TanGo: A Cost Optimization Framework for Tenant Task Placement in Geo-distributed Clouds

Luyao Luo[1,2] Gongming Zhao[1,2] Hongli Xu[1,2] Zhuolong Yu[3] Liguang Xie[4]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Suzhou Institute for Advanced Research, University of Science and Technology of China
[3] Johns Hopkins University, USA
[4] Futurewei Technologies, USA

***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 [1], Disney+ [2]) 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 [3]). This geo-distributed deployment model requires cloud providers (*e.g.*, AWS [4], Microsoft Azure [5], and Google Cloud [6]) 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.

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

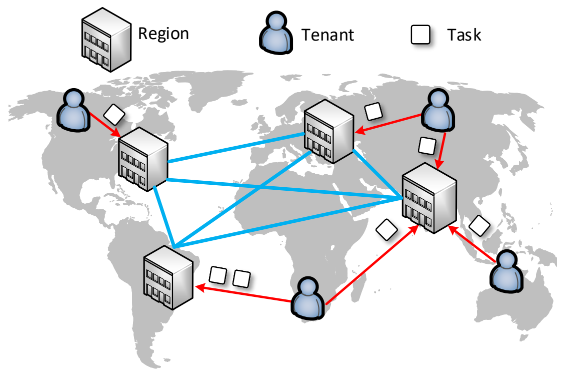


Fig. 1: An example of global tenants placing tasks based on a geographically distributed cloud.

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 [7]. 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, 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 [20].
- *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-

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.*, [8]–[11])	✗	✗	✗	✗
Inter-region Power-aware (*e.g.*, [12]–[16])	✗	✓	*partial*	✓
Inter-region Price-aware (*e.g.*, [17]–[19])	✓	✗	*partial*	✗
TanGo	✓	✓	✓	✓

insensitive), and the actual latency varies substantially based on the geographic distance between clients and cloud regions [21]–[23].

- *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 [24] and distributed machine learning [25] 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 [15].

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 [12]–[19], 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 [17]–[19]. 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 $(1 - 1/e)$.
3) We conduct extensive experiments based on real-world regional topology, electricity pricing map, and tenant datasets including Alibaba Cluster Trace [26] and Google Cluster Trace [27]. The results show that TanGo can reduce the electricity cost by up to 10%-30% compared with existing solutions.

II. Background and Motivation

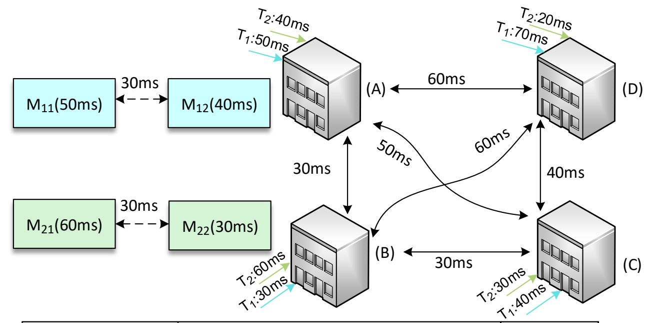
A. Current Cost-effective Solutions and Limitations

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 [9]) or workload (*e.g.*, Workloadcompactor [10]). Some other studies consider adopting eco-friendly energies [8, 11] 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 [12]–[16] make efforts to reduce power consumption through multi-region task placement in a geo-distributed cloud while offering low access delay to end users [13] or reducing inter-DC traffic volume [15, 16]. However, the objective of cloud providers like Azure [5], AWS [4], 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 [17]–[19] 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 [18] 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



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.

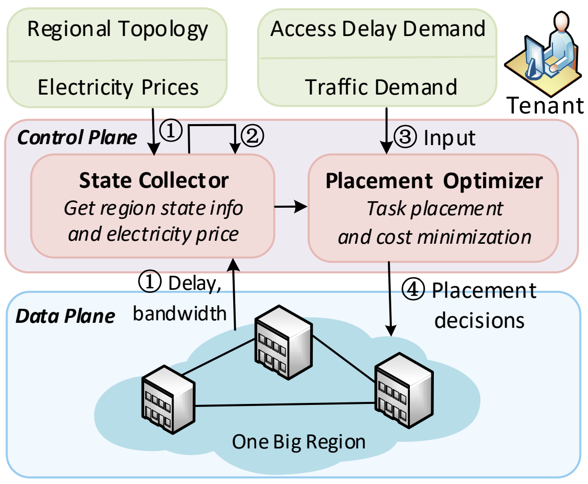


Fig. 3: Overview and workflow of TanGo. TanGo is mainly composed of two parts: the control plane and the data plane. Specifically, the control plane consists of two components: the state collector and the placement optimizer. The data plane consists of data centers located in different regions.

not appropriate for the current large-scale distributed cloud system.

In practice, there are mainly two strategies of task placement in current public clouds [28]. 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 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 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 ($50×2 + $60×2).
- *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 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"

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

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 [29]. 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

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 $\mathcal{R} = \{k_1, \dots, k_K\}$ to represent the set of regions, where $K = |R|$ is the number of regions. The set of tenants in the cloud is denoted as $\mathcal{T}$=$\{t_1, \dots, t_{|\mathcal{T}|}\}$. As tenants deploy different kinds of tasks in clouds, we use $\mathcal{I}$ to denote the set of tasks and $\mathcal{I}^t = \{I_1^t, \dots, I_{|\mathcal{I}^t|}^t\}$ 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 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 [30, 31]. 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 [32]). $b_{i,i'}^t$ 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 $x_{i,k}^t \tau_k^t \le \tau_i^t, \forall I_i^t \in \mathcal{I}^t, t \in \mathcal{T}, k \in \mathcal{R}$.
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 $x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \le \tau_{i,i'}^t, \forall I_i^t, I_{i'}^t \in \mathcal{I}^t, t \in \mathcal{T}, k, k' \in \mathcal{R}$.
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.*, $\sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \le b_{k,k'}, \forall k, k' \in \mathcal{R}$.
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, $\sum_t \sum_i x_{i,k}^t r_i^t \le r_k, \forall k \in \mathcal{R}$.

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:

$$\min \sum_k E_k$$

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

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, $\sum_k E_k$.

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 [33]), it still needs the order of hours to solve even for relatively small scales (*e.g.*, 1000 tasks) [34]. 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 $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_K\}$, 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.

$$\max C_{max} - \sum_{k\in K} C(\mathcal{I}_k)$$

$$S.t \begin{cases} \sum_k x_{i,k}^t = 1, & \forall i, t \\ x_{i,k}^t \tau_k^t \le \tau_i^t, & \forall i, t, k \\ x_{i,k}^t x_{i',k'}^t \tau_{k,k'} \le \tau_{i,i'}^t, & \forall (i,i'), t, (k,k') \\ \sum_t \sum_{i,i'} x_{i,k}^t x_{i',k'}^t b_{i,i'}^t \le b_{k,k'} & \forall (k,k') \\ \sum_t \sum_i x_{i,k}^t r_i^t \le r_k, & \forall k \\ x_{i,k}^t \in \{0,1\} & \forall i, t, k \end{cases} \qquad (2)$$

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 we determine the placement location for a task, we update the available set of each task based on the traffic demand among tasks.

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:*

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

Where k_n is the region with the lowest electricity price that can accommodate all tasks in Φ_n. It is determined in Alg. 1.

Next, we give the definition of submodularity and prove that the function H is submodular in Section III-B3.

Definition 3 *(Submodularity [35]): 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,

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

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 [36] 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. 1. 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 1 *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

Algorithm 1 SM-CTP: Submodular-based Algorithm for CTP

```
1:  Step 1: Initialization
2:  Compute the available region set for each task, and the set of feasible task sets $B_k$ for each region $k$
3:  $\Phi \leftarrow \emptyset$
4:  Step 2: Iterative Selection
5:  while $|\Phi| \leq K-1$ do
6:      Set $tmp \leftarrow 0, opt \leftarrow 0$
7:      for $k \in \mathcal{R}$ do
8:          for $S \in B_k - \Phi$ do
9:              $tmp \leftarrow H(\Phi \cup \{S\})$
10:             if $tmp > opt$ then
11:                 $opt \leftarrow tmp$, $S^* \leftarrow S$
12:     $\Phi \leftarrow \Phi + \{S^*\}$
13:     Update available region sets and feasible task sets
14:     Update the feasible task sets based on the updated available region sets
15: $\Phi \leftarrow \Phi + \{\mathcal{I} - \bigcup_{S\in\Phi} S\}$
16: Step 3: Assignment of tasks and regions
17: for $S \in \Phi$ do
18:     Set $x_{i,t}^k = 1$ if $I_i^t \in S$ if $S$ is taken out from $B_k$
```

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

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

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

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

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

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

From Eq. (7), we can get:

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

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

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

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

Lemma 2 *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 [35].*

Theorem 3 *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 1. 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:

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

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 2, we prove that our proposed algorithm can reach a $(1-1/e)$ approximation factor for the CTP problem in Eq. (2). ■

In fact, Nemhauser [37] 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 4 *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 [38] 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 number of tasks $|\mathcal{I}|$) of feasible sets for each region. Under this condition, the function H is calculated $O(K|\mathcal{I}|)$ times in each iteration and the algorithm runs in $K-1$ iterations. As a result, the time complexity of SM-CTP is $O(K^2|\mathcal{I}|)$.

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, called Lowest-Delay-First (LDF), places all tasks of a tenant in the same region. LDF simulates the preference behavior of tenants in the cloud and selects the region with the lowest access delay to place tasks [39].
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 [27]. Specifically, it chooses the region with the lowest electricity price that meets tenants' access delay demand.

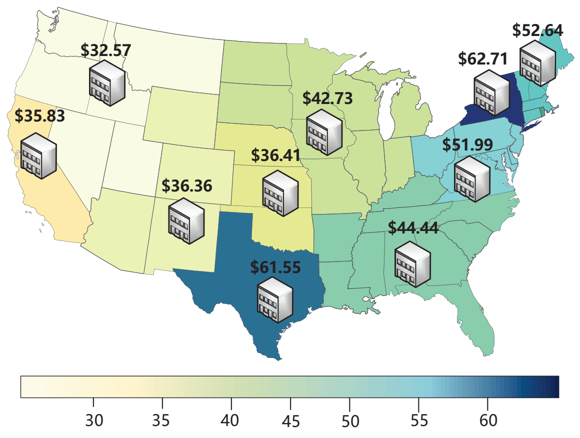


Fig. 4: The U.S. cloud region selection and electricity price ($/MWh) [20].

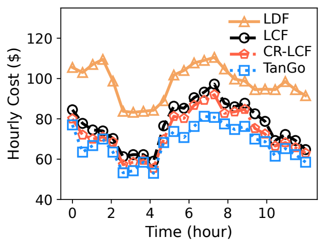
3) The last one is Cross-Region Lowest-Cost-First (CR-LCF). Since no existing work fully takes into account both regional diversity in electricity prices and tenant requirements in region-wide task placement, to better illustrate the benefits of our solution, we employ a greedy cross-region strategy based on LCF for task placement. Specifically, the CR-LCF strategy 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. Different from both LDF and LCF, CR-LCF may place tasks of a tenant in multiple regions.

B. Simulation Settings

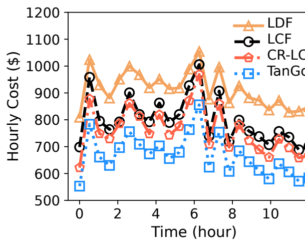
Regional Topology. We assume that the regions are spread across the continental U.S. in order to account for the geographic diversity of the cloud infrastructure and electricity prices as in [19]. For the sake of convenience, we suppose that each region, as depicted in Fig. 4, has a single data center in a randomly selected location. We first choose 10 cities as the precise locations of data centers in the United States. The tenants access these regions (*i.e.*, data centers) randomly from another 100 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 [20]. 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 [16].

Task Sets. We conduct simulations on two different real-world task datasets: (a) Alibaba Cluster Trace [26] and (b) Google Cluster Trace [27]. 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 [40].

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.



(a) Alibaba Cluster Trace

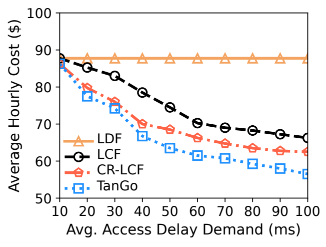


(b) Google Cluster Trace

Fig. 5: Hourly Cost vs. Time.

C. Evaluation Results

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. 5. Specifically, Fig. 5(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. 5(b). For example, at the eighth hour, the hourly cost of TanGo, LCF, LDF, and CR-LCF is $608, $718, $862, and $697, respectively.


(a) Alibaba Cluster Trace

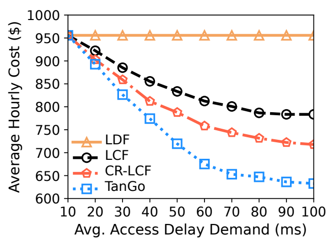
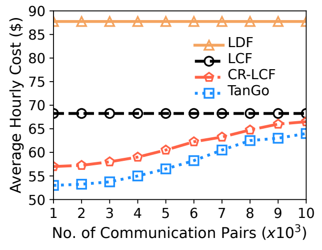

(b) Google Cluster Trace

Fig. 6: Average Hourly Cost vs. Average Access Delay Demand.


(a) Alibaba Cluster Trace

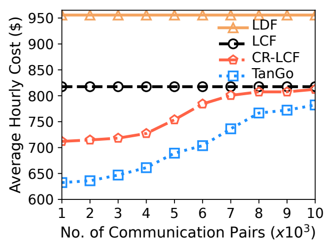
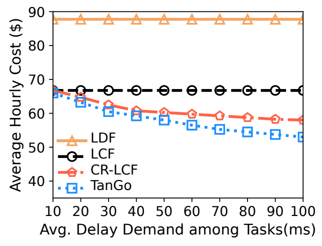

(b) Google Cluster Trace

Fig. 7: Average Hourly Cost vs. Number of Communication Pairs.


(a) Alibaba Cluster Trace

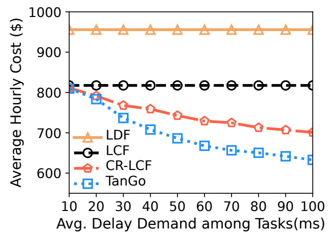

(b) Google Cluster Trace

Fig. 8: Average Hourly Cost vs. Avg. Delay Demand among Tasks.

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.

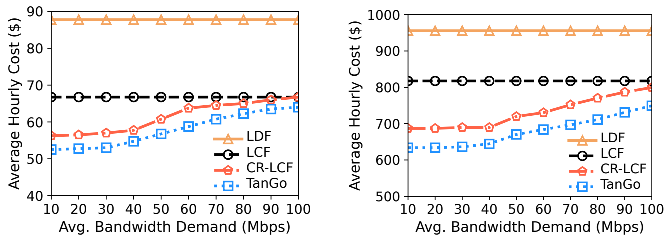
To further demonstrate the high applicability of TanGo under different scenarios, we compare TanGo, LCF, LDF, and CR-LCF by changing the settings of tenant requirements. The results are shown in Figs. 6-9, where we change separately the access delay, the number of communication pairs, and the traffic delay and bandwidth demands, respectively.

Cost versus access delay demand. Fig. 6 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. 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. 6(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. 6(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. 7, 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. 7(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.

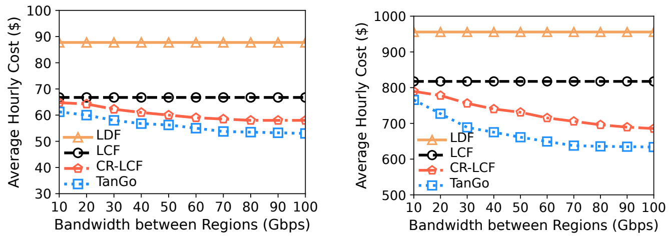
Cost versus delay demand among tasks. We also test the average hourly cost by changing the average delay demand between any two tasks. Fig. 8 shows the average hourly cost with the average delay demand among tasks increasing from 10ms to 100ms. Note that when the average delay demand is 10ms, the average 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 Fig. 8(a), 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, 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. 9, where the horizontal axis is the average bandwidth demand



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

Fig. 9: Average Hourly Cost vs. Avg. Bandwidth Demand among Tasks.



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

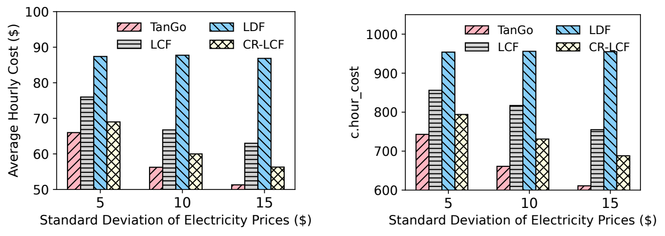
Fig. 10: Average Hourly Cost vs. Bandwidth between Regions.

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. 7(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. 10-11.

Cost versus bandwidth between regions. In this set of experiments, we change the bandwidth among regions, and the results are shown in Fig. 10. The average hourly cost 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. 11, where



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

Fig. 11: Average Hourly Cost vs. Standard Deviation of Prices.

the standard deviation of electricity prices is adjusted to $5, $10, and $15, respectively. From Figs. 11(a) and 11(b) we can 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. 5, TanGo can achieve superior performance in terms of electricity cost compared with the other three methods. Second, as shown in Figs. 6-9, 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 Fig. 10-11. 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.

V. Conclusion

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.

Acknowledgement

The corresponding authors of this paper are Gongming Zhao and Hongli Xu. This article was supported in part by the National Science Foundation of China (NSFC) under Grant 62102392, the Hefei Municipal Natural Science Foundation under Grant No. 2022013 and the National Science Foundation of Jiangsu Province under Grant BK20210121.

References

[1] Netflix streaming service. [Online]. Available: https://www.netflix.com/

[2] Disney+ streaming service. [Online]. Available: https://www.disneyplus.com/

[3] General data protection regulation (gdpr). [Online]. Available: https://gdpr-info.eu/

[4] Amazon web services. [Online]. Available: https://aws.amazon.com/

[5] Microsoft azure. [Online]. Available: https://azure.microsoft.com/en-us/

[6] Google cloud. [Online]. Available: https://cloud.google.com/

[7] Data center white paper from caict. Accessed: July. 20, 2022. [Online]. Available: https://pdf.dfcfw.com/pdf/H3_AP202204241561314215_1.pdf?1650898389000.pdf

[8] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," in *2020 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2020, pp. 769–778.

[9] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "Trafficshaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1193–1206, 2018.

[10] T. Zhu, M. A. Kozuch, and M. Harchol-Balter, "Workloadcompactor: Reducing datacenter cost while providing tail latency slo guarantees," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 598–610.

[11] W. Deng, F. Liu, H. Jin, C. Wu, and X. Liu, "Multigreen: Cost-minimizing multi-source datacenter power supply with online control," in *Proceedings of the fourth international conference on Future energy systems*, 2013, pp. 149–160.

[12] R. Eyckerman, S. Mercelis, J. Marquez-Barja, and P. Hellinckx, "Requirements for distributed task placement in the fog," *Internet of Things*, vol. 12, p. 100237, 2020.

[13] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay cloud video conferencing," in *2015 IEEE 35th International Conference on Distributed Computing Systems*. IEEE, 2015, pp. 103–112.

[14] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang, "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1785–1796, 2016.

[15] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 201–216.

[16] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, "Cost-minimizing bandwidth guarantee for inter-datacenter traffic," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 483–494, 2016.

[17] 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 *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[18] 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 Transactions on Computers*, vol. 64, no. 7, pp. 2049–2059, 2014.

[19] H. Xu and B. Li, "Cost efficient datacenter selection for cloud services," in *2012 1st IEEE International Conference on Communications in China (ICCC)*. IEEE, 2012, pp. 51–56.

[20] Federal energy regulatory commission. U.S. electric power markets. [Online]. Available: http://www.ferc.gov/market-oversight/mkt-electric/overview.asp

[21] 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 *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 253–266.

[22] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 1269–1272.

[23] Y. Feng, B. Li, and B. Li, "Airlift: Video conferencing as a cloud service using inter-datacenter networks," in *2012 20th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2012, pp. 1–11.

[24] D. Dahiphale, R. Karve, A. V. Vasilakos, H. Liu, Z. Yu, A. Chhajer, J. Wang, and C. Wang, "An advanced mapreduce: cloud mapreduce, enhancements and applications," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 101–115, 2014.

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

[26] Alibaba cluster data. [Online]. Available: https://github.com/alibaba/clusterdata/

[27] Google cluster data. [Online]. Available: https://github.com/google/cluster-data/

[28] I. Pelle, J. Czentye, J. Dóka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on aws," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 2019, pp. 272–280.

[29] S. Lenhart and D. Fox, "Participatory democracy in dynamic contexts: A review of regional transmission organization governance in the united states," *Energy Research & Social Science*, vol. 83, p. 102345, 2022.

[30] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 2009, pp. 202–208.

[31] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 139–152.

[32] A. Soltanian, D. Naboulsi, R. Glitho, and H. Elbiaze, "Resource allocation mechanism for media handling services in cloud multimedia conferencing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1167–1181, 2019.

[33] B. Bixby, "The gurobi optimizer," *Transp. Re-search Part B*, vol. 41, no. 2, pp. 159–178, 2007.

[34] A. Agarwal, Z. Liu, and S. Seshan, "{HeteroSketch}: Coordinating network-wide monitoring in heterogeneous and dynamic networks," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 719–741.

[35] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, pp. 71–104, 2014.

[36] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[37] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of operations research*, vol. 3, no. 3, pp. 177–188, 1978.

[38] L. Luo, G. Zhao, H. Xu, L. Xie, and Y. Xiong, "Vita: Virtual network topology-aware southbound message delivery in clouds," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 630–639.

[39] 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 Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, 2017.

[40] 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," *Information Sciences*, vol. 468, pp. 47–62, 2018.