Optimal edge gateway deployment in internet of things based on simulated annealing with adaptive external penalty

Xiaorong Zhu1 | Fang Xiao1 | Yue Wang1 | Yong Wang2

1 Nanjing University of Posts & Telecommunications, Nanjing, China
2 Nanjing Vocational University of Industry Technology, Nanjing, China

Correspondence
Xiaorong Zhu and Fang Xiao, Xin Mo Fan Road No.66, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210003, PR. China.
Email: xrzhu@njupt.edu.cn, 1184483504@qq.com
Yong Wang, is with the Nanjing Vocational University of Industry Technology, Nanjing, China.
Email: 2018100906@niit.edu.cn

Funding information
Natural Science Foundation of China, Grant/Award Numbers: 61871237, 92067101; Program to Cultivate Middle-aged and Young Science Leaders of Universities of Jiangsu Province and Key R&D plan of Jiangsu Province, Grant/Award Number: BE2019017; Jiangsu Agriculture Science and Technology Innovation Fund, Grant/Award Number: CX(19)3008

Abstract
In the large scale Internet of things, edge gateway (EG) deployment is used to find the minimal number of gateways required in the network and their optimal locations under the design constraints to meet different service requirements, which is one significant issue for improving network performances. We formulate the EG deployment problem as a k-median problem with some constraints, which is known as a NP-hard problem. For that, we propose a new heuristic based on simulated annealing with external penalty function (SA-AEP) for its solution. The external penalty function is used to transform the multi-constrained optimization problem into a single constraint one. Also the complexity of proposed heuristic algorithm is analyzed. In addition, in order to evaluate the proposed algorithm, we compare it with an adaptive variable-length particle swarm optimization algorithm with varying lengths. Simulation results show that the proposed SA-AEP algorithm has much better performances on network cost and efficiency than other algorithms when the higher service rate is required.

1 | INTRODUCTION

Smart Internet of Things (IoT) applications such as intelligent manufacturing and autonomous driving, military and smart city are emerging whereby machine-, terminal-, and human-intelligence are integrated to form a large-scale machine-object fusion system [1]. A typical IoT system includes the cloud or edge cloud platform, numerous IoT terminals and edge gateways (EGs). The IoT has a variety of network connection technologies, such as NB-IoT, LoRa, 5G etc. The IoT platform shields the differences in connections for IoT applications, which will also open up network capabilities and some basic capabilities, such as big data analysis and location-based service capabilities. The IoT terminals generate data and can communicate with the network in two ways, directly through NB-IoT or traditional cellular networks, or indirectly through EGs, using short-range wireless technologies such as ZigBee, WiFi, Bluetooth, and RFID. In some industrial applications, wired connections may also be used.

The EG in IoT network near the users is deployed to implement routing, communication and security protocol functions to collect sensor information or perform on-site actions, and connect to the network through standard cellular technology. In addition, the platform may also put most of the computing volume at the EG, so that the decoupling between different IoT terminals, coupled with SDN control, enables the Internet of everything. The EG may adopt virtual container or microserver technology, which can be deployed the corresponding SDK/Service according to different service requirements, and has dynamic support for different protocols and heterogeneity access network functions. Also as a computing unit, it can provide scalable, dynamically deployable IoT terminal localization management and real-time data analytics, processing and storage capabilities, making mobile service deployment easier and more accessible to a wider range of nodes. Therefore, it can
provide differentiated services such as massive terminal access, very low latency and ultra-bandwidth etc. for different types of typical applications including intelligent manufacturing and autonomous driving, military and smart city.

As the number of IoT terminals grows exponentially and the demand for different types of services increases, the effective and adaptive EG deployment will become more and more important for improving network performance and also can contribute to achieve smart IoT. The deployment of a container-based application over a geo-distributed computing infrastructure is a key task that has a significant impact on the application non-functional requirements (e.g. performance, security, cost) [2]. The optimal EG deployment contributes to find out the minimal number of EGs needed in the network and their optimal locations they should be placed. The network throughput, network topology and traffic distribution can be enhanced and simultaneously the cost may be decreased if EGs are well placed. Therefore, it has attracted much more and more attention [4–17].

Therefore, in this paper, we propose an adaptive EG deployment model. Our contributions are as follows:

- We formulate the EG deployment as a cost optimization problem with coverage constraint, capacity constraint, and diverse service requirements of terminals.
- We transform the delay and capacity constraints into an adaptive external penalty (AEP) function for inclusion into the objective function. The new simulated annealing algorithm can have simple constraints for the complex optimization problem. Also, a complexity analysis is given.
- We use simulation to perform a thorough comparison of the proposed algorithm with other algorithms. We demonstrate that the proposed algorithm has superior performances.

The remainder of this article is organized as follows. Section II gives the related work. Section III introduces the system model and the formulation of edge gateway deployment problem. Section IV and V present the improved simulated annealing algorithm and particle swarm optimization algorithm, respectively. Section VI shows the implementation and evaluation results. Finally, Section VII concludes this article and discusses future works.

2 | RELATED LITERATURES

EG deployment problem is usually to calculate the total number of gateways required in the network and the locations where they should be placed under the constraints designed to meet the different requirements of given IoT terminals. The most common design constraints are delay, backhaul capability and load balance. The delay constraint comes from the quality of service (QoS) requirements, which limits the maximum number of hops, thereby limiting the coverage radius. The limitation of the backhaul capability is caused by the wireless rate limitation provided by the backhaul link. Furthermore, the load of each wireless node should not exceed its capacity, which is determined by the transmission rate and computing capability. In [3], the authors provide a systematic literature review on the resource management approaches in fog environment, which include six main fields: application placement, resource scheduling, task offloading, load balancing, resource allocation, and resource provisioning. Nowadays, it has become an important research topic and many literatures have studied the deployment of edge nodes including wireless EGs [4–10] and edge computing servers [11–17].

In [4], the authors considered the gateway deployment and low-end transmission equipment allocation problem and proposed an optimal integer linear programming formulation and a heuristic algorithm to minimize the IoT network installation cost. This paper only considers the communication capability of gateway without other capabilities such as computation and storage. [5] modeled the wireless gateway deployment as a multi-objective linear programming problem and proposed a two-stage load balancing deployment algorithm, which balances the connection to the gateway while selecting it. However, this algorithm does not consider the number of deployed gateways and route hops. Reference [6] carried out the gateway deployment with the aim of optimizing the network connection and coverage rate, and proposed a new gateway deployment method to solve the bottleneck problem of the wireless mesh network on the gateway and optimize the network performance. However, in large scale networks, the cost will greatly increase with the increase in the number of gateways that maintain high connectivity and coverage. Also the authors in [7] proposed a gateway deployment method based on cross-layer throughput optimization to maximize the throughput in multi-hop wireless mesh networks while ensuring fairness among all nodes. However, this method does not take into account the number of routers connected to a specific gateway and hence the balance of the number of routers per gateway is ignored. [8] conducted the gateway deployment for the purpose of minimizing cost, modeled the gateway deployment problem as an integer linear programming problem considering QoS constraints, and used a new heuristic algorithm to solve it, but this method only focuses on cost minimization and does not improve gateway performance. Reference [9] deployed gateways with the goal of optimizing QoS and communication delay, and proposed a clustering algorithm to solve the problem of gateway deployment, but the algorithm does not comprehensively consider the global network information and hence generated clusters are not optimal. [10] implemented an intelligent hybrid system for node placement problem in wireless mesh network by particle swarm optimization, hill climbing and simulated annealing, which mainly consider the network connectivity requirement.

In addition, some literatures have studied the deployment of edge computing servers for IoT applications [11–17]. Reference [11] addressed the deployment problem of Virtual Machine Replica Copies (VRCs) that support multiple applications
among multiple mobile edge computing (MEC) servers in the edge network. In the case of limited MEC server capacity and different application resource requirements, a delay-aware heuristic deployment algorithm by optimizing the deployment of VRGs to minimize the average response delay is proposed. The authors in [12] proposed a framework for placing edge servers, which comprehensively considers traffic constraints, fault tolerance and cost constraints through a simple optimization model, thereby significantly improving deployment efficiency and reducing the cost of edge configuration. [13] modeled the deployment of fog nodes in a 5G use case scenario as a mixed integer linear programming problem. While considering the requirements of 5G mobile networks, the locations of fog nodes are optimized to reduce service access delays. In this model, the distance between the fog node and the terminal to represents the delay constraint, which is not reasonable enough. [14] studied the deployment of an intelligent computing system consisting of cloud center, gateways, fog terminals, edge terminals, and sensors connected to a logistics center facility. An integer programming model is established for this problem, which considers the capacity requirements of gateway, fog and edge terminals and the maximum delay, and installation cost. And the discrete monkey population algorithm and genetic algorithm are combined to solve this NP-hard facility positioning problem, which improves the performance of intelligent computing systems in medium-sized intelligent logistics centers.

In [15], the authors formulated the edge server placement problem as a multi-objective constraint optimization problem with the objective to make balance the workloads of edge servers and minimize the access delay between the mobile user and edge server. In this model, it is not reasonable that the edge server access delay is assumed to be proportional to the distance between the base station and edge server. [16] studied the minimum gateway deployment in 802.15.4-based wireless sensor networks with NB-IoT, which considers the reliability, end-to-end latency, and network lifetime. But it does not consider the coverage capability of gateway. In [17], a general model to support QoS-aware deployment is given, where the IoT application is deployed across fog infrastructure. Here the fog infrastructure includes edge and cloud environments. However, it only gives the abstraction model and does not address how to get the optimal deployment solution and how to perform the physical deployment.

Hence, it is seen that the existing literatures have considered a few parameters for EG deployment optimization and the capital expenditure (CAPEX) and operational expenditure (OPEX) of EGs are separately considered for cost optimization. We therefore propose an optimal EG deployment method for smart IoT to find out the number and locations of deployed EGs can be optimized to minimize the deployment cost under the constraints of different QoS requirements. Furthermore, in contrast to the previous works, we consider the different EGs have different computation and communication capabilities. In particular, we propose a novel heuristic algorithm to resolve the deployment problem and also make complexity analysis of the proposed algorithm and provide a thorough performance evaluation.

3 SYSTEM MODEL AND EG DEPLOYMENT FORMULATION

We assume the system architecture is shown in Figure 1, where the IoT terminals may generate many computationally intensive and delay-sensitive application requests such as object recognition, blockchain consensus etc., and they need to offload the computing tasks to nearby EGs by WiFi to reduce response delay. Then EGs may access the edge clouds or clouds by 4G/5G/Internet or other broadband access technology. We model the IoT terminals as traffic generators and consider edge-to-end collaboration, that is, terminals may send the generated data or tasks to EGs for processing. In this paper, comprehensively considering the coverage, capacity, and delay constraints, the number and locations of deployed EGs can be optimized to minimize the deployment cost, improve the resource utilization of EGs, and simultaneously achieve their load balancing.

Let \( T \) be the set of terminals and \( E \) be the set of possible EG locations. Furthermore, let \( |T| \) and \( |E| \) be the number of elements in sets \( T \) and \( E \) respectively. Under the constraints of delay, coverage and capacity, we need to find the optimal locations of EGs from the set of candidate positions \( E \) so that the overall cost is minimum. Let binary variable \( \varepsilon_e = 1 \) denote a gateway is deployed in location \( e \) and \( \varepsilon_e = 0 \) denote otherwise.

During the deployment of EGs, we must ensure that each device is covered by at least one gateway. We use a binary variable \( C_{j|e} \) to indicate whether terminal \( j \) is covered by gateway \( e \). If terminal \( j \) is covered by edge node \( e \), then \( C_{j|e} = 1 \), otherwise \( C_{j|e} = 0 \), where \( \forall j \in T, \forall e \in E \). We denote the distance between EG \( e \) and terminal \( j \) as \( d_{j|e} \), and denote the coverage radius of EG \( e \) as \( r \). Therefore, \( C_{j|e} = 1 \) if and only if \( d_{j|e} \leq r \). In addition, to conserve resources, we require each terminal is affiliated with only one EG. The definition of other symbols are shown in Table 1.

In this paper, we try to minimize cost by reducing the number of deployed EGs while satisfying coverage, latency, and capacity constraints.

![FIGURE 1 System architecture](image.png)
TABLE 1 Symbol definition

| Symbol | Meaning |
|--------|---------|
| T      | The set of terminals |
| E      | The set of candidate EGs |
| T      | Traffic generator of a device |
| e      | Location of a certain candidate EG |
| ∊      | Whether EG is deployed or not |
| Tje   | Whether terminal j is covered by EG e or not |
| Ije   | Whether terminal j is served by EG e or not |
| dje   | Distance between EG e and terminal j |
| r      | Coverage radius of EG |
| Rj     | Uplink data rate for offloading processing tasks from terminal to EG |
| W      | Channel bandwidth |
| Pj     | Transmit power of terminal j |
| Hje   | Channel gain between terminal j and EG e |
| N0     | Background noise power |
| Dj     | The size of the input data for processing |
| Cj     | CPU cycles required to complete the calculation task of terminal j |
| ttra   | Transmission delay for offloading task from terminal to EG |
| fe     | CPU cycles per second |
| texe   | Execution time of the task offloaded by terminal j on EG e |
| Tmax   | The maximum delay that terminal t with service type i can tolerate |

3.1 Coverage constraint

We let r denote the coverage radius of EG e and it may cover many terminals. And each terminal j may also be within the coverage of multiple EGs at the same time, but each terminal can only be served by one EG. Here we use a binary variable Tje to indicate whether terminal j is covered by EG e and that is

\[ Tje = \begin{cases} 1, & \text{if } dje \leq r \\ 0, & \text{if } dje > r \end{cases} \]  

(1)

Also we use a binary variable Ije to represent whether terminal t is served by EG e, that is, if terminal j is served by EG e, then Ije = 1, otherwise Ije = 0.

In order to ensure unique connectivity, each terminal must meet the following two coverage conditions:

\[ \sum_{\forall e \in E} Tje \geq 1, \forall j \in T, \]  

(2)

\[ \sum_{\forall e \in E} Ije = 1, \forall j \in T \]  

(3)

In addition, the premise that terminal t is served by EG e is that terminal j is covered by EG e, and the premise that terminal j is covered by EG e is that it is deployed. Therefore, the binary variables Ije, Tje and ∊, must meet the following conditions:

\[ Ije \leq Tje \leq \varepsilon, \forall j \in T, \forall e \in E, \]  

(4)

and

\[ Ije, Tje, \varepsilon \in \{0, 1\}, \forall j \in T, \forall e \in E. \]  

(5)

3.2 Delay constraints

The deployment of EGs mainly affects the number of serving terminals and load balancing, thereby affecting the offload delay of processing tasks, which includes the transmission delay of tasks from the terminal to EG, the queuing delay and the processing delay of EG.

The uplink data rate of terminal j offloading the processing task from the local to EG over the wireless channel can be written as

\[ Rj = W \log_2 \left( 1 + \frac{Pj Hje}{N0 + \sum_{q \in T \setminus j} Pq Hqe} \right), \]  

(6)

where W is the channel bandwidth, Pj is the transmission power of terminal j, Hje is the channel gain between terminal j and EG e, and N0 is the background noise power.

We model the computation task of terminal j as Mj = (Dj, Cj) that it can offload to EG, where Dj represents the size of the input data to compute, and Cj represents CPU cycles required by completing the processing task of terminal j. The transmission delay of terminal j uploading the data with size Dj to EG e by wireless link is:

\[ ttrans = \frac{Dj}{Rj}. \]  

(7)

If we assume the average service arrival rate, average service rate and service density of EG e are \( \lambda, \mu_e \) and \( \rho_e \), respectively, the average queuing delay of terminal t served by EG e can be expressed as

\[ tque = \frac{\rho_e}{1 - \rho_e} \times \frac{1}{\mu_e}. \]  

(8)

Here we define the computing power of EG e as \( f_e \), that is, the CPU cycle per second. EG will perform the processing tasks Mj = (Dj, Cj). Therefore, the execution time of the task generated by terminal j on EG e is calculated as follows:

\[ texe = \frac{Cj}{fe}. \]  

(9)

Since the size of the computation result is generally much smaller than the size of the raw input data, it is reasonable to assume that the delay for sending computation results back to terminal can be ignored. Therefore, the total delay for terminal
General simulated annealing algorithm

Processing capacity constraints

Since different IoT applications have different QoS requirements, we may divide the traffic into several types such as mobile streaming traffic, smart metering traffic, regular monitoring traffic, and emergency alert traffic etc. Assume each terminal generates a fixed type of traffic. Let the terminal \( j \) be responsible for the service type \( k \), the maximum delay it can tolerate is set at \( T_{jk}^{\text{max}} \). Therefore, the delay constraint for offloading processing tasks to EG can be written as follows:

\[
\sum_{\forall e \in E} I_{je} \left( t_{je}^{\text{pu}} + t_{je}^{\text{exe}} + t_{je}^{\text{que}} \right) \leq T_{jk}^{\text{max}}.
\] (11)

3.3 Processing capacity constraints

Since the processing capacity of EG \( \varepsilon \), that is, the CPU cycles per second, is defined as \( f_{\varepsilon} \), the total number of CPU cycles required for the tasks of all terminals served by EG cannot exceed its CPU processing capacity, that is,

\[
\sum_{\forall j \in T} I_{je} C_j \leq f_{\varepsilon}, \forall \varepsilon \in E.
\] (12)

3.4 Problem formulation

The optimization goal of this paper is to minimize the number of deployed EGs by adjusting their locations to meet the coverage, capacity, and delay constraints. Specifically, we have

\[
\min \cdot \sum_{\forall \varepsilon \in E} \varepsilon, \quad \text{s.t.:}
\]

\[
\begin{align*}
C1 : & \sum_{\forall \varepsilon \in E} I_{je} = 1, \forall j \in T \\
C2 : & T_{je} \leq \varepsilon, \forall j \in T, \forall \varepsilon \in E \\
C3 : & I_{je}, T_{je}, \varepsilon \in \{0, 1\}, \forall j \in T, \forall \varepsilon \in E \\
C4 : & \sum_{\forall \varepsilon \in E} I_{je} \left( t_{je}^{\text{pu}} + t_{je}^{\text{exe}} \right) \leq T_{jk}^{\text{max}}, \forall \varepsilon \in E, \\
& \forall j \in \{1, 2, \ldots, 5\} \\
C5 : & \sum_{\forall \varepsilon \in E} I_{je} C_j \leq f_{\varepsilon}, \forall \varepsilon \in E
\end{align*}
\]

Constraint C1 indicates that each terminal can only be served by one EG. Constraint C2 indicates that the relationship between variable \( I_{je} \), \( T_{je} \), and \( \varepsilon \), is that, it is possible to cover terminal only when the EG \( \varepsilon \) is deployed, and it is possible to serve terminal only when the edge node \( \varepsilon \) covers terminal. Constraint C3 states that \( T_{je}, I_{je} \) and \( \varepsilon \) are binary variables. C4 indicates that the delay for any terminal to offload its processing tasks to EG must be less than the maximum tolerable delay corresponding to its service type. Constraint C5 guarantees the sum of the processing loads at each EG should not exceed its capacity.

The optimization problem of (13) is similar to the \( k \)-median problem, but it adds some constraints such as delay and capacity limitations. Since the \( k \)-median problem is a NP-hard [15], EG deployment problem is also NP-hard. It is known that the Lagrange relaxation method [18] and the branch and bound method [19] are effective tools for small size problems. For the problem size we are considering, heuristic algorithms are needed.

4 Improved simulated annealing algorithm

In this section, we first introduce the general concept of simulated annealing algorithm. We then propose a new simulated annealing with an adaptive external penalty function (SA-AEP). The new algorithm to convert the multi-constrained optimization problem expressed by (13) into a simple constraint optimization problem and then solve it.

4.1 General simulated annealing algorithm

The simulated annealing algorithm is a meta-heuristic algorithm proposed by Metropolis [20]. It is a two-layer loop algorithm for solving large non-convex optimization problems. Let \( E(x) \) denote the internal energy when the configuration parameter is \( x \). And finding the optimal parameter configuration \( x \) in the parameter space \( \Omega \) is to minimize the internal energy \( E(x) \). The determination process of the configuration parameter \( x \) has five steps as follows:

Let \( T_k \) denote the current temperature of the \( k \)-th outer circulation step. Then \( T_0 \) is the initial temperature of the current annealing. Set the maximum number of iterations for each temperature as \( \Lambda_{\text{max}} \), that is, the number of inner cycles is \( \Lambda_{\text{max}} \), and the initial step \( s = 0 \), the corresponding configuration is randomly generated \( x_0(0) \in \Omega \), and this configuration is designated as the current solution and the historical optimal solution, that is \( x_0^*(0) = x_0(0) \) and \( x^{\text{opt}} = x_0(0) \), the internal energy corresponding to the current solution is calculated as \( E(x_0(0)) \), and the internal energy corresponding to the historical optimal solution is calculated as \( E(x^{\text{opt}}) \).

In most practical simulated annealing implementations, the temperature is updated according to the following rule:

\[
T_k = T_{k-1} \alpha^s, \alpha < 1,
\] (14)

where \( \alpha \) is the annealing rate and its value is a constant close to 1, and temperature gradually decreases to the target end value \( T_{\text{end}} \) with \( k \to +\infty \). Generate the next temperature according to formula (14), and let \( k = k + 1 \).

Suppose the current solution at temperature \( T_k \) is \( x_k^*(s - 1) \). At any step \( s \geq 0 \), we perturb the current state with a preset
neighborhood function to generate a new state and its corresponding configuration is $x_k(i)$. If the internal energy of the new state $E[x_k(i)]$ is less than the internal energy of the previous current solution $E[x_k^s(i-1)]$, set $x_k(i)$ to the new current solution, that is $x_k^s(i) = x_k(i)$; if the internal energy of the new state $E[x_k(i)]$ is greater than the internal energy of the previous current solution $E[x_k^s(i-1)]$, the new solution $x_k(i)$ will be accepted as the current solution of step $s$ with a certain probability, that is

$$P_{s_k}(x_k^s(i) \leftarrow x_k(i)) = \exp \left[ \frac{- (E[x_k(i)] - E[x_k^s(i-1)])}{\varepsilon T_k} \right],$$

where $\varepsilon$ is the Boltzmann constant and $T_k$ is the current temperature. Therefore, the probability that the new solution $x_k^s(i)$ is accepted as the current solution of step $s$ is:

$$P_{s_k}(x_k^s(i) \leftarrow x_k(i)) = \min\left\{ 1, \exp \left[ \frac{- (E[x_k(i)] - E[x_k^s(i-1)])}{\varepsilon T_k} \right] \right\}. \quad (16)$$

If $x_k^s(i)$ is not accepted, then $x_k^s(i) = x_k^s(i-1)$. If the internal energy of the current solution $E[x_k^s(i)]$ is smaller than the internal energy of historically optimal solution $E[x^\text{opt}]$, the current solution is set as the new historically optimal solution, that is, $x^\text{opt} = x_k^s(i)$, otherwise the historically optimal solution remains unchanged.

Under temperature $T_k$, the maximum number of iterations is $\Lambda_{\text{max}}$, repeat steps (2) and (3) for $\Lambda_{\text{max}}$ times. When the state is stable, the current solution is the optimal solution of the current state, and the temperature is reduced to the next temperature and continue iterating at the next temperature.

Determine whether $T_k$ has reached the target temperature $T_{\text{end}}$. If yes, stop the algorithm. Otherwise, go to step (2).

### 4.2 Improved simulated annealing algorithm with adaptive external penalty (SA-AEP)

Assuming that the feasible domain space formed by all constraints is $\Omega$, we need to find the configuration $\xi$ that minimizes the cost function (14) in $\Omega$. As the feasible domain space becomes smaller and smaller, the effectiveness of the simulated annealing algorithm may be reduced. Since the cost function surface of the algorithm may be split into several isolated regions, the algorithm may not be able to reach certain feasible regions. In addition, we hope that the optimal solution can strictly abide by the constraints and be in the feasible region $\Omega$. Therefore, the general simulated annealing algorithm cannot be directly applied to EG deployment scheme in this paper because the optimal solution it searches may not be in the feasible region.

This problem may be solved by using the outer point penalty function method [11]. This method can be used to deal with equality constraints and inequality constraints. It can extend the search beyond the feasible domain space $\Omega$, but will add a penalty term to the parameter configuration of cost function that violates the constraint. This method is an effective way for dealing with multi-constrained optimization problem. This method transforms the original constraint into a penalty function, and add it to the objective function. Suppose the original optimization problem is as follows:

$$\min_{\xi} F(\xi)$$

s.t. $f_{m}(\xi) \geq 0, \quad m = 1, \ldots, M$

$$f_{n}(\xi) = 0, \quad n = 1, \ldots, N \quad (17)$$

Using the outer point penalty function method, the problem is transform to

$$\min_{\xi} G(\xi, \tau) = F(\xi) + \tau H(\xi), \quad (18)$$

where $H(\xi)$ is a penalty function and $\tau$ is a large constant. The most common penalty function has the following form:

$$H(\xi) = \sum_{m=1}^{M} \phi(f_{m}(\xi)) + \sum_{n=1}^{N} \varphi(f_{n}(\xi))$$

where $\phi(f_{m}(\xi)) = \max[0, -f_{m}(\xi)]^{\mu}, \mu \geq 1$. (19)

$$\varphi(f_{n}(\xi)) = |f_{n}(\xi)|^{\nu}, \nu \geq 1$$

When configuration $\xi$ is not a feasible solution, a penalty is added to the objective function and forces the configuration to move towards the feasible domain during the optimization process. When configuration $\xi$ is a feasible solution, the penalty function value is equal to 0. This method can significantly shorten the search path.

There are three types of penalty functions, static, dynamic, and adaptive. In this paper we choose an adaptive external penalty function as follows:

$$\tau H(\varepsilon) = \tau \left( \sum_{j=1}^{\lceil \frac{T_{\text{max}}}{T_{j}} \rceil} \frac{F(\xi)}{T_{j}} \max\{0, \sum_{m=1}^{|E|} I_{m}^{*}\} + \sum_{j=1}^{\lceil \frac{T_{\text{max}}}{T_{j}} \rceil} \max\{0, \sum_{m=1}^{|E|} I_{m}^{*} C_{j} - f_{j}\} \right), \quad (20)$$

where $\tau$ is a penalty factor used to adjust accuracy and speed of search and $\varepsilon$ is the configuration parameter of $\varepsilon$:

$$\varepsilon = (\varepsilon_{1}, \varepsilon_{2}, \ldots, \varepsilon_{|E|}). \quad (21)$$

The first term in (20) corresponds to constraint $C4$. When the delay of the processing task offloaded to EG exceeds the maximum tolerable time delay, the penalty is proportionally increased with the violation. The second term in (20) corresponds to constraint $C5$. When the resources required by the tasks of all terminals served by EG exceed its capacity, penalty is also proportionally increased with the violations.

There are several benefits by using the penalty function (20). First, the penalty depends on the percentage of the constraint
violation. Multiplying the percentage of the violation of the constraint by the original objective function is added to the penalty function. This solves the problem that the constraint is often different in magnitude from the objective function \( F(\xi) \). Then, the penalty function is proportional to the objective function \( F(\xi) \). When the configuration outside the feasible domain space is not far from the feasible domain, we need to explore them. For example, the value of the objective function of a configuration, that is, the number of deployed EGs is significantly lower than the existing one. At this time, we need to perturb it and explore in its neighbourhood, which also reflects the adaptability of the penalty function (20). In addition, under the effect of parameter \( \tau \) we can adjust the acceptance of configurations outside the feasible domain space \( \Omega \). Finally, if the value of parameter \( \tau \) is too large, the minimum point configuration of the penalty function is still far from the optimal solution in the feasible region and the efficiency is very low. Hence, the penalty factor needs to choose an appropriate value.

With that, the optimization problem is transformed to:

\[
\min \left\{ \sum_{i=1}^{|E|} \varepsilon_i + \tau H(\xi) \right\},
\]

\[\begin{align*}
C1 & : \sum_{\forall e \in E} I_{\beta} = 1, \forall j \in T \\
\text{s.t.,} & \\
C2 & : I_{\beta} \leq T_{\beta} \leq \varepsilon, \forall j \in T, \forall e \in E \\
C3 & : I_{\beta}, T_{\beta}, \varepsilon \in \{0, 1\}, \forall j \in T, \forall e \in E
\end{align*}\]

Figure 2 shows the simulated annealing algorithm based on adaptive external penalty.

4.3 Complexity analysis

with As can be seen from the analysis of the SA-AEP algorithm, the number of internal cycles of the algorithm is \( A_{\text{inner}} \), the number of external cycles is determined by the temperature drop, from \( T_0 \) to \( T_{\text{end}} \), and the annealing rate is \( \alpha \), and hence the number of external cycles is \( \log \left( \frac{T_0}{T_{\text{end}}} \right) \). In each inner loop, \(|T|\) number of terminals are checked for service constraints violations. Therefore, the complexity of the improved simulated annealing algorithm based on adaptive external penalty is \( O(|T| \cdot \log \left( \frac{T_0}{T_{\text{end}}} \right)) \).

5 IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM

In addition, in order to compare the performances of the proposed SA-AEP, in this section, we use an adaptive variable-length particle swarm optimization algorithm (AVLPSO) [21] which is used to search for the near optimal solution.

The particle swarm algorithm is an evolutionary algorithm that simulates the foraging behaviour of a flock of birds. The general particle swarm algorithm assumes that there is a search space with a dimension of \( D \), where a group consists of \( m \) particles, and each particle searches for the optimal solution according to its own flying speed. The position of particle \( i \) is denoted as \( W(i) = (x_{i1}, x_{i2}, \ldots, x_{id}) \) and the velocity is denoted as \( V(i) = (v_{i1}, v_{i2}, \ldots, v_{id}) \). The fitness value calculated according to the particle’s position can reflect the pros and cons of the particle’s solution to a specific optimization problem. The particle’s velocity and position are updated according to the following formula:

\[
v_{id}(t + 1) = \omega v_{id}(t) + c_1 r_1 (p_{id} - x_{id}(t)) + c_2 r_2 (p_g - x_{id}(t)).
\]

\[
x_{id}(t + 1) = x_{id}(t) + v_{id}(t + 1).
\]

where \( l = 1, 2, \ldots, m \), \( d = 1, 2, \ldots, D \), \( -v_{\text{max}} \leq v_{id} \leq v_{\text{max}} \), that is, the velocity of the particle is limited to a certain range, \( t \) is the number of iterations, \( p_{id} \) represents the current individual optimal solution of particle \( i \), and \( p_g \) represents the current global optimal solution. \( \omega \) is the inertia factor describing the degree
of maintaining the previous speed, \( c_1, c_2, r_1, r_2 \) are all weights describing the distance between the particle’s own position to the individual optimal and the global optimal position, which are used to adjust the learning ability of individual and social experience.

The length of the particles in the general particle swarm algorithm is fixed, which represents the dimension of the search space. This paper assumes that the number of EGs deployed is \( N \), and each particle is a vector composed of the coordinates of \( N \) EGs. The plan is to find the optimal position and the minimum number of EGs at the same time. The number of EGs \( N \) is variable. Therefore, for the optimization problem in this paper, a particle swarm with varying length is used to search for the optimal solution. The initial particle length is \( N \), and then search for the optimal position to satisfy all constraints, and then reduce the particle length:

\[
N \leftarrow N - 1. 
\tag{25}
\]

Repeat the above search process until the smallest particle length is found to meet the service, delay and capacity requirements.

In order to avoid falling into a local optimum, we use an AVLPSO algorithm [21], where each particle is a one-dimensional vector composed of \( N \) EGs positions. Since the weight \( c_1 \) is used to adjust the learning ability of individual experience, we propose that as the number of iterations increases, \( c_1 \) can dynamically and nonlinearly increase so that the particles have strong self-awareness, that is, adaptively increase the diversities of solutions to solve the problem of premature convergence. The specific improved \( c_1 \) is expressed as follows:

\[
c_1 = c_{1_{\text{min}}} + \left( c_{1_{\text{max}}} - c_{1_{\text{min}}} \right) \left( \frac{1 - \cos \left( \frac{t}{M} \right)}{2} \right), 
\tag{26}
\]

where \( c_{1_{\text{min}}} \) is the minimum value of \( c_1 \), \( c_{1_{\text{max}}} \) is the maximum value of \( c_1 \), \( t \) is the current iteration number, and \( M \) is the set maximum iteration number.

Assuming that the number of deployed EGs is \( N \), that is, the search space dimension is \( 2N \), the initial generation of \( L \) particles is expressed as follows:

\[
W^{(l)}(t) = (x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_N),
\tag{26}
\]

where \( l = 1, 2, \ldots, L \). In the search process, the redundant EGs are continuously removed, and the particle length \( N \) also decreases with the number of EGs. The optimal position of the individual passing by the particle / is denoted as \( (p_{11}, p_{12}, \ldots, p_{12N}) \), the optimal position of the particle in the entire population is denoted as \( (p_{21}, p_{22}, \ldots, p_{22N}) \), and the position and velocity update formula of particle / is as follows:

\[
v_d^{(l)}(t + 1) = \omega v_d^{(l)}(t) + c_1 r_1 (p_{1l}(t) - W_d^{(l)}(t)) + c_2 r_2 (p_{2l}(t) - W_d^{(l)}(t)), \tag{28}\]

\[
W_d^{(l)}(t + 1) = W_d^{(l)}(t) + v_d^{(l)}(t + 1), \tag{29}\]

where \( \omega \) is updated according to formula (26).

During the solution process, the position and length of the particles need to be continuously adjusted to meet the QoS constraints, coverage constraints, and EG capacity constraints. Set a Boolean variable \( \xi_i \), and for any \( i \) in \( E \) if it satisfies the constraint \( C_1 \) in Equation (13), it is recorded as \( \xi_i = 1 \), otherwise, \( \xi_i = 0 \). Therefore, the constraint \( C_1 \) in Equation (13) is transformed into a utility function as follows:

\[
U_i = \sum_{i=1}^{M} |\xi_i - 1|, \tag{30}\]

where \( U_i \) is the difference between the number of service terminals of the particle / and the actual number of terminals, and the optimal utility function is recorded as \( U_{\text{opt}} \).

By analysis we know that if the constraint \( C_1 \) in formula (13) is satisfied, then the constraints \( C_2 \) and \( C_3 \) must be satisfied. Set Boolean variables \( \xi_i \) and \( \gamma_i \). For any \( i \), if it satisfies the constraint \( C_4 \), it is recorded as \( \xi_i = 1 \), otherwise it is recorded as \( \xi_i = 0 \); if it satisfies the constraint \( C_5 \), it is recorded as \( \gamma_i = 1 \), otherwise it is recorded as \( \gamma_i = 0 \). \( T_i \) represents the difference between the number of terminals that the particle / meets the delay constraint and the actual number of terminals, and \( S_i \) represents the difference between the number of EGs that the particle / meets the capacity constraint and the actual number of EGs. Then the constraints \( C_2 \) and \( C_3 \) can be transformed into:

\[
T_i = \sum_{i=1}^{M} |\xi_i - 1| = 0, \tag{31}\]

and

\[
S_i = \sum_{i=1}^{N} |\gamma_i - 1| = 0. \tag{32}\]

When the algorithm is iterated to meet the convergence requirements of formulas (30), (31) and (32), \( N \) is continuously reduced until the smallest \( N \) meets the coverage and capacity constraints. At this time, the number of EGs obtained is the smallest, and the EG location is the optimal location. Let the utility of the current global optimal particle position be expressed as \( U_{\text{opt}} \), the utility of the optimal position experienced by the particle / is expressed as \( U_{\text{opt}}^{(l)} \), and the global optimal particle position is recorded as \( W_{\text{opt}}^{(l)} \), the optimal position that particle / has experienced is recorded as \( W_{\text{opt}}^{(l)} \). The specific deployment algorithm steps are as follows:

The execution of the AVLPSO algorithm proposed in this paper is a circular iteration, and the number of iterations required to obtain the optimal solution is determined by the actual scenario. In each iteration, the \( L \) particle first calculates the current individual utility value and updates the individual
optimal utility value, then selects the particle with the best current utility value among all particles and records its position as the current global optimal position. For $L$ particles, the parameters $c_i$ of each particle are updated according to its speed and position. Therefore, the calculation complexity of each iteration of each particle length is $2L$. We set the maximum number of iterations per particle length to $I_{\text{max}}$ and after the initial number of EGs is set to $N_{\text{ini}}$, the utility function calculates a total of $2LI_{\text{max}}$ times. If the final deployed minimum number of EGs is $N_i$, the complexity of the AVLPSO algorithm is $O\left(L I_{\text{max}} (N_{\text{ini}} - N_i + 2)\right)$. Compared with the complexity of the improved simulated annealing algorithm based on adaptive external penalty $O\left(|T| \cdot \log \left(\frac{T_0}{T_{\text{end}}}\right)\right)$, it is not easy to evaluate which algorithm is less complex, and we will discuss in specific simulation experiments.

6 PERFORMANCE EVALUATION

In this section, we use simulation to evaluate the performances of the proposed simulated annealing algorithm with adaptive external penalty (SA-AEP) and simulated annealing (SA), adaptive variable-length particle swarm optimization algorithm (AVLPSO) and particle swarm optimization (PSO) algorithms. We set up a planning area of 1 km square. The simulation parameters are shown in Table 2. Two sets of experiments, as shown in Figure 3, there are two distributions of 30 and 60 random deployed terminals respectively in the planning area. According to the urban radio channel interference model, we set the channel gain $H_{ij} = d_{ij}^{-\gamma}$, where $\gamma$ is the path loss attenuation factor.

6.1 Performances of SA-AEP and SA algorithms

To compare the performance of SA-AEP and SA algorithms, we performed twenty experiments to obtain the average of the optimal number of EGs deployed in the twenty experiments. In both algorithms, the outer loop is reduced from temperature $T_0$ to $T_{\text{end}}$ and the number of inner loops is 1000. The random perturbation method in the inner loop works as follows. First, randomly select an unassigned EG, and randomly deploy another unassigned EG, and if it has not been deployed, deploy it or swap its status with one of the deployed EGs. If it is not deployed, deploy it or swap its status with one of the deployed EGs.

In Figure 4, the x-axis represents the outer loop iteration steps, and the y-axis represents the historically optimal number of deployed EGs at each iteration step. It can be seen that as the number of iteration increases, the number of historically optimally deployed EGs in both algorithms is decreasing, but due to the contribution of the AEP function, it can reach isolated feasible regions for further. In addition, when the proposed algorithm searches for a solution not far from the feasible region, we perturb it and explore in its neighbourhood. As the search progresses, the solution will move toward the optimal solution,
which increases the probability of finding a better solution than the general simulated annealing algorithm. Therefore, the proposed algorithm has a smaller number of deployed EGs than the original algorithm.

Figure 5 shows that the minimum number of deployed EGs obtained by the two algorithms varies with the change of the coverage radius of EG. As the coverage radius of a single EG increases, the required minimum number of deployed EGs becomes smaller and smaller, but as the coverage radius increases at a certain level, although the number of EGs required to meet the coverage constraints is small, the decrease in the number of EGs gradually slows down due to capacity and delay constraints, eventually getting to the stable state. In addition, the combination of adaptive external point penalty function algorithm may reach some isolated feasible regions that cannot be reached by general simulated annealing algorithms and increase the probability of finding a better solution. Thus, the minimum number of deployments is less than that of general simulated annealing algorithms.

We increased the number of terminals from 30 to 60 to verify whether the increase in the number of terminals would affect the performance of the two algorithms. We perform twenty experiments on these two algorithms and take average value of the optimal number of deployed EGs is shown in Figure 6. Since the general simulated annealing algorithm starts from the initial point and always searches in the feasible region, the path from the initial configuration to the optimal configuration may be longer at this time, but if the external search method is applied, it can make the optimal solution search extended from the inside to the outside, which can shorten the path from the initial configuration to the optimal configuration. Therefore, the proposed algorithm will converge earlier than the general simulated annealing algorithm and find a solution.
with a smaller number of EGs. The proposed algorithm has a faster downward trend of the target value at the time of 60 terminals than that of 30 terminals, because when the planning area size and EG coverage are unchanged, as terminals increase, more redundant coverage caused by randomly generated initial solutions and hence the target value decreases faster as the number of iterations increases in the early period.

6.2 Comparison of proposed SA-AEP and AVLPSO algorithm

In this part, firstly, we use simulation experiments to evaluate the performances of AVLPSO algorithm. In the experiments, we set $\omega = 0.8$, $c_{1\text{max}} = 2$, $c_{1\text{min}} = 2.5$, $v_{\text{max}} = 100w$, the number of terminals is set as 30, the number of particles in the particle swarm is 50, and the maximum number of iterations is set to $M = 100$.

The final result is shown in Figure 7, the x-axis represents the outer loop iteration steps, and the y-axis represents the service rate at each iteration step. The service rate is the ratio of the number of users served by all EGs to the actual number of users in a given planning area. Figure 7 shows the relationship between the service rate and the number of iterations of the AVLPSO algorithm when $N = 11$ and $N = 12$. It can be seen from that as the number of iterations increases, the service rate continues to increase, eventually tending to level off. Finally, the service rate of $N = 12$ is higher than the target service rate $\tau$, but the service rate of $N = 11$ is lower than the target service rate $\tau$. As the number of iterations increases, redundant EGs are continuously removed, and the number of EGs continues to decrease. When the number of EGs is reduced to 12, the service demand can still be met, but when the number of EGs is reduced to 11, no more to meet the service constraints, the minimum number of EGs that can meet all constraints is 12.

In order to verify the advantages of the AVLPSO algorithm in the deployment and planning of EGs, we compare it with the general PSO algorithm. If the general PSO algorithm is applied to the planning, the weight coefficient $c_{1}$ is set to a constant 2. Through iteration, the positions of EGs that meet all constraints are obtained and fixed, and then the EGs that have the least impact on the number of service users are removed one by one until the number of EGs that can meet the constraints is found.

Figure 8 shows the relationship between the service rate and the number of iterations of the PSO algorithm when $N = 11$ and $N = 12$ respectively. When the number of EGs is reduced to 13, the service demand can still be met, but when the number of EGs is reduced to 12, when the service constraints can no longer be satisfied, the minimum number of EGs that can meet all constraints is 13.

Figure 9 shows the comparison of the service rate of the AVPSO algorithm and the PSO algorithm with the increase in the number of EGs. It can be seen that when the number of EGs increases from 1 to 5, the service rates of the two are equal, and the number of EGs increases from when 5 is increased to 15, the service rate of AVPSO algorithm is higher than that of
Figure 9  Comparison of service rates of AVLPSO and PSO algorithms with different number of EGs

FIGURE 9  Comparison of service rates of AVLPSO and PSO algorithms with different number of EGs

The reason is that the population diversity of the general PSO algorithm decreases in the later iterations, and it is easy to fall into the local optimum, and its particle length remains unchanged. After eliminating redundant EGs one by one, the remaining EG positions remain unchanged. The EG location may not be the optimal location. In contrast, in the later stage of the AVLPSO algorithm, the coefficient $c_1$ is dynamically increased as the number of iterations increases, thereby increasing the population diversity, and then gradually reducing the particle length until the minimum number of EGs that meet the constraints and the EG position closest to the optimal solution are obtained.

Figure 10 shows the final solution after running the AVLPSO algorithm, in which the red dots represent randomly distributed terminals, the blue circle is the coverage of all deployed EGs, and the center of the circle is the deployment location of the EGs. As shown in Figure 10(a), there are 30 terminals, and the final solution has a total of 12 EGs. Figure 10(b) shows when there are 60 terminals, the final optimal solution is 15. Figure 11 shows the final solution after running the simulated annealing algorithm based on adaptive external penalty. As shown in Figure 11, there are 30 terminals in Figure 11(a). The final solution deployed a total of 11 EGs. There are 60 terminals in Figure 11(b). The final solution deploys 21 EG and it is seen that in Figure 11 how each terminal is covered.

Under the conditions that the initial conditions and basic parameter settings are the same, and the number of terminals is 30, we explore the minimum number of EGs required for different service rates by proposed SA-AEP and AVLPSO algorithms. Figure 12 shows the number of deployed EGs under different radius. When the EG radius is 50–200 m, the proposed SA-AEP algorithm has the least number of EGs, which is better than the other three algorithms.

Figure 13 shows when the service rate is greater than or equal to 0.95, the proposed SA-AEP algorithm is the least deployed, whereas when the service rate is less than 0.95, the proposed AVLPSO algorithm is less than that of SA-AEP algorithm. It is because that the proposed SA-AEP algorithm introduces the
adaptive external penalty function which can help to get the near optimal solution when the number of requesting service terminals is large.

Table 3 shows the minimum number of EGs required to be deployed by the four algorithms and the comparison of the complexity, where the service rate is 1, the TGs is 60 and the EG radius is 200 m. Here the complexity refers to the number of iterations required for the algorithm to reach the stable state. It can be seen from Table 3 that the SA-AEP proposed in this paper requires the smallest number of EGs, and its algorithm complexity is relatively low. Compared with PSO and AVLPSO, the required minimal number of EGs of proposed SA-AEP can be decreased by 27% and 20 respectively, and the complexity can be decreased by 37.5% and 61.5% respectively. In general, the complexity of the particle swarm algorithm is higher than that of the simulated annealing algorithm.

7 | CONCLUSION

This paper proposes an EG deployment model for the Internet of Things. We first analyse the conditions of coverage and capacity of EGs and the factors affecting the delay of offloading tasks, and establish the EG deployment optimization model. Then we propose an external penalty function, combined it with the PSO, to obtain the optimal solutions of EG deployment problem. For comparison, we use the AVLPSO algorithm to get the near optimal solution. The simulation results show that Compared with PSO and AVLPSO, the required minimal number of EGs of proposed SA-AEP can be decreased by 27% and 20 respectively, and the complexity can be decreased by 37.5% and 61.5% respectively.

In our future work, we will further build the EG deployment model which considers the communication capability from the EG to the cloud. Also, we will investigate the influence of collaborative computing among EGs on the deployment model. And we try to use the proposed mode in this paper in the actual IoT applications.

ACKNOWLEDGEMENTS

This work was supported by Natural Science Foundation of China (61871237, 92067101), Program to Cultivate Middle-aged and Young Science Leaders of Universities of Jiangsu Province and Key R&D plan of Jiangsu Province (BE2019017) and Jiangsu Agriculture Science and Technology Innovation Fund under Grant CX(19)3008.

ORCID

Xiaorong Zhu https://orcid.org/0000-0002-1521-8059

REFERENCES

1. Service Requirements for Machine-Type Communications, Sophia-Antipolis Cedex, France, document 3GPP TS 22.368 V11.5.0, 3GPP (2012)
2. Cardellini, V., et al.: Self-adaptive container deployment in the fog: A survey. In Algorithmic Aspects of Cloud Computing (ALGOCLOUD). Lecture Notes in Computer Science, vol 12041. Springer, Cham (2019)
3. Ghobaei-Arani, M., et al.: Resource management approaches in fog computing: A comprehensive review. J Grid Computing 18, 1–42 (2020)
4. Gravalos, P. M., et al.: Efficient network planning for internet of things with QoS constraints. IEEE Internet of Things Journal 5(5), 3823–3836 (2018). https://doi.org/10.1109/JIOT.2018.2849327.
5. USA: Abbrev. of Publisher, year, ch. x, sec. x, pp. xxx–xxx. Wenjia Wu, Junzhou Luo and Ming Yang. Gateway placement optimization for load balancing in wireless mesh networks. In: 13th International Conference on Computer Supported Cooperative Work in Design, pp. 408–413. IEEE, Piscataway (2009)
6. Peng, J., Zhou, Q.: Gateways placement optimization in wireless mesh networks. In: International Conference on Networking and Digital Society, pp. 221–226. IEEE, Piscataway (2009)
7. Li, F., et al.: Gateway placement for throughput optimization in wireless mesh networks. In: IEEE International Conference on Communications, pp. 4955–4960. IEEE, Piscataway (2007)
8. Ding, J., et al.: Gateway deployment optimization in wireless mesh network: A case study in China. In: IEEE/INFORMS International Conference on Service Operations, Logistics and Informatics, pp. 300–305 . IEEE, Piscataway (2009)
9. Bejerano, Y.: Efficient integration of multihop wireless and wired networks with QoS constraints. IEEE/ACM Trans. Netw. 12(6), 1064–1078 (2004)
10. Sakamoto, S., et al.: Implementation of intelligent hybrid systems for node placement problem in wms considering particle swarm optimization, hill climbing and simulated annealing. Mobile Netw Appl 23, 27–33 (2018).
11. Zhao, L., Liu, J: Optimal placement of virtual machines for supporting multiple applications in mobile edge networks. IEEE Trans. Veh. Technol. 67(7), 6533–6545 (2018)
12. Yin, H., et al.: Edge provisioning with flexible server placement. IEEE Trans. Parallel Distrib. Syst. 28(4), 1031–1045 (2017)
13. Santoyo-González, A., Cervelló-Pastor, C.: Latency-aware cost optimization of the service infrastructure placement in 5G networks. J. Netw. Comp. Applications 114, 29-37 (2018)
14. Lin, C., Yang, J.: Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0. IEEE Trans. Indus. Informatics 14(10), 4603–4611 (2018)
15. Wang, S. G., et al.: Edge server placement in mobile edge computing. J. Parallel Distrib. Computing 127, 160–168 (2019)
16. Li, H., et al.: Enhancing the performance of 802.15.4-based wireless sensor networks with NB-IoT. IEEE Internet of Things J. 7(4), 3523–3534 (2020). https://doi.org/10.1109/JIOT.2020.2972512
17. Brogi, A., Forti, S.: QoS-aware deployment of IoT applications through the fog. IEEE Internet Things J. 4(5), 1185–1192 (2017)
18. Wu, T., et al.: A Lagrangean relaxation approach for a two-stage capacitated facility location problem with choice of facility size. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 713–718. IEEE, Piscataway (2015)
19. Dupont, L.: Branch and bound algorithm for a facility location problem with concave site dependent costs. Int. J. Product. Economics 112, 245–254 (2008)

How to cite this article: Zhu, X., et al.: Optimal edge gateway deployment in internet of things based on simulated annealing with adaptive external penalty. IET Commun. 1–14 (2021). https://doi.org/10.1049/cmu2.12244