Hindawi
Security and Communication Networks
Volume 2020, Article ID 8822990, 13 pages
https://doi.org/10.1155/2020/8822990

Research Article

Deploying GIS Services into the Edge: A Study from Performance Evaluation and Optimization Viewpoint

Jianbing Zhang, Bowen Ma, and Jiwei Huang

Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China

Correspondence should be addressed to Jiwei Huang; huangjw@cup.edu.cn

Received 13 August 2020; Revised 27 September 2020; Accepted 16 October 2020; Published 31 October 2020

Academic Editor: Xiaolong Xu

Geographic information system (GIS) is an integrated collection of computer software and data used to view and manage information about geographic places, analyze spatial relationships, and model spatial processes. With the growing popularity and wide application of GIS in reality, performance has become a critical requirement, especially for mobile GIS services. To attack this challenge, this paper tries to optimize the performance of GIS services by deploying them into edge computing architecture which is an emerging computational model that enables efficient offloading of service requests to edge servers for reducing the communication latency between end-users and GIS servers deployed in the cloud. Stochastic models for describing the dynamics of GIS services with edge computing architecture are presented, and their corresponding quantitative analyses of performance attributes are provided. Furthermore, an optimization problem is formulated for service deployment in such architecture, and a heuristic approach to obtain the near-optimal performance is designed. Simulation experiments based on real-life GIS performance data are conducted to validate the effectiveness of the approach presented in this paper.

1. Introduction

Geographic information system (GIS) has been a hot technique for providing the tools for capturing, storing, analyzing, and displaying spatial data [1]. In order to provision GIS services with high Quality of Service (QoS), performance of the system is a critical issue [2]. In recent years, there have been several research works dedicating to optimizing the performance of GIS services from different aspects [2–4].

Edge computing is an emerging technique of optimizing computing systems by performing data processing at the edge of the network near the source of the original data [5]. It pushes applications, data, and services away from centralized points (i.e., the cloud) to the logical extremes of a network, and thus, the communication latency for processing user requests can be significantly reduced [6, 7], as well as fault-tolerance [8], privacy [9–12], and security [13] being enhanced. With edge computing architecture, the performance as well as scalability of GIS systems can be dramatically enhanced [14].

Although there have been some research studies focusing on improving the QoS of GIS services by applying edge computing techniques, few of them paid attention to the performance evaluation issue. There lacks of analytical approaches for evaluating as well as optimizing the performance of GIS systems which is able to quantitatively indicating the impact after deploying GIS services into the systems with edge computing paradigm. It is quite a challenging work to capture the dynamics of the GIS systems, especially after constructing them with edge computing architecture, since the introduction of the edge layer makes it quite complicated for task scheduling and request processing. Furthermore, whether to dispatch the request to the near-end edge servers or far-end cloud servers for obtaining the optimal QoS remains largely unexplored.

In this paper, we make an attempt at filling this gap by presenting a performance evaluation and optimization study of the GIS services deployed in the edge computing architecture. A theoretical model for capturing the dynamics of the edge computing systems running GIS services is presented, and its corresponding quantitative analysis is conducted. With the analytical results, an optimization problem is formulated and a service deployment scheme is designed for obtaining the near-optimal performance of GIS services. With performance data generated from real-world GIS systems, simulation experiments are conducted to validate the effectiveness of the approach.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work most pertinent to this paper. In Section 3, we present a theoretical model for formulating the GIS systems with edge computing architecture, and provide quantitative analysis of the model. In Section 4, we formulate an optimization problem and design a performance optimization approach. In Section 5, we conduct real-life data based experiments to validate the efficacy of our scheme. Finally, we conclude the paper in Section 6.

2. Related Work

2.1. Performance Evaluation. A straightforward approach of performance evaluation is to obtain the performance metrics by direct measurement. Due to the dynamics of the system and environments, a series of experimental measurements are commonly required and statistical techniques are applied for handling the original measurement data. Truong and Karan [15] designed a mobile application of performance measurement and studied the impact of performance and data quality for mobile edge cloud systems. Morabito et al. [16] constructed a real testbed to evaluate the container-based solutions in IoT environment at the network edge, and analyzed the power and resource consumption for performance evaluation. Chen and Kunz [17] combined measurement and emulation and designed a network emulator for performance evaluation of optimal protocols. Qi et al. [18] collected data from 18,478 real-world APIs and 6,146 real-world apps, and designed a data-driven approach for web service recommendation. Baptista et al. [19] deployed a web-based GIS and used two datasets as the benchmark to evaluate the performance of several optimization techniques in Web GIS.

Although the measurement-based approaches are effective in performance evaluation, their overhead is so expensive that sometimes especially in the design phase of a computing system, one may not be able to afford implementing all the feasible schemes for comparison in reality [20]. Therefore, an alternative type of approaches has emerged, which applied theoretical models to formulate a system and then provide quantitative analysis by solving the models. With significantly lower overhead, the model-based approaches are able to evaluate the performance of the schemes before their implementations, making them increasingly popular in system design and improvement. Wang et al. [21] applied queueing theory to formulate an edge computing system, based on which a near-optimal offloading scheme for the Internet of Vehicles was designed. Ni et al. [22] generalized Petri net models and conducted performance evaluation of resource allocation strategies in edge computing environments. Li et al. [23] presented a performance estimation approach using M/M/k queueing model in Internet of Things (IoT) environments, which further helped to explore the optimal QoS-aware service composition scheme.

2.2. Performance Optimization. The performance optimization is commonly based on the evaluation results and thus used to optimize the performance of a system by designing new policies, selecting the best candidate, or enhancing the existing ones. One popular way is to collect the performance data of the policies by either measurement-based approaches or model-based approaches and search for the optimal one. Sometimes due to the extremely large search space, such search-based optimization approaches may meet with search-space explosion problems, and thus how to search for the optimal solution with high efficiency has become a hot topic. Mebrek et al. [24] considered the QoS and energy consumption in edge computing for IoT, formulated a constrained optimization problem, and designed an evolutionary algorithm-based approach for searching the feasible solutions. Wu et al. [25] designed a service composition scheme for mobile edge computing systems by combining simulated annealing and genetic algorithm. Zhang et al. [26] used neural network models for search-based optimization and designed a proactive video push scheme for reducing bandwidth consumption in hybrid CDN-P2P VoD Systems. Xu et al. [27] designed a multiobjective evolutionary algorithm based on decomposition for adaptive computation offloading for edge computing in 5G-envisioned Internet of Connected Vehicles (IoCV).

Another feasible way is to build a mathematical model illustrating the relationships between the system parameters and the performance metrics, based on which optimization problems can be formulated and optimal policies can be obtained. Zhang et al. [28] presented a graph-based model for service composition and designed an optimization approach of service composition with QoS correlations. Mao et al. [29] formulated the resource management as a Markov decision process, and further applied deep reinforcement learning to construct an optimization algorithm. Chen et al. [30] applied queueing theory to capture the dynamics in the mobile edge computing environment, formulated a stochastic optimization problem and designed an energy-efficient task offloading and frequency scaling scheme for mobile devices.

2.3. Summary. Although there have been several cutting-edge research works dedicating to performance evaluation and optimization for edge computing systems, this topic remains largely unexplored in geographic information systems. Since it has been shown by the existing literature that edge computing is able to improve the performance of the computing systems, especially for real-time services, we believe that a comprehensive study on the performance evaluation and optimization of GIS services deployed in edge computing architecture will have theoretical reference and practical value for the design, management, and improvement of geographic information systems.

Previously, we have conducted some research works on the topic of model-based performance evaluation and optimization in edge computing service systems. We have applied queueing network model to the performance evaluation of IoT services deployed in edge computing paradigm [31], and further put forward a simulation-based optimization approach of efficient service selection [32]. With queueing theory, we also proposed a multiqueue approach of energy-efficient task scheduling for sensor hubs in IoT using Lyapunov optimization technique [33]. In [34], we investigated the task scheduling and resource management problem and designed an equivalent linear programming problem which could be efficiently and elegantly solved at polynomial computational complexity. In addition, we have explored generalized stochastic Petri net models for model-based performance evaluation and search-based optimization for both performance and reliability metrics [35]. However, the performance modeling, analysis, and optimization meet with new challenges in the background of GIS, due to the characteristics of different task arrivals and service procedures. This paper is our first attempt at studying the model-based evaluation and optimization issue for GIS services.

3. Analytical Model for Performance Evaluation

In this section, we apply queueing theory to construct an analytical model for performance evaluation of GIS services in edge computing paradigm. We firstly present the atomic queueing model of a GIS server and then propose a queueing network model for evaluating the overall performance of an edge computing system. The quantitative analyses of the performance metrics are also presented by solving the models mathematically. The main notations and definitions which will be used in the following discussions are provided in Table 1.

3.1. Queueing Model of a GIS Server. An atomic service represents a type of relationship-based interactions or activities between the service provider and the service consumer to achieve a certain business goal or solution objective [36]. In a GIS system, there are a number of atomic services that can provide different functionalities. For example, users upload requests to view satellite pictures of a certain area, sensors upload the temperature, humidity, and other data of a certain area in real time, and servers analyze and process a large amount of existing data. Due to the difference in the amount of calculation, some services with a small amount of calculation can be usually completed on the local devices, while some services with heavy computational workload should be deployed on more powerful edge servers.

The dynamic behavior of atomic services includes the following three basic parts. First, the request arrives at the service node and completes specific tasks according to their needs. These requests can be simple requests from users, routine sensing tasks on sensors, or complex data analysis in data centers. Second, because the resources on the service node are not unlimited, requests sometimes have to wait in the queue until the service is available. If the current queue is empty, the incoming request will be processed by the service immediately without waiting in line. Third, after the request is processed, it leaves the system.

In a real-life GIS service system, a single server can handle a number of different types of services, and the capacity of each queue should be finite. Thus, we consider a multiqueue, finite-capacity, and single-server queueing model, where each queue specifically deals with tasks of the same priority.

It has been shown that the task arrivals above the session level in distributed systems can be basically formulated by Poisson distribution [37]. And according to the known data, we can figure out that the service rate of GIS system obeys the general distribution. Therefore, we formulate a GIS server by a q-M/G/1/Ki queueing model [38].

We consider a scenario consisting of a set $\mathbb{Q}$ of q ($|\mathbb{Q}| = q$) queues. Each queue q_i, where $i \in \mathbb{Q} = \{1, 2, \ldots, q\}$ specifically deals with tasks with the same priority, is connected to the same server. Usually, tasks arrive to q_i according to the i.i.d. Poisson process with rate λ_i and are processed by the server under a general independent service rate μ_i. The order in which the server accesses the queue is determined by the queue selection rule (QSR) or the queue scheduler. To facilitate our analysis, we define the state of the multiqueue model as a q-tuple array $x = [n_1, n_2, \ldots, n_q]$, where $n_i \in [0, K_i]$ represents the number of tasks in q_i at the current moment.

With this description, we can clearly describe the current occupation of each queue with the state vector x. Furthermore, we have to introduce a secondary variable s to describe the queue currently being serviced. In this sense, another form of $[x; s] \in R^{q+1}, s \in \{1, 2, \ldots, q\}$, can give a more compact representation. Figure 1 illustrates an example of a queueing model where $x = [3, 0, 2; 1]$.

Since the service time follows the general distribution, the memoryless feature of state evolution in traditional Markovian queueing models does not hold. To facilitate the analysis, we choose our observation time for the moments when the task has just completed its service procedure. At these points, the Markovian attribute is retained and the arrival and service processes are restarted. For the sake of distinction, $[x; s^*]$ (s with a superscript $*$) is used to emphasize the observation of time as the state of the moment of departure. It should be noted that the corresponding state probabilities of $[x; s]$ and $[x; s^*]$ are denoted as $p_{x;s}$ and $\pi_{x;s^*}$, respectively.

3.1.1. Queue Transition Probability (QTP). Considering the state $[x; s^*]$, the state transitions to this state can be either (i) from any arbitrary states $[x; r^*]$ or (ii) from the null state $[0, 0, \ldots, 0; r^*]$. And the QTP is different in these two cases. In Case (i), the QTP is related to the queue selection rule (QSR). For example, in the case of the QSR is FCFS (first-come-first-served), the corresponding queue transfer probability is

$$\beta_{r \longrightarrow s}^{\mathrm{FCFS}} = \frac{n_s}{\sum_{j=1}^{q} n_j}. \tag{1}$$

Table 1: Notations and definitions.

Notations	Definitions
t_i	The i-th terminal
T	Number of terminals $\mathcal{T}$
h_j	The j-th type of tasks
H	Number of applications $\mathcal{H}$
q_j	Size of offloading request for h_j
s_j	Size of offloading response for h_j
c_j	Amount of h_j's computation
$p_{i,j}$	Probability for t_i to generate h_j
$h_{i,j}$	Task generated by t_i for h_j
$\alpha_{i,j,0}$	Probability that $h_{i,j}$ is executed by t_i
$\alpha_{i,j,1}$	Probability that $h_{i,j}$ is offloaded form t_i to edge server
λ_i	Task generation rate of t_i
$h_{i,j}^T$	Task $h_{i,j}$ which is executed by t_i
$\lambda_{i,j}^T$	Task arrival rate of $h_{i,j}^T$
$h_{i,j}^{\text{Edge}}$	Task $h_{i,j}$ which is offloaded form t_i to edge server
$\lambda_{i,j}^{\text{Edge}}$	Task arrival rate of $h_{i,j}^{\text{Edge}}$
μ_i	Service rate of t_i
μ^{Edge}	Service rate of the edge server
$\mu_{i,j}^T$	Service rate of each task $h_{i,j}^T$
$\mu_{i,j}^{\text{Edge}}$	The service rate of each task $h_{i,j}^{\text{Edge}}$
$L_{i,j}^T$	Average queue length of $h_{i,j}^T$
$L_{i,j}^{\text{Edge}}$	Average queue length of $h_{i,j}^{\text{Edge}}$
$t_{i,j}^T$	Average response time of $h_{i,j}^T$
$t_{i,j}^{\text{Edge}}$	Average response time of $h_{i,j}^{\text{Edge}}$
$r_i^{T\longrightarrow\text{Edge}}$	Uplink transmission rate
$t_{i,j}^{T\longrightarrow\text{Edge}}$	Transmission delay from t_i to edge server
$r_i^{\text{Edge}\longrightarrow T}$	Downlink transmission rate
$t_{i,j}^{\text{Edge}\longrightarrow T}$	Transmission delay from edge server to t_i
ξ_i	Energy consumption for completing each computational unit of t_i
$e_{i,j}^T$	Energy consumption caused by executing $h_{i,j}^T$
ξ^{Edge}	Energy consumption for completing each computational unit in edge server
$e_{i,j}^{\text{Edge}}$	Energy consumption caused by executing $h_{i,j}^{\text{Edge}}$
ω_i	Transmission energy consumption of t_i
$e_{i,j}^{T\longrightarrow\text{Edge}}$	Energy consumption from t_i to edge server
ω^{Edge}	Transmission energy consumption of edge
$e_{i,j}^{\text{Edge}\longrightarrow T}$	Energy consumption from edge server to t_i
$t_{i,j}$	Total time consumption for executing $h_{i,j}$
$e_{i,j}$	Total energy consumption for executing $h_{i,j}$
τ	Balance factor between energy and time

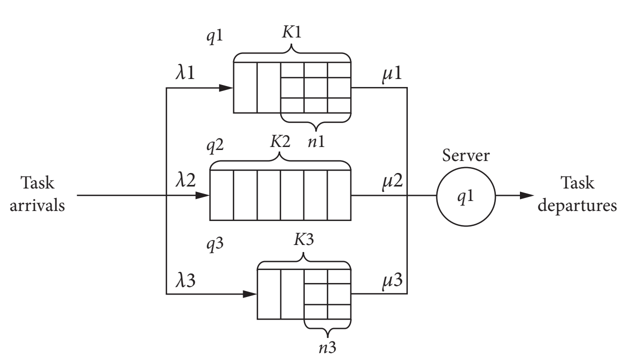

Figure 1: 3-M/G/1/Ki model when $K1 = 5$, $K2 = 6$, $K3 = 4$, $n1 = 3$, $n2 = 0$, $n3 = 2$, and $q1$ is being serviced.

However, in Case (ii), the QTP depends only on task arrival rates, which is represented as

$$\beta_{r\longrightarrow s} = \frac{\lambda_s}{\sum_{j=1}^{q}\lambda_j}. \tag{2}$$

In equation (2), the QSR is ignored since the QTP is merely related to the task arrival rates in Case (ii). For convenience, we do not need to label QSR unless it must be used.

3.1.2. Task Arrival Probability (TAP). The TAP of k arrival tasks during the service interval in the M/G/1/∞ model is represented as

$$\alpha_k = \int_0^\infty \frac{(\lambda t)^k}{k!} e^{-\lambda t} b(t) \mathrm{d}t, \quad 0 \le k < \infty, \tag{3}$$

where $b(t)$ is the probability density function (PDF) of the service time. When we solve the multiqueue model, the extension of α_k to multiqueue TAP is easily represented as

$$\begin{aligned}\alpha_{l1,l2,\ldots,l_q;s} &= \int_0^\infty \prod_{j=1}^{q} \left(\frac{(\lambda_j t)^{l_j}}{l_j!} e^{-\lambda_j t} \right) b_s(t) \mathrm{d}t, \quad 0 \le l_i < \infty, \\ &= \frac{1}{\prod_{i=1}^{q} l_i!} \int_0^\infty \left(\prod_{i=1}^{q} (\lambda_j t)^{l_j} \right) e^{-\sum_{m=1}^{q} \lambda_m t} b_s(t) \mathrm{d}t,\end{aligned} \tag{4}$$

where $\alpha_{l1,l2,\ldots,l_q;s}$ is expressed as the joint probability with l_k arrival tasks in q_k for $\forall k$ during the service interval of q_s, and $b_s(t)$ is the corresponding probability density function of the queue model. More specifically, the limited capacity of each queue should be taken into account. In the case of q-M/G/1/Ki, the formula in equation (4) needs to be modified properly further. Thus, since there are already n_i tasks in q_i, the maximum number of tasks allowed by q_i is $K_i - n_i$. And then the TAP can be expressed as $\sum_{m_i=K_i-n_i}^{\infty} \alpha_{l_1,\ldots,m_i,\ldots,l_q;s}$. Furthermore, assuming that the queues Q_{k+1} to Q_q are completely filled with tasks, $\alpha_{l_1,l_2,\ldots,l_q;s}$ is formulated as follows:

$$\sum_{m_{k+1}=K_{k+1}-n_{k+1}}^{\infty} \cdots \sum_{m_q=K_q-n_q}^{\infty} \alpha_{l_1,\ldots,l_k,m_{k+1},\ldots,m_q;s}. \tag{5}$$

3.1.3. State Transition Equations (STEs). After we have solved the QTP and TAP, the state probability $\pi_{x;s^*}$ of $[x; s^*]$ can be satisfied as the following STE to govern the dynamic of the queueing system:

$$\begin{aligned}\pi_{x;s^*} &= \sum_{r=1}^{q} \pi_{0,\ldots,0;r^*} \beta_{0\longrightarrow s} \alpha_{n1,\ldots,n_q;s} \\ &\quad + \sum_{r=1}^{q} \left[\sum_{l_1=0}^{n_1} \cdots \sum_{l_q=0}^{n_q} \pi_{n_1-l_1,\ldots,n_q-l_q;r^*} \times \beta_{r\longrightarrow s} \alpha_{l_1,\ldots,l_q;s} \right].\end{aligned} \tag{6}$$

In equation (6), the first term in the right-hand side is the probability from the null state to $[x; s^*]$, while the second term is the probability from $[x; r^*]$ to $[x; s^*]$. Based on the above formulation, the STEs composed of all feasible states can be expressed more concisely as a matrix-vector form:

$$\begin{aligned} A_\pi \pi &= 0, \\ \sum_{i=0}^{N_{q,\pi}} \pi_i &= 1, \end{aligned} \tag{7}$$

where $N_{q,\pi}$ is the number of all feasible states, π is the aggregation of $\pi_{x;s^*}$, and the state transition matrix $A_\pi \in R^{N_{q,\pi \times q,\pi}}$ consists of multiplications of QTP and TAP.

3.1.4. State Balance Equations (SBEs). Based on the QTP, TAP, and π to set up the SBEs, the state probability $p_{x;s^*}$ of $[x; s]$ is easily to be solved. According to the fact that the task flows must be conserved in the equilibrium status, SBEs can be expressed in the following equation:

$$\mathrm{IA}_{x;s} + \mathrm{ID}_{x;s}^{\mathrm{EMC}} = \mathrm{OA}_{x;s} + \mathrm{OD}_{x;s}^{\mathrm{EMC}}. \tag{8}$$

The arrival process, including $\mathrm{IA}_{x;s}$ and $\mathrm{OA}_{x;s}$, is owing to new arrival task to the queue system, for example, from $[x_r^-; s]$ to $[x; s]$ or from $[x; s]$ to $[x_r^+; s]$, which results in an increment of task during the service interval of q_s. Similarly, the departure process, including $\mathrm{ID}_{x;s}^{\mathrm{EMC}}$ and $\mathrm{OD}_{x;s}^{\mathrm{EMC}}$, is owing to departure task from the queue system, for example, from $[x; r^*]$ to $[x; s]$ or from $[x; s]$ to $[x_s^-; r]$, which results in a decrement of task in the departure instant of q_s. Thus, all of the state probabilities $p_{x;s}$ can be obtained.

(i) Null state ($x_z = [0, 0, \ldots, 0]$) probability $p_{x_z;0}$:

$$p_{x_z;0} = \frac{\lambda_{\mathrm{eff}}}{\sum_{i=1}^{q} \lambda_i} \sum_{i=1}^{q} \pi_{x_z;i*} \beta_{i\longrightarrow 0}. \tag{9}$$

(ii) Full-loaded state ($x_F = [K_1, K_2, \ldots, K_q]$) probability $p_{x_F;s}, s \neq 0$:

$$p_{x_F;s} = \mu_s^{-1} \lambda_{\mathrm{eff}} \psi_s - \sum_{n_1 \neq K_1 \text{ or} \ldots \text{or } n_q \neq K_q} P_{n_1,n_2,\ldots,n_q;s}, \tag{10}$$

where $\psi_s = \sum_{\forall n_i} \pi_{n_1,\ldots,n_s,\ldots,n_q;s^*}$.

(iii) Arbitrary state probability $p_{x;s}$ where $s \neq 0$:

$$p_{x;s} = \frac{\sum_{i=1}^{q} \lambda_i p_{x_i^-;s} + \lambda_{\mathrm{eff}} \sum_{i=1}^{q} \left(\beta_{i\longrightarrow s} \pi_{x;i^*} - \beta_{s\longrightarrow i} \pi_{x_s^-;i^*} \right)}{\sum_{i-1}^{q} \lambda_i}. \tag{11}$$

And then several performance measures can be obtained. For example, the average queue length L_s can be calculated by

$$L_s = \sum_{m=0}^{K_s} m P_{m;s} - \lambda_{\mathrm{eff}} \mu_s^{-1}, \tag{12}$$

where $P_{m;s}$ is the probability that there are m tasks in q_s and can be expressed by

$$P_{m;s} = \sum_{i=1}^{q} \sum_{n_s=m} P_{n_1,\ldots,n_s,\ldots,n_q;i}, \quad m > 0. \tag{13}$$

In equation (13), $\lambda_{\mathrm{eff}} = \lambda(1 - P_{k_s;s})$. In particular, $P_{k_s;s}$ is the probability when q_s is completely filled with tasks.

3.2. Queueing Network Model of an Edge Computing System. With the rapid development of the Internet and its applications, the single server cannot meet the needs of the vast majority of users, which is now replaced by a two-tier or even multitier group of server architecture. Therefore, we introduce edge server into the GIS system to provide higher quality of service.

All the users and sensors and other individuals who can send requests are called terminals. In the GIS system, the edge server can overwrite all the tasks request of the terminals. We define t_i as the i-th $(i \in \mathcal{T} = \{1, 2, \ldots, T\})$ terminal covered by the edge server E.

A terminal can run multiple applications concurrently, and each application may contain many different tasks. We use a set $\mathcal{H}$ $(|\mathcal{H}| = H)$ to include all types of these tasks of all terminals in $\mathcal{T}$, and h_j $(j \in \mathcal{H} = \{1, 2, \ldots, H\})$ is expressed as the j-th type of tasks.

Each h_j is profiled by 3-tuple array $[q_j, s_j, c_j]$, which is characterized by the following: (i) q_j, the size of the task offloading request (including h_j's necessary description and parameters) for h_j sent by a terminal to the edge server; (ii) s_j, the size of the task offloading response (including h_j's execution result) for h_j received by a terminal from the edge server; (iii) c_j, the amount of h_j's computation.

t_i has a probability $p_{i,j}\,(p_{i,j} \in [0,1], \sum_{j\in H} p_{i,j} = 1)$ to generate h_j during its running period. And then we can use $h_{i,j}$ to express h_j generated by t_i. The total task generation rate of t_i is defined as λ_i.

There are two ways to completing $h_{i,j}$, i.e., (i) executing it locally, or (ii) offloading it remotely. On one hand, if $h_{i,j}$ is executed by t_i locally, time and energy consumption may be taken due to the low computing capability of m_i. On the other hand, if $h_{i,j}$ is offloaded to the edge server, it may suffer time and energy costs associated with the data transfer between t_i and the edge server although meanwhile it may benefit from edge server's powerful computing resources. Such tradeoff will be carefully balanced by an approach for obtaining global optimality which will be discussed in the next section.

We define $\alpha = \left\{\alpha_{i,j,k} \mid i \in T, j \in H, k = 1 \parallel k = 0\right\}$ as the selection probability to express the probability that terminal selects whether to execute the task locally or offload it to the edge server. For $h_{i,j}$, the value of $\alpha_{i,j}$ represents (i) the probability that $h_{i,j}$ is offloaded from t_i to the edge server, if $k = 1$; or (ii) the probability that $h_{i,j}$ is executed by t_i, if $k = 0$. And we have $\alpha_{i,j,0} + \alpha_{i,j,1} = 1$.

So far, we have been able to model the tasks generated by each terminal using the q-M/G/1/Ki model. For convenience, we define the task $h_{i,j}$ which is executed by t_i as $h_{i,j}^T$ and the task $h_{i,j}$ which is offloaded to the edge server as $h_{i,j}^{\text{Edge}}$. So, the task arrival rates $\lambda_{i,j}^T$ of $h_{i,j}^T$ can be expressed as

$$\lambda_{i,j}^T = \lambda_i p_{i,j} \alpha_{i,j,0}, \quad i \in \mathcal{T},\ j \in \mathcal{H}. \tag{14}$$

Similarly, the task arrival rates $\lambda_{i,j}^{\text{Edge}}$ of $h_{i,j}^{\text{Edge}}$ can be expressed as

$$\lambda_{i,j}^{\text{Edge}} = \sum_{i=0}^{T} \lambda_i p_{i,j} \alpha_{i,j,1}, \quad i \in \mathcal{T},\ j \in \mathcal{H}. \tag{15}$$

Then, we assume that the service rate for the terminal t_i is μ_i and the service rate for the edge server E is μ^{edge}. With μ_i μ^{edge} and the amount of h_j's computation c_j, the service rate $\mu_{i,j}^T$ of each task $h_{i,j}^T$ is easily obtained as

$$\mu_{i,j}^T = \frac{\mu_i}{c_j}. \tag{16}$$

Similarly, the service rate $\mu_{i,j}^{\text{Edge}}$ of each task $h_{i,j}^{\text{Edge}}$ is given by

$$\mu_{i,j}^{\text{Edge}} = \frac{\mu^{\text{edge}}}{c_j}. \tag{17}$$

Note that $\mu_i - \sum_{j\in\mathcal{H}} \lambda_i p_{i,j} \alpha_{i,j,0} c_j > 0, i \in \mathcal{T}$ and $\mu^{\text{Edge}} - \sum_{i=0}^{T} \sum_{j\in\mathcal{H}} \lambda_i p_{i,j} \alpha_{i,j,1} c_j > 0, i \in \mathcal{T}$ are the hard constraint, which means the service rate must be greater than the task arrival rate to make sure the queue is stable.

The average queue lengths of $h_{i,j}^T$ and $h_{i,j}^{\text{Edge}}$, represented by $L_{i,j}^T$ and $L_{i,j}^{\text{Edge}}$, respectively, can be obtained. Therefore, with Little's law, the average response time of $h_{i,j}^T$ and $h_{i,j}^{\text{Edge}}$ can be obtained from the following expression:

$$\begin{aligned} t_{i,j}^T &= \frac{L_{i,j}^T}{\lambda_{i,j}^T}, \\ t_{i,j}^{\text{Edge}} &= \frac{L_{i,j}^{\text{Edge}}}{\lambda_{i,j}^{\text{Edge}}}. \end{aligned} \tag{18}$$

In addition, the size of tasks' sending and receiving delays are so tiny that they can be ignored. And the time consumption caused by tasks to be offloaded on both terminals and edge server should be paid attention to. We define $r_i^{T \longrightarrow \text{Edge}}$ as the uplink data transmission rate from t_i to the edge server. Then, the transmission delay from t_i to the edge server can be given by

$$t_{i,j}^{T\longrightarrow\text{Edge}} = \frac{\lambda_i p_{i,j} q_j}{r_i^{T \longrightarrow \text{Edge}}}. \tag{19}$$

Similarly, the downlink data transmission rate from the edge server to t_i is denoted by $r_i^{\text{Edge} \longrightarrow T}$ delay:

$$t_{i,j}^{\text{Edge}\longrightarrow T} = \frac{\lambda_i p_{i,j} s_j}{r_i^{\text{Edge} \longrightarrow T}}. \tag{20}$$

4. Performance Optimization Approach for Service Deployment

4.1. Problem Formulation

4.1.1. Energy Consumption Analysis. In recent years, energy consumption has become a research hotpot in edge computing [39–41]. How to provide better services to meet the quality of service needs of users, while reducing the energy consumption of the systems and the operating cost of services, is one of the most important issues. It is different from [41], and we consider not only the energy consumption of mobile terminals, but also the energy consumption of edge server. In the GIS system, the energy consumption includes two aspects, i.e., task execution and task transmission.

We define the energy consumption caused by executing $h_{i,j}$ at t_i and caused by executing $h_{i,j}$ at the edge server as $e_{i,j}^T$ and $e_{i,j}^{\text{Edge}}$, respectively. And they can be expressed as follows:

$$e_{i,j}^{T} = \xi_i \lambda_i p_{i,j} c_j,$$
$$e_{i,j}^{\text{Edge}} = \xi^{\text{edge}} \lambda_i p_{i,j} c_j, \quad (21)$$

where ξ_i and ξ^{Edge} are the energy consumed for each calculation at t_i and at the edge server, respectively.

Considering the energy consumption in the uplink data transmission process from t_i to the edge server, the energy consumption of t_i for the transmission is

$$e_{i,j}^{T\longrightarrow\text{Edge}} = \omega_i t_{i,j}^{T\longrightarrow\text{Edge}}, \quad (22)$$

where ω_i is the transmission energy consumption per unit time of t_i.

Similarly, the energy consumption of the edge server for the transmission is

$$e_{i,j}^{\text{Edge}\longrightarrow T} = \omega^{\text{Edge}} t_{i,j}^{\text{Edge}\longrightarrow T}, \quad (23)$$

where ω^{Edge} is the transmission energy consumption per unit time of the edge server.

The energy consumption used by the t_i to receive an offloading response is very low that it can be ignored. So far, we have got the tasks' response time and the energy consumption of task execution and transmission.

4.1.2. Utility Function. With the help of time and energy consumption of each part, we can build the corresponding utility function.

The total time consumed in executing the task includes two aspects: (i) the time consumption of terminal executing tasks, and (ii) the time consumption of the edge server executing tasks. In Case (i), the time consumption is caused by executing $h_{i,j}^{T}$ at t_i, that is, $t_{i,j}^{T}$. In Case (ii), the time consumption is caused by transmitting the offloading request of $h_{i,j}^{\text{Edge}}$ from t_i to the edge server, executing $h_{i,j}^{\text{Edge}}$ at the edge server and transmitting the offloading request of $h_{i,j}^{\text{Edge}}$ from the edge server to t_i, that is, $t_{i,j}^{T\longrightarrow\text{Edge}} + t_{i,j}^{\text{Edge}} + t_{i,j}^{\text{Edge}\longrightarrow T}$.

In summary, the total time consumption for executing h_{i_j} is easily obtained:

$$t_{i,j} = \alpha_{i,j,0} t_{i,j}^{T} + \alpha_{i,j,1}\left(t_{i,j}^{T\longrightarrow\text{Edge}} + t_{i,j}^{\text{Edge}} + t_{i,j}^{\text{Edge}\longrightarrow T}\right). \quad (24)$$

The total energy consumed in executing the task includes two aspects: (i) the energy consumption of terminal executing tasks and (ii) the energy consumption of the edge server executing tasks. In Case (i), the energy consumption is caused by executing $h_{i,j}^{T}$ at t_i, that is, $e_{i,j}^{T}$. In Case (ii), the energy consumption is caused by transmitting the offloading request of $h_{i,j}^{\text{Edge}}$ from t_i to the edge server, executing $h_{i,j}^{\text{Edge}}$ at the edge server and transmitting the offloading request of $h_{i,j}^{\text{Edge}}$ from the edge server to t_i, that is, $e_{i,j}^{T\longrightarrow\text{Edge}} + e_{i,j}^{\text{Edge}} + e_{i,j}^{\text{Edge}\longrightarrow T}$.

In summary, the total energy consumption for executing $h_{i,j}$ is easily obtained:

$$e_{i,j} = \alpha_{i,j,0} e_{i,j}^{T} + \alpha_{i,j,1}\left(e_{i,j}^{T\longrightarrow\text{Edge}} + e_{i,j}^{\text{Edge}} + e_{i,j}^{\text{Edge}\longrightarrow T}\right). \quad (25)$$

In general, total time consumption and total energy consumption in the GIS system can be easily obtained as $\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} t_{i,j}$ and $\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} e_{i,j}$, respectively.

Therefore, the utility function can be built to evaluate the overall benefit of the GIS system. We normalize the energy consumption and time consumption, and thus the utility function is defined as follows:

$$f = \tau \frac{\tilde{t} - \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} t_{i,j}}{\tilde{t}} + (1-\tau)\frac{\tilde{e} - \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} e_{i,j}}{\tilde{e}}, \quad (26)$$

where $\tau \in [0,1]$ is the balance factor between energy consumption and time consumption and $\tilde{t} = \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} t_{i,j}^{T}$ and $\tilde{e} = \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} e_{i,j}^{T}$ are the total time consumption and total energy consumption when the all tasks are executed in terminal without offloading, respectively. We should note that the closer τ is to 1, the more weight we put on time consumption. On the contrary, the closer τ is to 0, the more attention we pay on energy consumption. Therefore, τ should be set properly by the system manager to balance the tradeoff between performance and energy consumption according to the requirements in real-life scenarios.

4.1.3. Optimization Problem Formulation. With all the analytical results presented in the above sections, we formulate an optimization problem in GIS systems as follows:

$$\max \quad f, \quad (27)$$

$$\text{s.t.} \quad \mu^{\text{Edge}} - \sum_{i=0}^{T}\sum_{j\in\mathcal{H}} \lambda_i p_{i,j}\alpha_{i,j,1}c_j > 0, \quad i \in \mathcal{T}, \quad (28)$$

$$\mu_i - \sum_{j\in\mathcal{H}} \lambda_i p_{i,j}\alpha_{i,j,0}c_j > 0, \quad i \in \mathcal{T}, \quad (29)$$

$$\alpha_{i,j,k} \in [0,1], \quad i \in \mathcal{T},\ j \in \mathcal{H},\ k = 0\|k = 1\|, \quad (30)$$

$$\alpha_{i,j,0} + \alpha_{i,j,1} = 1, \quad i \in \mathcal{T}, j \in \mathcal{H}, \quad (31)$$

where constraints (28) and (29) are the hard constraint of the GIS system of each terminal and the edge server, which is used to make the queue system stable, respectively. And constraint (30) is the value range of $\alpha_{i,j,k}, i \in \mathcal{T}, j \in \mathcal{H}, k = 0\|k = 1$. Constraint (31) limits the total probability of offloading to the edge server, and local execution of each task is 1:

$$f = \tau \frac{\widetilde{t} - = \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} t_{i,j}}{\widetilde{t}} + (1-\tau)\frac{\widetilde{e} - = \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}} e_{i,j}}{\widetilde{e}},$$

$$= 1 + \frac{1-\tau}{\widetilde{e}}\left(-\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}}\alpha_{i,j,0}e_{i,j}^{T} + \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}}\left(1-\alpha_{i,j,0}\right)\left(e_{i,j}^{T\longrightarrow \text{Edge}} + e_{i,j}^{\text{Edge}} + e_{i,j}^{\text{Edge}\longrightarrow T}\right)\right) \tag{32}$$

$$+ \frac{\tau}{\widetilde{t}}\left(\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}}\alpha_{i,j,0}t_{i,j}^{T} + \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{H}}\left(1-\alpha_{i,j,0}\right)\left(t_{i,j}^{T\longrightarrow \text{Edge}} + t_{i,j}^{\text{Edge}} + t_{i,j}^{\text{Edge}\longrightarrow T}\right)\right).$$

4.2. Optimization Approach. Due to the complexity of the utility function, we propose a heuristic algorithm based on differential evolution (DE) algorithm [42, 43] which has good convergence properties with few control variables.

DE is a parallel direct search method which utilizes NP D-dimensional parameter vectors, expressed as

$$\underline{x}_{i,G}, \quad i = 1, 2, \ldots, \text{NP}, \tag{33}$$

as a population for each generation G, where NP is the number of individuals in the population and does not change during the optimization process, i represents the i-th individual in the population, D is the dimension of the decision space. In a GIS system, if there are T terminals and H applications, the value of D is $T \times H$. And all the individuals in the population for the generation G is represented as $\underline{x}_G$. The j-th dimension variable of i-th individual in the population for the generation G is defined as

$$\underline{x}_{i,G}^{j}, \quad i = 1, 2, \ldots, \text{NP}, \ j = 1, 2, \ldots, D. \tag{34}$$

The DE algorithm includes the following four parts.

4.2.1. Initialization. As shown in Algorithm 1, if the system is unbeknown, the initial population should be chosen randomly.

4.2.2. Mutation. The core idea of DE is a new scheme for generating trial parameter vectors, which is called as mutation. DE generates new parameter vectors by using parameter F to add the weighted difference vector between two individuals to a third individual. For each vector $\underline{x}_{i,G}$ $(i = 0, 1, 2, \ldots, \text{NP} - 1)$, a perturbed vector $\underline{v}_{i,G+1}$ is generated according to Algorithm 2, with $r_1, r_2, r_3 \in [0, \text{NP} - 1]$, $i \neq r_1 \neq r_2 \neq r_3$. $F \in (0, 2)$ is a real and constant factor, which controls the amplification of the differential variation $(\underline{x}_{r_2,G} - \underline{x}_{r_3,G})$.

4.2.3. Crossover. In order to improve the diversity of the perturbed parameter vectors, crossover is introduced. To this end, the vector

$$\underline{u}_{i,G+1}, \tag{35}$$

with

$$\underline{u}_{i,G+1}^{j} = \begin{cases} \underline{v}_{i,G+1}^{j}, & \text{for } j = \langle n\rangle_D, \langle n+1\rangle_D, \ldots, \langle n+L-1\rangle_D, \\ \underline{x}_{i,G}^{j}, & \text{otherwise}, \end{cases} \tag{36}$$

is formed. The acute brackets $\langle\cdot\rangle_D$ denote the modulo function with modulus D. The starting index, $n \in [0, D-1]$, in equation (36) is a randomly chosen integer. The integer L, which represents the number of parameters that are going to be exchanged, is drawn from $[1, D]$ with the probability $Pr(L = v) = (CR)^{v-1}$, where $CR \in [0, 1]$ is the crossover probability. The random decisions for both n and L are made anew for each process of crossover. The crossover procedures are presented by Algorithm 3.

4.2.4. Selection. In order to decide whether the new vector $\underline{u}_{i,G+1}$ can become an individual in the population of generation $G+1$, it will be compared to $\underline{x}_{i,G}$. If vector $\underline{u}_{i,G+1}$ yields a larger objective function value which is the utility function in equation (34) than $\underline{x}_{i,G}$, $\underline{x}_{i,G+1}$ is set to $\underline{u}_{i,G+1}$, otherwise $\underline{x}_{i,G+1}$ retains $\underline{x}_{i,G}$. In addition, the optimal parameter vector $\underline{x}_{\text{best},G}$ is recorded for every generation G in order to keep track of the progress that is made during the optimization process. The selection scheme is formally presented in Algorithm 4.

Based on the following four parts, Algorithm 5 gives the main program of DE algorithm, which provides an approach on how to deploy the GIS services in the edge computing system. The near-optimal solutions for maximizing the utility function while satisfying the constraints can be obtained in an efficient way.

5. Evaluation

5.1. Experimental Setup. We conduct experiments based on the data collected from a real-world GIS system which has been deployed in reality providing real-time street view mapping services. The services are a kind of virtual reality service that provides end-users a 360-degree view panorama of the cities, streets, and other details. All the original data of the mapping services have been collected from real world by cars equipped with 3-dimensional laser scanners, global navigation satellite systems (GNSS), inertial measurement units (IMU), and panoramic cameras. Such original data have been stored in cloud data centers and processed by GIS servers. Upon the arrival of a task for users requesting a mapping service at a certain location, the task is firstly analyzed and initialized, and is divided into several subtasks

for $i = 1$ to NP **do**
  for $j = 1$ to D **do**
    $\underline{x}^{j}_{i,G} = \underline{x}^{j}_{\min} + \mathrm{rand}(0,1) \times (\underline{x}^{j}_{\max} - \underline{x}^{j}_{\min})$
  end for
end for

Algorithm 1: Initialization ($\underline{x}_G$).

$r_1, r_2, r_3 \in [0, NP-1]$
$i \neq r_1 \neq r_2 \neq r_3$
$F \in (0,2)$
$\underline{v}_{i,G+1} = \underline{x}_{r_1,G} + F \times (\underline{x}_{r_2,G} - \underline{x}_{r_3,G})$

Algorithm 2: Mutation ($\underline{x}_{i,G}$).

$n = \mathrm{rand}[0,\ D]$ integer
$L = 1$
while ($\mathrm{rand}() < CR$ and $L < D$) **do**
  $L = L + 1$
end while
for $i = n$ to $n + D$ **do**
  $j = i\%D$
  if $i \geq n$ and $i \leq n + L - 1$ **then**
    $\underline{u}^{j}_{i,G+1} = \underline{v}^{j}_{i,G+1}$
  else
    $\underline{u}^{j}_{i,G+1} = \underline{x}^{j}_{i,G}$
  end if
end for

Algorithm 3: Crossover ($\underline{v}_{i,G+1}, \underline{x}_{i,G}$).

if $f(\underline{u}_{i,G+1}) > f(\underline{x}_{i,G})$ **then**
  $\underline{x}_{i,G+1} \longleftarrow \underline{u}_{i,G+1}$
else
  $\underline{x}_{i,G+1} \longleftarrow \underline{x}_{i,G}$
end if
if $f(\underline{x}_{i,G+1}) > f(\underline{x}_{\mathrm{best},G+1})$ **then**
  $\underline{x}_{\mathrm{best},G+1} \longleftarrow \underline{x}_{i,G+1}$
end if

Algorithm 4: Selection ($\underline{u}_{i,G+1}, \underline{x}_{i,G}, \underline{x}_{best,G+1}$)

$G \longleftarrow 1$
Initialization ($\underline{x}_G$)
while $G < T$ **do**
  $\underline{x}_{\mathrm{best},G+1} = \underline{x}_{1,G}$
  for $i = 1$ to NP **do**
    $\underline{v}_{i,G+1}$ = *Mutation* ($\underline{x}_{i,G}$)
    $\underline{u}_{i,G+1}$ = *Crossover* ($\underline{v}_{i,G+1}, \underline{x}_{i,G}$)
    $\underline{x}_{i,G+1}$ = *Selection* ($\underline{u}_{i,G+1}, \underline{x}_{i,G}, \underline{x}_{best,G+1}$)
  end for
  $G \longleftarrow G + 1$
end while

Algorithm 5: Differential evolution of service deployment.

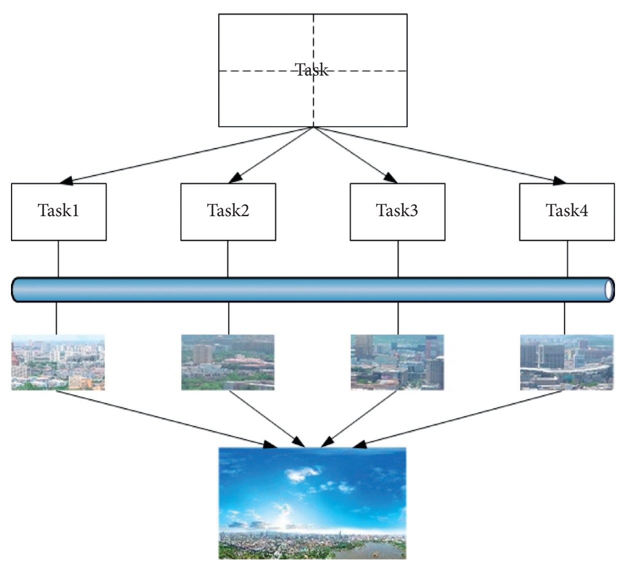

Figure 2: An example of GIS service workflow.

to be processed on a few cluster nodes in a parallel way. Each cluster node only processes a part of the original mapping data, and after completing the data processing, it returns the results to the centralized server for task convergence. The workflow of the GIS services is illustrated by Figure 2.

There are five nodes in our GIS systems. The centralized main server is equipped with an 8-core Intel Ice Lake CPU working at the maximum frequency of 4.7 GHz, and memory with capacity of 16 GB. Each cluster node has a CPU with 4 Intel Kaby Lake cores at maximum 3.8 GHz frequency as well as 16 GB or 8 GB memory.

The performance data are collected from such the GIS system during its service procedures for real-world users. We use the data to initialize the system parameters such as service rates and basic system architecture. Other parameters that we are not able to obtain from the system are set empirically shown as Table 2. Then, we apply our approach to analyze the impact of deploying the GIS services into edge computing architecture on the performance attributes, and validate our analytical results. During the experiments, we

Table 2: Parameter settings of the GIS system in experiments.

Name	Comment	Value
T	Number of terminals	[5, 25]
H	Number of tasks in each terminal	[10, 25]
λ_i	Task generation rate of t_i	[0, 1]/s
$p_{i,j}$	Probability for t_i to generate h_j	[0, 1]
q_j	Size of offloading request for h_j	[3, 5] MB
s_j	Size of offloading response for h_j	[1, 3] MB
c_j	Amount of h_j's computation	[200, 400] MI
μ_i	Service rate of t_i	$1.5\delta_i$ MIPS
μ^{Edge}	Service rate of the edge server	1000 MIPS
$r_i^{T\longrightarrow\text{Edge}}$	Uplink transmission rate	[1, 3] MB/s
$r_i^{\text{Edge}\longrightarrow T}$	Downlink transmission rate	[3, 5] MB/s
ξ_i	Each calculation energy consumption of t_i	[8, 10] × 10^{-3} J/MI
ξ^{Edge}	Each calculation energy consumption of edge	[3, 5] × 10^{-3} J/MI
ω_i	Transmission energy consumption of t_i	[100, 120] mW
ω^{Edge}	Transmission energy consumption of edge	[60, 80] mW
τ	Balance factor between energy and time	[0, 1]

also have to tune some system parameters for illuminating the effectiveness of our approach.

5.2. Experimental Results. In order to verify the applicability of the strategy, extensive simulations experiments are carried out to evaluate its efficacy. The simulation results demonstrate that the optimization approach based on the DE algorithm performs well in both utility function value and calculation time in different scenarios.

5.2.1. Efficacy Analysis. Although the DE algorithm cannot guarantee the global optimality, the simulation experiments show that the optimization algorithm has a strong global search ability. As shown in Figure 3, we illuminate the average utility values of population and their optimal values, which shows that our algorithm converges at about 300th generation.

We increase the dimension of decision space by increasing the number of terminals to 50. As shown in Figure 4, the algorithm converges at about 900th generation and the results are very close to the global optimal solutions.

With the further increase of the dimension of decision space by increasing the number of tasks in each terminal to 50, we find that the results converge over 1000th generations in Figure 5. The experimental results shown in Figures 3 to 5 validate that our approach performs well in solving large-scale optimization problems.

It has been well-known that, when the scale of the problem is small, the problem can be solved by some traditional optimization algorithms accurately. However, with the scale of the problem increases, the number of feasible solutions increases exponentially, which leads to the combination explosion of search space. And then, we analyze the calculation time of our algorithm in different dimension of decision space. Figure 6 shows that the computing time

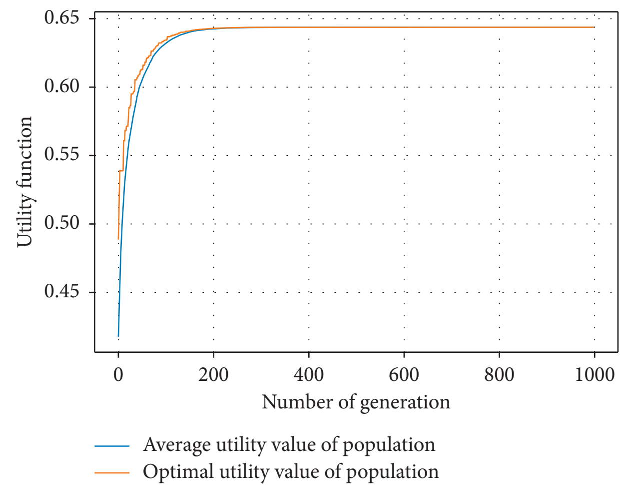

Figure 3: The utility function value of each generation by the DE algorithm, where $T = 5$ and $H = 10$.

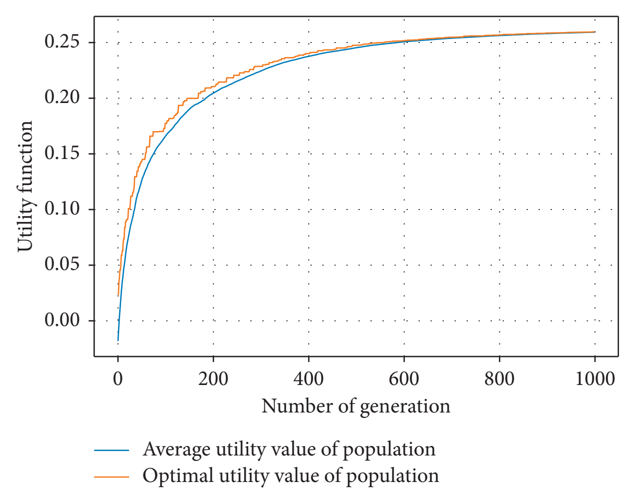

Figure 4: The utility function value of each generation by the DE algorithm, where $T = 50$ and $H = 10$.

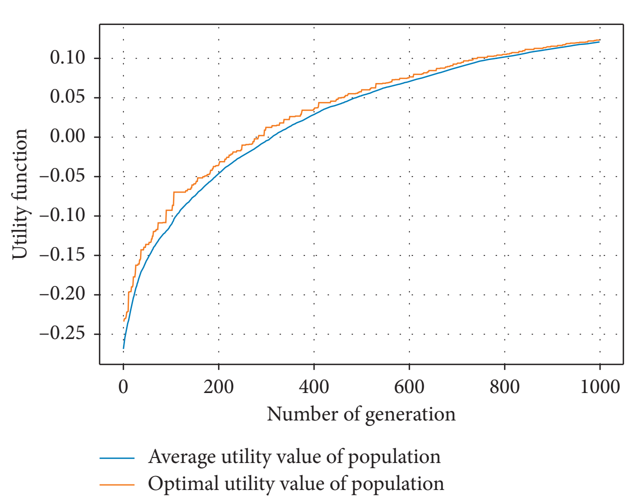

Figure 5: The utility function value of each generation by the DE algorithm, where $T = 50$ and $H = 50$.

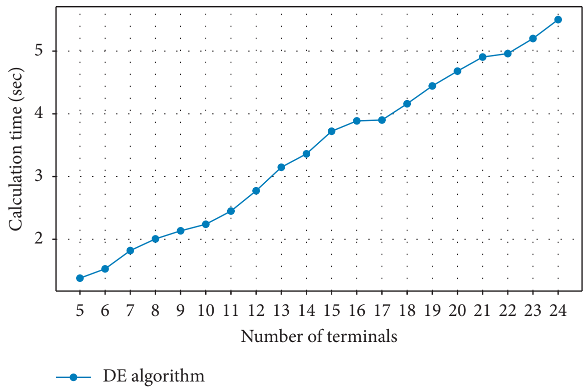

Figure 6: Empirical results of calculation time with the increase of the number of terminals.

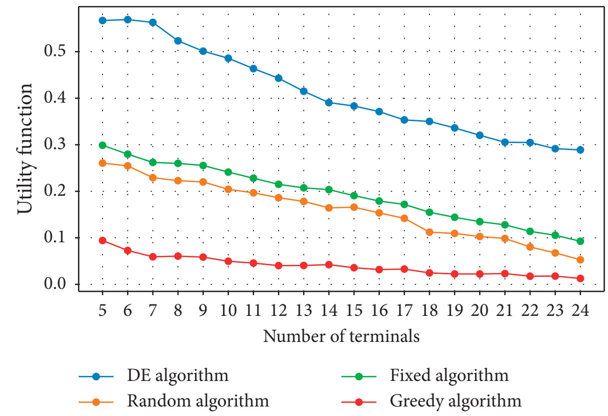

Figure 8: Empirical results of utility value with the increase of the number of terminals.

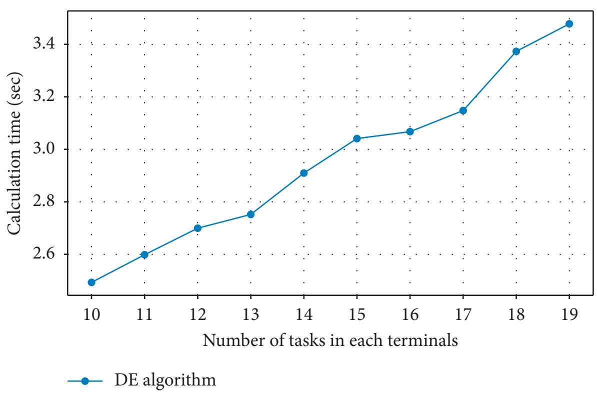

Figure 7: Empirical results of calculation time with the increase of tasks in each terminal.

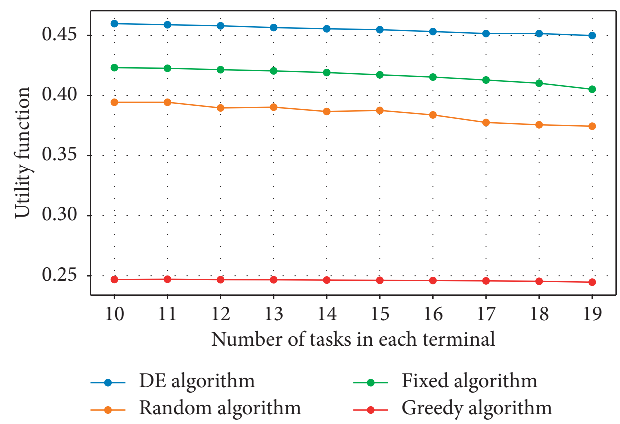

Figure 9: Empirical results of utility value with the increase of tasks in each terminal.

increases linearly with the number of terminals, where $H = 10$ and T increases from 5 to 24. Similarly, Figure 7 shows that the computing time increases linearly with the tasks of each terminal, where $T = 10$ and H increases from 10 to 19. Experimental results demonstrate that the DE algorithm is efficient in solving large-scale problems.

5.2.2. Comparison Analysis. Since there has been no existing well-developed scheme of service deployment optimization scheme for GIS services, we compare our approach with other three straightforward approaches which have been widely applied in practise. The first one is the random scheduling algorithm usually performs well in load balancing. The second approach is fixed algorithm which means that 50% of tasks are offloaded to the edge server. The third one is greedy algorithm in which tasks will be offloaded to the edge server as long as there is available resource.

We firstly tune the number of terminals T from 5 to 24, with fixed value $H = 10$ and $\tau = 0.5$, and the experimental results are shown in Figure 8. With the increase of T, the workload of the GIS system increases at both of the terminals and the edge server. Meanwhile, the time consumption and energy consumption increase so the utility function value decreases. Figure 8 also illustrates that our approach performs 50% better than the random algorithm, fixed algorithm, and greedy algorithm in terms of utility value.

We then tune the parameter H which is the number of tasks can be executed in each terminal from 10 to 19, and the empirical results are shown by Figure 9. We have similar conclusion that the scheme presented in this paper is 50% better than the random approach.

Finally, we discuss the impact brought by the balance factor τ, which trades off the weight between energy consumption and time consumption. The experimental results are shown in Figure 10. With the increase of τ, we pay more weight on optimizing the response time. In such scenario, introducing edge computing layer can benefit dramatically because of its additional computational capability. Since our algorithm is able to fully utilize the edge layer and optimize the global utility function, the utility values obtained by our DE approach are increasingly higher than random scheduling with the increase of τ.

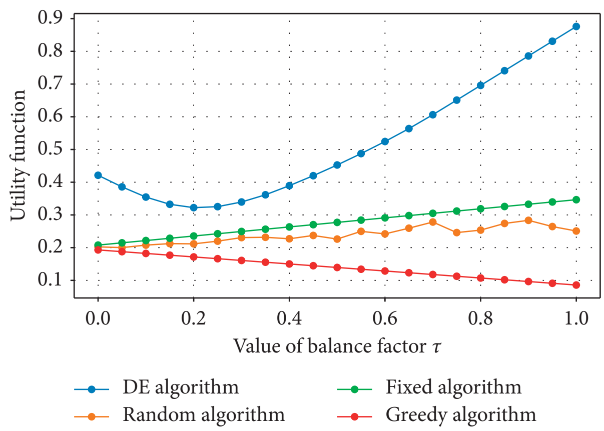

FIGURE 10: Empirical results of utility value with the increase of the balance factor between energy consumption and time consumption.

6. Conclusion

As GIS services become increasingly popular in daily life, the performance has drawn more and more attention. Deploying GIS services into edge computing architecture is an effective way for improving the performance. This paper conducts a quantitative study on the performance evaluation and optimization issue in deploying GIS services into the edge. Queueing models are presented for formulating the GIS services, and their corresponding analyses are provided in detail. Based on the analytical results, a heuristic approach is designed for obtaining the near-optimal solution of service deployment. Experiments based on the dataset collected from real-life GIS service systems are conducted, and the efficacy of the approach is validated. This work is expected to provide a theoretical reference of the evaluation and optimization of edge computing GIS systems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Plan (no. 2016YFC0303700), National Natural Science Foundation of China (no. 61972414), Beijing Natural Science Foundation (no. 4202066), Beijing Nova Program (no. Z201100006820082), and the Fundamental Research Funds for Central Universities (no. 2462018YJRC040).

References

[1] M. Ehlers, "Remote sensing and geographic information systems: towards integrated spatial information processing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, p. 763, 1990.

[2] S. Shekhar, S. Ravada, D. Chubb, and G. Turner, "Declustering and load-balancing methods for parallelizing geographic information system," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 4, pp. 632–655, 1998.

[3] W. Li, M. Song, B. Zhou, K. Cao, and S. Gao, "Performance improvement techniques for geospatial web services in a cyberinfrastructure environment—a case study with a disaster management portal," *Computers, Environment and Urban Systems*, vol. 54, pp. 314–325, 2015.

[4] N. Torres-Cruz, I. Villordo-Jimenez, and A. Montiel-Saavedra, "Analysis of the geographical-information impact on the performance of ABS-CRE HetNets," *IEEE Latin America Transactions*, vol. 18, no. 3, pp. 613–622, 2020.

[5] P. G. Lopez, A. Montresor, D. Epema et al., "Edge-centric computing," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[6] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4242–4251, 2019.

[7] M. Song, Y. Duan, T. Huang, and L. Zhan, "Inter-edge and cloud conversion accelerated user-generated content for virtual brand community," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 14, pp. 1–17, 2020.

[8] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6172–6181, 2020.

[9] L. Qi, C. Hu, X. Zhang et al., "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Transactions on Industrial Informatics*, In press.

[10] Y. Duan, Z. Lu, Z. Zhou, X. Sun, and J. Wu, "Data privacy protection for edge computing of smart city in a DIKW architecture," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 323–335, 2019.

[11] L. Qi, X. Wang, X. Xu, W. Dou, and S. Li, "Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing," *IEEE Transactions on Network Science and Engineering*, In press.

[12] Y. Zhang, J. Pan, L. Qi, and Q. He, "Privacy-preserving quality prediction for edge-based IoT services," *Future Generation Computer Systems*, vol. 114, pp. 336–348, 2021.

[13] Y. Duan, X. Sun, H. Che, C. Cao, Z. Li, and X. Yang, "Modeling data, information and knowledge for security protection of hybrid iot and edge resources," *IEEE Access*, vol. 7, pp. 99161–99176, 2019.

[14] R. Barik, H. Dubey, S. Sasane, C. Misra, N. Constant, and K. Mankodiya, "Fog2Fog: augmenting scalability in fog computing for health GIS systems," in *Proceedings of the 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pp. 241-242, Philadelphia, PA, USA, July 2017.

[15] H. Truong and M. Karan, "Analytics of performance and data quality for mobile edge cloud applications," in *Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD 2018)*, pp. 660–667, San Francisco, CA, USA, July 2018.

[16] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating performance of containerized IoT services for clustered devices at the network edge," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 1019–1030, 2017.

[17] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," in *Proceedings of the 2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT 2016)*, pp. 1–7, Cairo, Egypt, April 2016.

[18] L. Qi, Q. He, F. Chen, X. Zhang, W. Dou, and Q. Ni, "Data-driven web APIs recommendation for building web applications," *IEEE Transactions on Big Data*, In press.

[19] C. d. S. Baptista, C. P. Nunes, A. G. de Sousa, E. R. da Silva, F. L. Leite, and A. C. de Paiva, "On performance evaluation of web GIS applications," in *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05)*, pp. 497–501, Copenhagen, Denmark, August 2005.

[20] Y. Zhang, K. Wang, Q. He et al., "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, In press.

[21] Z. Wang, Q. Zhao, F. Xu, H. Dai, and Y. Zhang, "Detection performance of packet arrival under downclocking for mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 9641712, 7 pages, 2018.

[22] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed Petri nets," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[23] L. Li, S. Li, and S. Zhao, "QoS-aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 1497–1505, 2014.

[24] A. Mebrek, L. Merghem-Boulahia, and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the IoT-fog-cloud computing," in *Proceedings of the 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA 2017)*, pp. 1–4, Cambridge, MA, USA, October 2017.

[25] H. Wu, S. Deng, W. Li, M. Fu, J. Yin, and A. Y. Zomaya, "Service selection for composition in mobile edge computing systems," in *Proceedings of the 2018 IEEE International Conference on Web Services (ICWS 2018)*, pp. 355–358, San Francisco, CA, USA, July 2018.

[26] Y. Zhang, C. Gao, Y. Guo et al., "Proactive video push for optimizing bandwidth consumption in hybrid CDN-P2P VoD systems," in *Proceedings of the 2018 IEEE Conference on Computer Communications (INFOCOM 2018)*, pp. 2555–2563, Honolulu, HI, USA, April 2018.

[27] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, In press.

[28] Y. Zhang, G. Cui, S. Deng, F. Chen, Y. Wang, and Q. He, "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, In press.

[29] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets 2016)*, pp. 50–56, Atlanta, GA, USA, November 2016.

[30] Y. Chen, N. Zhang, Y. Zhang et al., "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, p. 1. In press.

[31] J. Huang, S. Li, Y. Chen, and J. Chen, "Performance modelling and analysis for IoT services," *International Journal of Web and Grid Services*, vol. 14, no. 2, pp. 146–169, 2018.

[32] J. Huang, Y. Lan, and M. Xu, "A simulation-based approach of QoS-aware service selection in mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 5485461, 10 pages, 2018.

[33] J. Huang, C. Zhang, and J. Zhang, "A multi-queue approach of energy efficient task scheduling for sensor hubs," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[34] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-To-Peer Networking and Applications*, vol. 13, no. 5, pp. 1776–1787, 2020.

[35] J. Huang, J. Liang, and S. Ali, "A simulation-based optimization approach for reliability-aware service composition in edge computing," *IEEE Access*, vol. 8, pp. 50355–50366, 2020.

[36] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*, Springer, Berlin, Germany, 2007.

[37] E. Chlebus and J. Brazier, "Nonstationary poisson modeling of web browsing session arrivals," *Information Processing Letters*, vol. 102, no. 5, pp. 187–190, 2007.

[38] M.-S. Chen and H.-W. Yen, "A two-stage approach in solving the state probabilities of the multi-queueM/G/1 model," *International Journal of Systems Science*, vol. 47, no. 5, pp. 1230–1244, 2016.

[39] Y. Chen, Y. Zhang, Y. Wu, L. Qi, X. Chen, and X. Shen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8419–8429, 2020.

[40] X. Xu, S. Fu, W. Li, F. Dai, H. Gao, and V. Chang, "Multi-objective data placement for workflow management in cloud infrastructure using NSGA-II," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 605–615, 2020.

[41] W. Fan, Y. a. Liu, B. Tang, F. Wu, and Z. Wang, "Computation offloading based on cooperations of mobile edge computing-enabled base stations," *IEEE Access*, vol. 6, pp. 22622–22633, 2018.

[42] R. Storn and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 23, no. 1, 1995.

[43] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of the Fuzzy Information Processing Society, Nafips Biennial Conference of the North American*, Berkeley, CA, USA, June 1996.