Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

# Energy-Efficient and Reliable Deployment of IoT Applications in a Fog Infrastructure Based on Enhanced Water Strider Algorithm

**Huda Hasan Alsaabri**

Biomedical Engineering Department, College of Engineering and Technologies, Al-Mustaqbal University, Hillah 51001, Babil, Iraq. E-mail: E-mail: huda.hasan.hatif@uomus.edu.iq

**Hamza Mohammed Ridha Al-Khafaji ***

*Corresponding author, Biomedical Engineering Department, College of Engineering and Technologies, Al-Mustaqbal University, Hillah 51001, Babil, Iraq. E-mail: hamza.alkhafaji@uomus.edu.iq

## Abstract

Fog computing is considered a promising solution to minimize processing and networking demands of the Internet of things (IoT) devices. In this work, a model based on the energy consumption evaluation criteria is provided to address the deployment issue in fog computing. Numerous factors, including processing loads, communication protocols, the distance between each connection of fog nodes, and the amount of traffic that is exchanged, all have an impact on the re-search system's overall energy consumption. The power consumption for implementing each com-ponent on the fog node as well as the power consumption for information exchange between the fog nodes are taken into account when calculating each fog node's energy use. Each fog node's energy consumption is closely correlated to how its resources are used, and as a result, to the average normalized resource utilization of a fog node. When the dependent components are spread across two distinct fog nodes, the transfer energy is taken into account in the computations. The sum of the energy used for transmission and the energy used for computational resources is the entire amount of energy consumed by a fog node. The goal is to reduce the energy consumption of the fog network while deploying components using a novel metaheuristic method. Therefore, this work presents an enhanced water strider algorithm (EWSA) to address the problem of deploying application components with minimum energy consumption. Simulation experiments with two scenarios have been conducted based on the proposed EWSA algorithm. The results show that the EWSA algorithm achieved better performance with 0.01364 and 0.01004 optimal energy consumption rates.

**Keywords:** Internet of things; fog computing; energy-efficient; applications deployment; courtship learning-based water strider algorithm; metaheuristic.

Journal of Information Technology Management, 2023, Vol. 15, Issue 4, pp. 179-204
Published by University of Tehran, Faculty of Management
https://doi.org/ 10.22059/jitm.2023.94931
Article Type: Research Paper

Received: June 02, 2023
Received in revised form: July 28, 2023
Accepted: September 20, 2023
Published online: November 15, 2023

## Introduction

Cloud computing provides smart systems with useful solutions for managing Internet of things (IoT) devices (Stojmenovic, 2014; Al-Khafaji, 2022). Currently, a wide range of IoT devices around the world, generate a significant amount of data. Processing this data is a complex process, time-consuming and expensive. In this way, scientists introduced fog computing. In contrast to cloud computing, fog computing accelerates data processing and can send information more quickly. Along with the growth and penetration of IoT devices in different sectors, the use of fog computing services also became more important.

In the fog infrastructure, any device having processing power, storage, and network connectivity qualifies as a node and may be placed anywhere in the fog network. These nodes may be placed in vehicles or in offices that serve as target locations (Al-Khafaji et. al, 2019). Data generated by an IoT device may be received by one of these nodes, processed inside the network, and then sent to cloud data centers. Fog computing facilitates the location of IoT technology near the data source by supporting the integrated use of edge and cloud resources. Deploying, managing and updating applications in such a layered environment creates new challenges. The large-scale fog network includes a large number of heterogeneous nodes with separate computing resources, such as processing, storing, and memory. Deploying the components of an application on minimal nodes in the fog network leads to the reduction of energy consumption and the optimal use of computing resources, as well as reducing the delay between the application components, but this deployment plan leads to the strengthening of the single-point failure phenomenon, in which failure of one node disrupts communication through the whole system. Therefore, the single-point failure has a negative effect on the reliability of the customer's use, and as a result, a solution must be adopted for the proper deployment of components to provide reliable services.

This work focuses on one of the major fog computing challenges that is application components placement in fog computing environments. Finding the best deployment strategy is not a trivial task, due to the heterogeneous nature of fog nodes. In recent years, studies have been done regarding the distribution of components in the fog infrastructure. Samani et al. (2021) suggested a multilayer resource-aware partitioning mechanism. Using a multilayered

network graph, the approach represented the heterogeneous Fog components as subsets depending on network topology and resource properties. The appropriate device partitions to put an application in are then determined depending on its requirements; these partitions must overlap in the same network topology partition. Simulations showed that the multilayer resource-aware partition technique can put 2 times so many more operations, satisfy deadlines for 3 times so many more operations requests, and minimize waste of material besides up to 15-32 times compared to two published availability-aware and resource-aware methods. To find a cloud computing environment with fog an efficient scheduling method for connecting application elements. Arshed et al. (2022) provided a genetic algorithm-based method. The suggested approach bases its module scheduling on the implementation time as a fitness function, taking into account the accessible fog devices. In terms of execution time, financial cost, and bandwidth, the suggested technique was assessed and contrasted against baseline algorithms. The suggested technique provides a superior scheduling strategy than the current scheduler, according to extensive simulation data. To reduce network consumption and latency, Hassan et al. (2022) created an optimization algorithm that dynamically allocates suitable sensor equipment among fog nodes. The suggested method computed the volume of information sensed by the edge device based on the speed of sensing frequency of the sensors connected to the edge device. The recommended strategy took device heterogeneity and computing power into account while joining the network nodes. For the evaluation of the suggested method, many evaluations were carried out on various scales. The assessment results demonstrated that the suggested technique is successful in reducing network usage and end-to-end latency.

In general, the component placement optimization problem is an NP-hard problem that cannot be solved in polynomial time. The metaheuristic method has been widely used to solve NP-hard problems, due to its ability to achieve a feasible solution in a reasonable time. Furthermore, a metaheuristic method can be used to solve a variety of optimization problems with less modification. Examples of such metaheuristics include arithmetic optimization (AO) (Abualigah et al., 2021a), aquila optimizer (Abualigah et al., 2021b), world cup optimization (WCO) (Razmjooy et al., 2018) and cat swarm optimization (Ahmed et al., 2020).

The water strider algorithm (WSA) is one of the recently proposed metaheuristics (Kaveh and Eslamlou, 2020). This algorithm has shown competitive performance in solving several optimization problems. However, based on the no-free-lunch theorem, no algorithm guarantee success in solving all optimization problems. Furthermore, the searchability of the WSA algorithm is weak, and it is difficult to jump out of the local optimum when optimizing complex problems. For this reason, in this study, an enhanced WSA (EWSA) has been proposed. The proposed EWSA is used for the reliable distribution of application components in fog networks with different scenarios such as theft alarm systems, elderly care, and fire extinguishing system. In these scenarios, reliability and reliance on the system is very important for the customer. For optimal energy consumption, full mesh subnets are extracted

from the fog network. Among them, the most appropriate sub-network is selected, and the application components are distributed on the nodes of the selected sub-network according to the resources of the nodes and communication traffic between the components. The proposed EWSA presents an efficient balance between exploration and exploitation strategies that Yielded good results in terms of reducing energy consumption for running applications in fog landscape as well as fast convergence.

**System Modeling**

This section presents a fog-based approach for the deployment of application components and also suggests a framework for component management. The issue with installing the application components on the fog node is finally described. The application components management framework is presented in Figure 1 and according to the figure, an organizer is utilized on the highest of the fog. One of the responsibilities of the organizer is to extract complete mesh subnets from fog nodes.

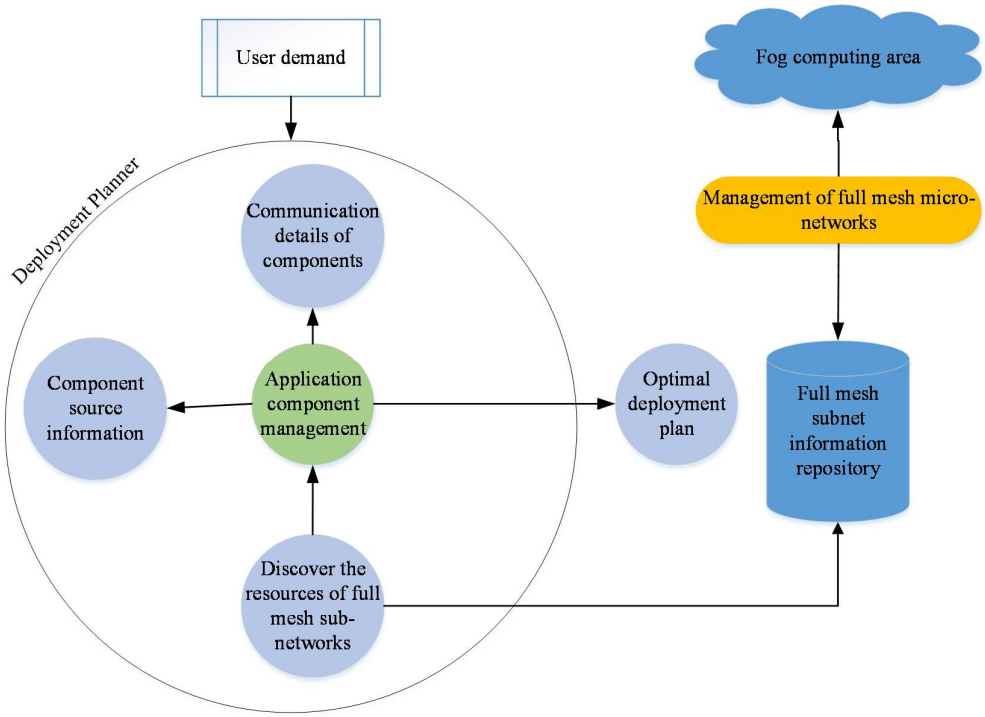


Figure 1. Application components management framework

The communication architecture of nodes in each subnet is similar to the architecture of a wireless mesh network. The computing model in each full mesh network is different from the traditional mesh network. The mesh network of fog nodes (switches and fog servers) is used for distributed operations inside the network. After extracting the complete meshes, a

suitable subnet is selected from among them and the organizer decides to deploy them in the selected complete mesh according to the characteristics of the application components. This planner is technically centralized and may be deployed for fault tolerance and the central point of component failures. The main priority in the suggested framework is the extraction of the deployment plan according to the selected complete meshes and the distribution of components based on the plan. In the distribution of components, only those that are not sensitive to delay are transferred to the cloud infrastructure for deployment and are communicated with them in rounds for information processing.

To properly manage and deploy the application components in the fog nodes according to the system efficiency, a deployment planning framework is used in the fog organizer. As shown in Figure 1, the programming part includes the application component manager and also modules that help in component management. Next to the deployment scheduler, there are modules for storing and retrieving network information and other resources of full mesh subnets. The collected information is used by the deployment planner module to manage the application components and provide an optimal deployment plan. In the following, the details related to these components are explained in the proposed framework.

The first component is the application element administrator. This component is crucial because it leverages other framework modules to determine how to distribute application element in fog or cloud networks. In a multi-component application, due to the dependence of the components on each other, the decision to deploy is made based on several factors such as the network structure, availability of resources, load distribution, and service quality requirements for the application. Component placement might be based on objectives like lowering energy usage, cutting down on network connections, and also lowering the total delay in use.

The second element is component resource data, which is used to determine how to deploy application components by extracting from user requests the processing and memory needs of the application components. The third component is communication information of components that has a major contribution in consuming the resources of fog nodes used in the IoT. Fog node handling of application modules comprises a memory, processing, and communication optimization. This module extracts the communication information of the application components from the request sent by the users and makes it available to the application components manager.

The fourth component is to discover the resources of full mesh subnets that are based on the information received from the application components manager, the information repository monitors the complete mesh subnets and sends the desired full mesh information for component deployment to the application components manager. The last component is the full mesh subnet manager. According to the information received from the fog, this

component extracts the complete mesh subnets from the fog nodes and stores the information of these subnets in the information repository. It also validates the status of complete meshes in the tank by periodically monitoring the fog infrastructure.

**Application Components**

The architecture of apps that handle user data on a wide scale has altered recently in response to demand and now has a multi-component structure to accommodate shifting users' needs and new expectations from Internet-based services. These components are interdependent and work together to satisfy the customer's needs. For example, consider a simple elderly care IoT application provided by a smart health service provider to its customers. This use of three components of control center (cmp1) for interpretation of collected data and manual control of the system, condition manager (cmp3) to monitor the condition of elderly and disabled people and provide immediate information to the nearest medical center in case of physical and mental problems and the learning machine (cmp2) was created to record the history of people's information and estimate their future health status.

The cmp2 component is not sensitive to delay and can be deployed in the cloud or data centers in fog infrastructure. The hardware resources and software capabilities needed by each component are shown in Figure 2 and the relationship between the components is depicted through links.

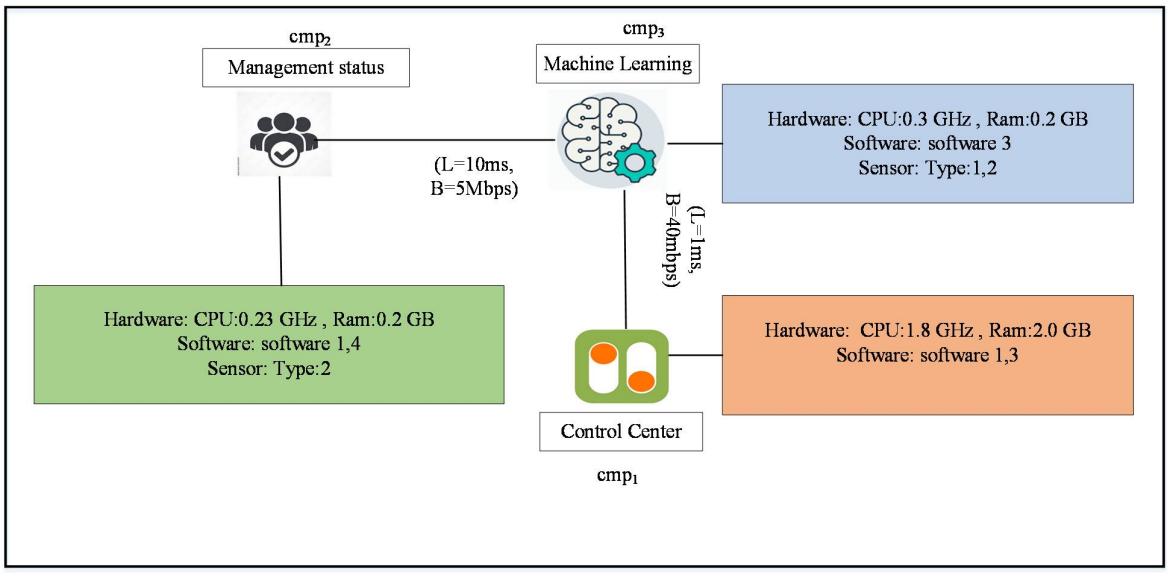


Figure 2. Specifications of application components

For the timely management of the condition of the elderly, the cmp3 component must access the required sensors (physical condition control sensors) and a trigger that activates the mechanisms of preliminary operations and notification to the treatment centers, and this must be within 10 milliseconds from the location of the cmp3 to the location. Installation of sensors and actuators will take place.

It should be noted that the APIs offered by the fog service layer are meant to allow fog or cloud units to remotely access items in surrounding nodes. How to distribute this module in such a manner that the resources it requires are met is the issue that has to be resolved for the activation of the aforementioned application components.

Even for this straightforward example, it is necessary to compare several deployment strategies to choose the best mapping for the application's components. Because several components might be deployed on a single node depending on the resources available, it is vital to employ sophisticated search algorithms to determine the best deployments as topologies and the number of application units develop.

In this study, it is assumed that $R$ number of IoT applications exist and each application $r \in R$ demonstrations represented using a vector $[M, cmp_l]$. Each application has several modules, where $M$ and $cmp_l$ are, in turn, the number and the list of the application components. The user application is modeled using a graph $G = (cmp_l, T)$, where, $T = [t_{11}, \dots, t_{nn}]$ is a matrix in this graph, $cmp_l = [cmp_1, cmp_2, \dots, cmp_N]$ is the traffic sent between $cmp_j$ and $cmp_i$. In the following formula, the traffic matrix between application components is presented in Equation (1).

$$\begin{array}{c} \\ cmp_1 \\ cmp_2 \\ \vdots \\ cmp_n \end{array} \begin{array}{c} \begin{array}{cccc} cmp_1 & cmp_2 & \dots & cmp_n \end{array} \\ \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \dots & t_{nn} \end{bmatrix} \end{array} \quad (1)$$

**Fog Model**

In this study, a network consisting of $N$ fog nodes with heterogeneous processing power and energy is assumed, which are capable of storing and executing application components. Fog nodes in the assumed network are members of one or more subnets of the complete mesh. Each of the fog nodes has direct or indirect access to a variety of sensor nodes through wired or wireless connections. A fog network $fn \in F$ is represented using a vector $[Mid, SL, ID, SW, HW]$, in which, $Mid$ is the ID in the complete mesh, $SL$ is the list of available sensors, $ID$ is the node ID in the fog network, $SW$ is the software, and $HW$ is the hardware.

The components that are distributed in the nodes of a complete mesh have access to the software capabilities and sensors of other fog nodes in the same mesh. The communication link 1 between fog nodes can be shown using the vector $[D, B_L]$. In which, $D$ stands for the delay amount and $B_L$ is the link bandwidth. Figure 3 demonstrates the details of a complete mesh.

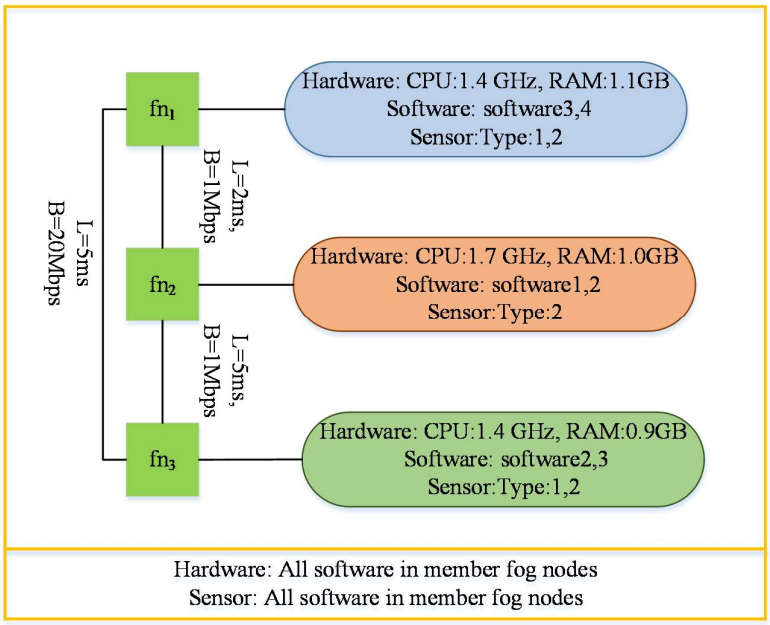


Figure 3. Specifications of a complex subnet and member nodes

Communication network model: The communication network in a full mesh subnet is modeled using a graph $G = (FN, Dst)$. In this graph, $FN = [fn_1, fn_2, \dots, fn_M]$ is a set of fog nodes and $Dst = [dst_{ji}]$ is the distance between $fn_j$ and $fn_i$. In each complete mesh subnet, if $fn_j = fn_i$, then $dst_{ji} = 0$, otherwise $dst_{ji} = 1$. The communication network and distance matrix of fog nodes can be shown as follows:

$$\begin{array}{c} \\ fn_1 \\ fn_2 \\ \vdots \\ fn_m \end{array} \begin{array}{c} \begin{array}{cccc} fn_1 & fn_2 & \dots & fn_m \end{array} \\ \begin{bmatrix} dst_{11} & dst_{12} & \dots & dst_{1m} \\ dst_{21} & dst_{22} & \dots & dst_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ dst_{m1} & dst_{m2} & \dots & dst_{mm} \end{bmatrix} \end{array} \tag{2}$$

**Reliability Model**

One of the problems that lie ahead is trying to strengthen the single goal of failed phenomenon for application programs. When an application's components are deployed on the fewest possible fog nodes, objectives like cutting down on energy use and making the best use of computational power are maximized. As a result, a minimum constraint for the number of fog nodes is taken into consideration for the deployment of application components to satisfy the aims of fog computation owners in attaining the ideal goals and to also lower the level of vulnerability of services in centralized transmission in fog structure. The number of nodes in the chosen complete mesh subnet is used as the least quantity of nodes for element distribution for this purpose. The distribution of the components in the event of non-fulfillment is carried out over the shortest possible distance in relation to the total number of nodes in the mesh subnet.

**Deployment Model**

To install the components from the extracted complete meshes, a complete mesh is selected according to the needs of the application. The fog nodes in a complete mesh provide the resources required by the components (delay, bandwidth and sensors). Our assumption in the proposed model is that the sensors or software required by the application components are shared by the fog nodes. will be available in a complete mesh subnet. In the procedure of distributing application modules, $i$ on fog nodes, the computation resources, the distance of fog nodes and the quality-of-service parameters required by the application components should be considered. To decrease the traffic load, the space matrix should be considered. between the fog nodes in the network graph and also calculated the traffic matrix between the components of an application. It should be noted that the communication link between the fog nodes m and n has a fixed ability in terms of delay and bandwidth, and consequently the traffic amount of the modules of the applications with the capacity of the link is limited between fog nodes, so we have:

$$\sum_{cmp_j \in fn_n} \sum_{cmp_i \in fn_m} b_{ji} \times l_{ji} < B_{nm} \times L_{nm} \tag{3}$$

In the above expression, $b_{ji}$ and $l_{ji}$ represent, in turn, the bandwidth and desired delay between $j$ and $i$ components, and $B_{nm}$ and $L_{nm}$ are, in turn, the bandwidth and delay of the communication link between fog nodes $n$ and $m$. We define a binary variable, $x_{cmp}^{fn}$, to determine the deployment status of the application components $cmp \in U_{app}$ on the fog node fn $fn \in F$, if the $cmp$ component is placed on the $fn$ node, $x_{cmp}^{fn}$ will be equal to 1 and otherwise, it will be 0. A module can be located on a fog node when the fog node stands active, that is, $y_{fn} = 1$ and as a result we have (Arshed et al., 2022):

$$x_{cmp}^{fn} \leq y_{fn}, \forall cmp \in U_{app}, fn \in F \quad (4)$$

It should also be noted that all the hardware resources needed by the components located in the fog node $fn$ should not exceed the capacity of that node and therefore we have (Chegini et al., 2022):

$$\sum_{cmp \in U_{app}} x_{cmp}^{fn} \times sr_{cmp} \leq SR_{fm}, \forall fm \in F \quad (5)$$

In the above expression, $SR_{fm}$ and $sr_{cmp}$ represent the software resources capacity of the complete mesh subnet and the requested software resources of the application components. We also have (Arshed et al., 2022):

$$\sum_{cmp \in U_{app}} x_{cmp}^{fn} \times s_{cmp} \leq S_{fm}, \forall fm \in F \quad (6)$$

where, $S_{fm}$and $S_{fm}$represent the available sensors through the fully selective mesh subnet and the requested sensor resources of the application components. Each of the components is executed only on one of the fog nodes, so we have (Xiao et al., 2022):

$$\sum_{fn \in F} x_{cmp}^{fn} = 1, \forall cmp \in U_{app} \quad (7)$$

## Methodology

The study system's overall energy usage is generally influenced by several variables, including processing loads, communication technologies, the range between each connection of fog nodes, and the volume of traffic that is exchanged. To compute the energy usage of each fog node, the energy consumption related to the implementation of each of the components on the fog node and the power consumption for exchanging information between the fog nodes are considered. The energy consumption of each fog node directly depends on the use of its resources, and therefore the average use of normalized resources from a fog node is calculated as shown in Equation (8).

$$U_{fn_i}^{res} = \frac{1}{2} \times \left( w_1 \times \sum_{i}^{fn_j} \frac{r_{cmp_i}^{CPU}}{R_{fn_j}^{CPU}} + w_2 \times \sum_{i}^{fn_j} \frac{r_{cmp_i}^{RAM}}{R_{fn_j}^{RAM}} \right) \quad (8)$$

In the above equation, the coefficients $w_1$ and $w_2$ have real values, such that $0 \leq w_1, w_2 \leq 1$ where, $w_1 + w_2 = 1$. These coefficients are used to determine the importance of combined resources in total energy consumption. Since most of the energy usage is related to the processing units, in this article, $w_1 = 0.8$ and $w_2 = 0.2$ are considered. Therefore, the energy usage related to each host node of different components is calculated using Equation (9).

$$P_{fn}^{res} = \underline{P} + y_{fn} \times \left(\overline{P} - \underline{P}\right) \times U_{fn}^{res} \quad (9)$$

where the parameters $\underline{P}$ and $\overline{P}$ are utilized to determine the least and most energy usage of each processing node in the lowest and highest operating conditions, respectively, and the binary variable $y_{fn}$ is utilized to indicate whether the preceding node is active or inactive. Also, the energy usage resulting from the transmission is done through communication links as shown in Equation (10).

$$P_{fn}^{tr} = P_{tr} \times \sum_{fn_j \neq fn_i} t_{cmp_j, cmp_i} \quad (10)$$

In the above expression, $P_{tr}$ describes the data transmission energy in the fog node and $t_{cmp_j, cmp_i}$is the traffic of the dependent components. According to the above relationship, the transfer energy is used in calculations when the dependent components $j$ and $i$ are located in two different fog nodes. Finally, the total consumed energy in a fog node, which is the sum of the transmission energy and the consumed energy of computing resources, is obtained from Equation (11).

$$P_{fn} = P_{fn}^{tr} + P_{fn}^{res} \quad (11)$$

The proposed optimization model for minimizing the fog network total power consumption (TPC) in the deployment of components is formulated as shown in Equation (12) (Sofia et. al, 2018).

$$min\ TPC = min \sum_{fn \in F} P_{fn} \quad (12)$$

Such that:

$$n_h > \sum fn_{fm} \quad (13)$$

$$x_{cmp}^{fn} \in [0, 1], \quad (14)$$

$$y_{fn} \in [0, 1] \quad (15)$$

$$\sum_{cmp_j \in fn_n} \sum_{cmp_i \in fn_m} b_{ji} \times l_{ji} < B_{nm} \times L_{nm} \quad (16)$$

$$\sum_{cmp \in U_{app}} x_{cmp}^{fn} \times hr_{cmp} \leq HR_{fn}, \ \ \forall fn \in F \quad (17)$$

$$\sum_{cmp \in U_{app}} x_{cmp}^{fn} \times sr_{cmp} \leq SR_{fm}, \ \ \forall fm \in F \quad (18)$$

$$\sum_{cmp \in U_{app}} x_{cmp}^{fn} \times s_{cmp} \leq S_{fm}, \ \ \forall fm \in F \quad (19)$$

$$x_{cmp}^{fn} \leq y_{fn}, \ \forall cmp \in U_{app}, \ fn \in F \tag{20}$$

$$\sum_{fn \in F} x_{cmp}^{fn} = 1, \ \forall cmp \in U_{app} \tag{21}$$

**Water Strider Algorithm**

The WSA is inspired by the territorial behavior, mating manner, ripple communication, foraging and succession of water striders. This algorithm includes five main steps namely, birth, territory establishment, mating, feeding, and death. In the birth step, the WSA mimics the female egg hatching to create a uniform distribution of the water strider population. The initial position of the $i^{th}$ water strider, $x_i^0$, is expressed as shown in Equation (22).

$$x_i^0 = Lb + rand \times (Ub - Lb); \ i = 1,2, \dots, N. \tag{22}$$

where $N$ is the total number of water striders. $Ub$ and $Lb$ for the maximal and minimal bounds of the problem, respectively. rand is a random number between 0 and 1. The best solutions are then selected by calculating the fitness values of water striders using an objective function. To establish nt number of territories, the population of water striders, nws is then divided into $\frac{nws}{nt}$ groups based on their fitness value. To catch and flirt with females in every location, the keystone (male water strider) sends out a specific wavy pattern to the females. Any female can either respond to the signs or ignore them. Consequently, mating can only take place if the females receive and accept the request. In this case, the keystone may mate or be repelled, either way. This behavior is formulated as shown in Equation (23).

$$\begin{cases} x_i^{t+1} = x_i^t + R \times \delta, & if \ mating \ occurs \ (with \ probability \ of \ p) \\ x_i^{t+1} = x_i^t + R \times (1 + \delta), & Otherwise \end{cases} \tag{23}$$

where $\delta$ is a random value between 0 and 1 the position off the ith water strider is denoted by $x_i^t$, and R is the distance between the male position, $x_i^{t-1}$ and female position, $x_F^t$, which is calculated as shown in Equation (24).

$$R = x_F^{t-1} - x_i^{t-1} \tag{24}$$

The water striders relax by eating and updating their position following the mating ritual because the fish used a significant amount of energy. If the previous position is greater than the objective quantity in these situations, this should travel through the ideal area with the best fitness ($x_{BL}^t$), but if the previous position is less than the objective quantity, the meal is not provided for reconstruction. This strategy may be expressed mathematically as follows:

$$x_i^{t+1} = x_i^t + 2 \times rand \times \left(x_{BL}^t - x_i^t\right) \tag{25}$$

The purpose of the update formula, as it is presented following, is to place restrictions on the search process within the targeted range.

$$x_i^{t+1} = Lb_j^t + rand \times \left(Ub_j^t - Lb_j^t\right) \tag{26}$$

The algorithm usually ends if the terminating conditions have been met. Most iteration serves as the cutoff point for the optimization algorithm.

**Courtship Learning-based Water Strider Algorithm**

Although in the WSA, a male transmits specific signals to attract a female, it does not offer a specific method for distinguishing males and females in the population. Hence, it cannot effectively use the gender information of water striders, which restricts the global search ability of WSA. Therefore, to preserve the global search ability and increase its effectiveness, the courtship learning approach has been integrated with WSA. In the courtship learning approach, an individual with a lower fitness value is more likely to be selected. To select a female from the archive, each individual is ranked based on their fitness value as shown in Equation (27).

$$M_i = \frac{1}{f\left(x_i^t\right)} \tag{27}$$

where, $f(x_i^t)$ specifies the fitness value of the ith female in the archive. The selection probability of a female from the female archive is calculated as shown in Equation (28) (Zheng and Li, 2018).

$$S_i = \frac{M_i}{\sum_{j=1}^{N} M_i} \tag{28}$$

According to Equation (28), a female water strider of a lower fitness value will have a higher probability of being selected from the female archive. The water strider could get stuck in the local optimal solution if there is no probabilistic selection. Therefore, after calculating the selection probability (Equation (28)), the roulette wheel selection method is employed to select a female individual and avoid local optima.

**Movement Strategy**

Once the selected water stride's goal quantity is lower than the existing water stride, the current water stride will initiate the moving controller (Ramezani et al., 2020). Even if the distance between the 2 water striders is less appealing, the moving process will likely conclude sooner and the optimal answer won't be found. The following formula is proposed as a solution to this problem:

$$\alpha = \left(\frac{r}{1600}\right) \times logsig\ \left((-\beta)^{\frac{t}{600}}\right) \tag{29}$$

$$x_i^{t+1} = x_i^t + v \times rand \times \left(x_{BL}^t - x_i^t\right) \tag{30}$$

where the epoch time is indicated by $t$, the regression analysis function $logsig(,)$ is specified in the range [0,1], and the attraction variable $v$ is denoted by, if $r = 0$, $v$ is equivalent to 0.

**Initialization by Chaos Theory**

The typical WSA primarily employs a random distribution strategy to produce the initial population. When the search space is large, the beginning population finds it difficult to supply a high ergodic degree, which affects how successfully the water striders solves problems. To improve the quality of the vectors, the beginning locations are begun using a pseudo-random chaotic sequence. The chaotic logistic map creates the chaotic sequences, and the following may be said about the link between the maps:

$$x_i^{t+1} = \mu \times x_i^t \times \left(1 - x_i^t\right) \tag{31}$$

where, $\mu$ determines bifurcation coefficient which is in the interval [3.57, 4], $x_i^t$ signifies the $i^{th}$ chaotic candidate in the interval [0, 1], where, $x_i \notin [0, 0.25, 0.5, 0.75, 1]$ (Liu et al., 2005). Straight away, a certain chaotic variable tracking that is considered for the entire search space.

**Performance Evaluation**

To validate the performance of the proposed EWSA, six common benchmark functions have been employed that illustrated in Table 1. The proposed EWSA has been compared to four recent optimizers namely, the billiard-based optimization algorithm (BOA) (Kaveh et al., 2020), black hole (BH) (Hatamlou, 2013), locust swarm optimization (LS) (Cuevas et al., 2020), and the original WSA (Kaveh and Eslamlou, 2020). Simulations have been conducted using MATLAB version R2017b on a laptop with AMD A4 3600 processor and 8GB RAM. The algorithms' parameters are shown in Table 1.

Table 1. Algorithms' parameters utilized in this simulation

| Algorithm | Parameters | Values |
|---|---|---|
| BOA (Kaveh et al., 2020) | No. of pockets | 19 |
| | $w$ | 0.6 |
| | $ES$ | 0.5 |
| BH (Hatamlou, 2013) | $a$ | 0.7 |
| | Number of stars | 60 |
| LS (Cuevas et al., 2020) | F | 0.4 |
| | L | 1 |
| | g | 23 |

Table 2 lists the mathematical equation, dimension ($D$), range, and fitness value ($F_{min}$) of the optimal solution for each of the benchmark functions under investigation.

Table 2. The information about the utilized test functions

| Type | Function Name | Function | D | Range | $F_{min}$ |
|---|---|---|---|---|---|
| Unimodal | Sphere | $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 30 | [-100,100] | 0 |
| | Schwefel2.22 | $F_2(x) = \sum_{i=1}^{n} \lvert X_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ | 30 | [-10,10] | 0 |
| Multimodal Basic Functions | Rosenbrock's | $F_3(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | 30 | [-30,30] | 0 |
| | Quartic | $F_4(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$ | 30 | [-128,128] | 0 |
| Multimodal Benchmark Functions | Schwefe | $F_5(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$ | 30 | [-500,500] | -418.99 |
| | Ackley | $F_6(x) = -20\, exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$ | 30 | [-32,32] | 0 |

For a fair comparison, the population size is set to 30, and the maximum number of iterations for all optimization algorithms is set to 200 (Razmjooy et al., 2016). The average (AVG) of the 45 (Razmjooy et al., 2019, 2021) experimental results and standard deviation (SD) are used as evaluation indicators, and the statistical results are shown in Table 3.

Table 3. Comparison results of the algorithms

| Algorithms / Function | BOA (Kaveh et al., 2020) | | BH (Hatamlou, 2013) | | LS (Cuevas et al., 2020) | |
|---|---|---|---|---|---|---|
| | AVG | SD | AVG | SD | AVG | SD |
| Sphere | 5.246 E-10 | 3.734E-11 | 5.123 E-12 | 4.705-13 | 6.154 E-12 | 3.158E-14 |
| Schwefel2.22 | 4.128E-8 | 0.42E-8 | 2.3011E-10 | 2.730E-11 | 2.493 E-12 | 1.853E-13 |
| Rosenbrock's | 0.760 | 0.00574 | 1.96 | 0.081 | 1.723 | 1.0528 |
| Quartic | 1.437E-5 | 0.084 | 0.007 | 0.004 | 0.0024 | 0.0084 |
| Schwefe | -115.507 | 24.1 | -153.364 | 24 | -195.351 | 19.15 |
| Ackley | 1.213E-9 | 3.43-10 | 3.13E-9 | 2.25E-10 | 2.65E-10 | 1.64E-11 |
| Algorithm / Function | WSA (Kaveh and Eslamlou, 2020) | | EWSA | |
| | AVG | SD | AVG | SD |
| Sphere | 4.241 E-15 | 3.156E-16 | 4.124 E-16 | 2.031E-17 |
| Schwefel2.22 | 1.098 E-13 | 1.238E-14 | 1.650E-12 | 1.095E-13 |
| Rosenbrock's | 0.847 | 0.040 | 0.2162 | 0.915 |
| Quartic | 0.00953 | 0.0056 | 0.00125 | 0.00001 |
| Schwefe | -186.705 | 18.15 | -245.730 | 13 |
| Ackley | 1.06 E-11 | 1.14E-12 | 2.78 E-11 | 2.53E-12 |

From the comparison of the statistical results in Table 3, it can be seen that the performance of EWSA is better than the WSA and other algorithms in solving all benchmark functions.

**Module Placement Using Courtship Learning-based Water Strider Algorithm**

This algorithm receives the application component deployment problem data, including application component request resource information, component communication, fog network infrastructure configuration, amount of population members, and the greatest number of iterations as input parameters, and then an optimal deployment plan as output. Algorithm 1 demonstrates the pseudocode of the proposed method.

| Algorithm 1: Pseudocode of the proposed method |
|---|
| 1) Initializing: input Application components specification, Fog Network specification and the EWSA |
| 2) Apply preprocessing |
| 3) Perform initial step of Courtship Learning-based Water Strider Algorithm |
| 4) Calculate cost function |
| 5) If stopping criteria is reached, go to (9) and return the results |
| 6) Else |
| 7) Update Fog data structure based on the algorithm updating formulas |
| 8) Go to (3) |
| 9) End |

Figure 4. Pseudocode of the proposed method

Each water strider can be considered a possible solution to the deploying application components optimization problem in fog nodes. A water strider consists of $N$ components, which represent a used component. The allocation value given to each component is an integer between 1 and $M$, which indicates the location of the components. As can be observed from Algorithm 1, Before executing the main loop of the program, preprocessing steps are performed. Then, some algorithms are used to extract full mesh subnets. Then in the main loop, two parts are executed. The impossible solutions are then corrected. The $P_a$ percent of a less valuable solution is then updated based on the cost function. To do this, $P_a$% has been chosen based on the worst-ranked solutions. Then, they updated by running the random-step procedure. Merit values are then calculated to update the solutions and after the execution of the last round of the iteration loop, the optimal solution is returned.

**Preprocessing Stage**

Figure 5 shows the proposed method for the pre-processing step. In this figure, the Fog net is the Fog Network Communication Matrix with dimension $j \times i$, such that $j, i = 1,2, \dots, n$, and $n$ is the number of fog nodes.

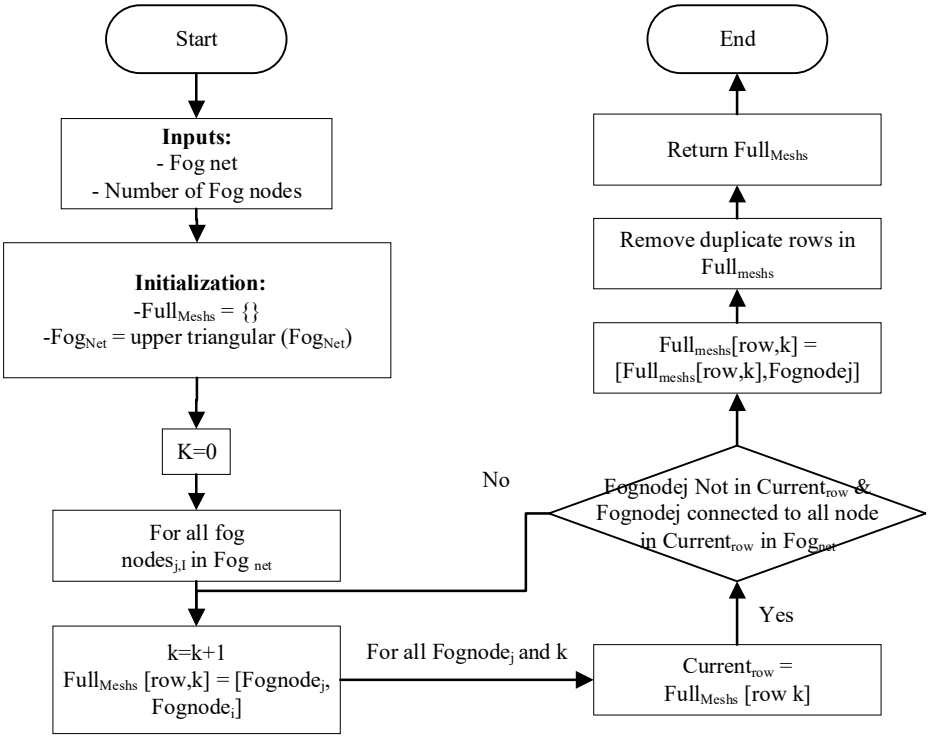


Figure 5. Preprocessing stage

As can be observed from Figure 5, in the pre-processing stage, a set of complete mesh subnets are selected in the fog network for the deployment of components. The extraction of these subnets has several advantages: the first advantage is the decrease of the search space for extracting the optimal deployment plan, and the second advantage is the sharing of sensors and software required by the components of the complete mesh subnets.

In the proposed algorithm, at the beginning of entering the condition, all the nodes that are connected are extracted and each pair is stored in the $Full_{meshs}$ array. For the loop, all fog nodes are compared with lines from $Full_{meshs}$ that do not exist in it, and if it is connected to all the nodes in it, it is added to the node list of the current line. Then, the repeated lines from $Full_{Meshes}$ are deleted. This process continues until the end of the condition, and finally, the $Full_{meshs}$ array, which contains a set of $full_{mesh}$ subnets are returned.

The stopping criteria of the algorithm are the desired number of $k$ clicks, and in other words, the key loop is repeated $k$ times. After extracting the complete meshes, several

complete meshes are selected for the deployment of components. Figure 6 shows the pseudocode of extracting candidate complex subnets.

| Algorithm2: Pseudocode of finding candidate $Full_{mesh}$ subnetworks |
|---|
| 1) Initializing |
| Application components list $Appcmp = (1, \dots, n)$ |
| Fog node list $(fn = (1, \dots, m))$ |
| Bandwidth between components $(Appcmp_{BW} = array(n \times n))$ |
| Bandwidth between fog nodes $(FN_{BW} = array(m \times m))$ |
| Latency between fog nodes $(FN_{Latency} = array(m \times m))$ |
| Latency between components $Appcmp_{Latency} = array(n \times n))$ |
| AppcmpDataset is Application components resource requirements |
| FNDataset is Fog nodes resources |
| $Full_{meshs}$ is $Full_{meshs}$ list |
| 2) For each $Full_{meshs}$ as $row_i$: |
| $Latency_{BW_{status}} = Check_{Latency_{BW}}(Appcm\_Latency, Appcmp_{BW}, row_i, FNDataset, FN_{BW}, FN_{Latency})$; |
| 3) $HR_{status} = HR_{chk}(AppcmpDataset, FNDataset, row_i)$ |
| 4) $SR_{status} = SR_{chk}(AppcmpDataset, FNDataset, row_i)$ |
| 5) $S_{status} = S_{chk}(AppcmpDataset, FNDataset, row_i)$ |
| 6) If $Latency_{BW_{status}}$ & $HR_{status}$ & $SR_{status}$ & $S_{status}$ are true, go to (7) |
| 7) $Candidate_{Full_{meshs}} = [Candidate_{Full_{meshs}}; row_i]$ |
| 8) Return $Candidate_{Full_{Meshs}}$ |

Figure 6. Pseudocode of extracting candidate complex subnets

## Results

Several tests are planned and carried out to evaluate the success of the proposed EWSA in determining the best deployment strategy for application components in the fog infrastructure. We assessed the suggested strategy using the energy consumption evaluation criterion. Fog node energy consumption is often influenced by several variables like the volume of exchange traffic, communication technology, the distance between nodes, and the number of computations.

A fog node's energy consumption is determined by taking into account both the energy used for processing application components on the node and the energy used for information transfer between fog nodes. The outcomes produced by the suggested algorithm are compared with those of some other metaheuristic approaches, including the accelerated particle swarm optimization algorithm (APSO) (Ouyang et al., 2022), the multiswarm algorithm (MSA) (Hasan and Al-Rizzo, 2019), accelerated particle swarm optimization algorithm (APSO)

(Ouyang et al., 2022), the multiswarm algorithm (MSA) (Hasan and Al-Rizzo, 2019), and the original WSA (Kaveh & Eslamlou, 2020).

To evaluate the proposed framework, two scenarios were implemented. In each scenario, the number of modules and the amount of fog nodes changes. In the first scenario, 20 mesh nodes and 30 modules were considered, and in the second scenario, 25 mesh nodes and 40 components were considered. We performed all the experiments by MATLAB version R2017b on a laptop with AMD A4 3600 processor and 8GB RAM as aforementioned. To calculate the capability of the proposed algorithm in providing the optimal deployment plan of application components, fog nodes are assumed to be heterogeneous in terms of storage capacity, processing ability, delay and bandwidth. For better display, an example of data set information, a fog network with 5 nodes is shown in Table 4 to Table 6.

Table 4. Sources of fog nodes

| Fog nodes | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| The central processing unit (GHz) | 1.03 | 1.016 | 1.39 | 1.45 | 1.05 |
| Memory size (GB) | 1.29 | 1.58 | 1.19 | 1.35 | 1.29 |
| Consumption threshold | 0.94 | 0.90 | 0.94 | 0.90 | 0.89 |
| Memory consumption threshold | 1.00 | 0.97 | 0.90 | 0.91 | 0.95 |
| Minimum energy consumption | 92 | 80 | 95 | 79 | 75 |
| Maximum energy consumption | 125 | 125 | 128 | 135 | 130 |
| Sensors | 0.95 | 1.1 | 1.1 | 1.8 | 0 |
| Software | 0 | 1.1 | 1.8 | 0 | 1.8 |
| transfer energy | 0.15 | 0.15 | 0.15 | 0.8 | 0.8 |

As seen in Table 4, CPU and memory consumption thresholds, computing resources, as well as minimum and maximum energy consumption, supported software, transmission energy, and types of sensors, are considered different for each node. The value of 0 in the sensors and software means that the sensors and software are not supported, and the values of 1 and 2 show the supported types of software and sensors.

In Tables 5 and 6, the bandwidth and delay in the direct connection between nodes are shown, and the values in these tables are normal numbers in the range [0, 1]. The value 0 means no connection between node $j$ and $i$ and bandwidth 1 means that node $j$ is the same as node $i$.

Table 5. Width of the band between fog nodes

| Fog knots | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.99 | 0.92 | 0.89 | 0.82 | 0.93 |
| 2 | 0.80 | 0.98 | 0.81 | 0.91 | 0.90 |
| 3 | 0.91 | 0.92 | 0.99 | 0.96 | 0.99 |
| 4 | 0.87 | 0.91 | 0.89 | 0.99 | 0.99 |
| 5 | 0.90 | 0.95 | 0 | 0.89 | 0.98 |

In Table 6, a delay of 1 between two nodes means that there is no connection between the two nodes and a delay of 0 means that the two nodes are the same.

Table 6. The delay between fog nodes

| Fog knots | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0.15 | 0.18 | 0.09 | 0.09 |
| 2 | 0.11 | 0 | 0.09 | 0.1 | 0.16 |
| 3 | 0.17 | 0.13 | 0 | 0.11 | 0.9 |
| 4 | 0.15 | 0.17 | 0.13 | 0 | 0.13 |
| 5 | 0.1 | 0.13 | 0.99 | 0.15 | 0 |

An example of the specifications of the application components, including the type of sensor, CPU, memory, and the required software, as well as the connections and delays between them, are shown in Tables 7, 8, and 9.

Table 7. Required resources of application components

| Fog knots | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| CPU | 0.14 | 0.18 | 0.21 | 0.25 | 0.28 |
| Memory | 0.15 | 0.15 | 0.1 | 0.15 | 0.1 |
| Sensors | 0.98 | 0 | 0.98 | 0 | 1.9 |
| Software | 0 | 0.9 | 0 | 1.1 | 0.9 |

Table 8: Bandwidth required by application components

| Components | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.98 | 0.98 | 0 | 0 | 0 |
| 2 | 0 | 0.98 | 0.30 | 0 | 0.31 |
| 3 | 0 | 0 | 1 | 0.29 | 0.19 |
| 4 | 0 | 0 | 0 | 0.98 | 0.35 |
| 5 | 0 | 0 | 0 | 0 | 0.98 |

Table 9: The required delay of application components

| Components | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0.98 | 0.98 | 0.98 | 0.98 |
| 2 | 0 | 0 | 0.19 | 0.98 | 0.25 |
| 3 | 0 | 0 | 0 | 0.21 | 0.19 |
| 4 | 0 | 0 | 0 | 0 | 0.20 |
| 5 | 0 | 0 | 0 | 0 | 0 |

It should be mentioned that due to the absence of a standard benchmark in the literature on this subject¸ here, we produced a data set with a range of values similar to Table 4 to Table 9. These values, processing power and storage capacity of fog nodes have been selected based on the internal processors, smartphones, and digital assistants for individuals that make up today's fog nodes as well as their topologies and computational capabilities. To compare the performance of the studied algorithms, graphs of the state of the cost function in the repetition of the algorithms (i.e., convergence profile), and the minimum rate of energy consumption have been used. The nodes in the fog network are scattered randomly and with a normal distribution.

**Scenario 1**

A network with 20 fog nodes and 30 components. Figure 7 shows the superiority of the proposed EWSA over the proposed method in some of the iteration steps, but in the end, the proposed EWSA meets our goal in comparison with WSA (Kaveh and Eslamlou, 2020), MSA (Hasan and Al-Rizzo, 2019), and APSO (Ouyang et al., 2022).

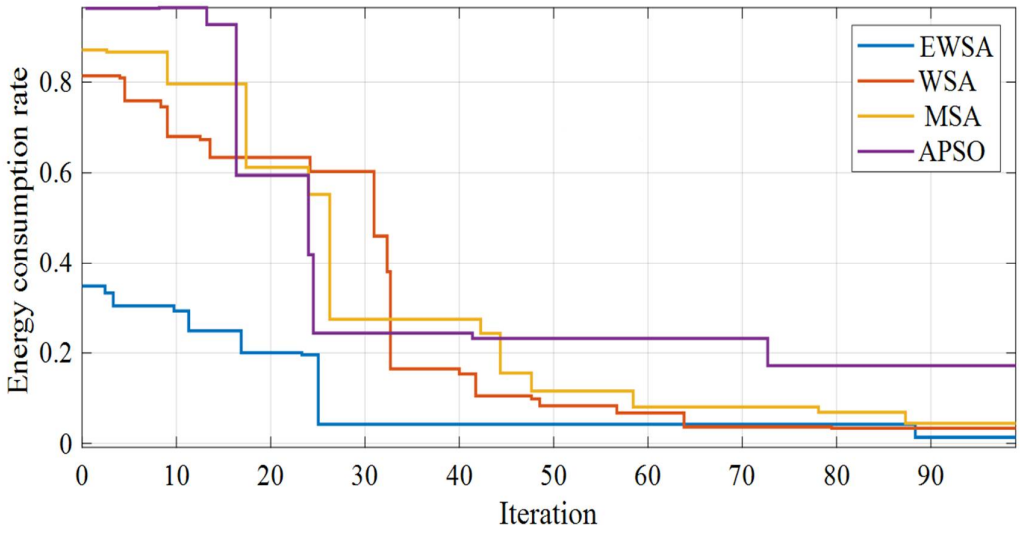


Figure 7. The convergence analysis for the energy consumption rate based on the studied algorithms for scenario 1.

As can be seen, the proposed EWSA provided the best convergence. Indeed, we need a balance between the accuracy and the convergence profile to show the method's effectiveness. At the end of the simulation, the optimal solution obtained according to the objective function (Eq. (12)) by the proposed EWSA and the original WSA shows better results in comparison with MSA and APSO.

Table 10 shows the optimal rate of energy usage at the end of the simulation based on the studied methods.

Table 10. The optimal rate of energy usage at the end of the simulation for scenario 1

| Algorithm | The optimal rate of energy consumption |
|---|---|
| EWSA | 0.01364 |
| WSA (Kaveh and Eslamlou, 2020) | 0.03372 |
| MSA (Hasan and Al-Rizzo, 2019) | 0.04432 |
| APSO (Ouyang et al., 2022) | 0.1719 |

The optimal solution obtained in Table 10 are achieved according to the objective function (Eq. (12)). As can be observed, the proposed EWSA and the original WSA show better results in comparison with MSA and APSO algorithms.

With the increase in the number of nodes in the fog, the number of complete meshes that can be extracted for the distribution of components increases, but due to the heterogeneous nature of the nodes and application components, the candidate complete meshes for the distribution of components are limited. In Figure 8, the position of fully selected mesh nodes is shown in the network map.

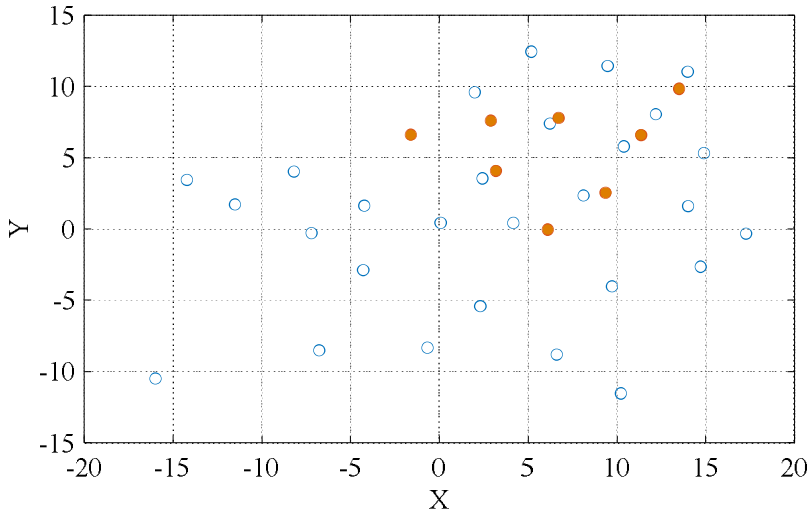


Figure 8. Position of fully selected mesh nodes for scenario 1

As can be observed from Figure 8, after applying the proposed optimization methodology, the complete selection for the distribution of components includes nodes 16, 15, 14, 8, 5, and 18.

**Scenario 2**

A network with 25 fog nodes and 40 modules. Figure 9 shows the close competition of the algorithms can be seen in extracting solutions with the minimum value in the TPC objective function, but finally, in the final stages, the repetition of the EWSA overcomes the other algorithms with a small difference.

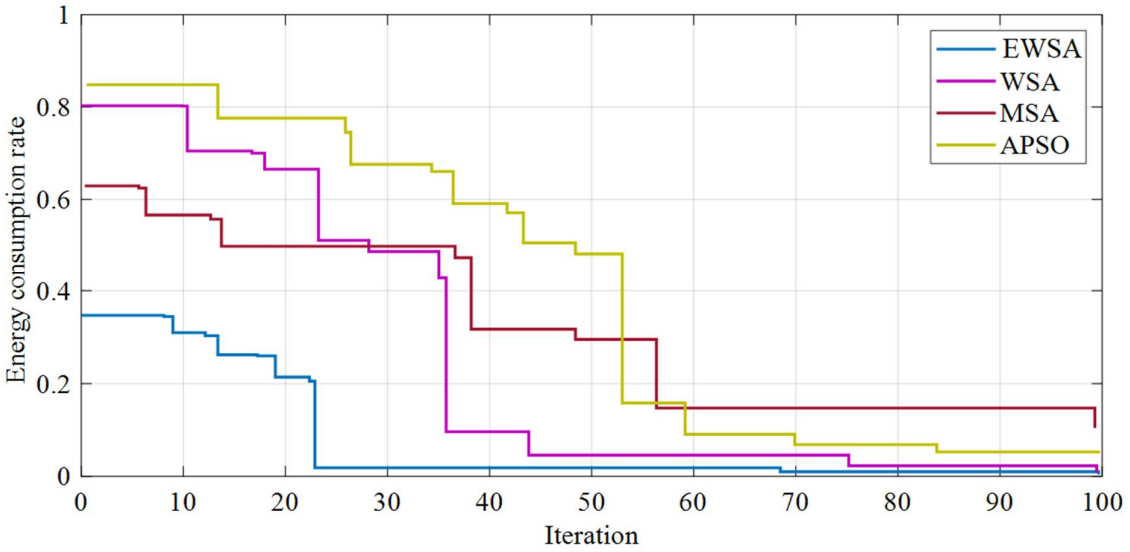


Figure 9. The convergence analysis for the energy consumption rate based on the studied algorithms for scenario 2

The optimal solution obtained according to the objective function (12) shows the difference between the EWSA and WSA (Kaveh and Eslamlou, 2020), MSA (Hasan and Al-Rizzo, 2019), and APSO (Ouyang et al., 2022) at the end of the simulation. Table 11 shows the optimal rate of energy usage at the end of the simulation based on the studied methods.

Table 11. The optimal rate of energy usage at the final of the simulation for scenario 2

| Algorithm | Optimal rate of energy consumption |
|---|---|
| EWSA | 0.01004 |
| WSA (Kaveh and Eslamlou, 2020) | 0.02277 |
| MSA (Hasan and Al-Rizzo, 2019) | 0.4991 |
| APSO (Ouyang et al., 2022) | 0.1472 |

Figure 10 shows the position of fully selected mesh nodes is shown in the network map.

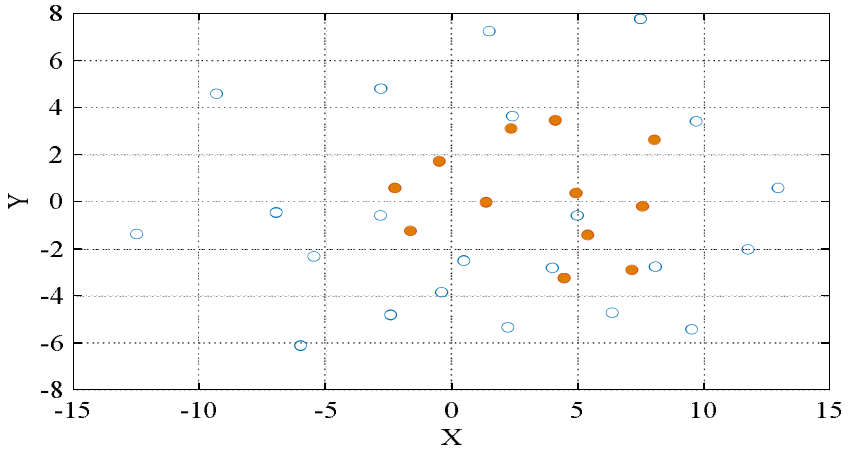


Figure 10. Position of fully selected mesh nodes for scenario 2

The full mesh subnet selected by the EWSA for the distribution of components includes nodes 18, 14, 11, 12, 9, 7, 6, 5, 1 and 20.

## Conclusion

With the aid of internet infrastructure, the IoT enables remote management and control of devices used for various purposes, enhances productivity scalability and connection, and helps organizations save money and time. Fog computing and the IoT are expanding quickly in the world of networking. The traditional centralized cloud computing paradigm will confront several difficulties due to the explosive expansion of IoT applications, including limited capacity, network failure, and excessive latency. Instead of using the cloud, IoT devices may process, safeguard, and locally store data by using fog. In actuality, fog offers IoT users more outstanding performance and quicker solutions than the cloud. The utilization of computer resources in the cloud architecture is effectively reduced by deploying IoT applications as a supplement to the cloud. Application components deployed inefficiently in the fog waste bandwidth, energy, and resources. Additionally, dispersing an application's elements among the fewest amount of fog nodes feasible to save energy results in decreased service dependability. In this study, an improved metaheuristic approach based on WSA, chaos theory and courtship learning were designed for the static distribution of application elements on fog structure to strike a balance between optimal energy consumption, minimizing the impact of a single goal of failure, and enhancing the dependability of the application against damage. The simulations of the proposed method were compared with those from Accelerated Particle Swarm Optimization Algorithm, the multiswarm algorithm, and the basic Water Strider Algorithm. The consequences demonstrated that the approach described in this article lowers the energy usage in the fog network and offers the high dependability needed to meet the service quality criteria of an IoT application. Simulation results showed that the proposed EWSA with 0.01364 and 0.01004 for scenarios one and two provided the most efficient results.

## Conflict of interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

## Funding

The authors would like to thank Al-Mustaqbal University for the financial support provided to carried out this research.

## References


Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021a). The arithmetic optimization algorithm. *Computer methods in applied mechanics and engineering*, *376*, 113609.

Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021b). Aquila optimizer: a novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering*, *157*, 107250.

Ahmed, A. M., Rashid, T. A., & Saeed, S. A. M. (2020). Cat swarm optimization algorithm: a survey and performance evaluation. *Computational intelligence and neuroscience*, *2020*.

Al-Khafaji, H. M. R. (2022). Improving Quality Indicators of the Cloud-Based IoT Networks Using an Improved Form of Seagull Optimization Algorithm. *Future Internet*, *14*(10), 281.

Al-Khafaji, H. M. R., Alomari, E. S., & Majdi, H. S. (2019, October). Secured environment for cloud integrated fog and mist architecture. In *2019 IEEE International Conference on Electrical Engineering and Photonics (EExPolytech)* (pp. 112-116). IEEE.

Arshed, J. U., Ahmed, M., Muhammad, T., Afzal, M., Arif, M., & Bazezew, B. (2022). GA-IRACE: Genetic Algorithm-Based Improved Resource Aware Cost-Efficient Scheduler for Cloud Fog Computing Environment. *Wireless Communications and Mobile Computing*, *2022*.

Chegini, H., Naha, R. K., Mahanti, A., & Thulasiraman, P. (2021). Process automation in an IoT–fog–cloud ecosystem: A survey and taxonomy. *IoT*, *2*(1), 92-118.

Cuevas, E., Fausto, F., & González, A. (2020). The locust swarm optimization algorithm. *New advancements in swarm algorithms: operators and applications*, 139-159.

Hasan, M. Z., & Al-Rizzo, H. (2019). Optimization of sensor deployment for industrial internet of things using a multiswarm algorithm. *IEEE Internet of things journal*, *6*(6), 10344-10362.

Hassan, S. R., Ahmad, I., Rehman, A. U., Hussen, S., & Hamam, H. (2022). Design of resource-aware load allocation for heterogeneous fog computing environments. *Wireless Communications and Mobile Computing*, *2022*, 1-11.

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information sciences*, *222*, 175-184.

Kaveh, A., & Eslamlou, A. D. (2020, June). Water strider algorithm: A new metaheuristic and applications. In *Structures* (25), 520-541. Elsevier.

Kaveh, A., Khanzadi, M., & Moghaddam, M. R. (2020, October). Billiards-inspired optimization algorithm; a new meta-heuristic method. In *Structures* (27), 1722-1739. Elsevier.

Liu, B., Wang, L., Jin, Y. H., Tang, F., & Huang, D. X. (2005). Improved particle swarm optimization combined with chaos. *Chaos, Solitons & Fractals*, *25*(5), 1261-1271.

Ouyang, M., Xi, J., Bai, W., & Li, K. (2022). Band-area resource management platform and accelerated particle swarm optimization algorithm for container deployment in Internet-of-Things cloud. *IEEE Access*, *10*, 86844-86863.

Ramezani, M., Bahmanyar, D., & Razmjooy, N. (2020). A new optimal energy management strategy based on improved multi-objective antlion optimization algorithm: applications in smart home. *SN Applied Sciences*, *2*, 1-17.

Razmjooy, N., Ashourian, M., & Foroozandeh, Z. (Eds.). (2021). Metaheuristics and optimization in computer and electrical engineering.

Razmjooy, N., Estrela, V. V., & Loschi, H. J. (2019). A study on metaheuristic-based neural networks for image segmentation purposes. In *Data science* (25-49). CRC Press.

Razmjooy, N., Estrela, V. V., Padilha, R., & Monteiro, A. C. B. (2020). World cup optimization algorithm: Application for optimal control of pitch angle in hybrid renewable PV/wind energy system. In *Metaheuristics and Optimization in Computer and Electrical Engineering* (25-47). Cham: Springer International Publishing.

Razmjooy, N., Khalilpour, M., & Ramezani, M. (2016). A new meta-heuristic optimization algorithm inspired by FIFA world cup competitions: theory and its application in PID designing for AVR system. *Journal of Control, Automation and Electrical Systems*, *27*, 419-440.

Samani, Z. N., Saurabh, N., & Prodan, R. (2021, May). Multilayer resource-aware partitioning for fog application placement. In *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)* (9-18). IEEE.

Sathya Sofia, A., & GaneshKumar, P. (2018). Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. *Journal of Network and Systems Management*, *26*, 463-485.

Stojmenovic, I. (2014, November). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *2014 Australasian telecommunication networks and applications conference (ATNAC)* (117-122). IEEE.

Xiao, Y., Sun, X., Guo, Y., Cui, H., Wang, Y., Li, J., & Li, S. (2022). An enhanced honey badger algorithm based on Lévy flight and refraction opposition-based learning for engineering design problems. *Journal of Intelligent & Fuzzy Systems*, *43*(4), 4517-4540.

Zheng, Z. X., & Li, J. Q. (2018). Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption. *Energy and Buildings*, *161*, 80-88.

**Bibliographic information of this paper for citing:**

Alsaabri, Huda Hasan, & Al-Khafaji, Hamza Mohammed Ridha (2023). Energy-Efficient and Reliable Deployment of IoT Applications in a Fog Infrastructure Based on Enhanced Water Strider Algorithm. *Journal of Information Technology Management,* 15 (4), 179-204https://doi.org/10.22059/jitm.2023.94931