I_i^t) + k$
9: **else**
10: Transfer the demand $\tau_i^{t,n}$ to duplicated task $I_{i'}^t$
11: **end if**
12: Calculate the available set $A(I_{i'}^t)$ by steps 1 and 2
13: **end if**
14: **end for**
15: **Step 3: Assignment of tasks and regions**
16: Output the assignment of tasks and regions by Alg. 2

Alg. 3 first initializes each available set for each task by Alg. 1 (Lines 1-2). If the available region set $A(I_i^t)$ is empty for any task I_i^t, we know that placing this task in only one region cannot fully meet all the access delay demand. Then the algorithm adds a duplicated task $I_{i'}^t$ into the task set $\mathcal{I}$ (Lines 4-6). Then it re-inits the available set of I_i^t and tries to satisfy access delay demand as much as possible, and other unmet demands are transferred to the duplicated task (Line 7-10). The algorithm runs iteratively steps 1 and 2 until there is no empty available set (Lines 12). At last, we run algorithm Alg. 2 on the task set $\mathcal{I}$ to get the final results. Considering that there are at most K duplicated tasks of each task, steps 1 and 2 run at most $K|\mathcal{I}|$ times. As a result, the overall time complexity of duplicated placement algorithm is $O(K^2|\mathcal{I}|^3)$.

D. Discussion

This section discusses some issues to enhance our TanGo mechanism and extends it to adapt a more general situations.

A more general optimization goal. Although electricity cost is a major component of cloud service providers' operating costs, there may be other factors that impact the cost and benefit of cloud providers in some practical scenarios [46]. For example, carbon emissions associated with electricity consumption have become an increasingly important concern in recent years. Besides, in scenarios where billing is based on the size of traffic, the cost of bandwidth cannot be ignored, especially in high-traffic situations. In addition, load balancing and user experience may also need to be taken into account as they may impact the willingness of tenants to pay for the service.

To address such multi-objective optimization problems, a feasible approach is to define the optimization objective as a utility function, which can be formulated to balance different goals, such as minimizing the energy and bandwidth cost, reducing carbon footprint, improving performance, or maximizing user satisfaction. Specifically, the utility function can be defined as a weighted sum of the different goals,

where the weights reflect the importance of each goal to the overall objective. By optimizing the utility function as shown in Eq. (12), cloud providers can make trade-offs between different objectives and achieve a more balanced and efficient solution.

$$\max \sum_{j} U_j(f_j) \tag{12}$$

where f_j is j-th factor and U_j is its corresponding utility function. For example, if the cloud provider considers including bandwidth costs in the optimization objective, a possible function definition can be defined as $-\sum_{k,k'} d_{k,k'} \sum_t \sum_{i,i'} x^t_{i,k} x^t_{i',k'} b^t_{i,i'}$, where $d_{k,k'}$ is the unit bandwidth cost between region k and k' and the negative sign indicates an attempt to minimize the cost. Moreover, if the cloud provider tries to minimize the total access delay of tenants, another function could be defined as $-\alpha \sum_k \sum_t \sum_i x^t_{i,k} \tau^t_k$, where α represents the proportion of this factor in the overall objective function. Our proposed algorithm can be adapted to handle such scenarios by modifying the definition of set function H and feasible task sets, and its optimality is preserved as long as H remains submodular.

However, the respective weights of these factors are subjective and may require domain-specific knowledge. Besides, some utility functions cannot be defined simply as linear (*e.g.*, P95 for bandwidth billing [16]). We leave these issues as our future work.

Workload replication. As outlined in Algorithm 3, our objective is to strategically place duplicated tasks in order to accommodate various access delay requirements. It is natural to inquire about the bandwidth and delay requirements for individual tasks across different replicas. However, we should note that our proposed solution operates on the principle of workload replication, where each copy of the task is scheduled and executed independently in its respective region, without the need for inter-replica communication.

For example, we consider a scenario involving high-load conditions. In such scenarios, it is common to deploy a load balancer along with multiple duplicated backend servers. These backend servers are responsible for distributing the incoming workload and ensuring high availability. The load balancer acts as a traffic manager, receiving incoming requests from clients and effectively distributing them across the available backend servers. It is worth emphasizing that, in this specific setup, there is no requirement for communication among the backend servers. Each backend server operates autonomously, handling its assigned requests without any direct communication with other replicas. Upon receiving a request, a server independently processes the task, performs necessary computations or data retrieval, and generates the corresponding response for the client. This self-contained execution enables efficient parallel processing of requests without the need for inter-replica coordination.

By leveraging this approach, we can achieve workload replication and optimize task placement without introducing communication overhead among replicated tasks. The independence of backend servers ensures that each task replica can operate in isolation, resulting in improved scalability, fault tolerance, and response time.

Overall, our solution facilitates the efficient distribution of tasks by strategically placing duplicated replicas while allowing them to operate independently without the need for inter-replica communication. This approach harnesses the advantages of workload replication and enables effective utilization of resources in diverse geographical regions.

IV. Performance Evaluation

A. Performance Metrics and Benchmarks

We evaluate the effectiveness of TanGo through the most important metric, *i.e.*, the electricity cost, including the hourly cost and the average hourly cost. The hourly cost is the electricity price multiplied by the total electricity consumption of all regions in an hour. The average hourly cost is calculated over the entire period. Moreover, we compared TanGo with the following three practical benchmarks.

1) The first one, known as Lowest-Delay-First (LDF), places all tasks of a tenant in the same region to meet the traffic demand among tasks. LDF simulates the preference behavior of tenants in the cloud and selects the region with the lowest access delay to place tasks [47]. However, this placement strategy does not consider the cost factor. The computational complexity of LDF is $O(K|\mathcal{T}|)$, as it selects one region k out of $|K|$ regions for each tenant t in $\mathcal{T}$.
2) The second one, called Lowest-Cost-First (LCF), also places tasks of the same tenant in the same region, which mimics the preference decision-making process of cloud providers such as Google [28]. Specifically, it chooses the region with the lowest electricity price that meets tenants' access delay demand. The computational complexity of LDF is also $O(K|\mathcal{T}|)$.
3) The third one, referred to as Cross-Region Lowest-Cost-First (CR-LCF), uses a greedy cross-region strategy based on LCF for task placement. Since no existing work fully takes into account both regional diversity in electricity prices and tenant requirements in region-wide task placement, we employ this approach to better illustrate the benefits of our solution. CR-LCF places a task in a region with the lowest prices while satisfying all the tenant requirements, including the access delay demand and the traffic demands among tasks. Unlike both LDF and LCF, CR-LCF may place tasks of a tenant in multiple regions. The computational complexity of CR-LCF is $O(K|\mathcal{I}|)$, as it develops a placement strategy for each task of $\mathcal{I}$ individually.

B. Simulation Settings

Regional Topology. We assume that the regions are spread across the continental U.S. in order to account for the geographic diversity of the cloud infrastructure and electricity prices as in [20]. For the sake of convenience, we suppose that each region, as depicted in Fig. 5, has a single data center in a randomly selected location. In accordance with the

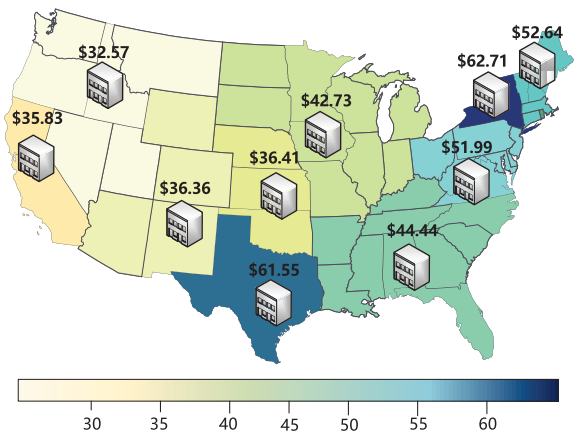
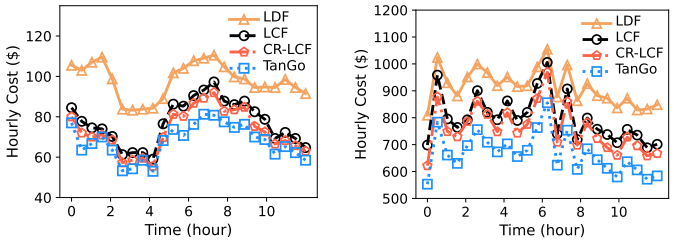


Fig. 5. The U.S. cloud region selection topology and electricity price ($/MWh) [21].



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 6. Hourly cost vs. time.

topology settings established in previous studies [18], [20], we selected 10 specific cities in the United States to serve as the locations for our data centers. Similarly, the tenants access these regions (*i.e.*, data centers) randomly from another 100 cities. A random selection of cities was made to ensure that our simulation results are not specific to any particular city and that our results can be generalized to other cities. Following the Euclidean distance which is easy to calculate, we map the transmission latency between tenants and data centers to the interval of 10ms to 100ms. We utilize the Federal Energy Regulatory Commission's annual average day-ahead on peak pricing ($/MWh) as the electricity price for each region [21]. For example, the electricity price in California is $35.83/MWh while that of New York is $62.71/MWh. The bandwidth between data centers is set to 50Gbps by default [17].

Task Sets. We conduct simulations on two different real-world task datasets: (a) Alibaba Cluster Trace [27] and (b) Google Cluster Trace [28]. These two datasets both contain the arrival time, the requested resources, and the duration of tasks. More specifically, Alibaba Cluster Trace covers up to 76 thousands tasks submitted by 12 thousands tenants over 12 hours. We utilize this dataset to simulate low workload situations, *i.e.*, when the system is running well below the total capacity it can handle. Google Cluster Trace covers up to 25 million tasks submitted by 2 million tenants over 29 days. We use this dataset to simulate high workload scenarios to study the performance of TanGo in high load clouds. For fairness, we select tasks within 12 hours from both datasets. Without loss of generality, we assume that a server at full load consumes 1 KWh electricity per hour [48].

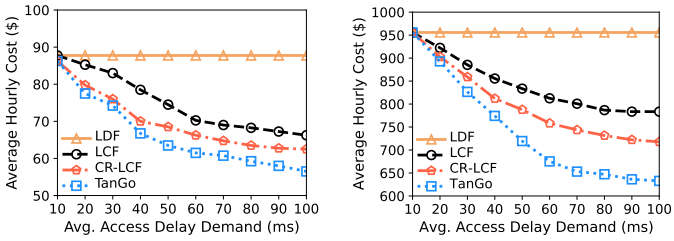
Tenant Requirements. To demonstrate the efficiency of our algorithm in various settings, we thoroughly test how the tenant requirements affect both our algorithm and the comparison algorithms in the experiments. By default, we assume that each tenant requires an average of 50ms access delay following the uniform distribution. For the inter-task traffic demands, we specify 5k task communication pairs from the task datasets, where the delay and bandwidth demands are also taken from the uniform distribution with an average bandwidth demand of 50Mbps and an average delay demand of 50ms. While these parameters may deviate somewhat from actual values, we perform multiple sets of tests to ensure the robustness of our approaches, as shown in Figs. 7-10.

C. Evaluation Results

1) Evaluation on Overall Costs: We first test the performance of TanGo against other comparison algorithms in different scenarios without duplicated placement. The evaluation results are shown in Figs. 6-12 as follows.

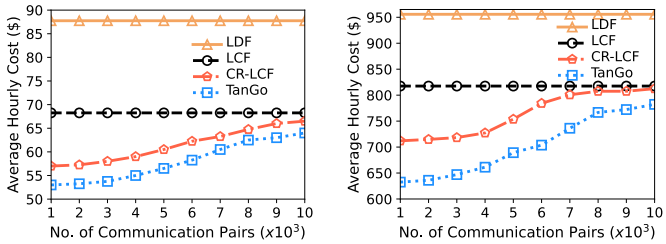
Hourly cost over time. To analyze the performance of these algorithms intuitively, we test the hourly cost over 12 hours on the two task datasets. The results are shown in Fig. 6. Specifically, Fig. 6(a) shows that the hourly cost of all methods varies with a time offset, where LDF has the highest cost while TanGo achieves the lowest cost all over the time. For example, at the eighth hour, the hourly cost of TanGo, LCF, LDF, and CR-LCF is $77.5, $87.7, $104.8, and $83.6, respectively. It means that TanGo can reduce the overall cost by over 11%, 26%, and 7% compared with LCF, LDF, and CR-LCF, respectively. This is due to the fact that TanGo provides a more advantageous placement decision from a global standpoint and can handle tenant demands and regional resource restrictions more equitably. We also observe that, despite the fact that both LCF and CR-LCF favor regions with lower electricity costs, the cost of CR-LCF will ultimately be less than that of LCF. It implies that, if the tenant requirements and resource constraints are satisfied, allowing tasks to be placed across regions will be more likely to take advantage of regional variances in electricity prices to minimize expenses. These findings are likewise true with Google cluster trace as in Fig. 6(b). For example, at the eighth hour, the hourly cost of TanGo, LCF, LDF, and CR-LCF is $608, $718, $862, and $697, respectively. More specifically, TanGo reduces the hourly cost by about 15%, 30%, and 13% compared with LCF, LDF, and CR-LCF. Moreover, we discover that TanGo can reduce more cost than other methods in the case of high workload (*i.e.*, with Google cluster trace). The result suggests that task placement in high workload scenarios has a greater impact on performance than that in low workload scenarios. Simple methods (*e.g.*, CR-LCF) would lead to inefficient usage of computing power or bandwidth among regions.

To further demonstrate the high applicability of TanGo under different scenarios, we compare TanGo, LCF, LDF, and CR-LCF by changing the settings of tenant requirements. The results are shown in Figs. 7-10, where we change separately



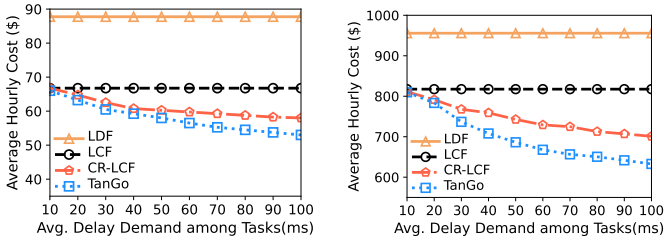
(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 7. Average hourly cost vs. Average access delay demand.



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 8. Average hourly cost vs. Number of communication pairs.



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 9. Average hourly cost vs. Avg. delay demand among tasks.

the access delay, the number of communication pairs, and the traffic delay and bandwidth demands, respectively.

Cost versus access delay demand. Fig. 7 analyzes the impact of the access delay demand on the average hourly cost. As the average access delay increases, the average hourly cost decreases for all methods except LDF. That is because LDF tends to select the closest region (with the lowest delay) for each tenant. As for other methods, relaxing the average access delay would increase the number of possible task placement choices and lead to lower overall costs. In comparison, TanGo achieves a lower cost than the other three methods. For example, given the average access delay demand as 60ms under low workload (*i.e.*, with Alibaba cluster trace in Fig. 7(a)), the average hourly cost of TanGo is $60.1 while that of LCF and CR-LCF are $69.7 and $64.8. More specifically, TanGo reduces the cost by about 13% and 7% compared with LCF and CR-LCF. This difference is more pronounced with the Google cluster trace in Fig. 7(b), where the cost of TanGo, LCF, and CR-LCF are $675, $813, and $759, respectively. That is to say, TanGo reduces the cost by about 17% and 11% compared with LCF and CR-LCF.

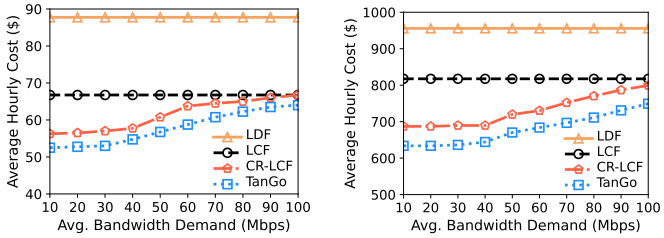
Cost versus the number of task communication pairs. We test the average hourly cost by changing the number of communication pairs among tasks. The results are shown in Fig. 8, where the horizontal axis is the number of communication pairs, ranging from 1k to 10k. With more tasks communication pairs, the average hourly cost of TanGo and CR-LCF increases while that of LCF and LDF remains the same, since these two algorithms do not consider the inter-task traffic and place all the tasks of a tenant in the same region. For example, as shown in Fig. 8(a), when the number of task communication pairs reaches 6k, the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $58.2, $66.7, $62.6, and $87.7, respectively. That is, TanGo can reduce the average hourly cost by about 13%, 7%, and 33% compared with LCF, CR-LCF, and LDF, respectively. As for the Google cluster trace, the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $703, $818, $791, and $956, respectively. That means TanGo reduces the cost by about 15%, 11%, and 26% compared with LCF, CR-LCF, and LDF.

Cost versus delay demand among tasks. We also test the average hourly cost by changing the average delay demand between any two tasks. Fig. 9 shows the average hourly cost with the average delay demand among tasks increasing from 10ms to 100ms. Note that when the average delay demand is 10ms, the average hourly cost of TanGo and CR-LCF is close to that of LCF, since the delay between regions is more than 10ms, and we can only place related tasks in the same region. The average hourly cost of both TanGo and CR-LCF decreases as the average access delay increases. For example, as shown in Figs. 9(a) and 9(b), when the average delay demand among tasks reaches 70ms with Alibaba cluster trace, the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $54.5, $66.7, $58.8, and $87.7, while the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $656, $818, $728, and $956 with Google cluster trace, respectively. More specifically, TanGo reduces the average hourly cost by about 25%, 10%, and 31.3% compared with LCF, CR-LCF, and LDF, respectively.

Cost versus bandwidth demand among tasks. The last tenant requirement we study is the average bandwidth demand between any two tasks. The results are shown in Fig. 10, where the horizontal axis is the average bandwidth demand between tasks, ranging from 10Mbps to 100Mbps. With the average bandwidth demand increasing, the average hourly cost of TanGo and CR-LCF increases while that of LCF and LDF remains the same. In particular, the average hourly cost rises significantly when the average bandwidth demand surpasses 40ms. This is because the bandwidth between regions is constrained, and bandwidth preemption may occur. For example, as shown in Fig. 8(b), when the average bandwidth demand reaches 70Mbps, the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $681, $817, $755, and $956, respectively. More specifically, TanGo reduces the average hourly cost by about 17%, 10%, and 29% compared with LCF, CR-LCF, and LDF, respectively.

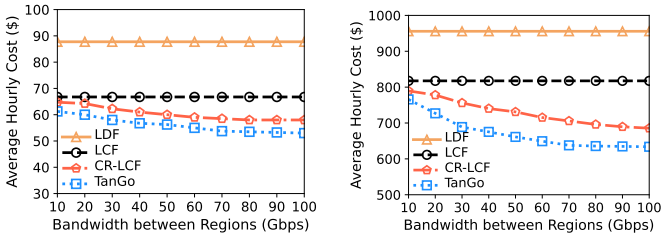
To gain a deeper understanding, we extend the experiments by changing the bandwidth and the variation in electricity prices among those regions, as shown in Figs. 11-12.

Cost versus bandwidth between regions. In this set of experiments, we change the bandwidth among regions, and the results are shown in Fig. 11. The average hourly cost



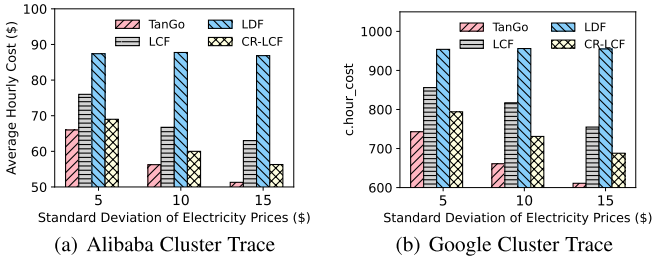
(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 10. Average hourly cost vs. Avg. Bandwidth demand among tasks.



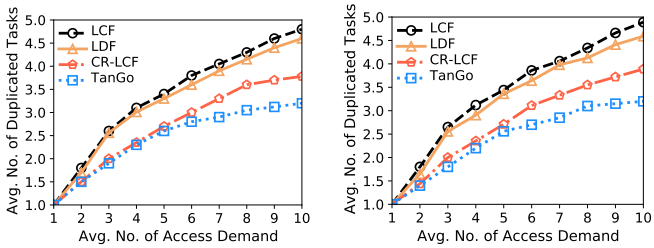
(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 11. Average Hourly cost vs. Bandwidth between regions.



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 12. Average Hourly cost vs. Standard deviation of prices.



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 13. Number of Duplicated tasks vs. Number of access demands.

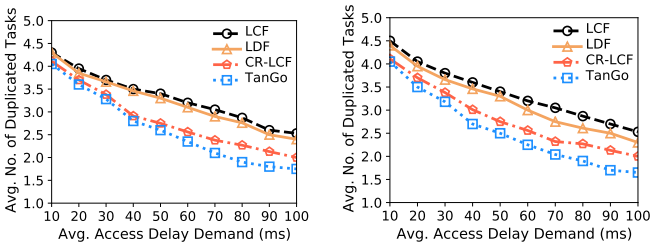
of TanGo and CR-LCF decreases as the bandwidth among regions increases from 10Gbps to 100Gbps. For example, when the bandwidth among regions achieves 70Gbps, the average hourly cost of TanGo, LCF, CR-LCF, LDF is $53.7, $66.8, $58.5, $87.8 with Alibaba cluster trace, and $637, $817, $706, $956 with Google cluster trace. That means, TanGo can reduce the average hourly by about 22%, 10%, and 33% compared with LCF, CR-LCF,and LDF, respectively.

Cost versus standard deviation of prices. The last set of experiments tests the average hourly cost by changing the standard deviation of electricity prices while maintaining a constant average. The results are shown in Fig. 12, where the standard deviation of electricity prices is adjusted to $5, $10, and $15, respectively. From Figs. 12(a) and 12(b) we learn that the average hourly cost gap of each method gradually widens as the standard deviation of the pricing rises. For example, when the standard deviation of electricity prices is $15, the average hourly cost with Alibaba cluster trace of TanGo, LCF, CR-LCF, and LDF is $51.3, $63, $57.3, and $86.8, respectively. More specifically, TanGo reduces the cost by about 18.6%, 10.5%, and 41% compared with LCF, CR-LCF, and LDF, respectively. The same is true for Google cluster trace, where the average hourly cost of TanGo, LCF, CR-LCF, and LDF is $611, $755, $688, and $955, respectively. That means TanGo reduces the cost by about 19.1%, 11.2%, and 36% compared with LCF, CR-LCF,and LDF, respectively.

From these simulation results, we can draw some conclusions. First, as shown in Fig. 6, TanGo can achieve superior performance in terms of electricity cost compared with the other three methods. Second, as shown in Figs. 7-10, our algorithm proves its effectiveness in different scenarios with various tenant demands and workloads. Third, TanGo performs well when conditions such as inter-region bandwidth and electricity prices fluctuate, as shown in Figs. 11-12. Fourth, compared with LCF and LDF, cross-region task placement methods like TanGo and CR-LCF can reduce electricity costs since they can better capitalize on regional differences in electricity prices. Meanwhile, TanGo is superior to CR- LCF, particularly in high workload scenarios.

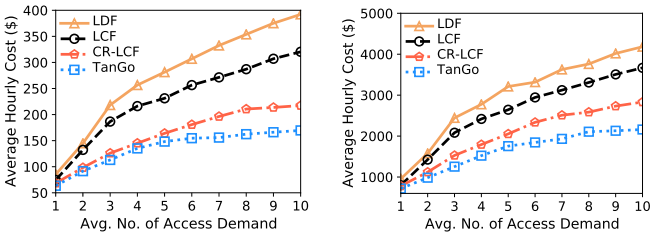
2) Evaluation on Duplicated Placement: In this section, to better demonstrate TanGo's performance in more general scenarios, we modify the tenant requirements as described in Section IV-B, *i.e.*, each tenant may have different access demands at different locations. Similarly, we modify the benchmark algorithms (LDF, LCF, and CR-LCF) to support duplicated placement. More specifically, these algorithms prioritize placing tasks in regions where they can satisfy the most access demands considering the resource consumption of duplicated placements. For example, the LCF algorithm selects the region that can cover the most access demands of tenants. If there are multiple options, LCF chooses the region with the lowest electricity price. The algorithm keeps greedily selecting until all access demands are satisfied.

Number of duplicated tasks versus number of access demands. In this set of experiments, we change the average number of access demands for each task from 1 to 10, and test the average duplicated number of each task. The results are shown in Fig. 13. When each tenant has only one access demand for each task, there is no need to duplicate the placement, *i.e.*, the average number of duplicated tasks is 1. However, we learn from Fig. 13 that the average number of duplicated tasks gradually increases with the increasing number of access demands for each task. For example, when the average number of access demands reaches 5, the average number of duplicated tasks of TanGo, LCF, LDF, and CR-LCF is 2.61, 3.42, 3.33, and 3.04 with Alibaba cluster trace, respectively. The reason why LCF and LDF require a significantly higher number of duplicated tasks is that these two methods place tasks with traffic demands in the same region. Therefore, when a task is duplicated, some other tasks



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 14. Number of Duplicated tasks vs. Access delay demand.



(a) Alibaba Cluster Trace (b) Google Cluster Trace

Fig. 15. Average hourly cost vs. Number of access demands.

must also be duplicated to meet the traffic demand. The same is true for Google cluster trace, where the average number of duplicated tasks of TanGo, LCF, LDF, and CR-LCF is 2.56, 3.44, 3.36, and 3.11, respectively. That means TanGo reduces the number of duplicated tasks by about 34.3%, 23.8%, and 17.7% compared with LCF, LDF, and CR-LCF, respectively.

Number of duplicated tasks versus access delay demand. We also test the number of duplicated tasks by changing the average delay demand. Fig. 14 shows the number of duplicated tasks with the average delay demand increasing from 10ms to 100ms. Meanwhile, the average number of access demands for each task is set to 5 for ease of comparison. The average number of duplicated tasks of all the methods decreases as the average access delay increases since these access delay demands can be more easily met by placing tasks in the same region. For example, as shown in Figs. 14(a) and 14(b), when the average delay demand reaches 70ms with Alibaba cluster trace, the average number of duplicated tasks of TanGo, LCF, LDF, and CR-LCF is 2.12, 3.11, 2.89, and 2.38, while that of TanGo, LCF, LDF, and CR-LCF is 2.04, 3.05, 2.75, and 2.32 with Google cluster trace, respectively. More specifically, TanGo reduces the number of duplicated tasks by about 33.1%, 25.8%, and 14.3% compared with TanGo, LCF, LDF, and CR-LCF, respectively.

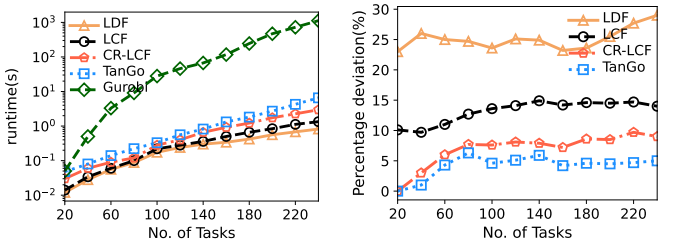
Cost versus number of access demands. Since duplicated tasks consume more electric power, we test the trend of average hourly cost by changing the number of access demands in this set of experiments. The results are shown in Fig. 15, where the number of access demands increases from 1 to 10. The difference in electricity costs between TanGo and the other three methods gradually widens with the increasing number of access demands due to the placement of fewer duplicated tasks. For example, given five access demands, the average hourly cost with Alibaba cluster trace of TanGo, LCF, LDF, and CR-LCF is $148.2, $231.2, $281.3, and $164.9, respectively. More specifically, TanGo reduces the cost by about 35.9%, 47.3%, and 10.2% compared with LCF, LDF, and CR-LCF, respectively. As for Google cluster trace, the average hourly cost of TanGo, LCF, LDF, and CR-LCF is $1754.9, $2643.6, $3211.3, and $2053.5, respectively. It means TanGo reduces the cost by about 33.6%, 45.3%, and 14.5% compared with LCF, LCF, and CR-LCF, respectively. There are two reasons why TanGo performs significantly better than other methods. One is that our proposed SM-CTP algorithm is better at finding the most cost-effective region for each task. The other is the ability of TanGo to generate as few duplicated tasks as possible.

We can infer several conclusions from the results of these three sets of experiments. First, Figs. 13-14 show that TanGo can place fewer duplicated tasks to meet all the tenant access demands compared with the other three methods. Besides, our algorithm demonstrates its efficacy in different scenarios with various tenant access demands. Second, as shown in Fig. 15, TanGo achieves superior performance in terms of electricity cost compared with the other three methods with duplicated placement. Third, compared with cross-region task placement methods like TanGo and CR-LCF, LCF and LDF place more duplicated tasks to meet the traffic among tasks.

3) Evaluation on Runtime and Optimality: In this section, we provide a detailed comparison of the efficacy and accuracy of Tango in comparison to the state-of-art LP solver (Gurobi [37] in our experiment), demonstrating the computational efficiency of Tango without sacrificing much optimality. We conducted benchmark testing on a system equipped with an Intel Core i5-10400F processor and 32GB of RAM.

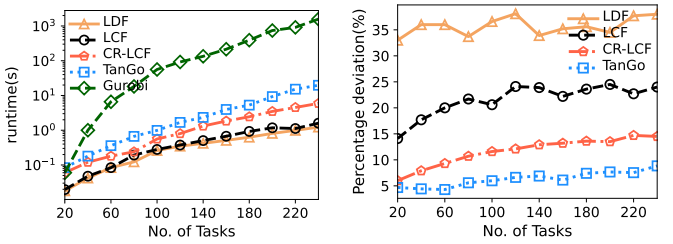
Performance comparison with Gurobi and its optimal solution. In this set of experiments, we change the number of tasks from 20 to 240, and test the runtime and percentage deviation from the optimal obtained by the IP solver (Gurobi). The overall cost percentage deviation from the optimal solution represents the performance metric. The results are depicted in Fig. 16. As shown in Fig. 16(a), the running time of LP solver significantly increases as the number of tasks increases. This is due to the fact that the number of variables to be solved increases proportionally with the number of tasks. For example, when the number of tasks is 200, a total of 2000 variables (with 10 regions) need to be solved. Furthermore, the nonlinearity of the problem brings more complexity to direct solving procedure. Although TanGo takes relatively longer runtime compared to other benchmarks, as shown in Fig. 16(b), TanGo provides a relatively good performance guarantee, with the difference between its solution and the optimal solution within 5% at most time, which demonstrates the practicality and effectiveness of our proposed algorithm. It should be noted that the runtime of these algorithms is dependent on the performance of the device. As the device's performance increases, the execution time of these algorithms is expected to decrease.

We also test the algorithms' runtime and performance compared to the optimal solution in the scenarios of duplicated placement. The results are shown in Fig. 17 with the number of tasks varying from 20 to 240. We observe that TanGo also achieves relatively good performance within an acceptable



(a) Runtime (b) Percentage deviation from optimal

Fig. 16. Performance comparison with gurobi and its optimal solution.



(a) Runtime (b) Percentage deviation from optimal

Fig. 17. Performance comparison with gurobi and its optimal solution (Duplicated Placement).

runtime with duplicated placement. Moreover, the performance gap between TanGo and other benchmarks is more significant, which is attributed to the fact that TanGo deploys fewer duplicate tasks.

V. Related Works

In this section, we summarize recent related works on geo-distributed cloud computing, task placement strategies, and prior efforts on economy efficiency in clouds.

Geo-distributed Cloud Computing. The study of cloud computing has gained popularity in recent years, and cloud infrastructure has increasingly shown a propensity toward geographic dispersal to offer tenants throughout the world connectivity at any time and from any location. Considering the characteristics of geographically distributed clouds, Valley [49] proposes a system that can place application data across the geo-distributed cloud while dealing with different business constraints. GA-Par gives [50] a dependable microservice orchestration framework to reduce the discrepancy between user security requirements and actual service provision. Some other works [51], [52] propose new geo-distributed cloud storage to support low-cost and latency storage services. Distributed services produce significant network traffic inside clouds. To address it, some cloud traffic management frameworks [15], [53], [54] are proposed, ranging from geographical load balancing to backbone traffic engineering. However, the resource constraints of the cloud region itself are often overlooked in these works.

Task Placement. As a cloud provider, how to place tasks properly is important, as this can increase the utilization, productivity, or profit of a cloud. To achieve the optimal task placement, authors in [55] offload dependent tasks to edge nodes with limited service caching in order to minimize the Job Completion Time (JCT), and NEAT+ [56] gives a task scheduling framework that leverages information from the underlying network scheduler and available compute resources to make task placement decisions. However, as geographically distributed clouds grow, cloud tenants' access demands for tasks become broader in scope. As a result of access delay demands and computing power constraints, cloud providers often cannot restrict the task distribution on their servers to a particular data center or region. Some previous works make efforts to reduce power consumption through multi-region task placement in a geo-distributed cloud while offering low access delay to end users [14] or reducing inter-DC traffic volume [16], [17]. However, these works often ignore the regional differences in resource prices, which may result in increased operating costs.

Economy Efficiency in Clouds. The objective of cloud providers is to lower overall operating costs, which is based on not only how much power they use but also the price of the power source, which varies greatly in different regions. Some studies consider adopting eco-friendly and economy efficient energies [9], [12], [57] to reduce the overall cost inside a data center or a region. Authors in [58] propose a cloud-based electricity consumption analysis approach using neural networks to help understand the energy consumption pattern. The authors in [46] consider optimizing overall costs associated with cloud resource, bandwidth, and deployment by proactively planning cloud resource allocation based on peak traffic awareness. Some other works, such as [18], [19], and [20], are motivated by the geographical diversity in electricity prices and concentrate on the issue of lowering the electricity cost of data centers by rerouting user requests to various data centers while taking into account regional electricity price diversity. For instance, the authors in [19] study the problem of task placement over geo-distributed data centers while ensuring user quality-of-service, and authors in [20] consider the bandwidth constraints between data centers. The majority of these efforts, however, assume that tasks are carried out individually and do not account for the traffic demands between tasks. As a result, they are inapplicable to the current large-scale distributed cloud system.

VI. Conclusion and Future Work

In this paper, we explore the challenges faced by region-wide distributed task placement and formulate the electricity cost minimization problem for task placement in a geo-distributed cloud. Then we solve this problem with an effective submodular-based algorithm. Results of in-depth analyses based on real-world electricity prices and task datasets show the efficacy of our algorithm compared with other solutions. In the future, we will endeavor to refine our cloud task placement optimization by exploring diverse cost considerations and incorporating realistic network modeling for inter-datacenter communication.

References

[1] L. Luo, G. Zhao, H. Xu, Z. Yu, and L. Xie, "TanGo: A cost optimization framework for tenant task placement in geo-distributed clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2023, pp. 1–10.

[2] Netflix Streaming Service. *Unlimited Films, TV Programmes and More*. Accessed: Jul. 20, 2022. [Online]. Available: https://www.netflix.com/

[3] Disney+ Streaming Service. *The Home of Disney, Pixar, Marvel, Star Wars, National Geographic, and Star*. Accessed: Jul. 20, 2022. [Online]. Available: https://www.disneyplus.com/

[4] General Data Protection Regulation (GDPR). *Harmonize Data Privacy Laws Across Europe*. Accessed: Jul. 20, 2022. [Online]. Available: https://gdpr-info.eu/

[5] Amazon Web Services. *Build, Deploy, and Manage Websites, Apps or Processes On AWS Secure, Reliable Network*. Accessed: Jul. 20, 2022. [Online]. Available: https://aws.amazon.com/

[6] Microsoft Azure. *Invent With Purpose, Realize Cost Savings, and Make Your Organization More Efficient With Microsoft Azure's Open and Flexible Cloud Computing Platform*. Accessed: Jul. 20, 2022. [Online]. Available: https://azure.microsoft.com/en-us/

[7] Google Cloud. *Build, Deploy, and Scale Applications, Websites, and Services on the Same Infrastructure as Google*. Accessed: Jul. 20, 2022. [Online]. Available: https://cloud.google.com/

[8] *Data Center White Paper From CAICT*. Accessed: Jul. 20, 2022. [Online]. Available: https://pdf.dfcfw.com/pdf/H3_AP202204241561314215_1.pdf?1650898389000.pdf

[9] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2020, pp. 769–778.

[10] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "TrafficShaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1193–1206, Jun. 2018.

[11] T. Zhu, M. A. Kozuch, and M. Harchol-Balter, "WorkloadCompactor: Reducing datacenter cost while providing tail latency SLO guarantees," in *Proc. Symp. Cloud Comput.*, Sep. 2017, pp. 598–610.

[12] W. Deng, F. Liu, H. Jin, C. Wu, and X. Liu, "MultiGreen: Cost-minimizing multi-source datacenter power supply with online control," in *Proc. 4th Int. Conf. Future Energy Syst.*, May 2013, pp. 149–160.

[13] R. Eyckerman, S. Mercelis, J. Marquez-Barja, and P. Hellinckx, "Requirements for distributed task placement in the fog," *Internet Things*, vol. 12, Dec. 2020, Art. no. 100237.

[14] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay cloud video conferencing," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, Jun. 2015, pp. 103–112.

[15] P. Li et al., "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1785–1796, Jun. 2017.

[16] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *Proc. NSDI*, 2021, pp. 201–216.

[17] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, "Cost-minimizing bandwidth guarantee for inter-datacenter traffic," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 483–494, Apr. 2019.

[18] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed Internet Data Centers in a multi-electricity-market environment," in *Proc. IEEE (INFOCOM)*, Mar. 2010, pp. 1–9.

[19] L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic, "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2049–2059, Jul. 2015.

[20] H. Xu and B. Li, "Cost efficient datacenter selection for cloud services," in *Proc. 1st IEEE Int. Conf. Commun. China (ICCC)*, Aug. 2012, pp. 51–56.

[21] *Federal Energy Regulatory Commission. U.S. Electric Power Markets*. Accessed: Jul. 20, 2022. [Online]. Available: http://www.ferc.gov/market-oversight/mkt-electric/overview.asp

[22] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proc. 9th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2012, pp. 253–266.

[23] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. 19th ACM Int. Conf. Multimedia*, Nov. 2011, pp. 1269–1272.

[24] Y. Feng, B. Li, and B. Li, "Airlift: Video conferencing as a cloud service using inter-datacenter networks," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2012, pp. 1–11.

[25] D. Dahiphale et al., "An advanced MapReduce: Cloud MapReduce, enhancements and applications," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 1, pp. 101–115, Mar. 2014.

[26] D. Pop, "Machine learning and cloud computing: Survey of distributed and SaaS solutions," 2016, *arXiv:1603.08767*.

[27] Alibaba Cluster Data. *Cluster Data Collected From Production Clusters in Alibaba for Cluster Management Research*. Accessed: Jul. 20, 2022. [Online]. Available: https://github.com/alibaba/clusterdata/

[28] Google Cluster Data. *Borg Cluster Traces From Google*. Accessed: Jul. 20, 2022. [Online]. Available: https://github.com/google/cluster-data/

[29] I. Pelle, J. Czentye, J. Dóka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on AWS," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 272–280.

[30] S. Lenhart and D. Fox, "Participatory democracy in dynamic contexts: A review of regional transmission organization governance in the United States," *Energy Res. Social Sci.*, vol. 83, Jan. 2022, Art. no. 102345.

[31] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 73–86.

[32] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with NetStitcher," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 74–85.

[33] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 202–208.

[34] C. Guo et al., "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 139–152.

[35] A. Soltanian, D. Naboulsi, R. Glitho, and H. Elbiaze, "Resource allocation mechanism for media handling services in cloud multi-media conferencing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1167–1181, May 2019.

[36] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2005.

[37] B. Bixby, "The Gurobi optimizer," *Transp. Res. B*, vol. 41, no. 2, pp. 159–178, 2007.

[38] A. Agarwal, Z. Liu, and S. Seshan, "HeteroSketch: Coordinating network-wide monitoring in heterogeneous and dynamic networks," in *Proc. 19th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2022, pp. 719–741.

[39] A. Krause and D. Golovin, "Submodular function maximization," *Tractability*, vol. 3, no. 19, pp. 71–104, 2012.

[40] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, Jun. 1972.

[41] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Math. Oper. Res.*, vol. 3, no. 3, pp. 177–188, Aug. 1978.

[42] L. Luo, G. Zhao, H. Xu, L. Xie, and Y. Xiong, "VITA: Virtual network topology-aware southbound message delivery in clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2022, pp. 630–639.

[43] V. K. Adhikari et al., "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1620–1628.

[44] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 9, pp. 2674–2688, Sep. 2017.

[45] H. Xu, J. Fan, J. Wu, C. Qiao, and L. Huang, "Joint deployment and routing in hybrid SDNs," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–10.

[46] V. Eramo and F. G. Lavacca, "Optimizing the cloud resources, bandwidth and deployment costs in multi-providers network function virtualization environment," *IEEE Access*, vol. 7, pp. 46898–46916, 2019.

[47] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 554–568, Sep. 2017.

[48] W. Lin, H. Wang, Y. Zhang, D. Qi, J. Z. Wang, and V. Chang, "A cloud server energy consumption measurement system for heterogeneous cloud environments," *Inf. Sci.*, vol. 468, pp. 47–62, Nov. 2018.

[49] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," in *Proc. NSDI*, 2010, pp. 1–16.

[50] Z. Wen et al., "GA-Par: Dependable microservice orchestration framework for geo-distributed clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 129–143, Jan. 2020.

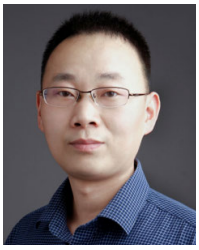
[51] H. Wang, H. Shen, Z. Li, and S. Tian, "GeoCol: A geo-distributed cloud storage system with low cost and latency using reinforcement learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 149–159.

[52] K. Oh, N. Qin, A. Chandra, and J. Weissman, "Wiera: Policy-driven multi-tiered geo-distributed cloud storage system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 2, pp. 294–305, Feb. 2020.

[53] A. Yassine, A. A. N. Shirehjini, and S. Shirmohammadi, "Bandwidth on-demand for multimedia big data transfer across geo-distributed cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1189–1198, Oct. 2020.
[54] C. Feng, H. Xu, and B. Li, "An alternating direction method approach to cloud traffic management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2145–2158, Aug. 2017.
[55] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2777–2792, Nov. 2021.
[56] A. Munir, T. He, R. Raghavendra, F. Le, and A. X. Liu, "Network scheduling and compute resource aware task placement in datacenters," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2435–2448, Dec. 2020.
[57] B. Liang, X. Dong, Y. Wang, and X. Zhang, "Memory-aware resource management algorithm for low-energy cloud data centers," *Future Gener. Comput. Syst.*, vol. 113, pp. 329–342, Dec. 2020.
[58] N. Kumar, V. H. Gaidhane, and R. K. Mittal, "Cloud-based electricity consumption analysis using neural network," *Int. J. Comput. Appl. Technol.*, vol. 62, no. 1, pp. 45–56, 2020.

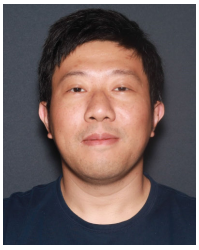
Luyao Luo (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree in computer science with the University of Science and Technology of China. His main research interests are software defined networks and cloud computing.

Gongming Zhao (Member, IEEE) received the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2020. He is currently an Associate Professor with the University of Science and Technology of China. His current research interests include software-defined networks and cloud computing.



Hongli Xu (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China (USTC), China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, USTC. He has published more than 100 articles in famous journals and conferences, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, International Conference on Computer Communications (INFOCOM), and International Conference on Network Protocols (ICNP). He has held more than 30 patents. His research interests include software defined networks, edge computing, and the Internet of Things. He was awarded the Outstanding Youth Science Foundation of NSFC in 2018. He has won the best paper award or the best paper candidate in several famous conferences.

Zhuolong Yu received the bachelor's and master's degrees from the University of Science and Technology of China and the Ph.D. degree from the Department of Computer Science, Johns Hopkins University. His research interests include programmable networking and high-performance networking.



Liguang Xie (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2013. He is currently a Senior Principal Architect and the Senior Director of Engineering with the Futurewei Seattle Research Center, where he leads the Cloud Networking Research and Development Team and oversees cloud networking open-source, research, and engineering efforts. Prior to Futurewei, he was a Senior Software Engineering Lead with Microsoft Azure Networking and an Adjunct Assistant Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. His research interests are cloud networking, software-defined networking, distributed systems, edge computing, and cloud computing.