EBA: An Adaptive Large Neighborhood Search-Based Approach for Edge Bandwidth Allocation

Qinghong Hu[1,2], Qinglan Peng[3], Jiaxing Shang[1,2(✉)], Yong Li[1,2], and Junjie He[4]

[1] College of Computer Science, Chongqing University, Chongqing, China
qinghonghu@yeah.net, {shangjx,yongli}@cqu.edu.cn
[2] Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing, China
[3] School of Artificial Intelligence, Henan University, Zhengzhou, China
qinglan.peng@hotmail.com
[4] Information and Data Center, Guizhou Minzu University, Guiyang, China
18212727047@163.com

Abstract. As a promising computing paradigm, edge computing aims at delivering high-response and low-latency computing, storage, and bandwidth resources to end-users at the edge of the network. However, those edge-based novel applications (e.g., live broadcast, edge cloud game, real-time AR/VR rendering, etc.) are usually bandwidth-consuming, which has made a considerable contribution to the operating costs of edge application providers. Meanwhile, the bandwidth pricing modes of edge infrastructure providers are also complicated. Therefore, how to allocate the bandwidth demands of edge users to suitable edge servers to minimize the monetary cost of edge application providers becomes an important issue. In this paper, we consider the widely adopted 95th-percentile bandwidth billing mode and Quality-of-Service constrained edge bandwidth allocation problem, and propose a neighborhood search-based approach, shorts for EBA. It firstly employs a network flow-based heuristic to find a feasible initial solution quickly. Then, an adaptive large neighborhood search-based method is utilized to perform iterative optimization, which contains hill climbing and simulated annealing mechanisms. Therefore, the proposed EBA approach can expand the searching space, accelerate the optimization convergence speed, and avoid falling into local optimal. Experiments based on a real-world edge bandwidth consumption dataset illustrate the effectiveness of the proposed approach.

Keywords: Edge computing · Edge bandwidth allocation · 95th-percentile billing

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 249–268, 2022.
https://doi.org/10.1007/978-3-031-24383-7_14

1 Introduction

With the rapid development of advanced communication technologies and novel Internet of Things (IoT) technologies, nowadays we are surrounded by various kinds of smart mobile devices, which continuously generate a large amount of data day and night, thus creating the demand for processing local data using local computing resources [22]. To this end, the multi-access edge computing (MEC) paradigm emerged and has been widely used to support those computing-intensive, latency-sensitive, and privacy-preserving applications. The key idea of MEC is to sink the computing, storage, and bandwidth resources to the edge of the network, i.e., closer to the end-users [14,19]. Benefitting from edge computing-offloading technologies, the MEC paradigm has successfully supported the realization and prosperity of various novel applications, such as extreme-low-latency live broadcasting, online video game rendering, and immersion metaverse.

However, different from the traditional cloud computing paradigm with infinite resources, edge computing is usually constrained by limited resources [9] and high prices, especially for bandwidth resources. According to Gartner's prediction [2], by 2025 bandwidth cost will be the primary driver for new edge computing deployments, especially for those heavy content delivery applications (e.g., live streaming, cloud games, etc.). For these edge application providers, bandwidth accounts for a significant portion of their operating costs. Therefore, for edge application providers, it is of great practical significance to study how to properly allocate the bandwidth demands of edge users to appropriate edge servers while fulfilling the Quality-of-Service (QoS) constraint to reduce the cost.

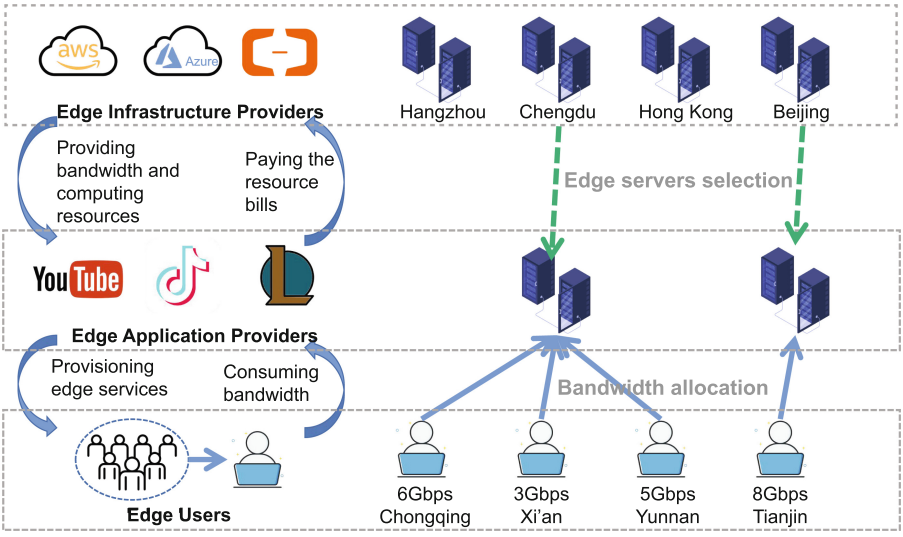

Fig. 1. Edge bandwidth allocation problem.

As shown in Fig. 1, where edge user refers to a group of end-users located in a certain area (e.g., all Tiktok users in Singapore), and its bandwidth demands are the aggregation of all individual end-users in that group at a certain time point. Note that, different user groups could have different bandwidth demands, which fluctuate over time. Edge application providers need to employ suitable edge servers, and properly schedule edge users' bandwidth demands to minimize their operating costs to improve their competitiveness. In this study, we consider the 95th-percentile billing mode, widely adopted by edge cloud infrastructure providers to provision bandwidth resources in real-world application scenarios [4, 15,21]. As shown in Fig. 2, the billing period of the 95th-percentile billing mode is usually a month. Bandwidth providers sample the real-time bandwidth usage every 5 min (i.e., 8640 checkpoints for a month), discard the top 5% checkpoints, and choose the rest highest value as the actual billing bandwidth amount. The 95th-percentile billing is also called burstable billing. In this mode, since the top 5% of bandwidth usage is ignored, a great deal of cost could be saved if the bandwidth demands are properly allocated, and this is the edge bandwidth allocation problem.

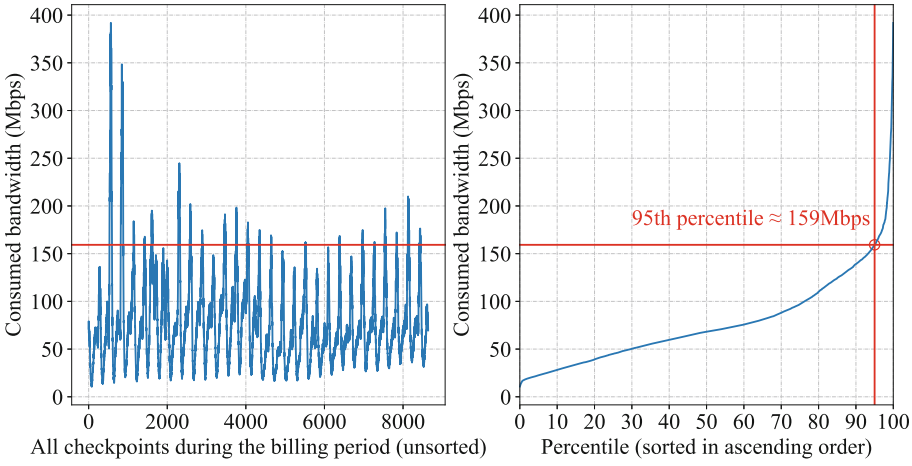

Fig. 2. The 95th-percentile billing mode.

The challenge of this problem also comes from the 95th-percentile billing mode, which has been proved to be an NP-hard problem [11]. Although many previous studies have been carried out, the existing methods rarely consider the network latency constraint in the edge bandwidth allocation problem. To fill this gap, in this paper we propose an adaptive large neighborhood search-based approach named EBA to solve the edge bandwidth allocation problem. Our approach mainly consists of two phrases. In the first phase, to quickly generate a good enough and feasible initial solution, we introduce intuitive sorting strategies which take full advantage of the top 5% non-billing checkpoints, and gradually

increase the billing volume of the servers using the previously allocated information. In the second phase, to achieve load consolidation, we propose an iterative continuous optimization method which combines hill climbing and simulated annealing mechanisms based on adaptive large neighborhood search. In order to efficiently verify the feasibility of the neighborhood of the current solution, we design OptDinic suitable for this optimization based on the Dinic algorithm.

In sum, the main contributions of this paper are as follows:

1) Different from the traditional works of considering on-demand bandwidth billing [5] or static quota bandwidth billing, we consider 95th-percentile billing, which is more complicated and closer to the reality.
2) In our system model, we take the geographical distribution of edge users and edge servers, the various communication distances, and the resulting latency as QoS constraints.
3) Our EBA approach can expand the searching space, accelerate the optimization convergence speed, and avoid falling into local optimal.
4) We conducted a series of experiments based on a real-world edge application bandwidth demand tracking dataset to demonstrate the effectiveness of the proposed approach.

2 Related Work

For edge application providers, their goal is to minimize the total cost of bandwidth paid to the edge infrastructure providers while maintaining the QoS to the edge users. Extensive studies have been carried out on this problem of great practical significance till now.

For Delay-Tolerant Bulk (DTB) data, a lot of works minimized the 95th percentile cost through job-level scheduling, specifically by delaying some data to future time slots [8,10,13]. Wang *et al.* [23] made a trade-off between cost and delayed performance by considering both the cost charged by ISP and the penalty for delayed service. Wang *et al.* [24] studied how to selectively delay some traffic demands to save Internet access costs without sacrificing much service quality. An offline optimal solution based on dynamic programming is proposed.

Goldenberg *et al.* [7] considered how to allocate users' traffic demands to different ISPs to minimize the cost based on percentile billing without delay. For this purpose, they designed a series of intelligent routing algorithms. Adler *et al.* [3] provided an optimal offline algorithm that routes traffic to minimize the total bandwidth cost incurred in ISPs with AVG and MAX contracts. Peng *et al.* [16,25] took the long-term edge user allocation rate and edge server leasing cost as scheduling targets and proposed a decentralized collaborative and fuzzy-control-based approach to yielding real-time user-edge-server allocation schedules. Khare *et al.* [12] proposed NetReq to assign users' requests to reduce servers' billing volume. To solve the 95th percentile billing problem, Zhan *et al.* [27] proposed a mixed-integer linear programming (MILP) problem. Singh *et al.* [20] transformed percentile cost into K-max problem and developed an efficient online TE framework, CASCARA. Yang *et al.* [26] expressed the problem of network bandwidth allocation as an integer programming model.

However, methods such as [8,10,13,23,24] do not apply to real-time scenarios such as live streaming, because it is unrealistic to delay sending users' data. In addition, [3,7,12,20,26,27] do not take into account the importance of network latency in bandwidth allocation, which is closely related to geographical distribution. For example, to ensure the real-time and QoS, it is not practical to allocate the demands of an edge user in Guangxi to an edge server in Beijing, which has a high network latency.

3 System Model and Problem Formulation

In this section, we first present the system model of the proposed edge bandwidth allocation problem, and then give its corresponding problem formulation. Table 1 has listed the notations frequently used in this study.

Table 1. List of notations

Notation	Description
t_k	k-th time slot in T, also known as checkpoint
T	Set of checkpoints, also known as the billing period
s_i	i-th edge server in S
S	Set of edge servers
m	The number of edge servers
b_i	The bandwidth capacity of s_i
B	The bandwidth capacity of all edge servers
u_j	j-th edge user in U
U	Set of edge users
n	The number of edge users
y_j^k	Bandwidth demands of the u_j at t_k
Y	Bandwidth demands of all edge users in the billing period T
l_{ij}	Network latency between s_i and u_j
L	Network latency matrix between all edge servers and all edge users
τ	The upper limit of network latency the edge servers must meet
x_{ij}^k	The amount of bandwidth allocated by s_i to u_j at t_k
X	Bandwidth allocation matrix during the billing period
w_i^k	Total bandwidth allocated by s_i to all edge users at t_k
W_i	All allocation records of s_i during the billing period
$O_{95}(A)$	The $\lceil 95\% \times \lvert A \rvert \rceil$-th value in A_{sorted}, where A_{sorted} is A sorted in non-decreasing order, and $\lvert A \rvert$ is the number of elements in A
q_i	The billing volume of s_i
p	The unit price of bandwidth provided by edge servers (i.e., USD per Mbps)
C	Total cost during the billing period
Θ	The initial temperature of simulated annealing
Γ	Number of the outer loop of simulated annealing
Φ	Number of the inner loop of simulated annealing
α	Temperature attenuation coefficient of simulated annealing
λ	Frequency of simulated annealing in neighborhood search

3.1 System Model

In the 95th-percentile billing mode, for every employed edge server, the billing period T is usually a month, and the bandwidth usages are sampled every 5 min as shown in Fig. 2. Thus the billing period T can be split into 8640 time slots. Suppose that there are m edge servers available for edge application providers to rent. We use $S = \{s_1, s_2, \cdots, s_m\}$ to denote the set of edge servers, b_i is the maximum bandwidth that the i-th server can provide at each time slot. Since the edge servers in S could be geographically distributed in different locations, they may present different communication latency for the same edge user.

We use $U = \{u_1, u_2, \cdots, u_n\}$ to represent the set of edge users, where u_j is the j-th edge user which refers to a group of end-users located in a certain area. And we use l_{ij} to denote the network latency between edge server s_i and edge user u_j. At time slot t_k, we use y_j^k to denote the bandwidth demands of edge user u_j. To guarantee the QoS, edge user u_j can only connect to edge servers whose communication latency to u_j is less than τ. We use x_{ij}^k to denote the amount of bandwidth allocated by s_i to u_j at t_k. Thus, the total bandwidth allocated by s_i at t_k can be calculated as:

$$w_i^k = \sum_{j=1}^{n} x_{ij}^k \tag{1}$$

In this way, the consumed bandwidth trace can be represented as $W_i = \left\{w_i^1, w_i^2, \cdots, w_i^{|T|}\right\}$. Thus, the billing volume of s_i can be given as:

$$q_i = O_{95}(W_i) \tag{2}$$

Therefore, the total cost of an edge application provider renting edge servers to meet the needs of edge users can be calculated as:

$$C = \sum_{i=1}^{m} q_i \times p \tag{3}$$

3.2 Problem Formulation

Based on the above formulation, the proposed edge bandwidth allocation problem can be formulated as follow:

$$Min : C \tag{4}$$

$$s.t : \sum_{i=1}^{m} x_{ij}^k = y_j^k \tag{5}$$

$$\sum_{j=1}^{n} x_{ij}^k \leq b_i \tag{6}$$

$$l_{ij} < \tau, \quad \forall x_{ij}^k > 0 \tag{7}$$

$$x_{ij}^k \in \mathbb{N}, \quad i \in [1, m], \quad j \in [1, n], \quad k \in [1, |T|]$$

As shown in Eq. (4), the target of this problem is to minimize the total bandwidth overhead of edge application providers. Equation (5) indicates that every edge user's bandwidth demands need to be met, and Eq. (6) is the bandwidth capacity constraint for every edge server. Meanwhile, Eq. (7) is the constraint that implies the available candidate serves for every edge user.

For the 95th-percentile billing optimization problem, Jalaparti *et al.* [11] have proved it is NP-hard. Similarly, for the problem we formulated above, if we relax the 95th-percentile billing mode to the 100th-percentile billing mode (i.e., billing amount is equal to the highest bandwidth usage), the problem discussed above reduces into an integer linear programming, which is well-known to be an NP-hard one. Thus the proposed edge bandwidth allocation problem is also an NP-hard one.

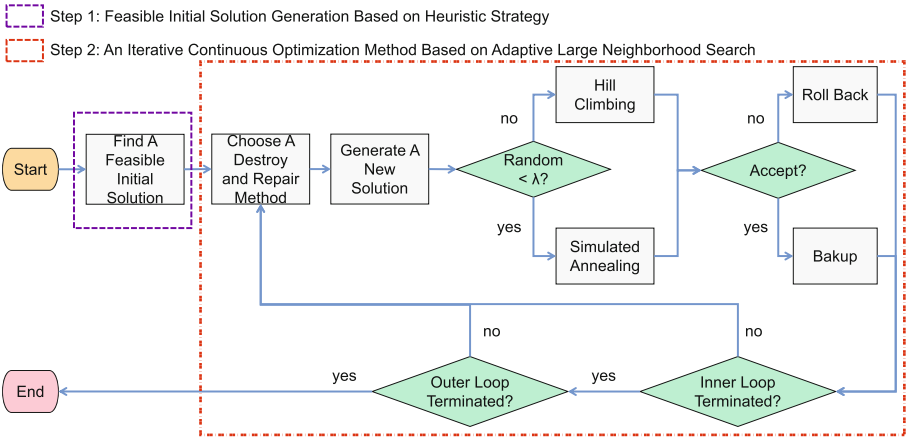

Fig. 3. The process of the proposed EBA approach.

4 Proposed EBA Approch

To address the aforementioned problem, we propose an adaptive large neighborhood search (ALNS) based approach, shorts for EBA. ALNS [18] is a kind of heuristic search method. It brings the adaptive operators (destroy and repair) selection feature to traditional local search, and thus expands the searching space and accelerates the optimization convergence speed. Figure 3 illustrates the process of the proposed EBA approach. It firstly employs a heuristic strategy to find a feasible initial solution quickly. Then, an ALNS-based method, which contains a Hill Climbing (HC) and Simulated Annealing (SA) mechanisms, is utilized to perform iterative optimization.

Algorithm 1: Feasible Initial Solution Generation (FISG)

Input : Billing period T; Edge servers S; Edge users U; Bandwidth capacity B; Bandwidth demands Y; Network latency matrix L; Network latency constraint τ; Unit price of bandwidth p;
Output: Bandwidth allocation matrix X; Total cost C;

1 $\widehat{Y} \leftarrow Y$; $f \leftarrow |T| - \lceil |T| \times 0.95 \rceil$;
2 sort edge servers in ascending order of connectivity;
3 **for** $i \leftarrow 1$ **to** m **do**
4 sort checkpoints in descending order by the sum of remaining bandwidth demands;
5 **for** $k \leftarrow 1$ **to** f **do**
6 $h \leftarrow b_i$;
7 sort edge users in ascending order of connectivity;
8 **for** $j \leftarrow 1$ **to** n **do**
9 **if** $h = 0$ **then** break;
10 **else if** $l_{ij} < \tau$ **then**
11 $x_{ij}^k \leftarrow \min\left(y_j^k, h\right)$; $y_j^k \leftarrow y_j^k - x_{ij}^k$; $h \leftarrow h - x_{ij}^k$;
12 **for** $k \leftarrow 1$ **to** $|T|$ **do**
13 $H \leftarrow \varnothing$;
14 **for** $i \leftarrow 1$ **to** m **do**
15 **if** t_k *is a peak checkpoint of* s_i **then** $h_i \leftarrow b_i$;
16 **else** $h_i \leftarrow$ calculate the billing volume according to Eq. (2);
17 $H \leftarrow H \cup \{h_i\}$;
18 $\widehat{X} \leftarrow \texttt{Dinic}(\widehat{Y}, H, X, k)$;
19 $H \leftarrow B$;
20 $X \leftarrow \texttt{Dinic}(\widehat{Y}, H, \widehat{X}, k)$;
21 $C \leftarrow$ calculate the total cost according to Eq. (3);
22 **return** X, C;

4.1 Feasible Initial Solution Generation

For the first phase of the EBA approach, we develop a network flow-based feasible initial solution generation method, shorts for FISG. As shown in Algorithm 1, FISG has two main steps: 1) Allocating as much bandwidth as possible to each edge server at peak checkpoints. Here we have three sorting strategies: ascending order for edge servers and edge users according to connectivity while descending order for checkpoints according to the sum of remaining demands. These ideas are intuitive, because less connected servers or users are more difficult to meet the conditions later on and peak checkpoints are better suited for high demands (as shown in lines 2–11); 2) For the remaining user demands at each checkpoint, we preset the bandwidth capacity of each server and try to solve it by Dinic,

an efficient network flow algorithm to find the max flow in a graph [6]. If Dinic fails to find a feasible max flow, FISG will increase the bandwidth capacity of servers to the upper limit before trying again (if successful, this step has no impact, as shown in lines 12–20). The time complexity of the algorithm is $O\left(|T|\, nm\, (n+m)^2\right)$.

4.2 ALNS-Based Iterative Continuous Optimization

Obviously, the initial solution obtained by FISG satisfies all constraints defined in Eqs. (5–7). However, its total cost can be further improved by performing load consolidation. Therefore, in the second phase of the proposed EBA approach, we develop an ALNS-based method to carry out the iterative continuous optimization. To avoid falling into local optimum, the proposed method also includes a SA mechanism to explore more feasible solutions.

Algorithm 2: ALNS-based Iterative Continuous Optimization (ICO)

Input : Initial solution X; Initial total cost C; Initial temperature Θ; Number of the outer loop Γ; Number of the inner loop Φ; Temperature attenuation coefficient α; Simulated annealing frequency λ;
Output: Bandwidth allocation matrix X; Total cost C;

1 $\bar{X}, \bar{C} \leftarrow X, C$; `// current best solution`
2 $\widehat{X}, \widehat{C} \leftarrow X, C$; `// backup solution`
3 $\boldsymbol{\rho}^- \leftarrow (1, \cdots, 1)$; $\boldsymbol{\rho}^+ \leftarrow (1, \cdots, 1)$;
4 **for** $e \leftarrow 1$ **to** Γ **do**
5 **for** $g \leftarrow 1$ **to** Φ **do**
6 select destroy method `d()` $\in \Omega^-$ and repair method `r()` $\in \Omega^+$ using $\boldsymbol{\rho}^-$ and $\boldsymbol{\rho}^+$;
7 $s_i \leftarrow$ `d()`; $O \leftarrow$ `r()`; $\delta \leftarrow$ `random(0,1)`; $\psi \leftarrow$ false;
8 **if** $\delta \geq \lambda$ **then**
9 **if** `HCO`$(\Theta, X, C, \widehat{X}, \widehat{C}, s_i, O) = true$ **then** $\psi \leftarrow$ true ;
10 **else**
11 **if** `SAO`$(\Theta, X, C, \widehat{X}, \widehat{C}, s_i, O) = true$ **then** $\psi \leftarrow$ true ;
12 **if** $\psi = true$ **then** $\widehat{X}, \widehat{C} \leftarrow X, C$;
13 **else** $X, C \leftarrow \widehat{X}, \widehat{C}$;
14 **if** $C < \bar{C}$ **then** $\bar{X}, \bar{C} \leftarrow X, C$;
15 update $\boldsymbol{\rho}^-, \boldsymbol{\rho}^+$ according to Eq. (9);
16 $\Theta \leftarrow \Theta \times \alpha$;
17 **return** $\bar{X}, \bar{C}$;

As shown in Algorithm 2, the detailed steps of ICO are shown below:

1) Copy the initial solution to the current best solution and backup solution (for the rollback operation). And then initialize the weights of all destroy and repair methods to 1 (as shown in lines 1–3).

2) After initialization, ICO searches Φ times under the current SA-related temperature Θ. For each inner loop, ICO selects a destroy method and a repair method in a roulette manner. We use λ to determine the proportion of two neighborhood search mechanisms (e.g., HC and SA). Whichever mechanism is chosen, if it performs successfully (i.e., leads to cost reduction), ICO will update the backup solution. If it fails to optimize cost, ICO will roll back from the backup solution. The best solution is constantly updated if a solution with a lower total cost than the current best solution is found. The weights of the destroy and repair methods are dynamically adjusted based on their performance in the search process (as shown in lines 5–15).
3) After a cooling down event occurs (i.e., the inner loop of ICO is finished), ICO goes to step 2) and continues until the termination condition is reached (as shown in lines 4 and 16).

Destroy and repair operators are the key components of ALNS. Here we have designed three ways to destroy and repair the allocation solutions to explore further improvement. Specially, the destroy operators in ICO are to select which server to reduce its cost, i.e., the billing volume. The details of the three destroy methods employed by ICO are as follows:

1. Select a server randomly.
2. Select a server that has recently successfully updated the current solution.
3. Select a server with the highest billing volume.

After the destroy method has selected a server s_i, we need to reallocate the bandwidth allocated to s_i at some checkpoints to other servers. We can modify the order of building the edges in the Dinic algorithm to determine which servers take on first. The details of the three employed repair methods are as follows:

1. The priority of the servers is sorted randomly.
2. The priority of the servers is sorted by connectivity.
3. The priority of the servers is the order in which data files are read.

For each destroy and repair method, we maintain a count η and a score Ψ, which are initialized to 0. For each selected method, η is increased by 1, and the amount of Ψ increased is based on the following rules [17]:

$$\Psi = \Psi + \max \begin{cases} \omega_1 & \text{if } C < \bar{C} \\ \omega_2 & \text{if } C < \widehat{C} \\ \omega_3 & \text{if the new solution is accept} \\ \omega_4 & \text{if the new solution is rejected} \end{cases} \tag{8}$$

Given a weight attenuation coefficient $\sigma \in [0, 1]$, the weight of each method can be updated as follows:

$$\rho = \begin{cases} \sigma \times \rho & \text{if } \eta = 0 \\ \sigma \times \rho + (1 - \sigma) \times \dfrac{\Psi}{\eta} & \text{if } \eta > 0 \end{cases} \tag{9}$$

Algorithm 3: Hill Climbing-based Optimization (HCO)

Input : Current temperature Θ; Current solution X; Current total cost C; Backup solution $\widehat{X}$; Backup total cost $\widehat{C}$; Selected server s_i; Servers' priority list O;

Output: Whether the optimization is successful

1 $step \leftarrow$ `random(1, Θ)`; $f \leftarrow 0$; $G \leftarrow \varnothing$;
2 $q_i \leftarrow$ calculate the billing volume according to Eq. (2);
3 **for** $k \leftarrow 1$ **to** $|T|$ **do**
4 **if** $\sum_{j=1}^{n} x_{ij}^{k} = q_i$ **then**
5 $G \leftarrow G \cup \{t_k\}$;
6 **if** t_k *is not a peak checkpoint of* s_i **then** $f \leftarrow f + 1$;
7 **while** $step > 0$ **do**
8 $cnt \leftarrow 0$;
9 **foreach** $t_k \in G$ **do**
10 **if** $\sum_{j=1}^{n} x_{ij}^{k} \geq step$ **then**
11 $H \leftarrow \varnothing$;
12 **for** $e \leftarrow 1$ **to** m **do**
13 **if** t_k *is a peak checkpoint of* s_e **then** $h_e \leftarrow b_e$;
14 **else** $h_e \leftarrow$ calculate the billing volume according to Eq. (2);
15 $H \leftarrow H \cup \{h_e\}$;
16 **if** `OptDinic(`$H, X, s_i, O, step, k$`)` **then**
17 $cnt \leftarrow cnt + 1$;
18 **if** $cnt = f$ **then**
19 **return** true;
20 $X, C \leftarrow \widehat{X}, \widehat{C}$; $step \leftarrow \frac{step}{2}$;
21 **return** false;

Given a specific edge server s_i, the HCO aims to reduce the billing volume by at least 1Mbps without increasing other edge servers' costs. As shown in Algorithm 3, the HCO includes two main steps: 1) Find all checkpoints of s_i such that the load at that checkpoint is equal to the billing volume, and calculate the number of checkpoints that need to be lowered (as shown in lines 1–6). 2) Try to reduce the billing volume by *step*. If it fails, halve the

current *step* before trying again. The OptDinic proposed by us is used to verify whether it is feasible (as shown in lines 7–20). The time complexity of the HCO is $O\left(|T|\, nm\, (n+m)^2 \log \Theta\right)$.

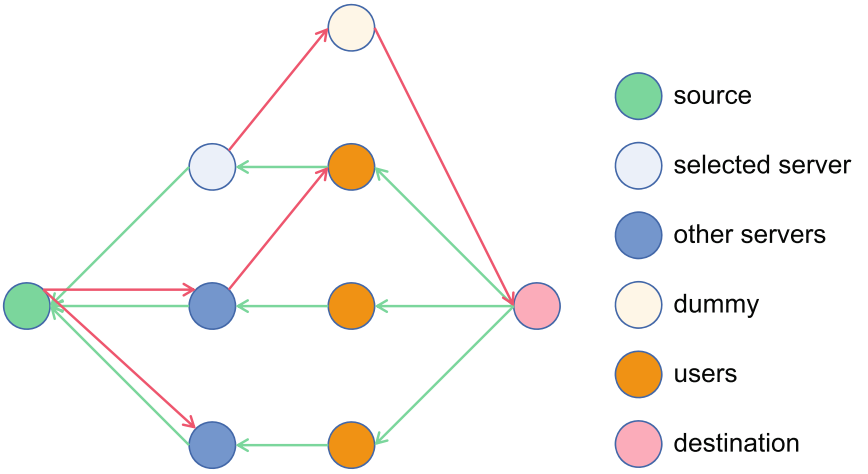

Fig. 4. Residual network of OptDinic. (Color figure online)

To verify the feasibility of the current solution, OptDinic is proposed based on the original Dinic algorithm [6]. As shown in Fig. 4, there are six kinds of nodes and two kinds of edges. The green edges represent the bandwidth that has been allocated, i.e., the current solution at one checkpoint. In contrast, the red edges denote the unallocated bandwidth. To reduce the actual load of the selected server by *step* at the checkpoint, we add the dummy node and let its bandwidth demands equal to *step*. It is worth noting that there is no unallocated bandwidth from the source to the selected server, while the dummy node is only connected to the selected server but not other servers. Therefore, when the bandwidth traffic from the source to the destination increases by *step*, it must be that the selected server allocates the *step* size bandwidth to the dummy node, and other servers increase the *step* size altogether to meet the users' demands. As for which server tries to increase its load first, we can control the order of edge building through the servers' priority.

Given a specific edge server s_i, the SAO aims to select a peak checkpoint to reduce its load at that checkpoint to below the billing volume. As shown in Algorithm 4, the SAO involves two main steps: 1) Try to reduce the load by *step* at a peak checkpoint without increasing the others' cost. If the current checkpoint t_k is the peak checkpoint of a server, its bandwidth capacity is defaulted to the upper limit. Otherwise, it will be set to the billing volume. Here we also use OptDinic to verify the feasibility of the optimized solution (as shown in lines 2–9). 2)Increase the bandwidth capacity of servers to the

Algorithm 4: Simulated Annealing-based Optimization (SAO)

Input : the same as HCO
Output: Whether the optimization is successful

1 **foreach** $t_k \in$ *peak checkpoints of* s_i **do**
2 $q_i \leftarrow$ calculate the billing volume according to Eq. (2);
3 $step \leftarrow \sum_{j=1}^{n} x_{ij}^{k} - \max(q_i - 1, 0)$; $H \leftarrow \varnothing$;
4 **for** $e \leftarrow 1$ **to** m **do**
5 **if** t_k *is a peak checkpoint of* s_e **then** $h_e \leftarrow b_e$;
6 **else** $h_e \leftarrow$ calculate the billing volume according to Eq. (2);
7 $H \leftarrow H \cup \{h_e\}$;
8 **if** `OptDinic`$(H, X, s_i, O, step, k)$ **then**
9 **return** true
10 $H \leftarrow B$; $\delta \leftarrow$ `random`$(0, 1)$;
11 **if** `OptDinic`$(H, X, s_i, O, step, k)$ *and* $\delta < \exp\left(\frac{\widehat{C}-C}{\Theta}\right)$ **then**
12 **return** true
13 **else** $X, C \leftarrow \widehat{X}, \widehat{C}$;
14 **return** false;

upper limit before trying again. If the total cost rises, accept it with a certain probability. If it fails, roll back from the backup solution and try the next peak checkpoint (as shown in lines 10–13). The time complexity of the SAO is $O\left(|T|\, nm\, (n+m)^2\right)$. Therefore, the total time complexity of proposed EBA approach is $O\left(\Gamma\Phi\, |T|\, nm\, (n+m)^2 \log\Theta\right)$.

5 Experiments and Analysis

In this section, we conducted a series of experiments based on a real-world edge bandwidth load dataset to verify the effectiveness of the proposed approach.

5.1 Experiment Settings

For a real-world short video and live streaming application, we have tracked the bandwidth demands within a monthly billing period. Here edge user refers to a group of end-users located in a certain area (i.e., end-users in the same province), and its bandwidth demands are the aggregation of all individual end-users in that group at a certain time point. Figure 5 shows the overall bandwidth demands of edge users. It can be seen that the demands present a noticeable periodical change, the period is one month, and the highest whole network bandwidth demand is about 4 Tbps. In our experiment, we consider 110 edge servers distributed among different provinces in China, in which the bandwidth capacity of edge servers follows the normal distribution with 512 Gbps bandwidth expectation, and the network latency between edge servers and edge users is set to be

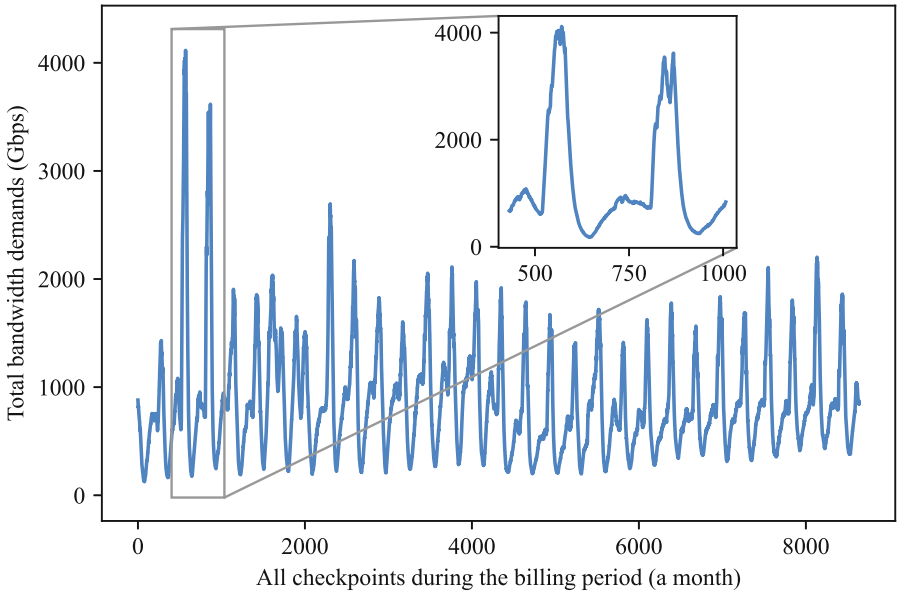

Fig. 5. Bandwidth demands during the billing period.

uniformly distributed from 140 ms to 540 ms. The bandwidth price is obtained from the latest Aliyun ECS price table [1].

Some hyperparameter settings during the experiment are shown in Table 2.

Table 2. Parameter setting

Hyperparameter	Value
Θ	8192
Γ	400
Φ	50
α	0.95
λ	0.25
ω_1	100
ω_2	10
ω_3	1
ω_4	0

We compare EBA against the following baseline algorithms:

- Load Balance (LB): For each edge user at every checkpoint during the billing period, the LB heuristic randomly selects d feasible edge servers. Note that, those servers with remaining bandwidth capacity at peak checkpoints will be preferentially selected. And then, edge users' bandwidth demands are allocated to the selected server with the lowest load.
- Round robin (RR): For each edge user at every checkpoint during the billing period, the target server is rotated by a polling strategy. If the selected server does not have enough capacity, the remaining demands will be allocated to other servers in the same way of polling.
- Equal Split (ES): For each edge user at every checkpoint during the billing period, the servers are sorted in ascending order in terms of capacity. And in this order, the amount of bandwidth allocated to the current server is less than the remaining capacity and the average remaining demands.
- FISG: Our proposed network flow-based approach described in Sect. 4.1.
- EBA-ALNS: Our proposed EBA approach without the ALNS module. Specifically, the destroy and repair method always choose the random strategy.
- EBA-SA: Our proposed EBA approach without the SA mechanism.

5.2 Performance Comparison

Results Under Different Demands Constraints: In Fig. 6, we evaluate the performance of the proposed EBA approach and its peers in terms of bandwidth monetary costs. It can be seen that LB, ES, and RR algorithms achieve the top 3 highest cost. The FISG algorithm we proposed obviously outperforms them, whose cost is only 19.82% of the RR algorithm with the scaling of demands on average, and 22.27% of the RR algorithm with the scaling of edge users on average. We also learn from the Fig. 6 that compared with the FISG, the proposed EBA approach is capable of reducing by 37.45% and 35.31% of the total cost on average. Besides, in most cases of Fig. 6(b), the total cost shows a rising trend regardless of whether the ALNS or SA are removed. In addition, with the increase in demands and the number of edge users, the total cost tends to rise in all cases.

Analysis: Because the LB algorithm randomly selects servers each time, the billing volume of each server is relatively high. The cost of the ES algorithm is higher than that of the RR algorithm because the former tends to employ more servers to fulfill each user's demands. While our FISG algorithm takes into account three intuitive sorting strategies in the first phase and gradually updates the billing volume of the servers in the second phase. In this way, each server is fully utilized in both peak and non-peak time slots. Compared with the native local search method, the ALNS-based approach can expand the neighborhood

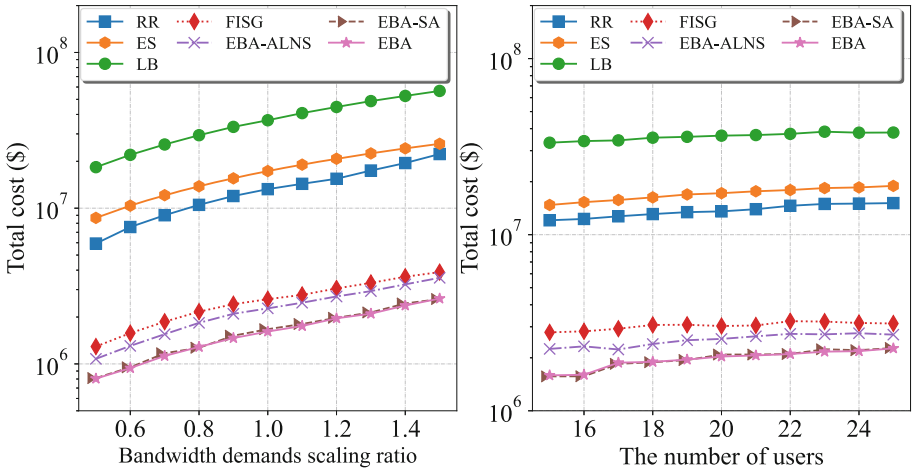

Fig. 6. (a) Comparison of the total cost under different bandwidth demands. (b) Comparison of the total cost under different numbers of users.

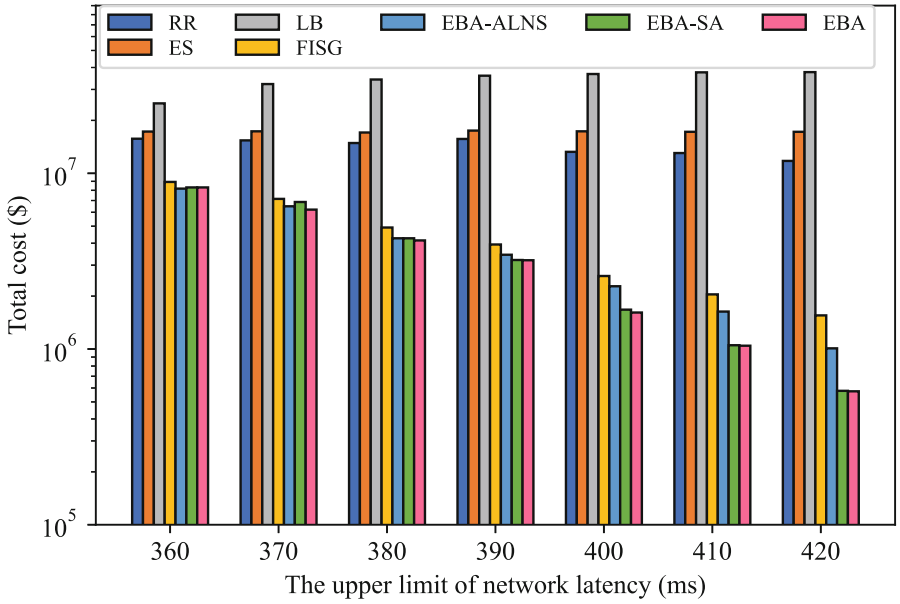

Fig. 7. Comparison of the total cost under different latency constraints.

search range, and SA can jump out of the local optimum. That is why the proposed EBA outperforms other baselines in all experimental cases.

Results Under Different Latency Constraints: In Fig. 7, we compare the total cost under different latency constraints. It can be seen that LB, ES, and

RR are still the three algorithms with the highest total cost. Moreover, our EBA achieves the lowest cost. In addition, with the relaxation of the latency constraint, our proposed FISG algorithm and EBA algorithm show a gradual increasing advantage over other baselines. When the latency constraint is set to 360 ms, the total cost of the FISG algorithm is 56.78% of the RR algorithm, and the EBA reduces by 7.15% compared to FISG. However, when the latency constraint is set to 420 ms, the total cost of FISG is 13.18% of RR, and the EBA reduces by 63.02% compared to FISG.

Analysis: Our FISG algorithm is improving because the number of servers available increases when the latency constraint is relaxed. Besides, the importance of taking full advantage of 95th-percentile billing becomes apparent and helps a lot. When the number of available servers increases, it also means that more neighborhood solutions meet the conditions for ALNS-based approaches, which result in a high success rate of optimization and thus improve the quality of yield solutions. Therefore, the total cost of the EBA approach drops more than other baselines with the increase in latency constraints.

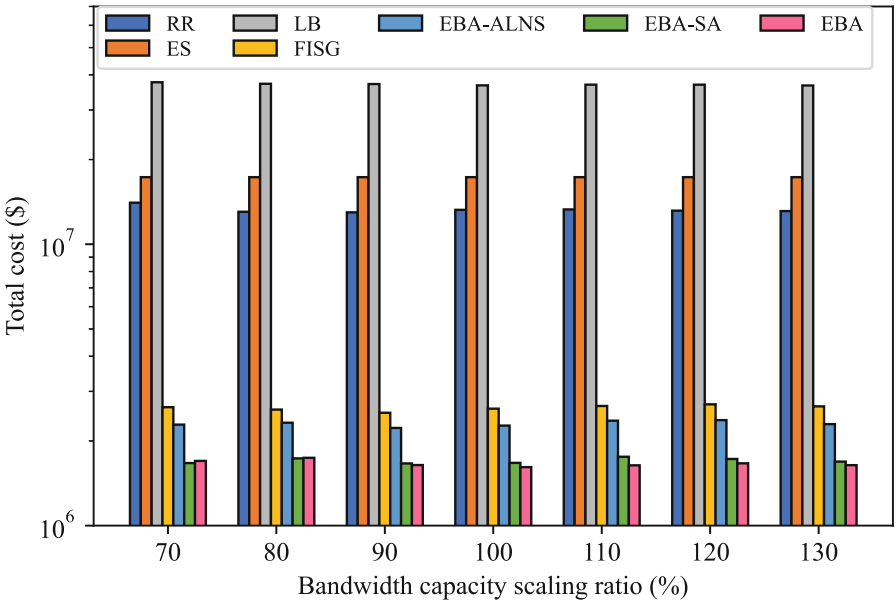

Fig. 8. Comparison of the total cost under different bandwidth capacities.

Results Under Different Capacity Constraints: Figure 8 compares the total cost under different bandwidth capacities. The top four algorithms with the highest total cost are LB, ES, RR, and FISG. The total cost of EBA-ALNS decreased by 12.16% based on the FISG algorithm on average, while EBA-SA

decreased by 35.11% on average. EBA achieves the lowest cost, and the total cost is reduced by 36.57% on average compared to the FISG algorithm.

Analysis: Due to the large search space resulting from the complex 95th-percentile optimization problem, it is difficult for a single search strategy to find an appropriate descent route. In addition, the SA mechanism has a certain probability of accepting worse solutions during the search. Thus it is capable of jumping out of the local optimum. However, restricted by a single destroy and repair operator, SA fails to improve the quality of the solution further. In contrast, the ALNS-based approach expands the optimization range by introducing multiple operators and an adaptive switching mechanism.

The above experimental results also show that our proposed FISG can obtain high-quality initial solutions. No matter how to scale users' bandwidth demands, the number of users, latency constraints, or servers' bandwidth capacity, the proposed EBA approach always achieves considerable optimization based on the initial solution. We also verified the effectiveness of ALNS and SA. Since the hill climbing mechanism does not lead to a cost increase, it is not included in the ablation experiment.

6 Conclusion and Future Work

In this paper, we proposed an approach to the edge bandwidth allocation problem based on 95th-percentile billing in edge computing, shorts for EBA, and considered the geographical heterogeneity of network latency between edge users and edge servers. The proposed EBA includes a heuristic-based method for feasible initial solution generation and an iterative optimization method based on adaptive large neighborhood search. Hill climbing and simulated annealing mechanisms were utilized to achieve continual optimization and jumping out of the local optimum. Experiments based on a real-world edge application bandwidth demand tracking dataset illustrated the effectiveness of our approach.

For our future work, the online edge bandwidth allocation problem will be investigated and studied. We will consider using machine learning methods to predict future edge bandwidth demands. At each checkpoint, the EBA approach proposed in this paper is used to obtain the solution according to the current state, and reinforcement learning can be employed to learn the allocation strategy.

Acknowledgements. This work was supported in part by: National Natural Science Foundation of China (Nos. 61966008, U2033213), Sichuan and Chongqing Joint Key R&D Project (No. 2021YFQ0058).

References

1. Aliyun ECS pricing-calculator. https://www.aliyun.com/pricing-calculator?#/commodity/vm. Accessed 12 June 2022

2. Predicts 2022: The distributed enterprise drives computing to the edge. https://www.gartner.com/en/documents/4007176. Accessed 12 June 2022
3. Adler, M., Sitaraman, R.K., Venkataramani, H.: Algorithms for optimizing the bandwidth cost of content delivery. Comput. Netw. **55**(18), 4007–4020 (2011)
4. Dimitropoulos, X., Hurley, P., Kind, A., Stoecklin, M.P.: On the 95-percentile billing method. In: Moon, S.B., Teixeira, R., Uhlig, S. (eds.) PAM 2009. LNCS, vol. 5448, pp. 207–216. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00975-4_21
5. Ding, J.W., Deng, D.J., Wu, T.Y., Chen, H.H.: Quality-aware bandwidth allocation for scalable on-demand streaming in wireless networks. IEEE J. Sel. Areas Commun. **28**(3), 366–376 (2010)
6. Dinic, E.: Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math. Doll. **11**(5), 1277–1280 (1970). English translation by RF. Rinehart (1970)
7. Goldenberg, D.K., Qiuy, L., Xie, H., Yang, Y.R., Zhang, Y.: Optimizing cost and performance for multihoming. ACM SIGCOMM Comput. Commun. Rev. **34**(4), 79–92 (2004)
8. Golubchik, L., Khuller, S., Mukherjee, K., Yao, Y.: To send or not to send: reducing the cost of data transmission. In: 2013 Proceedings IEEE INFOCOM, pp. 2472–2478. IEEE (2013)
9. Hong, C.H., Varghese, B.: Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. ACM Comput. Surv. (CSUR) **52**(5), 1–37 (2019)
10. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network. In: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 145–158 (2004)
11. Jalaparti, V., Bliznets, I., Kandula, S., Lucier, B., Menache, I.: Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In: Proceedings of the 2016 ACM SIGCOMM Conference, pp. 73–86 (2016)
12. Khare, V., Zhang, B.: CDN request routing to reduce network access cost. In: 37th Annual IEEE Conference on Local Computer Networks, pp. 610–617. IEEE (2012)
13. Laoutaris, N., Smaragdakis, G., Stanojevic, R., Rodriguez, P., Sundaram, R.: Delay-tolerant bulk data transfers on the internet. IEEE/ACM Trans. Netw. **21**(6), 1852–1865 (2013)
14. Liyanage, M., Porambage, P., Ding, A.Y., Kalla, A.: Driving forces for multi-access edge computing (MEC) IoT integration in 5G. ICT Express **7**(2), 127–137 (2021)
15. Mukerjee, M.K., Naylor, D., Jiang, J., Han, D., Seshan, S., Zhang, H.: Practical, real-time centralized control for CDN-based live video delivery. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 311–324 (2015)
16. Peng, Q., et al.: A decentralized collaborative approach to online edge user allocation in edge computing environments. In: 2020 IEEE International Conference on Web Services (ICWS), pp. 294–301. IEEE (2020)
17. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 399–419. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1665-5_13
18. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
19. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)

20. Singh, R., Agarwal, S., Calder, M., Bahl, P.: Cost-effective cloud edge traffic engineering with cascara. In: 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2021), pp. 201–216 (2021)
21. Stanojevic, R., Laoutaris, N., Rodriguez, P.: On economic heavy hitters: shapley value analysis of 95th-percentile pricing. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 75–80 (2010)
22. Tseng, C.W., Tseng, F.H., Yang, Y.T., Liu, C.C., Chou, L.D.: Task scheduling for edge computing with agile VNFs on-demand service model toward 5G and beyond. Wirel. Commun. Mob. Comput. **2018** (2018)
23. Wang, J.: Traffic regulation under the percentile-based pricing policy. In: Proceedings of the 1st International Conference on Scalable Information Systems, pp. 4-es (2006)
24. Wang, J., Chen, J., Yang, M., Zheng, S.: Traffic regulation with single-and dual-homed ISPS under a percentile-based pricing policy. J. Comb. Optim. **17**(3), 247–273 (2009)
25. Wu, C., et al.: Online user allocation in mobile edge computing environments: a decentralized reactive approach. J. Syst. Architect. **113**, 101904 (2021)
26. Yang, C., You, J., Yuan, X., Zhao, P.: Network bandwidth allocation problem for cloud computing. arXiv preprint arXiv:2203.06725 (2022)
27. Zhan, Y., Ghamkhari, M., Akhavan-Hejazi, H., Xu, D., Mohsenian-Rad, H.: Optimal response to burstable billing under demand uncertainty. arXiv preprint arXiv:1603.05752 (2016)