Dynamic Weighted Fog Computing Device Placement Using a Bat-inspired Algorithm with Dynamic Local Search Selection

Chun-Cheng Lin · Der-Jiunn Deng[*] · Sirirat Suwatcharachaitiwong · Yan-Sing Li

Abstract This work investigates the dynamical weighted deployment of mobile fog computing devices to support a mobile edge computing environment, in which each edge device is associated with a weight to reflect its importance based on the application. Since edge devices are mobile and could be switched off, it is challenging to dynamically optimize the deployment to adapt to dynamic change. This work further models the problem mathematically and solves it by a bat-inspired algorithm (BA), which searches the optimal solutions by simulating the food-searching behavior of bats via echolocation. Furthermore, three local search methods designed specifically for this problem are integrated into the BA, and a dynamic local search selection mechanism is proposed to adjust the probabilities of choosing the three local search methods iteratively in the BA main loop. Simulation results show outperformance of the proposed BA over the BA

C.-C. Lin[1, 2,3]
[1] Department of Industrial Engineering and Management,
National Chiao Tung University, Hsinchu 300, Taiwan;
[2] Department of Business Administration,
Asia Univesity, Taichung 413, Taiwan;
[3] Department of Medical Research,
China Medical University Hospital,
China Medical University, Taichung 404, Taiwan
e-mails: cclin321@nctu.edu.tw

S. Suwatcharachaitiwong · Y.-S. Li
Department of Industrial Engineering and Management,
National Chiao Tung University, Hsinchu 300, Taiwan
e-mails: sirirat.su@psu.ac.th, ocean79513@hotmail.com

D.-J. Deng
Department of Computer Science and Information Engineering,
National Changhua University of Education, Changhua 500, Taiwan
e-mail: djdeng@cc.ncue.edu.tw

[*] D.-J. Deng is the corresponding author of this paper (E-mail: djdeng@cc.ncue.edu.tw).

without local search and the previous approach.

Keywords Fog computing, mobile edge, nature-inspired algorithm, deployment, dynamics, node-weighted graph.

1. Introduction

Cloud/fog/edge computing and the Internet of Things (IoT) play a crucial role in Industry 4.0 [1]. Generally, in a smart factory (or other industrial applications defined within a limited geographical region), the cloud server is maintained in a centralized location, causing a too long latency time to respond to the requests from a huge number of IoT sensors distributed in a large working area. Fog computing extends cloud computing with distributing computing, storage, and networking resources to the network edge to support IoT applications with requirements of real-time and low-latency response, location awareness, end-device mobility, scalability, heterogeneity, transient storage, high-speed data dissemination, decentralized computation, and security [2], [3].

This work focuses on the optimization problem of deploying fog computing devices to support the requests from mobile edge devices in a geographical area. Deployment of fog systems has appeared in several applications, e.g., intelligent logistic center [4], local area networks [5], medical cyber-physical system [6], and vehicular network [7]. The deployment of other networking systems was widely studied, e.g., deploying cloudlets in wireless metropolitan area networks [8], deploying wireless sensor network in a 3D environment [9], and dynamically deploying router nodes in wireless mesh networks (WMN-dynRNP) [10].

The WMN-dynRNP in [10] is to find the optimal placement of mesh routers to cover mesh clients to adapt to a dynamic environment in which mesh clients can dynamically and autonomously move to any positions in the placement area and turn on or off their own network access. Inspired by [10], this work considers the dynamical deployment of a fog-computing system (FCS) consisting of fog-computing devices (fog device for short) and edge devices in a geographical placement area in which each edge device is a mobile device (e.g., smartphone, tablet, connected vehicles, embedded system, etc.) and can serve as an edge gateway (i.e., it could communicate with IoT sensors and other static edge devices through Bluetooth); the data collected by IoT sensors is transmitted to and preprocessed by an edge device, which then communicates with a fog device for further processing or being replayed to the cloud server.

It has been common in previous works to extend optimization problems into a weighted version to reflect the importance degree of each entity concerned in the system, e.g., routing in weighted-railway-station network graph [11], shortest path in node-weighted or edge-weighted graphs [12], [13], [14], IoT data placement in fog infrastructures based on weighted graph partition [15], gateway placement in WMNs based on node-weighted graphs [16], and channel assignment in WLANs based on edge-weighted graphs [17]. In real-world FCS applications, some edge devices that need shorter response time to achieve their requests (e.g., in smart factory, those edge devices that handles the data from crucial machine components) must have high priorities to connect the FCS. Therefore, this work further extends the FCS placement problem with assumption that each edge device has a weight to reflect its importance degree in the FCS, and should give a higher priority to connect the edge devices with a larger weight.

The concerned FCS placement problem is a facility location problem, which has been shown to be NP-hard [18]. Additionally, dynamics further complicates the problem to be solved. To solve this problem, this work proposes a bat-inspired algorithm (BA), which is a metaheuristic algorithm that simulates the food-searching behavior of bats via echolocation to find sufficiently good solutions [19], [20]. To increase the solution-searching ability of BA, the proposed BA is integrated with three local search (LS) methods that are designed specifically for the problem. Furthermore, this work proposes a dynamic LS selection mechanism based on [21] in which each iteration of the BA main loop dynamically choose one of the three LS methods proposed in this work, so as to choose an appropriate LS method more flexibly during the BA process. From simulation results, the proposed improved BA looks promising and is more stable than the previous PSO approach for a dynamic unweighted placement problem for WMNs [10].

2. Preliminaries

2.1. Dynamic weighted fog device placement in FCSs

The FCS consists of four types of device: cloud server, fog devices, edge devices, and IoT sensors. Each fog device has different-size radio coverage (represented as a circle centered at this fog device), and the edge devices within the radio coverage of certain fog device can communicate with one another. This work considers deploying only *n* fog devices and *m* edge devices on a two-dimensional geographical area, because it is supposed that each fog device at any positions in the placement area can always communicate with the cloud center through cellular networks; and positions of IoT sensors are fixed. Let the set of nodes in an FCS be represented by $U = F \cup C$, where $F = \{f_1, f_2, \ldots, f_n\}$ in which f_i is the fog device labeled by i with a radio coverage with radius γ_i; $C = \{c_1, c_2, \ldots, c_m\}$ in which c_j is an edge device labeled by j.

This work further assumes that each edge device is associated with a weight, and the FCS has a higher priority to address the requests from the edge device with a larger weight. Let $w(c_j)$ be the weight of edge device c_j. To associate each node with a weight consistently, this work also associates each fog device f_i with a weight $w(f_i)$. Thus, we have the weight function w: $U \rightarrow \mathbb{R}$.

The FCS considers a dynamic environment, i.e., all fog and edge devices have mobility (i.e., their positions can be not fixed), and edge devices could dynamically be turned on or off at different times. The concerned problem is to adjust the positions of fog devices with mobility periodically, so that the adjusted network topology of the FCS can adapt to dynamic network changes. This work divides time into multiple epochs, and makes adjustment decision at each epoch. At the τ-th epoch, each edge device $c_i \in C$ is placed at $D_\tau(c_i) \in \mathbb{R}^2$ on the placement area. Considering serving these edge devices, the solution to the problem at the τ-th epoch is to determine positions of fog devices that are represented by $D_\tau(F) = \{D_\tau(f_1), D_\tau(f_2), \ldots, D_\tau(f_n)\}$, in which $D_\tau(f_i)$ denotes the position of fog device f_i on the placement area at the τ-th epoch, for $i \in \{1, \ldots, n\}$. Let Υ_i^τ denote the radio coverage range centered at $D_\tau(f_i)$ with radius γ_i. To determine the positions of fog devices at each τ-th epoch, this work establishes a network topology graph $G_\tau = (U_\tau, E_\tau)$ for this epoch, in which $U_\tau = F \cup C \setminus S_\tau$, where S_τ denotes set of the edge devices that are turned off; for each pair of fog devices $f_i, f_j \in F$, a fog-fog link $(f_i, f_j) \in E_\tau$ exists if $\Upsilon_i^\tau \cap \Upsilon_j^\tau \neq \phi$; for any edge device $c_j \in C \setminus S_\tau$ and any fog device $f_i \in F$, if $D_\tau(c_j) \in \Upsilon_j^\tau$, an edge-fog link $(c_j, f_i) \in E_\tau$ exists.

For problem simplification, this work assumes that if radio coverages of two fog devices overlap, then the two fog devices can communicate with each other. This is a reasonable assumption because coverage overlap implies closeness of their positions, favoring their communications.

It is challenging to find a connected FCS topology graph G_τ, i.e., graph G_τ may include at least two graph components. Without loss of generality, suppose that G_τ has h subgraph components G_τ^1, …, G_τ^h in which

$G_\tau^i \cap G_\tau^j = \varnothing$ for $i, j \in \{1, \ldots, h\}$ and $h \geq 1$ (i.e., $G_\tau = G_\tau^1 \cup G_\tau^2 \cup \cdots \cup G_\tau^h$). This work considers the following measures for evaluating performance of the FCS topology graph. The first measure is the total weighted network connectivity of the FCS, to be maximized. Nonetheless, when maximizing the weighted network connectivity of a large FCS, fog devices may not be able to cover all edge devices. The second measure is the total weighted edge device coverage, to be maximized. Recall that the weight function w is introduced above. The *weighted network connectivity* is calculated as follows:

$$\varphi(G_\tau) = \max_{i\in\{1,\ldots,h\}} \{\textstyle\sum_{j\in G_\tau^i} w(c_j) + \sum_{j\in G_\tau^i} w(f_i)\}\,. \tag{1}$$

That is, it is the sum of all node weights of the largest subgraph component in G_τ. In general, the connectivity among fog devices constitutes the main network backbone, and hence each fog device has a larger weight value than all edge devices to reflect its higher importance degree. The *weighted edge device coverage* is represented as follows:

$$\psi(G_\tau) = \sum_{i\in\{1,\ldots,m\};\, d_\tau(c_i)>0} w(c_i) \tag{2}$$

where $d_\tau(c_i)$ represents the degree of node c_i in G_τ.

The concerned problem aims to dynamically determine a placement $D_\tau(F)$ of fog devices in the FCS at each τ-th epoch so that both the weighted network connectivity $\varphi(G_\tau)$ and the weighted edge device coverage $\psi(G_\tau)$ are maximized. Thus, the concerned problem is described as follows: Given an FCS consisting of n fog devices and m edge devices and a weight function w mapping each node to a real number so that the edge device with a larger weight should have a higher priority to be served, a network topology $G_\tau = (U_\tau, E_\tau)$ underlying the FCS at each τ-th epoch is constructed as described above. Consider a dynamic environment where each edge device has mobility, and could be turned on or off anytime. Consider a rectangular placement area of size $W\times H$, in which the position $D_\tau(f_i)$ of each fog device f_i is known at each τ-th epoch. The problem is to determine the placement of n fog devices $D_\tau(F) = \{D_\tau(f_1), D_\tau(f_2), \ldots, D_\tau(f_n)\}$ at each epoch τ, so as to simultaneously maximize the weighted network connectivity $\varphi(G_\tau)$ and the weighted edge device coverage $\psi(G_\tau)$.

2.2. Bat-inspired algorithm

When the concerned optimization problem is too difficult to be efficiently solved, metaheuristics may efficiently provide sufficiently good solutions to the problem, e.g., simulated annealing [22], genetic algorithm [23], and PSO [10]. Bat-inspired algorithm (BA) [19] is a nature-inspired metaheuristic algorithm which simulates a population of bats (agents) cooperatively searching for a food (optimal solution) through echolocation [24]. By echolocation (i.e., emitting sound pulses), bats can recognize the food in a fully dark environment, speculate the distance away from the food, and differentiate the food from environmental obstacles. The velocity of a sound pulse is generally v = 340 m/s, and the pulse frequency (where the frequency range of ultrasonic waves is between 25 kHZ and 150 kHz) follows the formula of wavelength: $\lambda = v / f$, such that the wavelength λ is between 2 mm and 14 mm, and the sound strength even reaches 110 dB.

The BA has been employed to tackle a lot of network optimization problems, e.g., wireless sensor network deployment problem [9], service allocation problem in fog server [25], and security and QoS optimization [26]. BA was also applied to various areas, e.g., engineering problems [27], UCAV path planning problem [28], economic dispatch problem [29], medical goods distribution problem [30], and price-forecasting problem for oil industry [31].

By following the physical characteristics of echolocation, the design of BA has the following assumptions:

- All bats can speculate the distance away from the food through echolocation, and they can identify the difference between the food and environmental obstacles.
- Each iteration t of the BA main loop considers each bat k to fly from position X_k^t to a new position at a velocity V_k^t, which is determined by a frequency f_k^t of the emitted pulse in $[f_{min}, f_{max}]$ = [0, 100], while the pulse wavelength is fixed.
- At the t-th iteration, each bat k locally searches for the food (i.e., the optimal solution) according to the pulse loudness $A_k^t \in [0, 1]$ and the pulse emission rate $r_k^t \in [0, 1]$.
- If the position of a bat is close to the food, the bat emits pulse with a smaller pulse loudness and a greater the pule emission rate.

Without loss of generality, consider to adopt the BA to solve a maximization problem, i.e., a fitness function is employed to evaluate the performance of a solution, which is encoded as a position of some bat. Let η be the number of bats. At first (i.e., $t = 0$), the position X_k^0, velocity v_k^0, frequency f_k^0, pulse loudness A_k^0, and pulse emission rate r_k^0 of each bat k for $k \in \{1, 2, \ldots, \eta\}$ are initialized randomly within the given ranges. Then, each iteration of the BA main loop randomly sets each bat k's pulse frequency f_k^t, velocity V_k^t, and position X_k^t as follows:

$$f_k^t = f_{\min} + (f_{\max} - f_{\min})\beta \tag{3}$$

$$V_k^t = V_k^{t-1} + (X_* - X_k^{t-1}) f_k^t \tag{4}$$

$$X_k^t = X_k^{t-1} + V_k^t \tag{5}$$

where β is a number generated randomly from the uniform distribution $U(0, 1)$ for $[0, 1]$; and X_* denotes the best position (solution) that has been searched by all bats so far.

The remaining steps of the BA are mainly divided into two parts. The first part considers each bat k. Check whether a random number from $U(0, 1)$ is greater than the pulse emission rate r_k^t of bat k. If true, one of the best positions is chosen randomly, and an LS on this best position is conducted as follows:

$$X_{new} = X_{old} + \varepsilon A^t \tag{6}$$

where X_{old} (resp., X_{new}) is the original (resp., updated) position at the t-th iteration; ε is a number generated randomly from $U(0, 1)$; A^t is the loudness average, i.e., $A^t = \sum_k A_k^t / \eta$.

The second part produce a new position of each bat k through the LS in (6). Then, it checks whether the new position has a better fitness and a random number from $U(0, 1)$ is less than the pulse loudness A_k^t of bat k. If true, this new position replaces the old position in the bat population, and its pulse emission rate and its pulse loudness are updated as follows, respectively:

$$r_k^{t+1} = r_k^0 (1 - e^{-\gamma t}) \tag{7}$$

$$A_k^{t+1} = \alpha A_k^t \tag{8}$$

where $\alpha = \gamma = 0.9$. At the end of each iteration, all the bats are sorted according to their fitness values, and the positions with the best fitness value are recorded.

3. The Proposed Improved BA

This section proposes an improved BA with dynamic LS selection to solve the problem concerned in this work, in which the solution representation is stated first; then, the fitness function is defined to evaluate the solution quality; then, three LS methods are proposed to improve the solution-searching ability; then, a dynamic LS selection mechanism is proposed for choosing the best one of the three LS methods at each iteration; finally, the details of the proposed algorithm is described.

3.1. Solution representation

The solution for the concerned problem is the positions of *n* fog devices on a two-dimensional $W \times H$ placement area, in which the origin is at the bottom left corner of the $x \times y$ plane. Hence, a solution candidate of the problem is the coordinates of *n* fog devices on the $x \times y$ plane. On the other hand, in the BA, the position of each bat *k* is corresponded to a candidate solution, consisting of two vectors as follows: vector $X_k^t = (x_{k1}^t, x_{k2}^t, ..., x_{k(2n)}^t)$ stores the current position of bat *k*, in which $(x_{k(2i-1)}^t, x_{k(2i)}^t)$ is the coordinate of fog device f_i on the placement area, for i = 1, 2, …, *n*; vector $V_k^t = (v_{k1}^t, v_{k2}^t, ..., v_{k(2n)}^t)$ stores the velocity of bat *k*, where v_{ki}^t is the velocity of fog device f_i adopted to search for the new position of f_i on the placement area, for i = 1, 2, …, *n*.

Since all the fog devices must be deployed within a $W \times H$ placement area, the corresponding decision variables are restricted to the following constraints: $\forall i \in \{1, \ldots, n\}$,

$$0 \le x_{k(2i-1)}^t \le W, \quad 0 \le x_{k(2i)}^t \le H, \tag{9}$$

$$-W \le v_{k(2i-1)}^t \le W, \quad -H \le v_{k(2i)}^t \le H. \tag{10}$$

To avoid from moving too drastically between two epochs, the velocity of bat *k* is restricted to the following constraint: $\forall i \in \{1, \ldots, 2n\}$,

$$-V_{\max} \leq v_{ki}^t \leq V_{\max} \tag{11}$$

where V_{max} is a given value and cannot be greater than bounds W and H. In practice, Constraint (11) makes sense because mobile edge devices cannot move too far between two epochs.

For the whole bat population, the best positions X_* found by all bats so far is stored at each iteration. Note that there may not be only one best position in the bat population. After the BA is terminated, X_* is the final output solution.

3.2. Fitness evaluation

At the t-th iteration, bat k finds the position vector X_k^t of fog devices, and then a topology graph $G_{t,k}$ underlying X_k^t is established as described in Subsection 2.1. To evaluate the quality of the topology graph, the concerned two main performance measures of FCSs are network connectivity $\varphi(G_{t,k})$ in (1) and edge device coverage $\psi(G_{t,k})$ in (2). Hence, in the proposed BA, the fitness value $g(X_k^t)$ of a position X_k^t is used to evaluate the performance of the position, and is represented as the following weighted sum of the two performance measures:

$$g(X_k^t) = \lambda \cdot \frac{\varphi(G_{t,k})}{\sum_{j=1}^m w(c_j) + \sum_{j=1}^n w(f_j)} + (1-\lambda) \cdot \frac{\psi(G_{t,k})}{\sum_{j=1}^m w(c_j)} \tag{12}$$

where topology graph $G_{t,k}$ is established according to the placement X_k^t; parameter λ is a given value within [0, 1], balancing the two normalized performance measures.

3.3. Three LS methods

The value of solving a problem by a novel metaheuristic algorithm is to propose some problem-specific operators to improve the solution-searching ability. Commonly, LS is adopted to improve the solutions produced by heuristics. In this work, three LS methods are adopted to the elite bats (i.e., the half of bats with better fitness values than the other half). The idea of the proposed LS methods is to move the positions of one or multiple fog devices locally while the other fog devices stay at the original positions.

The proposed three LS methods are based on two single-node LS types as follows. The first single-node LS type (Fig. 1(a)) is to move a fog device (the black node in Fig. 1(a)) to a random position in a circle area with center at the original position of the fog device with a given radius δ. This LS type is common, but the random position within the circle may be very close to the original position, such that there is almost no movement. To solve this issue, the second single-node LS type (Fig. 1(b)) is to move the concerned fog device to a random one of eight candidate positions (i.e., the eight gray positions in Fig. 1(b)). That is, the second single-node LS type forces the fog devices to have an obvious movement.

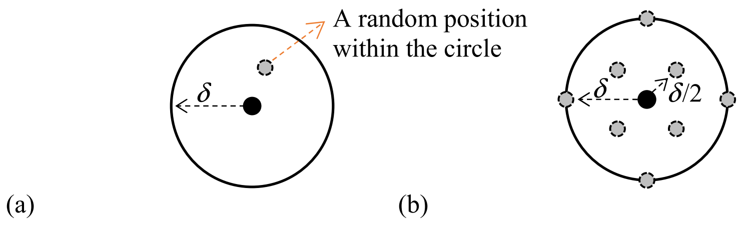

Fig. 1. Two single-node LS types. (a) Move the black node to a random position in the circle. (b) Move the black node to one of the eight gray positions.

Based on the above two single-node LS types, the proposed three LS methods are detailed as follows:

- ***Standard***: This method (Fig. 2(a)) randomly chooses a fog device and moves it by the second single-node LS type (Fig. 1(b)).
- ***Iterated***: This method (Fig. 2(b)) randomly chooses a fog device, then moves it to an arbitrary position in the placement area (i.e., along the blue arrow in Fig. 2(b)), and then moves it by the second one-node LS type in Fig. 1(b) (i.e., along the red arrow in Fig. 2(b)). This method is different from the *Standard* LS method because the chosen fog device searches for the local area of the new position, instead of that of the original position. By doing so, the LS method could avoid falling into a local optimal position.
- ***Random***: This method (Fig. 2(c)) moves each fog device to a random position by the first one-node LS type (Fig. 1(a)).

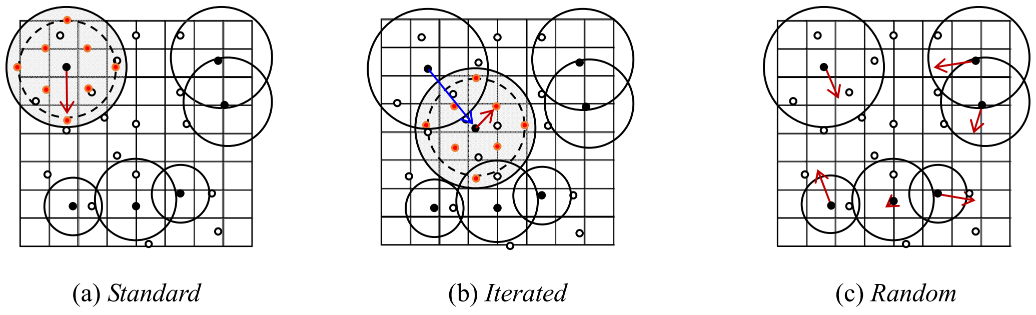

Fig. 2. Three LS methods.

3.4. Dynamic LS selection mechanism

Although three LS methods are proposed in the last subsection, it is hard to evaluate which one of the three LS methods is better. Hence, by analogy from [32], this work proposes a dynamic LS selection mechanism as follows. Let S_1, S_2, and S_3 store the probabilities of selecting the *Standard*, *Iterated*, and *Random* LS methods, respectively. Initially, all the probabilities are equal, i.e., $S_1 = S_2 = S_3 = 1/3$. At each iteration of the BA main loop, one of the three proposed LS methods is selected according to probabilities S_1–S_3. Let i be the chosen LS method, and S_i be the probability of selecting this LS method. Let $g(X_*)$ and $g_{new}(X_*)$ be the best fitness value before and after executing this LS method, respectively. Consider two cases: 1) $g_{new}(X_*) > g(X_*)$, and 2) $g_{new}(X_*) < g(X_*)$. In case 1), the probability of selecting LS method i is adjusted as follows:

$$S_i = S_i + \Delta variation \tag{13}$$

where

$$\Delta variation = \left(\frac{g_{new}(X_*) - g(X_*)}{g_{new}(X_*) + g(X_*)}\right). \tag{14}$$

Case 1) implies that the best fitness value becomes better after executing the LS method i. Hence, (14) is used to calculate $\Delta variation$ (i.e., the probability difference after LS), and then (13) is used to adjust the probability S_i. At the same time, the other two probabilities are adjusted to share $\Delta variation$ evenly as follows:

$$S_j = S_j - \frac{\Delta variation}{2} \tag{15}$$

$\forall j \in \{1, 2, 3\}$ and $j \neq i$. Note that the proposed BA does not have case 2) (i.e., $g_{new}(X_*) < g(X_*)$), because it only keeps better or equal solutions.

To summarize, one of the three LS methods is chosen and executed at each iteration of the proposed algorithm. After executing the LS method, the dynamic LS selection mechanism described above is adopted to readjust probabilities S_1–S_3. To avoid that the probability of selecting some LS method is too large such that other LS methods have no probability to be selected, the proposed mechanism sets the upper bound and the lower bound of each probability to be 0.8 and 0.1, respectively.

3.5. The proposed improved BA

BA works with a population of bats, where the position of each bat is corresponding to a candidate solution of the concerned problem. The proposed improved BA is given in Algorithm 1. The main steps of BA are explained as follows. Since the algorithm handles multiple epochs of a dynamic environment, the input of this algorithm is the problem at the τ-th epoch. If $\tau = 0$ (i.e., at the 0-th epoch), the position X_k^0 and pulse frequency f_k^0 of each bat k are set arbitrarily within the given ranges (Lines 3-4); otherwise (i.e., $\tau > 0$), they inherit the values at the ($\tau - 1$)-th epoch (Lines 5-8). Each bat k updates its velocity V_k^0, pulse emission rate r_k^0, and pulse loudness A_k^0 (Lines 9-10). Next, we compute the best position X_* (Line 12). Line 13 initializes three LS selections probabilities.

Next, bats iteratively search for the food (the best solution) through echolocation (Lines 14-36). Each iteration t considers each bat k to fly at a velocity V_k^t which is updated based on the frequency f_k^t in its previous iteration (Lines 16-20). Then, if a random number chosen from [0, 1] is less than the pulse emission rate r_k^t of bat k, we randomly choose a best position, and conduct the LS in (6) on this best position (Lines 21-24).

Then, Line 26 generates a new position of bat k by conducting the LS in (6) on the position X_k^t of bat k. If the pulse loudness is greater than a random number chosen from [0, 1], and this new solution performs better than X_* in terms of fitness, bat k accepts the new solution, and raises its pulse emission rate r_k^t and decrease its pulse loudness A_k^t by (7) and (8), respectively (Lines 26-29). Line 31 calculates the best positions X_* and their fitness value $g(X_*)$. The proposed dynamic LS selection mechanism is given in Lines 32-35. Line 32 generates a random number p from $[0, 1]$. Lines 33-34 use the random number p to find the adopted LS method according to the probabilities S_1–S_3, and record the updated fitness value $g(X_*)$ by $g_{new}(X_*)$. Finally, Line 35 adopts (13) and (15) (i.e., the proposed dynamic LS selection mechanism) to adjust S_1–S_3.

Algorithm 1. BA (the τ-th epoch)

```
1:  for each bat $k \in \{1, 2, \ldots, \eta\}$ do
2:      if $\tau = 0$ then
3:          bat $k$'s position $x_k^0 = (x_{k1}^0, x_{k2}^0, \ldots, x_{k(2n)}^0)$ is initialized randomly in which $x_{k(2i-1)}^0 \sim U(0, W)$ and
            $x_{k(2i)}^0 \sim U(0, H)$ for each $i \in \{1, 2, \ldots, n\}$ from (9), and its fitness value $f(X_k^0)$ is calculated
4:          bat $k$'s pulse frequency $f_k^0$ is initialized randomly
5:      else // i.e., $\tau > 0$
6:          $X_k^0$ inherits the position of bat $k$ at the $(\tau - 1)$-th epoch, and its fitness value $f(X_k^0)$ is
            calculated
7:          $f_k^0$ inherits the pulse frequency of bat $k$ at the $(\tau - 1)$-th epoch
8:      end if
9:      velocity $V_k^0$ is calculated by (4) under (10) and (11)
10:     bat $k$'s pulse rates $r_k^0$ and loudness $A_k^0$ is initialized randomly
11: end for
12: $g(X_*) \leftarrow \max_k f(X_k^0)$ and record $X_*$
13: initialize $S_1 = S_2 = S_3 = 1/3$
14: while ($t$ < the maximal number of iterations) do
15:     for each bat $k \in \{1, 2, \ldots, \eta\}$ do
16:         bat $k$'s frequency $f_k^t$ is set randomly by (3)
17:         bat $k$'s velocity $V_k^t$ is updated by (4)
18:         $V_k^t$ is truncated if violating (11)
19:         bat $k$'s position $X_k^t$ is updated by (5)
20:         $X_k^t$ is truncated if violating (9)
21:         if ($rand(0, 1) < r_k^t$) then
22:             one position among the best positions is chosen
23:             LS in (6) is conducted on this best solution
24:         end if
25:         conduct LS in (6) on $X_k^t$ to generate a new position of bat $k$
26:         if (a random number from [0, 1] is less then $A_k^t$ and this new position of bat $k$ is better
            than $X_*$) then
27:             the new solution is accepted
28:             $r_k^t$ and $A_k^t$ are adjusted by (7) and (8)
29:         end if
30:     end for
31:     after all bats are sorted according to fitness, (multiple) current best solutions $X_*$ are found,
        and are recorded as $g(X_*)$
32:     generate a random number $p \sim U(0, 1)$
33:     $\forall k \in \{1, 2, \ldots, \eta\}$, a new position of bat $k$ is generated by choosing and executing one of the
        Standard, Iterated, and Random LS methods by comparing the random number $p$ and their
        respective selection probabilities $S_i$, $i \in \{1, 2, 3\}$.
34:     $g(X_*)$ is updated by $g_{new}(X_*)$
35:     if the new position is better, then it is accepted by the bat population, and the corresponding
        selection probabilities are adjusted by (13) and (15)
36: end while
```

4. Implementation and Simulation Design

This section details the implementation of the proposed BA, and conducts various simulations to evaluate its performance. First, the simulation data and environment are described. Then, the simulation results of the proposed BA under various settings are analyzed, as compared with the previous PSO method in [10] and the BA without the proposed LS methods.

4.1. Simulation data and setting

As for the experimental instances, this work considers following problem instances by referring the previous work [10]:

- Case 1: $m = 16$, $n = 48$, $W = H = 32$, and $\gamma_i \sim U(3, 6)$ (i.e., placing 16 fog devices to serve 48 edge devices on a 32×32 placement area, in which the coverage radius of each fog device f_i follows distribution $U(3, 6)$).
- Case 2: $m = 32$, $n = 96$, $W = H = 64$, and $\gamma_i \sim U(4\sqrt{2}-2, 8\sqrt{2}-2)$ (i.e., placing 32 fog devices to serve 96 edge devices on a 64×64 placement area, in which the coverage radius of each fog device f_i follows distribution $U(4\sqrt{2}-2, 8\sqrt{2}-2)$).
- Case 3: $m = 64$, $n = 192$, $W = H = 128$, and $\gamma_i \sim U(7, 14)$ (i.e., placing 64 fog devices to serve 192 edge devices on a 128×128 placement area, in which the coverage radius of each fog device f_i follows distribution $U(7, 14)$.

Note that the three cases are of a small scale to a large scale; each case has 10 instances; edge devices are distributed uniformly on the placement area.

This work implements the proposed BA in C++ programming language. The parameter settings of simulation are given in Table 1. The simulation runs on a PC with an Intel Core i3-3210 CPU@3.2GHz and 4G memory. The average CPU times for the three cases are 1.446 s, 8.46 s, and 57.4 s, respectively.

Table 1. Parameter setting.

Parameter	Value
Maximal number of iterations	200
Number of epochs	20
Number of edge devices	0, 16, 32, 48
The probability that edge devices are switched to be turned on or off between epochs	1%
The maximal distance that edge devices can move to at one epoch	10
The λ value in the fitness function	
Maximal velocity V_{max}	0.3
Number of bats η	0.1
Pulse loudness decreasing rate α	50
Pulse emission rate γ	0.9
Bound for pulse frequency	0.9
Bound for the pulse loudness	[10, 100]
Bound for pulse emission rate	[1, 2]
Weight of each edge device	[0, 1]
Weight of each fog device	{1, 2, …, 10}
	10

4.2. Experiments for the dynamic environment of FCS

Besides the proposed BA performs better than the PSO approach in [10] for unweighted version in the static environment, this work also verifies whether the proposed BA also adapts to the dynamic environment more rapidly than the PSO approach. Since the PSO approach is designed for unweighted version, this subsection considers the dynamic environment of unweighted FCS, i.e., each edge device is weighted equally. Consider executing 200 iterations of the BA or PSO algorithm, in which the network topology has a dynamic change in every 20 iterations. Four different dynamic environments are considered, and their experimental results are plotted in Figs. 3(a)-(d), respectively, in which x-axis represents the number of iterations; y-axis represents the best fitness value of each iteration; solid and dashed broken lines are the results of the PSO and the proposed BA, respectively.

Furthermore, experimental results for the original BA (i.e., the BA without the proposed LS methods) are additionally included as a dotted broken line in Fig. 3(d). The difference of the four dynamical environments are that 0, 16, 32, 48 edge devices (i.e., 0, 1/3, 2/3, and all the edge devices, respectively) change their positions at different times. Additionally, the experiments allow 1% of the edge devices to be switched to turn on or off at different times. For experimental comparison, the initial input network topologies of the four environments in Figs. 3(a)-(d) are the same (i.e., the initial positions of edge devices are the same).

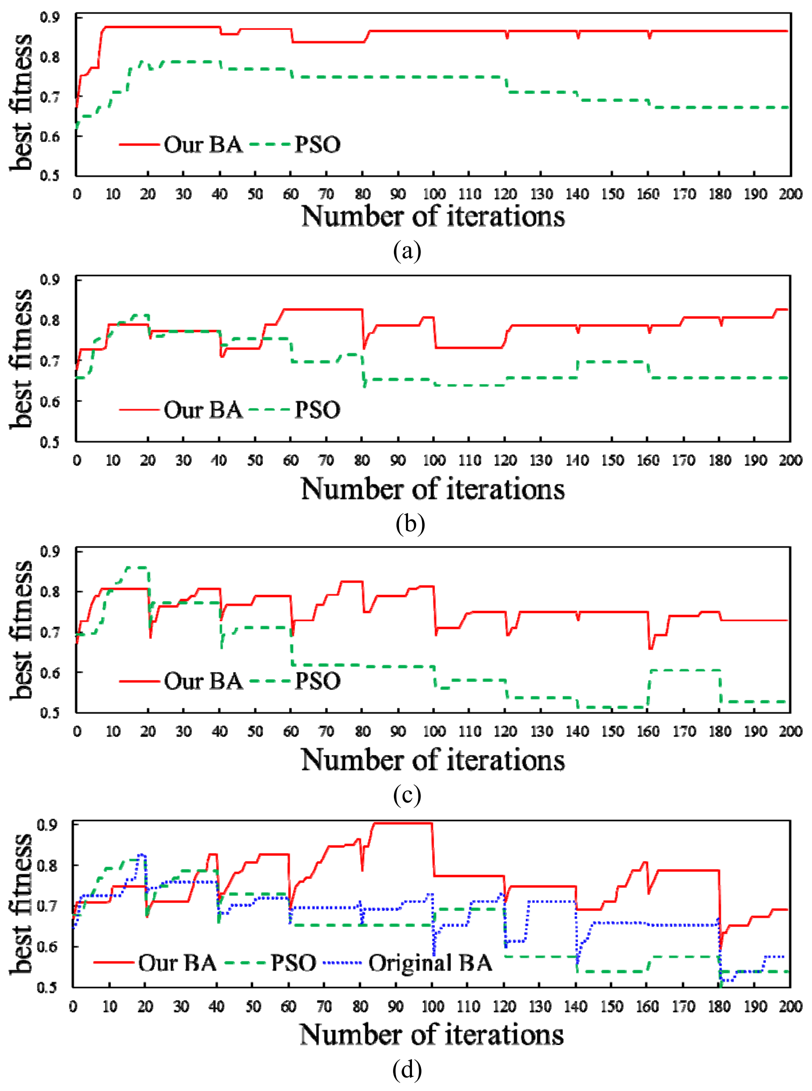

Fig. 3. The comparison of best fitness values versus number of iterations using the proposed BA and the PSO approach in different environments where edge devices are mobile and 1% of them can be switched on or off after every 20 iterations. The numbers of changeable edge devices are: (a) 0, (b) 16, (c) 32, and (d) 48.

In the first environment in Fig. 3(a) (i.e., the static environment), edge devices do not change their positions, but 1% of them can switch their network access. From Fig. 3(a), the proposed BA always outperforms the PSO approach, and can converge to a better fitness level. Additionally, after the 40-th iteration, the best fitness value of the PSO approach does not rise any longer, i.e., no improvement for the solution search. On the contrary, the proposed BA always has an improvement.

As for the second environment as shown in Fig. 3(b), the mobility of edge devices is of a small-scale. From Fig. 3(b), although the initial fitness values of the PSO approach and the proposed BA do not differ a lot and both of their convergence speeds are almost the same, the PSO approach is improved only slightly after the 40-th iteration, and has almost no adjustment for the later iterations of every 20 iterations. The proposed BA surpasses the PSO approach at about the 50-th iteration, and continues having improvements for later iterations.

As for the environments when more edge devices change their positions in Figs. 3(c) and 3(d), the best fitness values decrease drastically at the 20-th iteration, while the PSO approach has the same problem as described above, i.e., the best fitness values almost have no improvement after the 40-th iteration, such that its best fitness values continue decreasing. As for the proposed BA, although the best fitness values decrease a bit at the initial several iterations of each change, they are adjusted to the original high level after 7 iterations.

The last environment (Fig. 3(d)) is the severest, in which all edge devices can change their positions. Note that the experimental results of the original BA (i.e., the BA without the proposed three LS methods) are included in Fig. 3(d) for comparison. Although the original BA may not always outperform the PSO approach, it continues having improvement, and the final best fitness value before each change also performs better than the PSO approach. Additionally, by observing the results of the proposed BA and the original BA carefully in Fig. 3(d), the gap between the two plotted lines in Fig. 3(d) can be viewed as the effect of the proposed dynamic LS selection mechanism. In overall, the proposed BA performs better than the original BA and the PSO, no matter whether the environment is static or dynamic.

4.3. Experimental analysis for the dynamic LS selection mechanism

This subsection records the number of iterations of selecting the three LS methods in the proposed dynamic LS selection mechanism. We execute 20 times of the proposed BA, each of which has 200 iterations. And, each iteration selects one of the three LS methods to be executed. Hence, there are 4000 iterations for selecting LS methods. The ratios of the *Standard*, *Iterated*, and *Random* LS methods executed in the 4000 iterations are 0.282, 0.305, and 0.413, respectively. Hence, the *Random* LS method accounts for a significantly ratio more than the *Standard* and the *Iterated* LS methods. In the 4000 iterations, the *Standard*, the *Iterated*, and the *Random* LS methods are executed with 1652, 1220, and 1128 iterations. From the statistics, the numbers of iterations of the *Standard* and the *Iterated* LS methods are similar, while the number of iteration of the *Random* LS method accounts much more. It implies that the *Random* LS method has a better effect on solution-searching improvement than the *Standard* and the *Iterated* LS methods, such that the probability of selecting the *Random* LS method is adjusted to a high probability, and compresses the other two probabilities. Hence, we further observed the solution performance using each LS method, and found that this situation occurs because the *Standard* and *Iterated* LS methods are prone to fall into the local optimal solution and cannot be improved. Although the *Iterated* LS method can escape the local optimal solution more easily than the *Standard* LS method, the effect is not remarkable. Although in the *Random* LS method each fog device searches an arbitrary position within a fixed searching circle and may not move too far, it influences all the fog devices, such that the local optimal solutions could be escaped.

To express the proposed dynamic LS selection mechanism more clearly, and verify the above statistical results, the probability adjustments of the three LS methods of 200 iterations of 50 bats are illustrated in Fig. 4, in which the x-axis represents the number of iterations, and the y-axis represents the LS probabilities; the probabilities of choosing the *Standard*, the *Iterated*, and the *Random* LS methods are plotted by solid, dashed, and dotted broken lines, respectively.

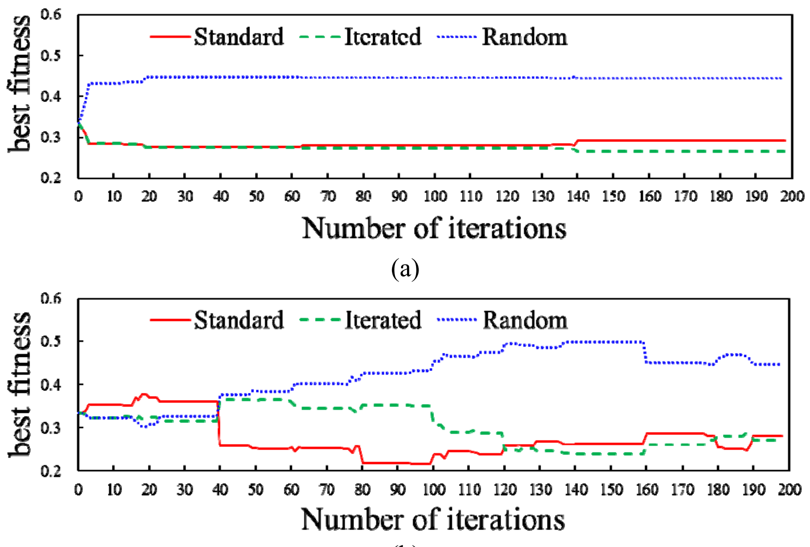

Fig. 4. The adjustment of the probabilities for selecting the three LS methods in 200 iterations. (a) The static environment. (b) The dynamic environment.

Figs. 4(a) and 4(b) show the static and the dynamic environments, respectively. In both environments, the final probability of selecting the *Random* LS method is adjusted to a level that is obviously greater than the other two LS methods. It implies that the *Random* LS method has the best solution-searching effect than the other two methods. On the other hand, for the static environment in Fig. 4(a), the convergence speed is relatedly rapid, because the convergence of fitness values leads to inefficiency of improving solutions with LS (i.e., the probability adjustment gap is decreasing). It can be seen that the probabilities almost have no adjustments after 20 iterations. As for Fig. 4(b), three probabilities are modified drastically, because in the dynamic environment the optimal placement would be modified a lot after each change of edge devices, such that all the three LS methods have the ability to improve the solution. But the *Random* LS method still has the best improvement effect than the other two methods at the later iterations.

5. Conclusion

This work has proposed an improved BA for dynamically placing weighted fog devices in FCSs in which three LS methods are designed specifically for the problem, and a dynamic LS selection mechanism is used for selecting the three LS methods, to enhance the solution-searching ability and the convergence speed of the proposed BA so as to find better solutions. In the experimental evaluation, the proposed BA is compared with the previous PSO approach to the unweighted problem and the BA without the proposed LS methods for the weighted problem. Simulation results show that the proposed BA with dynamic LS selection looks promising and can demonstrate a better solution improvement effect than the PSO approach.

In the future, we intend to consider the extension dynamic weighted fog device node placement problem in FCS by considering other devices inclusively in FCS [4] and several tiers in FCS combined cloud computing system such as cloud server, gateway which connects incompatible network facilities or access points [2], and so on. It is also of interest to continue improving the proposed algorithm, or applying a novel algorithm to achieve better solutions. Additionally, as multiple optimal solutions can be achieved, it is of interest to consider how to reduce the number of fog devices to achieve more economic benefits.

Acknowledgements

The authors thank the anonymous referees for comments that improved the content as well as the presentation of this paper. This work has been supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 106-2221-E-009-101-MY3 and Grant MOST 108-2628-E-009-008-MY3.

References

[1] Aazam M, Zeadally S, Harras K (2018) Deploying fog computing in industrial internet of things and industry 4.0. IEEE Transactions on Industrial Informatics 14(10), pp. 4674-4682.

[2] Mukherjee M, Shu L, Wang D (2018) Survey of fog computing: fundamental, network applications, and research challenges. *IEEE Communications Surveys & Tutorials* 20(3), pp. 1826-1857.

[3] Aazam M, Zeadally S, Harras K (2018) Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. Future Generation Computer Systems 87, pp. 278-289.

[4] Lin C, Yang J (2018) Cost-efficient deployment of fog computing systems at logistics centers in Industry 4.0. *IEEE Transactions on Industrial Informatics* 14(10), pp. 4603-4611.

[5] Lee J, Chung S, Kim W (2017) Fog server deployment considering network topology and flow state in local area networks. In: *Proc. Conf. Ubiquitous and Future Network*s, pp. 652-657.

[6] Gu L, Zeng D, Guo S, Barnawi A, Xiang Y (2017) Cost-efficient resource management in fog computing supported medical CPS. *IEEE Transactions on Emerging Topics in Computing* 5(1), pp. 108-119.

[7] Guo P, Lin B, Li X, He R, Li S (2016) Optimal deployment and dimensioning of fog computing supported vehicular network. In: *Proc. IEEE Trustcom/BigDataSE/I SPA*, pp. 2058–2062.

[8] Xu Z, Liang W, Xu W, Jia M, Guo S (2016) Efficient algorithms for capacitated cloudlet placements. *IEEE Trans. Parallel Distrib. Syst.* 27(10), pp. 2866–2880.

[9] Ng C, Wu C, Ip W, Yung K (2018) A smart bat algorithm for wireless sensor network deployment in 3-D environment. *IEEE Communications Letters* 22(10), pp. 2120-2123.

[10] Lin CC (2013) Dynamic router node placement in wireless mesh networks: A PSO approach with constriction coefficient and its convergence analysis. *Information Sciences* 232, pp. 294-308.

[11] Zwaneveld PJ, Kroon LG, van Hoesel SPM (2001) Routing trains through a railway station based on a node packing model. *European Journal of Operational Research* 128(1), pp. 14-33.

[12] Pettie S, Ramachandran V (2005) A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing* 34(6), pp. 1398-1431.

[13] Ding W, Qiu K (2017) Incremental single-source shortest paths in digraphs with arbitrary positive arc weights. *Theoretical Computer Science* 674, pp. 16-31.

[14] Ábrego B, et al. (2012) Proximity graphs inside large weighted graphs. *Networks* 61(1), pp. 29-39.

[15] Naas MI, Lemarchand L, Boukhobza J, Raipin P (2018) A graph partitioning-based heuristic for runtime IoT data placement strategies in a fog infrastructure. In: *Proc. of the ACM Symposium on Applied Computing*, pp. 767-774.

[16] Aoun B, Kenward G, Boutaba R, Iraqi Y (2006) Gateway placement optimization in wireless mesh networks with QoS constraints. *IEEE Journal on Selected Areas in Communications* 24(11), pp. 2127-2136.

[17] Mishra A, Banerjee S, Arbaugh WA (2005) Weighted coloring based channel assignment for WLANs. *ACM SIGMOBILE Mobile Computing and Communications Review* 9(3), pp. 19-31.

[18] Garey M, Johnson D (1979) *Computers and Intractability - A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

[19] Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: *Proc. of Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, vol. 284 of Studies in Computational Intelligence, pp. 65-74.

[20] Gandomi AH, Yang XS, Alavi AH, Talatahari S (2012) Bat algorithm for constrained optimization tasks. *Neural Computing & Applications* 22(6), pp. 1239-1255.

[21] Soza C, Becerra RL, Riff MC, Coello CA (2011) Solving timetabling problems using a cultural algorithm. *Appl Soft Computing* 11(1), pp. 337-344.

[22] Lin CC, Shu L, Deng DJ (2014) Router node placement with service priority in wireless mesh networks using simulated annealing with momentum terms. *IEEE Systems Journal* 10(4), pp. 1402-1411, 2014.

[23] Parker G, Zbeda R (2014) Learning area coverage for a self-sufficient hexapod robot using a cyclic genetic algorithm. *IEEE Systems Journal* 8(3), pp. 778-790.

[24] Yang XS (2011) Bat algorithm for multi-objective optimization. *International Journal of Bio-Inspired Computation* 3(5), pp. 267-274.

[25] Mishra S, Puthal D, Rodrigues J, Sahoo B, Dutkiewicz E (2018) Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. *IEEE Transactions on Industrial Informatics* 14(10), pp. 4497-4506.

[26] Zineddine M (2018) Optimizing security and quality of service in a real-time operating system using multi-objective bat algorithm. *Future Generation Computer Systems* 87, pp. 102-114.

[27] Yılmaz S, Küçüksille E (2015) A new modification approach on bat algorithm for solving optimization problems. *Applied Soft Computing* 28, pp. 259-275.

[28] Wang G, Chu H, Mirjalili S (2016) Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerospace Science and Technology* 49, pp. 231–238.

[29] Liang H, Liu Y, Shen Y, Li F, Man Y (2018) A hybrid bat algorithm for economic dispatch with random wind power. *IEEE Transactions on Power Systems* 33(5), pp. 5052-5061.

[30] Osaba E, et al. (2019) A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. *Swarm and Evolutionary Computation* 44, pp. 273-286.

[31] Naderi M, Khamehchi E, Karimi B (2019) Novel statistical forecasting models for crude oil price, gas price, and interest rate based on meta-heuristic bat algorithm. *Journal of Petroleum Science and Engineering* 172, pp. 13-22.

[32] Li S, Zhao S, Wang X, Zhang K, Li L (2014) Adaptive and secure load-balancing routing protocol for service-oriented wireless sensor networks. *IEEE Systems Journal* 8(3), pp. 858-867.