Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

SEFSD: an effective deployment algorithm for fog computing systems

Huan Chen[1], Wei-Yan Chang[1], Tai-Lin Chiu[1], Ming-Chao Chiang[2] and Chun-Wei Tsai[2*]

Abstract

Fog computing aims to mitigate data communication delay by deploying fog nodes to provide servers in the proximity of users and offload resource-hungry tasks that would otherwise be sent to distant cloud servers. In this paper, we propose an effective fog device deployment algorithm based on a new metaheuristic algorithm–search economics–to solve the optimization problem for the deployment of fog computing systems. The term "effective" in this paper refers to that the developed algorithm can achieve better performance in terms of metrics such as lower latency and less resource usage. Compared with conventional metaheuristic algorithms, the proposed algorithm is unique in that it first divides the solution space into a set of regions to increase search diversity of the search and then allocates different computational resources to each region according to its potential. To verify the effectiveness of the proposed algorithm, we compare it with several classical fog computing deployment algorithms. The simulation results indicate that the proposed algorithm provides lower network latency and higher quality of service than the other deployment algorithms evaluated in this study.

Keywords Mobile communication, Deployment problem, Metaheuristic algorithm, Search economics

Introduction

With the explosive growth of the internet of things, new applications aim to provide interactive and intelligent services to users; these services typically require lower delay for responsive user experience and greater computational resources for complex algorithms. However, determining how to reduce transmission and processing delay to provide responsive and intelligent services under various network traffic conditions is a challenging task. Fog computing is a method that involves distributed computing on fog nodes comprising numerous devices (e.g., sensors and appliances). It is different from uploading all computing tasks to the cloud and it can potentially be used to enhance the performance of internet of things (IoT) environments [1]. Fog computing architecture is typically hierarchical and can be divided into three layers, namely the terminal, fog, and cloud layers [2].

When the sensors require computational resources or run real-time tasks, the fog computing system provides the required services accordingly. Because the fog layer connects the low-level sensors and high-level cloud, it plays an essential role in such hierarchical architecture. The devices of the fog layer (e.g., fog servers) can generally be deployed in a fixed location or installed on a dynamic vehicle. The fog server analyzes the data collected from the terminal layer and then relays the analysis results to the terminal layer or uploads the data to the cloud layer for processing. In this type of system, the cloud layer comprises high-performance computing servers and storage devices that enable mass data processing and the performance of complex computational tasks. Fog nodes are typically deployed between lower-level devices and high-level cloud computing platforms. The fog servers share the cloud computing platforms'

*Correspondence:
Chun-Wei Tsai
cwtsai@mail.cse.nsysu.edu.tw
[1] Department of Computer Science and Engineering, National Chung Hsing University, Taichung, Taiwan
[2] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan

computational loads to reduce latency and increase the computational speed of the integrated IoT system.

Because the fog node deployment strategy influences a fog computing system's cost and performance, determining a suitable and efficient deployment strategy has become a critical optimization problem. One key research topic related to the development of such a strategy is the sufficient allocation of computational resources to the fog computing cluster to allow for the provision of customized services near users [3]. Some early studies on fog computer architecture [4, 5] regarded the fog node deployment optimization problem as nearly identical to the NP-hard degree-constrained minimum spanning tree (DCMST) optimization problem [6–8]. Because determining optimal solutions to most NP-hard optimization problems within a reasonable time is nearly impossible, metaheuristic algorithms are often used to determine approximate solutions to fog node deployment optimization problems. Some studies [9–11] have also attempted to use metaheuristic algorithms to solve the DCMST optimization problem in recent years because most can determine an approximate solution within a reasonable time.

Because most conventional meta heuristic algorithms [12–14] tend to converge to local optima, postponing the convergence of metaheuristic algorithms to allow searching of particular regions has become a key research topic. One approach to addressing this problem is to increase the search diversity of the metaheuristic algorithm during the convergence process by, for example, randomly creating additional candidate solutions to replace the current searched keys. However, using this method to increase the search diversity might also degrade the search performance of the metaheuristic algorithm because it erases some of the search experience. A new metaheuristic algorithm called search economics (SE) [15] was designed to increase search diversity in the convergence process by allocating searches to different regions according to the potential of each region instead of randomly creating new search directions.

The main goal of SE is to fairly allocate computational resources in each iterative process by computing the potential of each region (i.e., subspace). In this paper, we present an effective metaheuristic algorithm based on SE for solving the fog node deployment optimization problem. The main contributions of this paper are as follows:

- A new solution space division method is proposed for solving the fog node optimization problem based on SE.
- A new operator for SE called the trade operator and modification of other SE operators are proposed to solve the fog node optimization problem.

This paper is organized as follows. Related work section discusses fog computing systems, provides a definition of the optimization problem, and briefly discusses deployment algorithms used to address the fog node optimization problem. Proposed method section discusses the improved version of the SE algorithm. Experimental results section describes the experimental process and environment and presents the simulation results of the proposed algorithm and the other deployment algorithms compared in this paper. Finally, Conclusion section presents our conclusions and some suggestions for future research.

Related work

Problem definition

The performance of fog computational resource management can be evaluated on the basis of three factors discussed in [16] as follows: 1). Minimized latency: Reducing latency is a main goal of fog computing architecture. User-perceived latency strongly affects the service quality of a fog computing system. The locations of servers affect the speed of data transmission. 2). Minimized resource usage: Although increasing the number of fog servers might reduce the latency of the whole system, it also increases the cost. 3). Minimized service placement transitions: Because a user requests must often be completed by servers in different layers, reducing the data transmission requirements is another primary concern in such a system. In this paper, the problem definition is based on that employed in a recent study [17] and is defined as follows:

$$\min f(s) = C_l + C_s + P, \tag{1}$$

where s represents a candidate solution, C_l represents the total cost of deploying fiber to construct connections with devices, C_s is the total cost of device installation, and P represents penalty costs. The details of the calculation are described as follows:

$$C_l = c_f \sum_{(i,j)\in\{s\}\times\Omega_G\cup\Omega_G\times\Omega_F\cup\Omega_F\times\Omega_E} x_{i,j}\cdot d_{i,j}, \tag{2}$$

where c_f represents the unit cost of deploying fiber; $x_{i,j}$ represents the link installed between devices i and j; $d_{i,j}$ is the distance between devices i and j; z represents the location of the cloud center, and Ω_G, Ω_F, and Ω_E, represent the potential positions of the gateways, fog servers, and edge servers, respectively.

$$C_s = c_G \sum_{m\in\Omega_G} g_m + c_F \sum_{n\in\Omega_F} f_n + c_E \sum_{t\in\Omega_E} q_t, \tag{3}$$

where the c_G, c_F, and c_E are the costs of installing a single gateway, a fog server, and an edge server, respectively; g_m is a binary flag indicating the candidate location for

installation of gateway m; f_n is a binary flag indicating the candidate location for installation of fog server n; and q_t is a binary flag indicating candidate location for installation of edge sever t. The constraints of fog computing system deployment are difficult to evaluate independently of the solution space, but the constraints can increase the objective value by defining penalty costs. A fog computing system can incur considerable costs when deployment does not meet constraints. Penalty costs can be calculated using Eq. (4)

$$P = \kappa \cdot (\eta_l + \eta_d + \eta_\phi + \eta_o + \eta_u), \tag{4}$$

where the κ is the coefficient of the penalty cost and η_l, η_d, η_ϕ, η_o, and η_u are constrained terms in the fog computing system. The constrained terms are defined as follows:

- Constraint of link (η_l): The number of users or devices in the fog system without service. The optimal solution would allow all devices or users to obtain service from he top layer when they request resources. This constraint is calculated using Eq. (5)

$$\eta_l = \begin{cases} \eta_l + 1, & \text{if } x_{i,j} = 0, \\ \eta_l, & \text{otherwise,} \end{cases} \tag{5}$$

  where the $x_{i,j}$ indicates whether a connective link exists between devices i and j.
- Constraint of demand (η_d): The maximum number of services that devices can handle. Each user has its demand γ_A that requests resources from top layers to maintain service, and the demand is handled by the closest server. Each type of device has different demands (i.e., γ_t, γ_n, and γ_m are demands of edge server t, fog server n, and gateway m, respectively) and maximum demand factors (H_t^E, H_n^F, and H_m^G for edge server t, fog server n, and gateway m, respectively). This constraint is calculated using Eq. (6)

$$\eta_d = \begin{cases} \eta_d + 1, & \text{if } \sum_{k \in \omega_i} r_i \cdot x_{jk} > H_j^E, \\ \eta_d, & \text{otherwise,} \end{cases} \tag{6}$$

  where $\omega_{i,j}$ represents a device i's demands for a top-layer device j and k represents the type of device. In the deployment stage, the ideal is for each device to be able to handle the demands from the lower layer and the total loading demands of each device to be less than its maximum.
- Constraint of latency (η_ϕ): The total latency time of the whole fog computing system. The latency time of device i is calculated from its service provider device j according to data size L_j and transmission rate γ_j. This constraint is calculated using Eq. (7):

$$\eta_\phi = \begin{cases} \eta_\phi + 1, & \text{if } \sum_j \frac{L_j}{x_{i,j}} \cdot \gamma_j > D_{i,j}, \\ \eta_\phi, & \text{otherwise,} \end{cases} \tag{7}$$

  where $D_{i,j}$ represents the maximum latency time.
- Constraint of coverage (η_o): The distance of any user k to its service provider must be less than the radius of range for edge server t. In the deployment stage, the ideal is for all users to deploy at least one edge server nearby.

$$\eta_o = \begin{cases} \eta_o + 1, & \text{if } x_{t,k} \cdot d_{t,k} > R_E, \\ \eta_o, & \text{otherwise,} \end{cases} \tag{8}$$

  where $x(t, k)$ and $d(t, k)$ represent the connection flag and distance between the user k and edge server t, respectively. R_E represents the radius of service range for edge servers.
- Constraint of capacity (η_u): The maximum capacity of the edge servers, fog servers, and gateways are N_E, N_F, and N_G , respectively.

$$\eta_u = \begin{cases} \eta_u + 1, & \text{if } \sum_{k \in \omega_i} x_{i,k} \le N_j, \\ \eta_u, & \text{otherwise,} \end{cases} \tag{9}$$

  where ω_i represents the capacity of device k and i represents the type of device.

In summary, the design of the objective function defined in Eq. (1) takes into consideration the fiber deployment cost C_l, device installation cost C_s, and many other penalty costs P, which have been explained in detail in Eqs. (2) to (4), respectively. It's worth to mention that the penalty cost increases the objective value to avoid server overload, disconnection, high latency, and other problems. Since this deployment problem can be considered as a complex optimization problem, using the greedy or deterministic search algorithms might not be able to find out a good solution in reasonable time. To solve such complex optimization problems, the proposed metaheuristic algorithm can provide an alternative way to find out the approximate solution within reasonable time [18]. That is the major reason we apply the metaheuristic algorithm to solve this optimization problem in this study.

Proposed method

Basic idea

The SE algorithm is a new metaheuristic algorithm [15] that aims to adjust the computational resources allocated to different regions of the whole search space according to the potential profit of regions in the current search direction. Unlike other metaheuristic algorithms, the SE algorithm contains three essential components involved in the convergence process: Regions, goods, and searchers. The whole solution space (search space) can be regarded as an

investment market that can be further split into a certain number of submarkets, called regions, by domain knowledge. The solutions represent the goods in the market, and the objective value of the solution represents the profit. The searcher acts as an investor to identify and invest in the goods that can yield a high profit. The searcher collects historical profit information to improve product quality. The algorithm uses searchers to continually identify goods with high profit potential in each region. The goods retain characteristics that reflect their respective submarkets, remain in their original region, and do not exchange information with goods in other regions. If the algorithm can estimate the potential profit from investing in certain regions, it can dynamically adjust the allocation of computational resources. Therefore, the SE algorithm is suitable for solving problems with massive solution spaces. After the first version of SE was presented in [15], it was successfully applied to solve optimization problem in many fields, such as wireless sensors deployment problem [19], internet resource management [20], cell deployment in 5G wireless communication [21], hyper-parameter optimization for deep neural networks [22], and deep neural network pruning problem [23]. These results shown that SE can find better results than other metaheuristic algorithms (e.g., genetic algorithm), especially in complex optimization problems. Therefore, we are confident that using SE in our study is an efficient approach to obtain better deployment solution.

SE algorithms have the advantage over other metaheuristic algorithms in that they will first analyze the complex solution spaces and adjust the search direction accordingly. The algorithm collects three pieces of information from the process of searchers investing in goods: (1) The previous optimal profit of goods, (2) the region with the highest average profit, and (3) the regions not yet explored. For the searcher, if the indices of profit potential can be effectively classified by submarket, the efficiency of the search increases. For example, if the goods located in the same submarket yield the same profits and their profits are different from those of goods in other regions, the searcher can efficiently search for goods in submarkets with the highest profit potential. In this study, the SE algorithm was used to solve the fog computing system deployment problem. The algorithm generates several goods that represent the deployment of gateways, fog servers, and edge servers at candidate locations, and the profits of the goods are calculated using Eq. (1). These values are used to allocate computational resources and identify more profitable goods in the high-potential region. Finally, the algorithm generates the optimal deployment locations for the fog computing system in the final convergence stage. The notation presented in Table 1 is used throughout the remainder of this paper to simplify the discussion of the proposed algorithm.

Table 1 The notation of this paper

Notation	Description
k	Number of region.
r_i	The i-th region.
R	A set of region, $R = \{r_1, r_2, \ldots, r_k\}$.
m	Number of searchers.
s_j^i	The j-th searcher in the i-th region.
S	A set of searchers,$S = \{s_1, s_2, \ldots, s_m\}$.
n	Number of candidate goods in regions.
g_l^i	The l-th goods in i-th region.
G^i	A set of goods in the i-th region.
t_i^a	Number of consecutive invested in the i-th region.
t_i^b	Number of consecutive uninvested in the i-th region.
μ_i	Rates of consecutive invested in the i-th region.
ν_i	Average objective value obtained by searchers in the i-th region.
ρ_i	Average objective value obtained by goods in the i-th region.
E_i	The expected value of the i-th region.
E_i'	The adjusted expected value of the non-convergent region i.
o	Number of players.
γ	The weight of trade operator adjust the expected value of the non-convergent region.
t	Number of iteration.
d	The solutions with the best objective value.

Search economics for fog computing system

As described in Algorithm 1, the proposed algorithm consists of four primary operators: Resource arrangement, vision search, trade, and marketing research.

Input: regions quantity k, searchers quantity m, goods quantity n
Output: best solution d
$S, G \leftarrow \text{Initialization}(m, n)$
$R \leftarrow \text{Resource\_arrangement}(k, S, G)$
while *the termination criterion not met* **do**
  $S, G, E \leftarrow \text{Vision\_search}(S, G)$ $S, R \leftarrow \text{Trade}(S, R, E)$
  $G, d \leftarrow \text{Marketing\_research}(R, G)$
return d

Algorithm 1 Search economics for fog system deployment

In the algorithm, R, S, and G represent the sets of regions, searchers, and goods, respectively. At the initial stage (i.e., Initialization$(\cdot)$), the algorithm constructs solutions for each good and searcher at random, and the solution length is the total quantity of the gateways, fog servers, and edge servers. The role of Resource_arrangement$(\cdot)$ is to split the market to determine how to distribute limited resources to search the whole market evenly. The Vision_search$(\cdot)$ operator allows the searchers to exchange information with

goods to generate a new solution that combines the characteristics of searchers and goods and to evaluate the profit potential to obtain the optimal fitness value for each region. Trade(·) is a new operator for improved SE that is used to calculate the potential of each region to allow the searcher to move to a more suitable region. Subsequently, the proposed algorithm calculates and updates the market information using the Marketing_research(·) operator.

Figure 1 further shows a flowchart to explain how the proposed algorithm is applied to the deployment of the fog computing systems. This flowchart illustrates first the proposed system receives data from environment and sensors. As part of the proposed algorithm, the received data will be refined using the pre-processing procedure. Based on these data, a deployment plan (SEFSD) will be computed, which will then be used to deploy fog servers.

Resource arrangement

Resource arrangement involves splitting unknown markets into several submarkets and defining the specification of goods in each market. All invested interests (i.e., new solutions) must follow specifications for their respective markets. The specification of markets means that the goods generated in the same regions exhibit the same features. Each region keeps a certain number of goods in the search process. If the region produces more goods than this maximum, the region eliminates the goods exhibiting lower fitness. Searchers dynamically invest in different submarkets and exchange information to improve goods with high potential in their respective regions of investment. The method by which they estimate profit potential is introduced in a later section. For example, the SE depth-first search divides the market into four submarkets and selects two bits in the solution as identity bits. The market is divided into four regions and they are identified by bit pairs: (0, 0), (0, 1), (1, 0), and (1, 1). The identity bits comprise part of the solution and influence the search process and deployment situations. Therefore, the identity bits also affect the objective values.

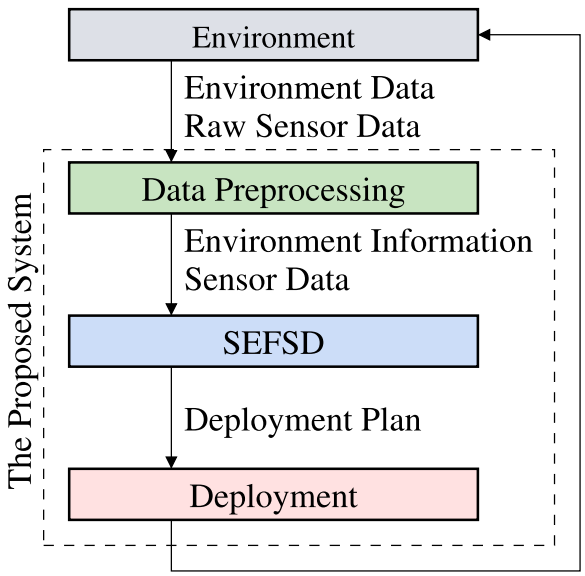

Fig. 1 The flowchart of the proposed algorithm for the fog computing system

In this paper, we propose an improved split-market mechanism for the deployment of fog computing systems to address the problem of large-scale solution spaces. In our algorithm, the solution space is divided into four regions, and the identity bit pairs extend to the segment of solution that represents the top layer devices installed in candidate locations. The high installation costs of top-layer devices and their distance to other connected devices substantially affect the devices in the next layer to be deployed. Moreover, solutions with the same number of deployed gateways have similar object values. The Resource_arrangement(·) operator is described as follows.

Input: Region quantity k, regions R, searchers $S = \{s_1, s_2, \ldots, s_m\}$, goods $G = \{g_1, g_2, \ldots, g_n\}$
Output: regions R
1 Divide R into k regions $R = \{r_1, r_2, \ldots, r_k\}$
2 **for all** r_i **in** R **do**
3 Define the range of quantity of gateways for r_i.
4 Allocate $\frac{m}{k}$ of s into r_i.
5 Allocate $\frac{n}{k}$ of g into r_i and set the gateways quantity within the range.
6 return R

Algorithm 2 Resource arrangement

When 30 candidate locations for gateways are present, the gateways are deployed in 25 or 21 candidate locations within the solution space rather than 5. If only five gateways are deployed, the gateways are far from each other. If the deployment pattern increases the number of gateways from 20 to 21, the gateways get closer to each other and the fiber installation cost decreases substantially. Therefore, the algorithm classifies similar numbers of gateways into the same submarket, and the objective values of solutions can therefore be expected to be different in each region. In Algorithm 2, the market R is split into k regions , and the number of regions k is used to allocate the goods G generated in the initial step according to the number of gateways in each region and to evenly assign the searchers S to each region. The number of gateways per region is obtained by dividing the candidate locations of gateways by the k regions. A region i only accepts the goods with numbers of gateways located within the interval of the region.

Following the aforementioned split-market method, the goods in the same region have the same number of gateways within the same interval. The number of regions into which the market is split can differ by problem or dataset. If the algorithm dynamically increases the number of regions, it will locate goods close to the optimal solution within the same region. However, if the number of regions remains low, the profit of goods will be uneven within the same region. Furthermore, a high number of regions will lead to several regions exhibiting the same profit potential. These two situations will lead to searcher misjudgment. Although this method splits the solution space by the number of gateways and generates sub-solution spaces of different sizes, the SE algorithm allocates resources according to each region's potential, and large regions are allocated more search resources. The details of the resource arrangement are introduced in the next section.

Vision search

After the resource arrangement stage, each region contains goods and searchers. In the first iteration, the region randomly allocates its searchers, and each searcher randomly generates new solutions using the crossover and mutation operators of the GA [24]. The main difference is that in the present algorithm, the searcher checks whether the identity segment of the new solution follows the specification of the region to which it belongs. If the identity segment does not follow the specification, the searcher randomly selects a bit of the new solution's segment to exchange with the identity segment of the original solution until the specification is met. When the new solution meets the specification and is superior to the original, it replaces the original solution, and the searcher evaluates the objective value of the new solution. The searchers do not exchange information with each other and do not reset the identity segments. Finally, the searcher updates the objective values of the new goods and estimates the expected profit for each region. The Vision_search(·) of the algorithm is described as follows.

Input: Regions $R = \{r_1, r_2, \ldots, r_k\}$, searchers $S = \{s_1, s_2, \ldots, s_m\}$, goods $G = \{g_1, g_2, \ldots, g_n\}$
Output: Searchers S, Goods G

```
Divide R into k regions R = {r_1, r_2, ..., r_k}
for all r_i in R do
    for all s_j in r_i do
        s_j randomly invest a goods g_l^i
        Evaluate objective value of each goods and searcher by Eq. (1)
for all r_i in R do
    Calculate expected value E_i by Eq. (10).
return S, G, E = {E_1, E_2, ..., E_i}
```

Algorithm 3 Vision search

Each searcher moves to the next search region according to the expected value, and the number of searchers for each region is dynamic. Searchers may not be allocated to a region because the region's expected value is low. The expected value E_i can be calculated using Eq. (10):

$$E_i = \mu_i \times \nu_i \times \rho_i, \tag{10}$$

where μ_i represents the investment situation for region i, ν_i represents the average profit of searchers in region i, and ρ_i represents the ratio of profit for the optimal goods to the total yield of all goods in region i. The investment situation μ_i can be calculated using Eq. (11):

$$\mu_i = \frac{t_i^a}{t_i^b}, \tag{11}$$

where t_i^a and t_i^b represent the number of iterations during which the searcher continuously invested and did not continuously invest in region i, respectively. How the investment times are counted and updated is introduced in the Marketing research section. The average profit for searchers ν_i can be calculated using Eq. (12).

$$\nu_i = \frac{\sum_{j=1}^{m} f(s_j^i)}{m \times \max(s_j^i)}, \tag{12}$$

where s_j^i represents searcher j in region i, m is the number of searchers in region i, and $f(\cdot)$ is the objective function for the problem, which is divided by the maximum profit and the number of searchers to normalize the value to within [0, 1]. It can avoid the expected value from the benchmark and prevent the other two indices at different levels from affecting the expected value. This index increases the probability of investment in the regions according to the experience of the searchers. The regions with potential profit become the investment target in the next iteration. The profit ratio for the optimal goods to the total yield of all goods ρ_i can be calculated using Eq. (13):

$$\rho_i = \frac{\min(g_l^i)}{\sum_{l=1}^{n} f(g_l^i)}, \tag{13}$$

where g_l^i represents the l-th good in region i,and n is the maximum number of goods in region i. This term calculates the ratio of the optimal profit to the total profit of all the products in the region. Although the searcher's strategy of accepting new solutions is greedy, the searcher uses the optimal solution of the previously searched region to exchange and generate new solutions, thus increasing search diversity. In addition, the expected value accounts for the potential profit of a region and

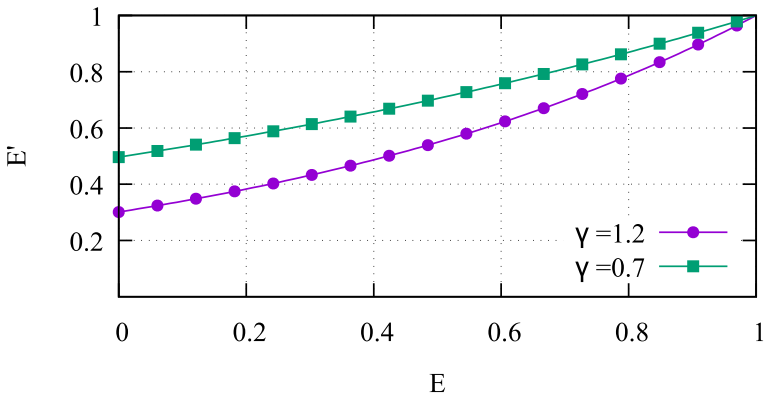

Fig. 2 Curves for transformed expected values when the parameter γ is 0.7 and 1.2

investment times. This mechanism can dynamically allocate computational resources to the high-potential regions to determine more favorable solutions.

Trade

The trade(·) operator is a new component of the proposed method used to improve upon the original SE. It determines the direction of the searcher because the original SE algorithm does not consider the potential of regions that are not converged. The trade(·) operator is used to determine the direction of the searcher in the next iteration in two stages: First it calculates the Hamming distances of regions and then uses tournament selection to determine the direction of the searcher. The goods in the same region are alike in the last search stage. To avoid wasting computational resources on regions with similar goods, the trade(·) operator increases the probability that the searchers invest in regions without convergence. The trade(·) of the algorithm is described as follows. The probability is modified by calculating the Hamming distances between goods in each region to determine the optimal solution. The total number of different bits in solutions is then calculated. If the Hamming distance is high, the region is not converged. The transformed expected value of the region according to Hamming distance is calculated using Eq. (14):

$$E_i' = (\frac{e^{E_i}}{e})^{\gamma}, \tag{14}$$

where the E_i', defined in Eq. (14), represents the new expected value and e is base of natural logarithm. The logarithmic base e of the originally expected values is divided by e to calculate the new expected value within the interval [0, 1]. γ is the parameter to adjust the expected value range. Adjustments to γ considerably affect the expected value of the nonconvergence region (Fig. 2). A lower parameter γ can obtain a higher range of expected values.

Input: Regions $R = \{r_1, r_2, \ldots, r_k\}$, searchers $S = \{s_1, s_2, \ldots, s_m\}$, goods $G = \{g_1, g_2, \ldots, g_n\}$, expected value $E = \{E_1, E_2, \ldots, E_i\}$
Output: Searchers S, goods G
for all r_i **in** R **do**
  Calculate the average hamming distance between each goods and the best goods.
Find the region with the largest average hamming distance, change its E by Eq. (14).
for all s_j **in** S **do**
  Change s_j's region by tournament selection based on E.
return S, R

Algorithm 4 Trade

After transformation based on the calculated Hamming distance, the expected value changes according to the convergence of each region. Each searcher s_j^i determines the search direction for the next iteration through tournament selection according to the transformed expected value E' of each region. The tournament selection involves randomly selecting regions o to determine the searcher's new search target region. Tournament selection was adopted as the selection method because it can prevent the region with the highest expected value from always winning. Avoiding unnecessary exploration increases search efficiency. Using the trade(·) operator reduces searching in the convergence region. The indices that affect search efficiency are the number of searchers continuously searching in region μ_i, the lowest average profit of investment in region ν_i, the lowest average profit of goods in region ρ_i, and the nonconvergence region with high Hamming distance.

Marketing research

After all the searchers finish investment, the Marketing research (·) operator is used to update the information for the whole market. This operator allows the operator of the next iteration to obtain new market information and the objective value of each region. The functions of this operator are to update the market information and to determine the new candidate solution in each region. The Marketing_research(·) operator of the algorithm is described as follows.

1) Determining the new candidate solution: Although the candidate goods in each region are updated, the goods are not influenced by each other because of their specifications. The previous section mentioned that the goods generated from the exchange would remain in their region and be compared with the original goods in the region. If one of the new goods is more profitable than the least-profitable original good, the operator replaces the less-profitable good with the new good. The operator compares the profitability of the least-profitable candidate goods with the new goods until all the new goods in the region have been evaluated.
2) Updating the market information: Before the solutions are exchanged and the objective values of the next iteration are evaluated, the Marketing_research(·) operator must update the investment record. The investment record r_i is updated as follows: If region i does not obtain any investment, the count of noninvestment t_b^i increases by 1. Otherwise, the count of investment t_a^i is set to 1 to avoid division by 0. If any searcher invests in the region, the count of investment t_a^i increases by 1. Otherwise, the count of noninvestment t_b^i is set to 1.

Input: Regions $R = \{r_1, r_2, \ldots, r_k\}$, searchers $S = \{s_1, s_2, \ldots, s_m\}$, goods $G = \{g_1, g_2, \ldots, g_n\}$
Output: Best solution d

```
for all r_i in R do
    Update goods {g_l ∈ r_i} by replace the worst goods in r_i.
    Find the best solution d among the searchers and goods.
    if |s_j^i| == 0 then
        t_b^i ← t_b^i + 1
        t_a^i ← 1
    else
        t_a^i ← t_a^i + 1
        t_b^i ← 1
return d
```

Algorithm 5 Marketing research

If all the new goods have been compared with the original goods and the market information for all regions has been updated, the algorithm proceeds to new iterations until the termination criterion is met.

Simplified example of the SE algorithm

To illustrate the proposed method, this section provides a simplified example of using SE to solve the fog computing deployment problem.

Step 1 consists of initialization of the searchers and goods by using randomly generated parameters from the whole solution spaces. In step 2, the solution spaces are split and the identity bits for each region are defined. The segment of the solutions for the goods are also constructed in this stage. Step 3 consists of the exchange of solution segments using the vision search operator. The searcher randomly selects goods to invest in and exchanges solutions with one another. The new solution generated in step 3 is illustrated in step 4 of Fig. 3. In step 4, the original solution is compared with the new one, and the solution with a lower objective value is eliminated. The goods generated from the investment remain in the region to which they belong to update the investment information and calculate the expected values. Step 5 involves the new operator of the SE algorithm. The searcher moves to another region according to the expected value. Finally, in step 6, the new goods are compared with the original goods in the same region, and the goods with higher profitability are retained.

Experimental results

Environment

The algorithm presented in this paper was run on a workstation with two Intel Xeon Silver 4410 cores of 2.1 GHz with 16 GB of memory each. The operating system was Ubuntu 18.04 LTS. The algorithms were developed in C++, and using the GNU Compiler Collection version 7.4.0. The proposed method was compared with the TF algorithm [25], GA [12], DBA [26], and DMGA [17]. Because the optimal solution to an NP-hard problem cannot be determined in a reasonable time using an exhaustive method, this study employed the rule-based TF algorithm as the baseline for comparison with other optimization algorithms. The TF algorithm prioritizes the deployment of servers in areas with the greatest workloads. In the ideal situation (i.e., the distribution of users is concentrated and enough candidate locations are available for deploying the devices.), the TF algorithm can be used to optimize the deployment of the fog systems. The GA was also used for comparison because it is longer established and well-known in

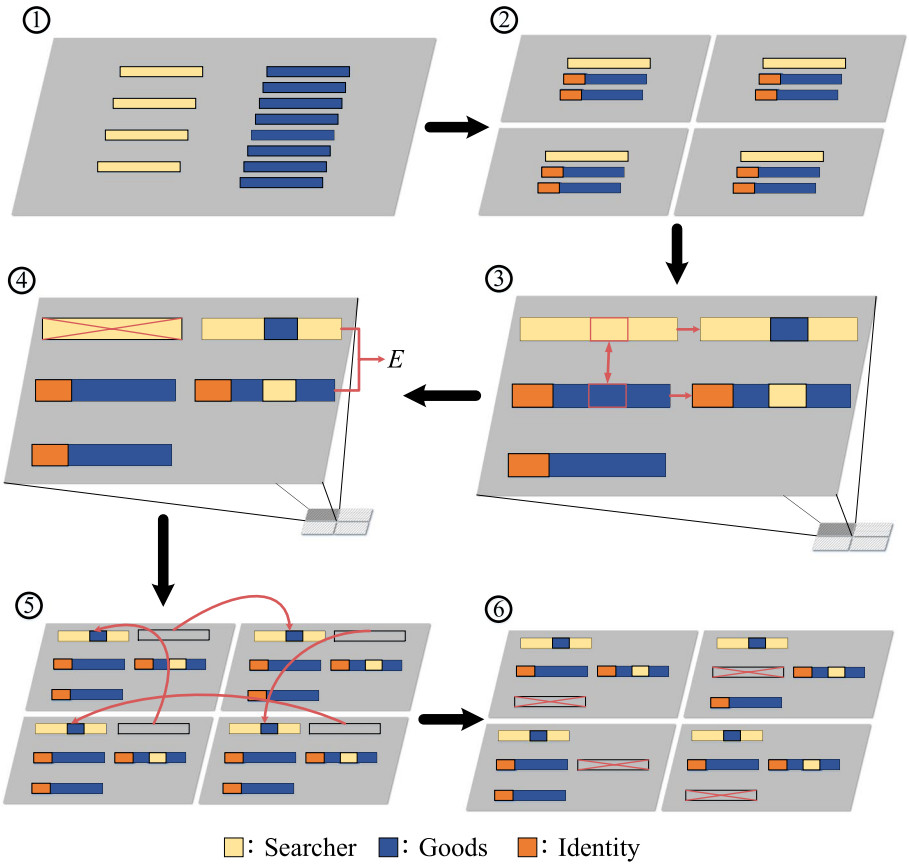

Fig. 3 Simplified illustration of using the improved search economics algorithm to solve the fog computing system deployment problem

the field of metaheuristic algorithms. Numerous studies have adopted the GA to solve deployment problems for wireless sensor networks, reporting adequate results. In this paper, the DBA and DMGA were selected for comparison because they can deliver more favorable results in complex solution spaces.

Datasets and encoding

The simulation of datasets is presented in study [17]. This study deployed 10 times as many devices in the same map size to analyze the performance of different algorithms in complex solution spaces. Therefore, dataset 1 (DS1) comprised 30 gateways, 150 fog servers, 700 edge servers, and 5, 000 users in a logistics center of 200 m $\times$ 180 m. The total number of devices was 880, which means that the solution space was 2^{880}. The visualization of DS1 is presented in Fig. 4. The details of the datasets are presented in Table 2. The purposes of adjusting the number of devices were to evaluate the performance of the algorithm using different devices and to observe the results for each algorithm in different solution spaces.

The proposed method adopts discrete encoding in addressing the fog computing system deployment problem. A simplified example of the encoding method is illustrated in Fig. 5. The solution comprises three parts; the first, second, and third parts represent the deployment pattern in candidate locations for the gateways, fog servers, and edge servers, respectively. A bit in the solution is represented in binary, in which a value of 1 indicates the installation of a device at the corresponding candidate location. A value of 0 indicates that no devices are installed at that candidate location.

Experiment for adjusting parameter

The proposed method integrates five adjustable parameters: The number of regions k, the number of searches m, the number of goods n in a region, the number of players for tournament selection o, and the parameter γ of trade operator. These parameters mainly comprise those present in the original SE algorithm. The algorithm analyzes and attempts to understand the features of a dataset by using those parameters. To analyze an unknown dataset, the algorithm compares the results of the greedy search and the search diversity to determine which search strategy is most suitable for the dataset. Subsequently, the algorithm uses this strategy to

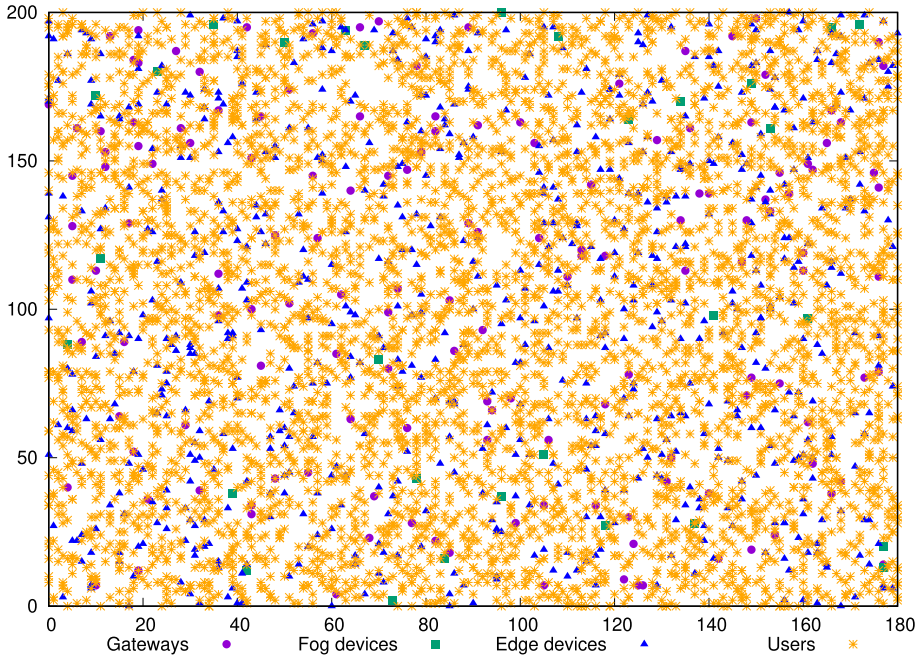

Fig. 4 Solution space for DS1, in which 880 devices are deployed in a 220 m × 180 m area

increase the greediness or diversity of the search. After the numbers of regions, goods, searches, and players are determined, the algorithm sets the aforementioned parameters and tests the effect of the parameter γ. The results of parameter adjustment are presented in Fig. 6, and the combinations of parameters are represented in the order (k, m, n, o).

The combinations in Fig. 6 can be divided into two types. The first type, which consists of different regions and numbers of searches, affects the search diversity. The second type, which consists of different numbers of goods and players, makes the algorithm greedily accept solutions. Through the adjustment of individual parameters, the greediness or diversity of a search can be increased. For example, the optimal objective value may be obtained using the set of parameters (4, 4, 4, 4). If the search diversity is increased and the parameters are set to (5, 5, 4, 4) or the search greediness is increased and the parameters are set to (4, 4, 5, 5) and a more favorable objective value is not obtained, the parameters are set to (4, 4, 4, 4) as a baseline before attempted adjustment of the numbers of searchers or goods. The results obtained using different combinations of searchers and goods are illustrated in Fig. 6. In the proposed method, when the number of goods increases, the search process converges rapidly because the number of goods is limited by the regional specification and increasing the number of goods increases the greediness of the algorithm. By contrast, increasing the number of searchers leads to slow convergence because the region may receive different information from multiple searchers from other areas to increase search diversity. As illustrated in Fig. 7, the objective values of the searchers and goods are similar after 40, 000 - 60, 000 evaluations. This is because the proposed method employs two complementary strategies: One that prioritizes greediness in global searches and one that prioritizes diversity in local searches.

Table 2 Details of each dataset

Dataset	Gateways	Fog Devices	Edge Devices	Users
DS1	30	150	700	5,000
DS2	30	150	1,000	5,000
DS3	30	500	700	5,000
DS4	200	150	700	5,000

This study employed the set of parameters (4, 4, 4, 4) as the baseline, and the numbers of searchers and goods were individually adjusted. The results are presented in Fig. 7 Increasing the number of goods and using the parameter set (4, 4, 8, 4) results in a more favorable objective value than does increasing the number for searchers and using the parameter set (4, 8, 4, 4). Increasing the number of goods improves the search diversity of the fog computing system deployment algorithm. The combination parameter γ was determined to affect the transformation of the expected value of the nonconvergence region. The results of the adjustment are illustrated in Table 3.

The parameter γ is initially set to 0.7. As it increases, the original expected value approaches 0, and the

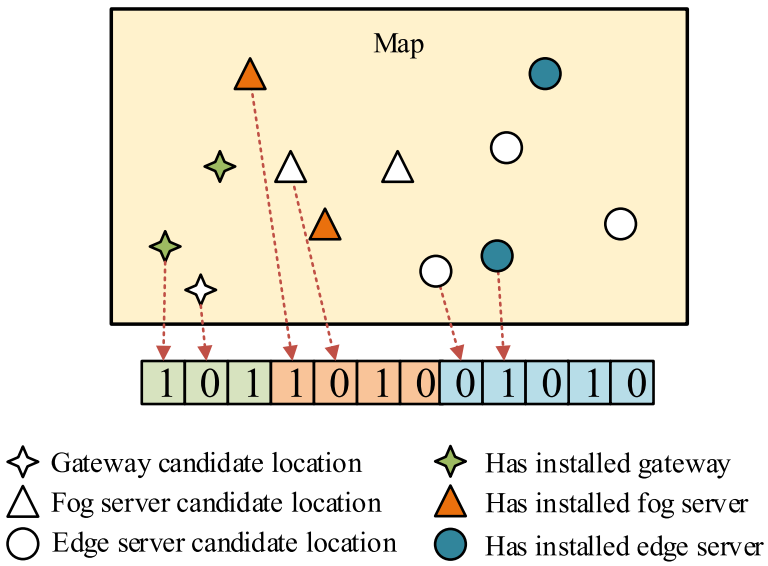

Fig. 5 Simplified example of the encoding method. Each solution is divided into three parts

transformed expected value approaches 0.5. The expected value for the nonconvergence region approaching 0 is unhelpful because it does not provide a more favorable objective value for the region. However, if the nonconvergence region exhibits a more favorable objective value, the region has more investment value. Therefore, it updates the original expected value to centering, and increasing or decreasing the value of parameter γ affects the objective value. The results indicate that the most favorable objective values were obtained when the parameter γ was set to 0.7.

Experimental results and discussion

This paper compares the performance of different algorithms, namely the TF algorithm, GA, DBA, DMGA, and SE algorithm, for addressing the fog computing system deployment problem. The results are presented in Table 4. Table 5 shows that the computation time of the proposed algorithm and deployment algorithms compared in this paper. These results show that TF and GA are faster than the other deployment algorithms for datasets DS1, DS2, DS3, and DS4. Since the design of the proposed algorithm considers much more factors in convergence process, its running time, of course, is larger than all the other deployment algorithms although it can find out better results than them. However, the good news is in that even though the other deployment algorithms use the same computation time with SEFSD they still cannot find the good deployment solution as the same as SEFSD can. This situation will be discussed

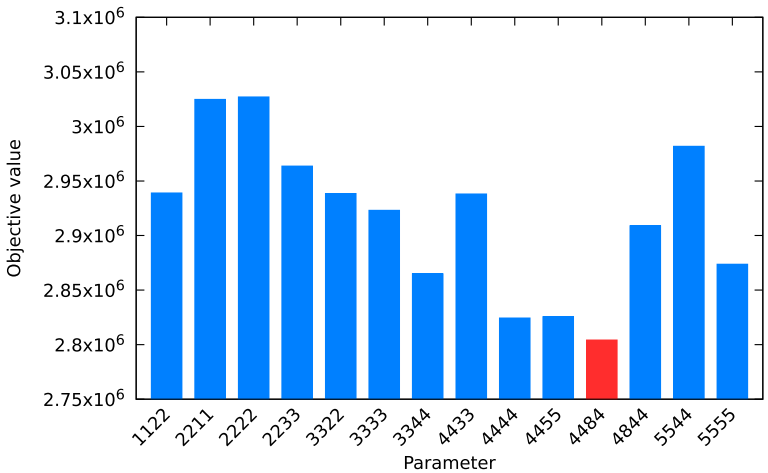

Fig. 6 Results of parameter adjustment of different parameter combinations

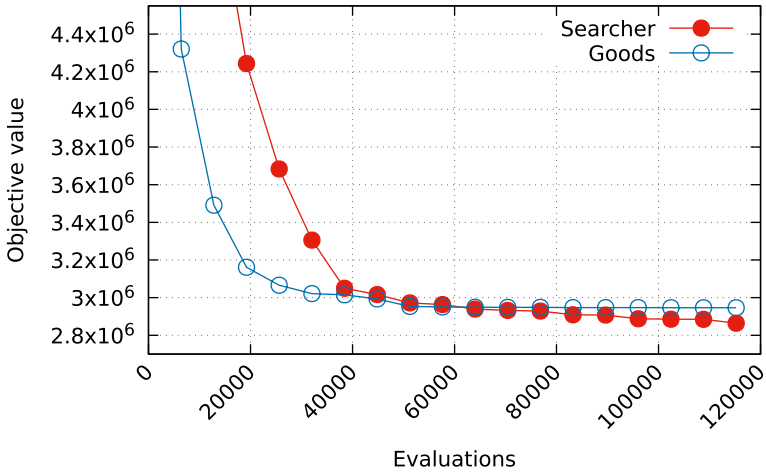

Fig. 7 Effect of the relationship between searchers and goods on the convergence speed of the algorithm

in detail shortly via the analysis of convergence curves of deployment algorithms.

Regardless of the complexity of the solution space, the proposed method eventually obtained the most favorable objective value. The results of the TF algorithm not only were less favorable than those obtained using the proposed method but also required a massive number of iterations to compute. For example, using dataset DS1, the TF algorithm obtained the same result, but only after 31, 500, 000 iterations. By contrast, other algorithms only required 1, 200, 000 iterations to obtain the same result. Compared with the conditional exhaustive search method, the proposed method obtained more favorable solutions for fog computing system deployment problems in the same number of iterations. As indicated in Table 6, the search efficiency of the SE algorithm did not change when the dataset changed, but the TF algorithm and GA did. However, the TF algorithm requires a massive number of iterations to complete, and the GA's s search strategy resembles a random search when used with complex datasets. The SE algorithm yielded results considerably superior to those of the DBA and DMGA using DS2 and DS3 because the solution lengths of DS2 and DS3 were greater than that of DS1, and their solution spaces were more complicated than those of DS1 and DS4.

Figure 8 illustrates the convergence of each algorithm using DS1 and the effects of algorithms employing different strategies. The TF algorithm always deploys servers with a massive workload preferentially and therefore does not search regions with unfavorable objective values. Although its convergence curve indicates that its search begins in the region with the least favorable objective value, the algorithm does not search the whole region. The proposed method obtains a more favorable solution, which is not located in the search space of the TF algorithm. The GA retains chromosomes in the convergence process through selection. A convergence strategy that does not require retaining the optimal chromosomes to receive the crossover operator may be inefficient for searches in a complex solution space. As indicated by its convergence curve, the GA cannot search for the optimal direction and initiate a local search when the optimal direction is located.

The GA's search diversity is similar to that employed in a random search of a complex solution space. The DMGA employs a greedy search strategy, which involves searching in the direction of the monkey with the highest objective value. The cooperation process integrated into the DMGA allows the algorithm to quickly locate the region with the most favorable objective value, but the algorithm does not have a mechanism to avoid local optima. The curve

Table 3 Effect of parameter γ

γ	Objective value
0.5	3,000,070
0.6	2,825,300
0.7	2,804,210
0.8	2,867,990
0.9	2,864,710

Table 4 Results of various algorithms for fog computing system deployment across different datasets

Dataset	TF	GA	DBA	DMGA	SEFSD
DS1	4,328,741	8,969,220	4,456,138	4,275,777	3,390,791
DS2	1,645,360	3,459,700	3,366,342	3,287,686	1,189,066
DS3	2,725,890	8,930,096	4,449,198	3,764,370	1,902,618
DS4	4,079,570	9,183,044	4,552,032	4,464,314	2,901,532

Table 5 Computation time of various algorithms for fog computing system deployment across different datasets

Dataset	TF	GA	DBA	DMGA	SEFSD
DS1	1,444.70	1,337.90	2,493.33	1,522.67	4,548.20
DS2	2,199.70	1,908.27	3,366.23	2,072.40	6,156.20
DS3	560.17	670.93	1,393.40	797.60	2,865.07
DS4	1,461.27	1,522.77	3,587.23	2,122.97	6,590.10

Table 6 Objective values of different percentages among the various algorithms compared with the proposed method

Dataset	TF	GA	DBA	DMGA
DS1	20.6%	66.0%	35.4%	35.2%
DS2	27.7%	65.6%	64.7%	63.8%
DS3	30.2%	78.7%	57.2%	49.5%
DS4	28.9%	68.4%	36.3$	35.0%

indicates that once DMGA has located the most favorable region, it no longer searches in less favorable regions.

To ensure that the search is thorough and conducted in the correct direction, the DBA uses loudness and pulse emission rates as parameters to dynamically switch between local and global search strategies. The convergence curve is between those of the GA and DMGA, but it also converges in regions with the most favorable objective values, as the DMGA does. Regarding the SE algorithm, in the initial iteration stage, the search strategy of goods involves rapidly moving to the regions with less favorable objective values and conducting a local search. The DBA and DMGA stop at the regions with the optimal objective value, whereas the SE algorithm is not limited to this region because it splits the solution space into several regions and allocates search resources to regions that have not yet been explored in a given period. Furthermore, the trend illustrated in Fig. 8 indicates that the searchers continually exchange solutions to enhance the search diversity even if all goods exhibit convergence in the region to which they belong.

The article [20] provides further information on the time complexity of the proposed algorithm. The overall time complexity of SEFSD is in the order of $O(nkt)$, where n is the number of searchers, k is the number of subsolutions, t is the number of iterations. Moreover, the time complexity of SEFSD is also similar to most metaheuristic algorithms.

In order to apply the proposed algorithm to real-world applications, two important considerations must be addressed. The first issue is about the problem definition which needs to take into account much more factors, such as power consumption, network topology, reliability, communication delay, collision, number of devices, as so forth. The second issue is about the period change of the system and environment. With these considerations, we can then let the proposed algorithm more useful for the deployment of fog system. It is important to note that due to the schema for encoding of the proposed algorithm is in the form of binary string, there will be a limitation to the proposed algorithm, which means that all the possible locations of the fog server will be limited to a specific number of locations.

Conclusion

The deployment of the fog computing systems must account for the cost, number of connected devices, workload of servers, latency time, cover rate, and service capacity of servers, which was formerly considered a complex problem. Therefore, this study employed an

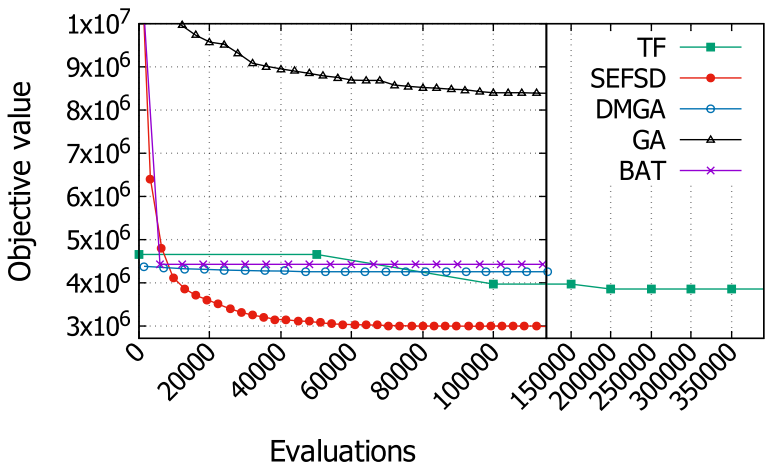

Fig. 8 Convergence curves for each algorithm simulated using DS1

SE algorithm suitable for handling complex solution spaces to solve the fog computing system deployment problem. This improved deployment algorithm was designed to address the hierarchical distribution architecture of the fog computing system by integrating split solution spaces and accounting for the number of gateways required. The solution spaces are logically divided according to the number of top-layer devices. The proposed method evaluates the potential of each region to dynamically allocate computational resources to subsolution spaces with high potential. According to the experimental results, the proposed method increases the objective value of each dataset by more than 50%. Compared with the conditionally exhaustive algorithms, the proposed method can deploy servers with the lowest workloads, obtain the lowest cost of deployment, and result in the lowest count of violated constraints. In future studies, the numbers of searchers, goods, and players should be adaptively adjusted according to the iteration and the features of solution spaces. In the search process, the algorithm may encounter situations in which its typical search strategy is unsuitable. If the algorithm can dynamically evaluate the features of solution spaces and immediately adjust the search strategy, it can obtain more favorable objective values and exhibit greater efficiency. Because the proposed algorithm is capable of determining a better deployment plan for fog servers, therefore we intend to apply it to other deployment problems in a variety of network environments in the future (for example, vehicular ad-hoc networks or sixth generation mobile systems).

Acknowledgements
The authors would like to thank the editor and anonymous reviewers for their valuable comments and suggestions on the paper.

Authors' contributions
Conception and design of study: Huan Chen, Tai-Lin Chiu, Ming-Chao Chiang, and Chun-Wei Tsai. Drafting the manuscript: Huan Chen, Wei-Yan Chang, and Tai-Lin Chiu. Implementation and acquisition of data: Wei-Yan Chang and Tai-Lin Chiu.

Funding
This research was partially supported in part by Qualcomm under the Taiwan University Research Collaboration Project, and National Science and Technology Council (NSTC) of Taiwan, R.O.C., under the grant numbers NSTC-110-2221-E-005-032-MY3, NSTC-111-2634-F-005-001, NSTC-111-2222-E-110-006-MY3, NSTC-112-2634-F-110-001-MBK, and NSTC-112-2628-E-110-001-MY3.

Availability of data and materials
The datasets generated during and/or analysed during the current study are available in https://ailab.cse.nsysu.edu.tw/datasets/fog_computing/.

Declarations

Competing interests
We declare the following:
- The work described has not been submitted elsewhere for publication, and all the authors listed have approved the manuscript that is enclosed.
- We have read and abided by the statement of ethical standards for manuscripts submitted to Journal of Cloud Computing.

Received: 25 November 2021 Accepted: 15 June 2023
Published online: 15 July 2023

References

1. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of Mobile Cloud Computing Workshop. ACM, pp 13–16
2. Hu P, Dhelim S, Ning H, Qiu T (2017) Survey on fog computing: Architecture, key technologies, applications and open issues. J Netw Comput Appl 98:27–42
3. Luan TH, Gao L, Li Z, Xiang Y, Sun L (2015) Fog computing: Focusing on mobile users at the edge. arXiv 1–11. http://arxiv.org/abs/1502.01815
4. Hong CH, Varghese B (2019) Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. ACM Comput Surv 52(5):1–37
5. Aditya S, Figueiredo JR (2017) Frugal: Building degree-constrained overlay topology from social graphs. In: Proceedings of IEEE International Conference on Fog and Edge Computing. IEEE, pp 11–20
6. Narula SC, Ho CA (1980) Degree-constrained minimum spanning tree. Comput OR 7(4):239–249
7. Ning A, Ma L, Xiong X (2008) A new algorithm for degree-constrained minimum spanning tree based on the reduction technique. Prog Nat Sci 18(4):495–499
8. Ribeiro CC, Souza MC (2002) Variable neighborhood search for the degree-constrained minimum spanning tree problem. Discret Appl Math 118(1):43–54
9. Singh K, Sundar S (2019) A hybrid steady-state genetic algorithm for the min-degree constrained minimum spanning tree problem. Eur J Oper Res 276(1):88–105
10. Singh K, Sundar S (2020) A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem. Soft Comput 24(1):2169–2186
11. Krishnamoorthy M, Ernst AT, Sharaiha YM (2001) Comparison of algorithms for the degree constrained minimum spanning tree. J Heuristics 7(6):587–611
12. Holland JH (1992) Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge
13. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
14. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of International Conference on Neural Networks, vol 4. IEEE, pp 1942–1948
15. Tsai CW (2015) Search economics: A solution space and computing resource aware search method. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Kowloon, China. IEEE; pp 2555–2560
16. Yang L, Cao J, Liang G, Han X (2015) Cost aware service placement and load dispatching in mobile cloud systems. IEEE Trans Comput 65(5):1440–1452
17. Lin CC, Yang JW (2018) Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0. IEEE Trans Ind Inform 14(10):4603–4611
18. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv 35(3):268–308
19. Tsai C-W (2016) An effective WSN deployment algorithm via search economics. Comput Netw 101:178–191
20. Tsai C-W, Liu S-J (2018) An effective iot service-to-interface assignment algorithm via search economics. IEEE Internet Things J 5(3):1708–1718
21. Tsai C-W, Liu S-J (2020) An effective hyper-dense deployment algorithm via search economics. J Ambient Intell Humanized Comput 11:2251–2262
22. Tsai C-W, Fang Z-Y (2021) An effective hyperparameter optimization algorithm for dnn to predict passengers at a metro station. ACM Trans Internet Technol 21(2):1–24
23. Tsai K-H, Tsai C-W, Chiang M-C (2022) An effective metaheuristic-based pruning method for convolutional neural network. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, pp 679–682
24. Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. Found Genet Algorithm 1(1):69–93

25. Li Y, Wang S (2018) An energy-aware edge server placement algorithm in mobile edge computing. In: Proceedings of IEEE International Conference on Edge Computing. IEEE, pp 66–73
26. Mishra SK, Puthal D, Rodrigues JJ, Sahoo B, Dutkiewicz E (2018) Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. IEEE Trans Ind Inform 14(10):4497–4506

Publisher's Note