International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 38, No. 4, August, 2024, pp. 1077-1088
Journal homepage: http://iieta.org/journals/ria

Discrete Black Widow Optimization Algorithm for Multi-Objective IoT Application Placement in Fog Computing Environments

Chouaib Maarouk*, Hichem Haouassi, Mohamed Mahdi Malik

ICOSI Laboratory, Computer Science Department, Abbes Laghrour University, Khenchela 4000, Algeria

Corresponding Author Email: maarouk.chouaib@univ-khenchela.dz

https://doi.org/10.18280/ria.380403

Received: 26 December 2023
Revised: 12 May 2024
Accepted: 9 July 2024
Available online: 23 August 2024

Keywords:
Internet of Things (IoT), fog computing, application placement, swarm intelligence, discrete black widow optimization

ABSTRACT

The “Internet of Things" describes a network comprising variedly distributed and heterogeneous devices that communicate by exchanging data to realize various applications with minimal human intervention. However, processing the massive amounts of data in the cloud environment becomes challenging. To address this issue, fog computing has appeared as a new paradigm that extends the capabilities of cloud computing to the edge of networks. The deployment of applications on diverse and dispersed nodes is one of the key issues in fog computing This article presents an approach to optimize application placement in fog computing infrastructure by formulating it as a combinatorial problem that aims to minimize both execution times and costs. Here, we propose a Discrete Black Widow Optimization (DBWO) algorithm specifically designed to tackle the discrete nature of the application placement in fog environments. Experimental results show that our approach demonstrates an average improvement of 9% compared to several recent approaches in the literature. In fog-only topology, DBWO demonstrated an improvement range from 4.30% to 9.87%, while in fog-cloud topology, it showed notable performance improvement, with fitness value enhancement ranging from 8% to 15.16%. This innovation represents a significant stride towards efficient and cost-effective application placement in fog computing environments.

1. INTRODUCTION

The twenty-first century has witnessed the transformative rise of IoT, a technology that has become a cornerstone of modern innovation. This technology entails the interconnection of physical objects, such as sensors, smart devices, and actuators, into a vast network. The sheer scale of this interconnected ecosystem is staggering, with the number of connected devices projected to reach a staggering 75 billion by 2025 [1]. With this exponential growth comes an unprecedented volume of data, necessitating substantial computing power for efficient processing and storage [2].

Traditionally, IoT devices have often been limited in terms of processing and storage capabilities. The data generated by these IoT devices has been managed and stored in a centralized cloud infrastructure. In terms of availability, processing performance, and storage capacity, the cloud has aptly served the needs of IoT applications. However, the centralization of cloud computing presents challenges, particularly for applications that require real-time processing and minimal latency.

To address the limitations of cloud-based systems for time-sensitive IoT applications, fog computing emerged as a solution in 2012 [3]. Fog computing involves a paradigm shift that enables the cloud to collaborate with distributed nodes located closer to IoT devices. These nodes, often referred to as "edge" devices, process data autonomously and directly, making IoT applications more secure, less reliant on distant cloud resources, and highly scalable when compared to relying solely on traditional cloud computing.

Fog computing has emerged as a promising approach to enhance network performance, mitigate overload, and reduce latency, thereby meeting the stringent requirements of time-sensitive applications. However, given the resource constraints of edge devices, a critical challenge lies in efficiently allocating resources and deploying applications to ensure swift access to available resources for time-sensitive applications.

The IoT application span a wide range of use cases each with unique requirement in terms of latency, reliability and resource consumption. For example, a smart healthcare application might prioritize low latency for patient monitoring [4], while a smart grid application might prioritize high availability for real time energy management [5]. The challenge of deploying IoT applications in fog computing systems revolves around efficiently allocating these applications to the available Fog nodes which serve as intermediaries between devices and the cloud, have limited computational and storage resources. Efficiently allocating these resources to meet the requirement of diverse IoT applications poses a significant optimization challenge and falls into the category of NP-hard problems [6, 7], which cannot be effectively solved using conventional optimization techniques.

In the literature, the IoT application placement problem in fog computing has garnered significant attention from researchers in both academia and industry [7]. Numerous studies have proposed optimization techniques to address various aspects of the problem. Three main approaches have emerged to address the application placement problem, each tailored to varying application numbers and physical resource requirements. The exact solution, while theoretically optimal, is impractical for large-scale environments due to its time-consuming nature. Heuristic methods offer a solution within a reasonable timeframe but lack performance guarantees. Lastly, meta-heuristic approaches, unrelated to the optimization problem itself, guide the search towards near-optimal solutions [3]. Meta-heuristic algorithms, such as swarm intelligence and evolutionary techniques, have demonstrated their efficacy in solving optimization challenges across diverse domains [8]. One such bio-inspired meta-heuristic approach is the Black Widow Optimization (BWO) algorithm, proposed by Hayyolalam and Kazem [9]. BWO draws inspiration from the intricate reproductive rituals of black spiders and has shown promising results in addressing continuous engineering optimization problems [9-11]. However, the application placement problem within fog-cloud infrastructure inherently involves discrete optimization, posing a challenge for the original BWO algorithm, designed primarily for continuous optimization.

To bridge this gap, we present DBWO, which aims to tackle the difficulties associated with optimizing application location in fog-cloud infrastructures. DBWO, specifically designed for discrete optimization tasks, places applications in the best possible order by taking execution time and total cost into account within a discrete parameter space. Unlike the original BWO algorithm, DBWO it is well-suited to the particular requirements of application placement in fog-cloud systems. By offering more distributed and responsive solutions, fog computing seeks to solve the drawbacks of conventional cloud-centric IoT designs. By concentrating on improving application placement in fog-cloud systems to achieve effective and economical execution, our study makes a contribution to this subject.

The main contributions of this paper are:

We have formulated the application placement in the fog-cloud system as a multi-criteria optimization problem to attain the optimal balance between execution time and total cost.

To the best of the authors' knowledge, none of the previous studies proposed a Discrete BWO algorithm. The focus of this work is to present an effective implementation of the discrete BWO algorithm, which is utilized to solve the problem of IoT application placement. The objective of the optimization problem is to minimize both the time and cost of running a given set of applications on the fog infrastructure. This is achieved by selecting the most suitable resources for each application to ensure efficient and cost-effective execution.

The experimental results of several tested scenarios indicate that the proposed DBWO approach delivers superior performance in execution time and total cost compared to four recently proposed approaches: Elitism Genetic Algorithm (EGA), Discrete Particle Swarm Optimization (DPSO), Bee Life Algorithm (BLA), and Gray Wolf Optimizer (GWO).

The rest of the paper is structured as follows: The following section reviews relevant literature in the field, providing a comprehensive overview of existing research. In Section 3, we present the background and motivation for our work. Section 4 introduces our system model and the formulation of application mapping within fog computing infrastructure. The proposed DBWO algorithm, a key component of our approach, is elaborated upon in Section 5. The experimental results and an in-depth discussion of the algorithm's performance can be found in Section 6. Finally, in Section 7, we conclude our paper and outline future directions for research.

2. RELATED WORKS

Due to the wide use of the cloud-fog environment, the researchers attach great importance to its performance. Because of the complexity of cloud fog, the optimal application placement strategy is one of the important factors influencing its performance [8, 12]. Different applications placement strategies are proposed in the literature with the aim to improve the computing efficiency of the cloud-fog environment [6-8]. In this section, we present the previous application placement strategies, focusing on their nature as exact, heuristic, and metaheuristic solutions.

2.1 Exact solutions

Previous studies have used Integer Linear Programming (ILP) solvers to obtain exact solutions for application placement in fog infrastructure. Skarlat et al. [13] employed the IBM CPLEX solver and Java ILP within the Ifogsim simulator [14] to address the service placement problem in a fog environment. Their optimization aimed to minimize the usage of the fog environment while considering the application's Quality of Service (QoS) requirements. The results showed that their optimization reduced the execution cost, and the solution did not violate the application deadline.

Similarly, Minh et al. [15] used an ILP solver to offer a service placement policy that maximizes the placement of services in a fog landscape, resulting in improvements in terms of delay and energy usage. Arkian et al. [16] formulated the service placement problem in a fog environment as mixed-integer nonlinear programming (MINLP), with the objective of lowering the total cost while meeting the application's QoS requirements. The simulation findings demonstrated cost, energy, and latency improvements.

Tran et al. [17] recommended service placement in a decentralized fog landscape based on context-aware data such as resource consumption, location, and reaction time. Their proposed method was found to be efficient for maximizing fog device utilization and decreasing latency. However, other optimization methods are less prevalent, and only a few scholars have investigated their application in fog computing. For example, in the study [12], the problem was formulated using constraint programming to satisfy the QoS criterion, and Choco-solver was used to solve it.

2.2 Heuristics approaches

Exact solution algorithms, such as ILP solvers, are frequently employed methods for tackling the issue of application placement in fog computing systems. However, due to their time-consuming nature, these algorithms are not suitable for large-scale infrastructure. As an alternative, heuristics algorithms, such as search-based strategies, are used to find feasible solutions within an acceptable timeframe.

In the literature, researchers have suggested and explored various search-based algorithms to optimize application placement in fog infrastructure. One such approach is the greedy backtracking heuristic algorithm proposed by Brogi and Forti [18], which employs fail-first or fail-last strategies to select the candidate node. Brogi et al. [19] extended the backtracking search algorithm [18] to estimate QoS assurance using the Monte Carlo method. Xia et al. [20] proposed a backtracking service placement solution that minimizes the response time of IoT applications.

Similarly, Lera et al. [5] proposed a first-fit heuristic algorithm to place services in fog device communities, optimizing QoS and service availability. Benamer et al. [21] proposed an exact and Latency Aware-Placement Heuristic (LAPH) algorithm to place IoT modules to reduce overall latency, with simulation results showing that the heuristic approach is significantly closer to the optimal solution within a short period of time. Finally, Azizi et al. [4] proposed a heuristic algorithm called most delay-sensitive application (MDAF) first for QoS-aware service placement, which prioritizes time-sensitive applications closer to the data source, resulting in improvements in latency and cost compared to the edge ward algorithm [14].

2.3 Meta-heuristics approaches

In the current big data era, which includes fields like social networks, health services, neuroscience, and eLearning, massive amounts of high-dimensional data are ubiquitous. The fast expansion of data and the need for responses in a short time create difficulties in effectively and efficiently managing applications and data, so it is desirable to apply intelligent optimization techniques as metaheuristics. Metaheuristics are optimization algorithms that are characterized by their simplicity, it obtains promising results in several optimization problems [22]. Metaheuristic algorithms can be simply changed to address specific issues. It efficiently examines the search space by balancing its two primary basic strategies, exploration and exploitation of the search space [23].

Several metaheuristic algorithms have been applied to the application placement issue throughout the last decade. Brogi and Forti [18] provided a framework for installation of IoT services in fog. To eliminate communication delays, they devised a Genetic Algorithm (GA). The suggested approach demonstrates a shorter deployment time than cloud-only placement, a first-fit solution, and an exact solution. Bitam et al. [24] suggested a multi-objective work scheduling issue in a fog environment using the bee life method to find a point where there is a compromise between the amount of memory available and the time it takes to process a task. The evaluation findings of their suggested method surpass those of Particle Swarm Optimization (PSO) and GA. Ayoubi et al. [25] presented an autonomous service placement using a four-phase methodology: monitoring, analysis, decision, and execution. The authors applied the Strength Pareto Evolutionary Algorithm II (SPEA-II) to make decisions in multi-objective optimization. Many performance criteria indicate that the suggested approach surpasses existing state-of-the-art approaches. Canali and Lancellotti [26] suggest a genetic algorithm for service placement by mapping data streams from the sensor to the fog node, the delay in transmission between sensors and nodes is the optimization aim of their research. Djemai et al. [27] introduced IoT application mapping as a dual-objective optimization problem to reduce system energy usage and boost QoS. The authors presented a method for placement based on DPSO. The results of simulations indicate that the DPSO approach decreases energy usage and reaction time overall.

Guerrero et al. [28] addressed a multi-objective service placement issue in a random fog network infrastructure, considering three optimization objectives: network latency, service dispersion, and resource consumption. The authors used three evolutionary algorithms. The experimental results showed the effectiveness of both NSGA-II and MOEA/D compared to WSGA. Salimian et al. [29] proposed an automatic application placement to optimize the system performance and the execution cost in a three-layer hierarchical architecture based on the GWO algorithm. According to the simulations, the proposed GWO algorithm outperforms the other five mentioned algorithms. Yadav et al. [30] developed a hybrid algorithm using traditional GA and PSO algorithms called GAPSO to optimize execution time and energy consumption. The results of the experiments demonstrate that GAPSO outperforms the GA and the PSO. Natesha and Guddeti [31] formulated the application placement as a multiobjective optimization problem and used the Elitism Genetic Algorithm (EGA) to optimize execution time, cost, and energy consumption. The results of simulations showed that the proposed EGA outperformed the GAPSO [30] and the other mentioned heuristic approaches.

Previous studies have utilized various techniques for modeling the IoT application placement problem, as summarized in Table 1. These include exact solutions such as ILP heuristic approaches and metaheuristic algorithms. While exact solution offer precision they can be impractical for large scale deployments. Heuristic approaches may provide fast solution but my not always guarantee optimality. Metaheuristic algorithms may struggle to balance between exploration and exploitation. In this paper, we propose a novel metaheuristic approach that aims to balance between solution quality and computational efficiency by considering application requirements and nodes capabilities.

Table 1. Summary of existing optimization approaches for application placement in fog computing environment

Ref.	Nature	Algorithm	Optimization Objectives	Findings
[13]	Exact Solution	IBM CPLEX, Java ILP	Minimize usage, meet QoS	Reduced cost, No violation
[15]	Exact Solution	ILP solver	Maximize placement, delay	Improved delay, Energy usage
[16]	Exact Solution	MINLP	Lower cost, meet QoS	Cost, Energy, Latency improvements
[17]	Exact Solution	Constraint Programming	Maximize utilization, reduce latency	Efficient utilization, Latency reduction
[12]	Exact Solution	Choco-solver	Satisfy QoS criterion	Satisfactory QoS compliance
[21]	Heuristic	LAPH	Reduce overall latency	Closer to optimal latency
[12]	Heuristic	MDAF	Prioritize time-sensitive apps	Improved latency, Reduced cost
[18]	Metaheuristic	GA	Minimize deployment time, QoS	Shorter deployment time, Improved

				QoS
[24]	Metaheuristic	BLA	Memory compromise, processing time	Superior to PSO and GA
[25]	Metaheuristic	SPEA-II	Various performance criteria	Outperforms state-of-the-art
[26]	Metaheuristic	GA	Minimize transmission delay	Reduced transmission delay
[27]	Metaheuristic	DPSO	Reduce energy usage, boost QoS	Decreased energy usage, Improved QoS
[28]	Metaheuristic	WSGA, NSGA-II and MOEA/D	Network latency, service dispersion, resource consumption	Effectiveness of NSGA-II and MOEA/D
[29]	Metaheuristic	GWO	Optimize performance, execution cost	Outperforms other algorithms
[30]	Metaheuristic	GAPSO (GA and PSO)	Optimize execution time, energy consumption	Superior to GA and PSO
[31]	Metaheuristic	EGA	Optimize execution time, cost, energy consumption	Outperforms GAPSO and other heuristic approaches

3. BACKGROUND AND MOTIVATION

In this section, we provide a comprehensive backdrop to the challenges and opportunities presented by fog computing in IoT landscape. We emphasize the need for advanced optimization techniques to address the intricate problem of IoT application placement in fog computing environments. Proper placement of IoT application ensures that the data is processed closer to devises, reducing latency and improving response time. Efficient placement technique that uses swarm intelligence and meta-heuristic algorithms, such as BWO algorithm, are crucial for achieving optimal performance.

3.1 Fog computing in IoT

Fog computing, a paradigm introduced to address the evolving landscape of IoT, has reshaped the way we process and manage data generated by interconnected devices. IoT encompasses a vast and growing network of sensors, smart devices, and actuators, with projections estimating an astonishing 75 billion connected devices by 2025 [1]. This proliferation of IoT devices has ushered in a new era of data generation, creating a pressing need for efficient data processing and storage solutions.

Traditionally, the centralized comprising infrastructure served as the primary hub for IoT data management. While cloud computing offered scalability and storage capabilities, it also posed significant challenges, particularly for applications requiring real-time processing and low latency [21]. The centralized nature of cloud computing meant that data had to traverse long distances, resulting in undesirable delays and potential performance bottlenecks.

3.2 Challenges in IoT application placement

The emergence of fog computing in 2012 [3] introduced a new solution to the limitations of the cloud architectures, by extending the capabilities of cloud computing to the edge of networks. Offering several advantages, including reduced reliance on distant cloud resources, and scalability that aligns with the dynamic nature of IoT applications.

Yet, a significant challenge arises in this distributed computing paradigm: efficiently allocating resources and deploying applications to ensure swift access to available resources for time-sensitive IoT applications. The core issue revolves around the deployment of IoT applications in fog computing environments, where applications must be allocated to available physical resources to meet performance and latency objectives. Given the sheer number of edge devices within IoT networks, this problem falls into the category of NP-hard problems [6, 7], making it impractical to solve using conventional optimization techniques. Fog nodes have resource constraints, such as computing power, storage and network, make it difficult to allocate resources efficiently while mating the applications requirements. The increasing number of IoT devices make scalability another concern. Additionally, Many IoT devices requires real-time processing [4] and the dynamic nature of IoT ecosystem [6] make the resources availability change constantly. Lastly Balancing multiple objectives such as minimizing execution time, cost simultaneously, adds to the complexity.

3.3 The need for optimization

Optimizing the placement of IoT applications in fog computing environments becomes imperative to unlock the full potential of this paradigm. It involves striking a delicate balance between execution time and total cost [32], two pivotal factors in fog computing scenarios. Achieving this balance ensures that applications run efficiently, delivering the desired performance while managing operational costs effectively.

The complexity of this optimization challenge cannot be overstated. Traditional optimization methods struggle to address the resource allocation and application placement problem effectively, particularly at the scale required for IoT deployments. Thus, the need for advanced optimization techniques, particularly meta-heuristic algorithms, becomes apparent.

3.4 Swarm intelligence and meta-heuristic algorithms

One class of optimization techniques that has demonstrated remarkable efficacy across diverse domains is swarm intelligence and, more specifically, meta-heuristic algorithms. These bio-inspired approaches draw inspiration from the collective behaviors of social organisms and natural processes to guide the search for near-optimal solutions [6].

Meta-heuristic algorithms offer the advantage of adaptability and robustness, making them well-suited for complex optimization challenges with no straightforward analytical solutions. These algorithms have successfully tackled optimization problems in various fields, from engineering to logistics and finance.

3.5 The role of BWO

One such meta-heuristic algorithm that has gained attention in optimization research is BWO algorithm, introduced by Hayyolalam and Kazem [9]. BWO takes inspiration from the intricate reproductive rituals of black widow spiders and has shown promise in addressing continuous engineering optimization problems [9-11].

However, an inherent limitation of the original BWO algorithm is its design for continuous optimization problems, which poses a challenge when dealing with discrete decision variables. This limitation necessitates the development of a specialized optimization approach tailored explicitly for discrete problems, such as the application placement challenge within fog-cloud infrastructure.

3.6 Research gap and motivation for DBWO

This research identifies a critical gap in the existing landscape of optimization techniques for fog computing application placement. While meta-heuristic algorithms, including BWO, have demonstrated their prowess in solving optimization problems, none have been tailored explicitly for discrete optimization in the context of IoT application placement.

Motivated by the unique demands of fog computing and the need to optimize application placement within a discrete parameter space, we introduce DBWO algorithm. DBWO is designed to excel at handling discrete decision variables, making it particularly well-suited to address the complex challenges posed by the discrete nature of IoT application placement in fog-cloud infrastructure.

4. SYSTEM MODEL

In this section, we discuss the structure of our fog computing system and define application placement as a combinatorial optimization approach.

4.1 System architecture

Fog computing is a system that is characterized by high levels of distribution and flexibility that allows data to be processed closer to the user's location, resulting in lower latency, greater efficiency, improved scalability, and better resource utilization. This system comprises cloud servers, fog servers, a fog broker, and IoT devices.

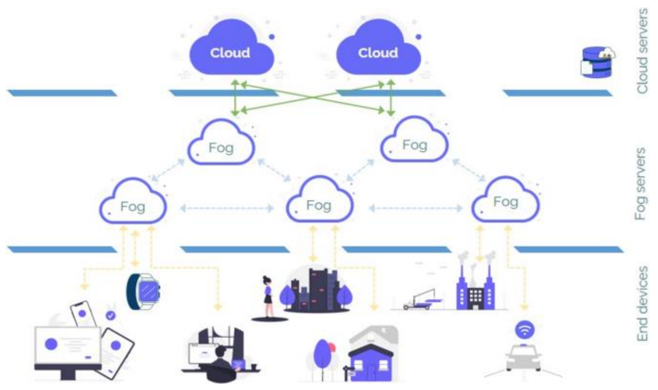


Figure 1. An overview of our system model

The fog broker receives applications from IoT devices and deploys them on either fog nodes or cloud nodes based on their requirements and the availability of resources. After execution, the results are sent back to the fog broker and then to the IoT devices. Fog computing allows for the dynamic distribution of applications across various resources while ensuring optimal performance. Figure 1 illustrates the typical system architecture of a fog computing infrastructure, which consists of IoT devices, fog nodes, and cloud nodes.

4.2 Application model

As shown in Figure 2, IoT applications are built on a concept of sense-process-act in which raw data is collected by IoT devices (sensors), processed by services running in fog and cloud nodes, and the processed data is sent to the actuators. The service placement in fog nodes depends on the resources requested by the services and the availability of these resources in the fog nodes.

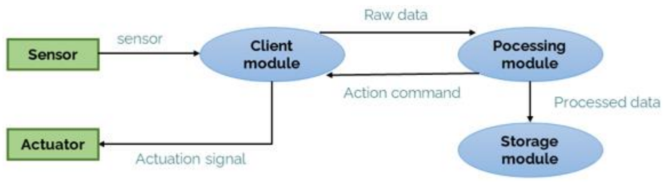


Figure 2. Application model [33]

We consider that IoT devices generate data that is sent to a variety of applications. Each one is encapsulated in a single processing module that may be separately implemented across the system nodes. In this context, the application can be modelled as a monolithic service [33]. The requested processing, memory, and bandwidth resources characterize each application. The amount of resources requested will vary depending on the type of application.

4.3 Application placement problem formulation

The placement of application entails assigning a set of applications to the best available nodes in a distributed system, while taking into account the different characteristics of each application and node. We assume a set of m fog-cloud nodes $FN = \{FN_1, FN_2, FN_3, \dots, FN_m\}$, each node i encompasses computing resources Ri {CPUi (MIPS), RAMi (Mb), bandwidthi (Mbps)}.

We also assume that exist n applications $A = \{A_1, A_2, A_3, \dots, A_n\}$ sent to the fog broker to be placed. Each application A_j is characterized by requested resources Req_j {CPUj (MIPS), RAMj (Mb), bandwidthj (Mbps)}.

The application placement in the system is mathematically modeled as follows:

Let R = [CPUi, RAMi, Bwi] be a matrix with m rows (fog nodes) and 3 columns (CPU, RAM, Bandwidth) representing the available resources of fog nodes .

$$R = \begin{pmatrix} CPU_1 & RAM_1 & Bw_1 \\ CPU_2 & RAM_2 & Bw_2 \\ \dots & \dots & \dots \\ CPU_m & RAM_m & Bw_m \end{pmatrix} \tag{1}$$

where, CPUi, RAMi, Bwi represent available CPU (in MIPS), RAM (in Mb) and Bandwidth (in Mbps) of the ith fog nodes respectively.

Let Req= [CPUj, RAMj, Bwj] be a matrix with n rows (applications) and 3 columns (resource requirements).

$$Req = \begin{pmatrix} CPU_1 & RAM_1 & Bw_1 \\ CPU_2 & RAM_2 & Bw_2 \\ \dots & \dots & \dots \\ CPU_n & RAM_n & Bw_n \end{pmatrix} \quad (2)$$

where, *CPUj, RAMj, Bwj* resource requirements CPU (in MIPS), RAM (in Mb) and Bandwidth (in Mbps) of the jth application respectively.

Let P be a binary matrix with n rows and m columns, where Pij = 1 if the jth application is executed in the ith node and Pij = 0 otherwise. Such as the sum of each, row equal to one.

$$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mn} \end{pmatrix} \quad (3)$$

Let Placement Cost Matrix (PCM) be three-dimensional matrix with n rows m columns and three slices (one for each resource).

$$PCM =$$

$$PCM(:;:;1) = \begin{pmatrix} PCM_{111} & PCM_{121} & \cdots & PCM_{1n1} \\ PCM_{211} & PCM_{221} & \cdots & PCM_{2n1} \\ \vdots & \vdots & \ddots & \vdots \\ PCM_{m11} & PCM_{m21} & \cdots & PCM_{mn1} \end{pmatrix}$$

$$PCM(:;:;2) = \begin{pmatrix} PCM_{112} & PCM_{122} & \cdots & PCM_{1n2} \\ PCM_{212} & PCM_{222} & \cdots & PCM_{2n2} \\ \vdots & \vdots & \ddots & \vdots \\ PCM_{m12} & PCM_{m22} & \cdots & PCM_{mn2} \end{pmatrix} \quad (4)$$

$$PCM(:;:;3) = \begin{pmatrix} PCM_{113} & PCM_{123} & \cdots & PCM_{1n3} \\ PCM_{213} & PCM_{223} & \cdots & PCM_{2n3} \\ \vdots & \vdots & \ddots & \vdots \\ PCM_{m13} & PCM_{m23} & \cdots & PCM_{mn3} \end{pmatrix}$$

where, PCMijk is the cost of placing the jth application on the ith node for the kth resource.

The objective function is to reduce executing time and cost by taking into account the available resources of the fog nodes and the application requirements.

The objective function can be defined as:

$$Minimize(F) = \alpha \times TotalTime + \beta \times Totalcost \quad (5)$$

where, α and β are weighting factors that determine the relative importance of the execution time and the total cost in the objective function where $(\alpha + \beta = 1)$.

The total time can be represented as:

$$TotalTime = \max\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{Req_{(j,1)}}{R_{(i,1)}}\right) \quad (6)$$

where, $Req_{(j,1)}$ represents the requested CPU by the jth application, $R_{(i,1)}$ represents the available CPU on the ith fog node. Total execution time means maximum processing time of all nodes, which is calculated as the sum of the execution time of all applications in a single fog node divided by the available CPU in the node.

The total cost of placing the application on fog nodes can be calculated as the sum of the execution cost, memory cost and bandwidth cost. Let's assume that CPT, CDP, and CPB are the cost per unit of CPU, RAM, and bandwidth respectively.

The total cost can be formulated as follows:

$$TotalCost = \sum_{i=1}^{m}\sum_{j=1}^{n}(Req_{j1} * P_{ij} * CPT + Req_{j2} * P_{ij} * CPD + Req_{j3} * P_{ij} * CPB) \quad (7)$$

where, Req_{j1}, Req_{j2}, Req_{j3} are the CPU, RAM, and Bandwidth requirements of application j respectively. Moreover, P_{ij} represents the binary placement decision with $P_{ij} = 1$ if the jth application is executed on the ith node, and $P_{ij} = 0$ otherwise. By summing up the cost for each application and each fog node, we get the overall cost of placing all the application on the fog nodes.

Resource constraint: each fog node has limited computer power, memory, and network bandwidth. Therefore, we need to ensure that the total resource requirement of all applications that run on a fog node do not surpass the resource of that node. This condition can be stated as:

$$\begin{cases} \sum_{j=1}^{n}(Req_{j1} * P_{ij}) \le CPU_i \\ \sum_{j=1}^{n}(Req_{j2} * P_{ij}) \le RAM_i \ \forall i = 1, 2, \dots, n \\ \sum_{j=1}^{n}(Req_{j3} * P_{ij}) \le Bw_i \end{cases} \quad (8)$$

Binary constraint: each application must be placed on only one fog node. this constraint can be mathematically expressed as:

$$\sum_{j=1}^{m} P_{ij} = 1, \forall i = 1, 2, \dots, n \quad (9)$$

5. THE DISCRETE BLACK WIDOW APPLICATION PLACEMENT ALGORITHM

In this section, we present our proposed DBWO algorithm for IoT application placement in fog computing infrastructure. The DBWO algorithm is an extension of the original BWO algorithm, customized to address the challenges of the IoT application placement problem.

5.1 Original BWO

The original BWO algorithm, proposed by Hayyolalam and Kazem [9], is a metaheuristic technique inspired by the predatory behavior of black widow spiders. This algorithm offers a powerful yet straightforward approach to tackle complex nonlinear optimization problems.

The BWO algorithm follows a series of steps:

-Population initialization: BWO begins with the creation of a population of candidate solutions represented as vectors of real values. These candidates, referred to as "widows," are evaluated using a fitness function specific to the optimization problem.

-Procreation: After population initialization, procreation occurs. Two parents are randomly selected from the population and mate, producing two offspring. The male spider is often consumed by the female during or after mating.

-Cannibalism: Cannibalism mechanisms are employed to eliminate weaker widows from the population. These include scenarios where females consume males, stronger spiders consume weaker siblings, and young spiders consume their mothers.

-Mutation: Following cannibalism, a mutation operation randomly exchanges two elements within a widow's vector. This promotes diversity within the population and can lead to the exploration of solutions with higher fitness.

-Natural selection and repetition: BWO algorithm iteratively repeats the procreation, cannibalism, and mutation stages across multiple generations. Through this process, only the fittest solutions persist and evolve, ultimately aiming to find an optimal solution.

5.2 DBWO for IoT applications placement

Here, we introduce a discrete version of the BWO algorithm, named DBWO, specifically designed to tackle the challenges of application deployment within the fog computing infrastructure. Figure 3 illustrates the general structure of the Algorithm.

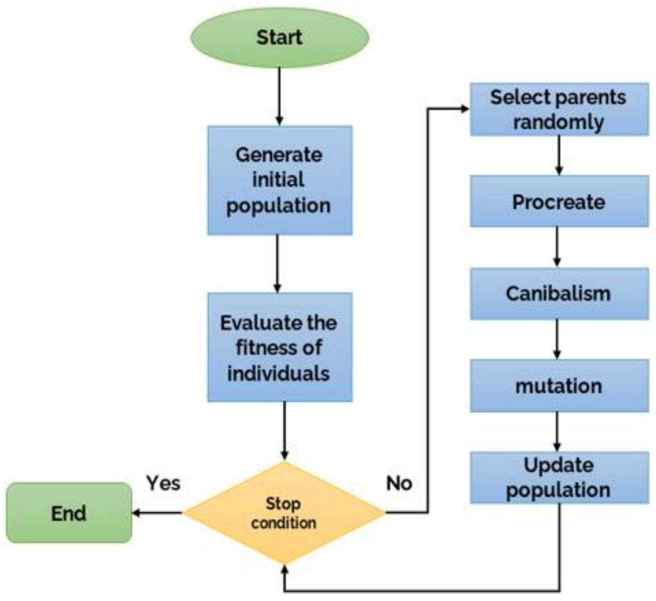
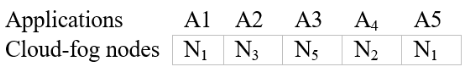


Figure 3. Flowchart of the black widow optimization algorithm [25]

5.2.1 Widow (solution) encoding

Instead of the original BWO algorithm, the proposed DBWO uses integer-based encoding to represent the application placement solution. In our representation, a widow is an array of $1 \times M$ (a set of M application) placed in fog nodes (devices). In the Figure 4, we present an example of placement of five applications in fog nodes.



Applications	A1	A2	A3	A_4	A5
Cloud-fog nodes	N_1	N_3	N_5	N_2	N_1

Figure 4. Example of application placement encoding

In DBWO algorithm, a widow w_i represents a feasible solution that represents applications placement in fog-cloud nodes. For example, in Figure 4, five application {A1, A2, A3, A4, A5} are placed on node1 (N1), node3 (N3), node5 (N5), node2 (N2) and node1 (N1) respectively.

5.2.2 Population initialization

In the proposed DBWO algorithm, each widow in the initial population is generated randomly by assigning to each variable the value of the fog-cloud node. As illustrated in Table 2, the initial population of four widows is randomly generated for the placement of seven applications.

Table 2. Example of initial population

	A1	A2	A3	A4	A5
W1	1	2	1	3	2
W2	3	3	2	1	1
W3	2	1	2	3	4
W4	4	3	2	1	1

5.2.3 Discrete procreation

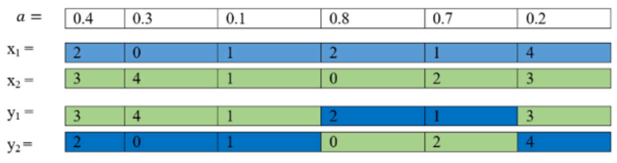
In the continuous optimization context of the original BWO [9], this operation seamlessly creates offspring solutions by blending elements from parent solutions. However, to adapt to our discrete optimization problem, we introduce a novel procreation mechanism, presented as Eq. (10):

$$\begin{cases} y_1 = \begin{cases} x1 & if\ \alpha \geq 0.5 \\ x2 & else \end{cases} \\ y_2 = \begin{cases} x2 & if\ \alpha \geq 0.5 \\ x1 & else \end{cases} \end{cases} \qquad (10)$$

In Eq. (10), the variables x1 and x2 represent the parents, α is a real random vector selected from the range [0, 1], and y1 and y2 are the resulting offspring solutions.

The modified procreation mechanism, as expressed in Eq. (10), fundamentally changes how offspring solutions are generated. It introduces a parameter α, which is a real random vector constrained within the range [0, 1]. This parameter is central to the procreation process and plays a decisive role in determining which elements from the parents (x1 and x2) are included in the offspring (y1 and y2).

In the Figure 5, an example of the discrete procreates operation is showed.



$\alpha =$	0.4	0.3	0.1	0.8	0.7	0.2
$x_1 =$	2	0	1	2	1	4
$x_2 =$	3	4	1	0	2	3
$y_1 =$	3	4	1	2	1	3
$y_2 =$	2	0	1	0	2	4

Figure 5. Example of discrete procreate operation

5.2.4 Cannibalism

Post-procreation, a selective cannibalistic process is employed to cull weaker widows from the population. Three types of cannibalism are enacted:

-Female cannibalism: The female spider terminates the male during or after mating, with fitness being the defining criterion for gender.

-Sibling cannibalism: The stronger spider consumes its weaker sibling, as determined by their respective fitness levels.

-Maternal cannibalism: Even the young spiders engage in cannibalism, devouring their mother.

Upon the removal of the population's least fit individuals, a rejuvenated population (Population 2) is formed.

5.2.5 Mutation

Subsequent to the cannibalism phase, a mutation operation is introduced, targeting a randomly chosen subset of black widow solutions. This mutation operation orchestrates a random exchange of two elements within each spider's vector, as depicted in Figure 6. This mutation is vital for infusing diversity into the spider population, thwarting premature convergence toward suboptimal solutions. The element exchange can potentially unveil novel combinations with higher fitness values than those of the parent spiders. The extent of mutation is governed by a predefined mutation rate, ultimately yielding a fresh population (Population 3) from the preceding one.

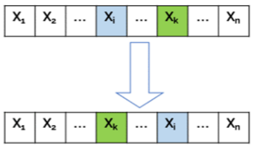


Figure 6. Example of mutation operation

The Algorithm 1 presents the outlines of the proposed DBWO algorithm.

Algorithm 1. Discrete black widow optimization Algorithm for application placement in cloud-fog environnement

```
Input: Applications , Nodes
Output: applications_placed
1.   // Initialization
2.   $N_{pop}$ =   // population size ; $M_{rate}$=
3.   // Mutation rate ;  $P_{rate}$ =    // procreation rate ;
4.   $C_{rate}$ =    // cannibalism rate ;
5.   $N_f$ =    //  dimension size ;   Iterations=  //
6.   //initial population
7.   Create a random initial population
8.   Using Eq. (5), determine the fitness of each solution.
9.   According to $P_{rate}$ determine $N_{rep}$  // offspring
10.  While nbr<iterations do
11.  // Procreation
12.  For Iter = 1 to $N_{rep}$ do
13.   Select two widows randomly
14.  from pop1 as parents.
15.  Generate D offspring using Eq. (10)
16.  Destroy father                              //cannibalism
17.  Destroy some children based on $C_r$     //cannibalism
18.  Preserve the remaining solution into population 2
19.  End for
20.  //Mutation
21.  According to $M_{rate}$ determine
22.   $N_m$ // count of mutation
23.  For Iter = 1 to $N_m$ do
24.  Choose a random individual from Pop1
25.  Employ mutation and create a new solution
26.  Save new generated solution in pop3
27.  End for
28.  // updating
29.  Population = Pop2+Pop3
30.  Calculate fitness value and evaluate solution
31.  Nbr = nbr + 1
32.  End while
33.  Return best_widow from pop
```

6. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present the experimental results and analysis of the performance of our proposed DBWO algorithm in simulated Fog environments. We compare our approach with recent literature methods in terms of application placement in cloud-fog environments, and provide details on the experimental setting, evaluation metrics, and comparative results in the following sub-sections.

6.1 Experimental settings

To evaluate the DBWO algorithm, experiments are performed using the Ifogsim [9] simulator, an open-source simulator based on Cloudsim, programmed in Java. The tests are conducted on a personal computer equipped with a 64-bit Windows 10 operating system, an Intel Core i5-4310u 2.00 GHz CPU, and 4 GB of memory, ensuring consistency and reproducibility of the experimental setup. The proposed DBWO algorithm has been tested using a set of fog-cloud nodes and their respective physical resource capabilities as detailed in Table 3. Additionally, we utilize the same application requirements as presented in Table 4 across all experiments for a fair comparison among the DBWO algorithm and the other algorithms under evaluation.

Table 3. Fog-cloud nodes characteristics

	CPU	**RAM**	**Bandwidth**
Fog node	[700-1900]	5000	10000
Cloud node	[5000-10000]	10000	10000
		Cost	
	CPU	**RAM**	**Bandwidth**
Fog node	[0,1 - 0,5]	[0,01-0,04]	[0,01- 0,03]
Cloud node	[1,0 - 3,0]	[0,05-0,10]	[0,05- 0,10]

Table 4. Applications requirements

	CPU	***RAM***	***File Size***	***Output Size***
Applications requirements	[10000-100000]	[50-200]	[20-100]	[20-100]

6.2 Evaluation metrics

The primary objective of this study is to achieve a balance between the execution time and total cost of application placement in the proposed system.

-The execution time that represents how long the system needs to execute all the applications, it is calculated using Eq. (6).

-The total cost that represents the required amount of money must be used to execute all the IoT applications in the system; it is calculated using Eq. (7).

6.3 DBWO evaluation and convergence study

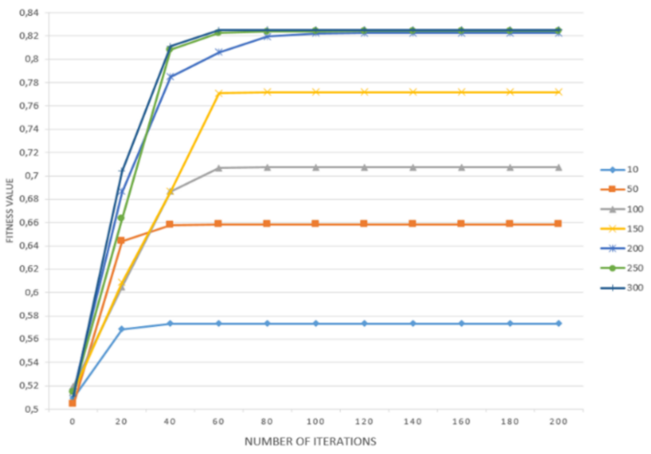


Figure 7. Convergence curve of the DBWO algorithm

The convergence of DBWO is experimentally analyzed. Figure 7 Presents the results of our algorithm in fog environment placing 100 applications in 10 fog nodes and 5 cloud the application requirement and the fog-cloud nodes characteristics are shown respectively in Table 3, Table 4. DBWO reached the best fitness value when the population size is 200 and the number of generations is 100. Hence, we use these values for the rest of the experiments.

6.4 Comparison with other approaches

Since our proposed DBWO is a Swarm-based metaheuristic, we compared it to the following four recently proposed swarm-based IoT application placement Algorithms. The comparison is made in terms of execution time and cost.

EGA [31], which is a multi-objective Elitism-based Genetic Algorithm, it aims to reduce service time, cost, and energy usage.

DPSO [27] is a discrete version of the particle swarm optimization algorithm that is proposed to the IoT services placement with the aims to minimizing the cost and maximizing the quality of experience.

BLA [24] is a placement approach based on bees life metaheuristic to solve scheduling problem in fog computing system. It aims to minimizing the CPU execution time and the allocated memory.

GWO [29] is a gray wolf optimization algorithm proposed to optimize the IoT application placement execution cost.

Table 5 displays the parameter settings used for the proposed algorithm and the other four algorithms. We have varied the parameter values to evaluate their effect on the performance of the proposed algorithm as shown in Figure 7. Our results indicate that the proposed algorithm achieves near-optimal performance when the number of individuals is 200 and the generation number is 100.

The DBWO's other parameters are set as in the original version of the BWO [9], where procreation rate is 0.6 , the mutation rate is 0.4, and the cannibalism rate is 0.44, and the parameters of the compared algorithms are fixed as in their papers. All these settings provide insight into how to tune the parameters in order to achieve the highest performance of the proposed algorithm for the problem at hand. For the rest of the experiment, we use the same parameters for all the algorithms in Table 5.

Table 5. Parameters setting for the DBWO and the compared algorithms

Algorithm	Parameter
DBWO	Procreation rate =0,6 Mutation rate = 0,4 Cannibalism rate = 0,44
EGA	Crossover probability = 0,5 Mutation rate = 0,3 Elitism rate = 0,08
DPSO	$C_1 = C_2 = 2$ P = 40

BLA	Queen = 1
	Drones = 50
	Workers = 149

GWO	wolf minimum position = 0 wolf maximum position = 100

We considered two different network topologies: fog-only and cloud-fog. In the first scenario, we used a fog-only topology, which only contained fog nodes. In the second scenario, we used a cloud-fog topology that contained both fog nodes and cloud nodes. This difference in the network topologies had a significant effect on the results of our study the fog-only topology was limited in its capabilities, while the cloud-fog topology was able to take advantage of the additional resources provided by the cloud nodes. We analyze the impact of varying the number of applications on the fitness value, the execution time, and the total cost.

6.5 First scenario: Application placement in fog only topology

The fitness value, which represents the quality of the solution in terms of both the execution time and total cost, was calculated as the mean of 10 independent runs in Figure 8. The results showed that the best performance was achieved with algorithm DBWO, followed by algorithms GWO, EGA, DPSO, and BLA respectively. It is clear that DBWO outperformed the other algorithms with a fitness value of 0.941 when compared to the other four algorithms, which is better than GWO, EGA, DPSO, and BLA by 4.30%, 5.35%, 5.77%, and 9.87% respectively.

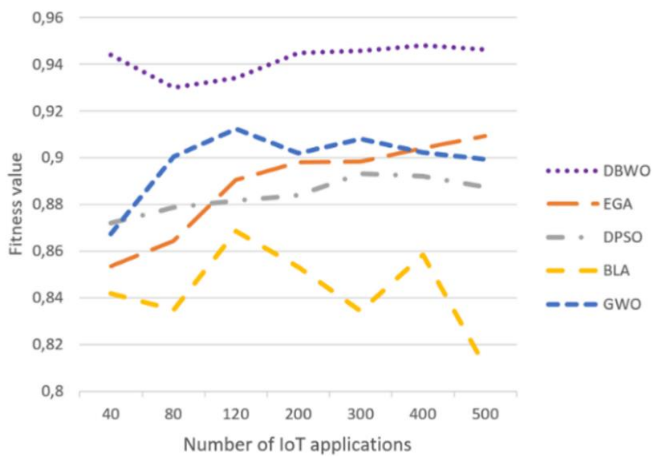


Figure 8. Fitness value of DBWO, GWO, EGA, DPSO and BLA algorithms in the first scenario

In the first scenario, we defined a fog-only topology with 10 fog nodes, available resource characteristics listed in Table 3, and varying application request numbers ranging from 40 to 500 requests. The application requirements were specified in Table 4. The balancing factors α and β of the fitness function were set to 0.5, which gave equal weight to execution time and overall cost. Each algorithm was then executed, the results obtained were compared in terms of fitness value, execution time, and total cost, as shown in Figures 9-11 respectively.

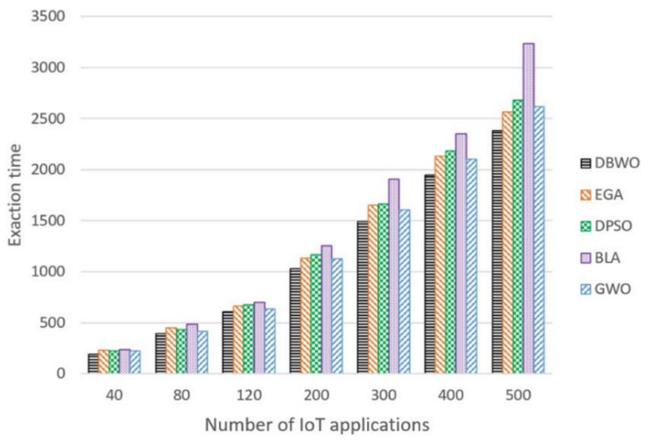


Figure 9. Applications placement exaction time in the first scenario

The execution time for each of the five algorithms was also evaluated, and the results in Figure 9 illustrate that the DBWO algorithm took the least amount of time to complete. The DBWO algorithm, for example, runs 400 applications in an average time of 1946.20, whereas the other algorithms took 2105.50, 2130.74, 2185.50, and 2352.14 for GWO, EGA, DPSO, and BLA respectively to complete the same data set. Overall, the DBWO algorithm outperformed the other algorithms in terms of execution time, with less execution time than GWO, EGA, DPSO, and BLA by 8.13%, 10.39%, 11.05%, and 19.12% respectively.

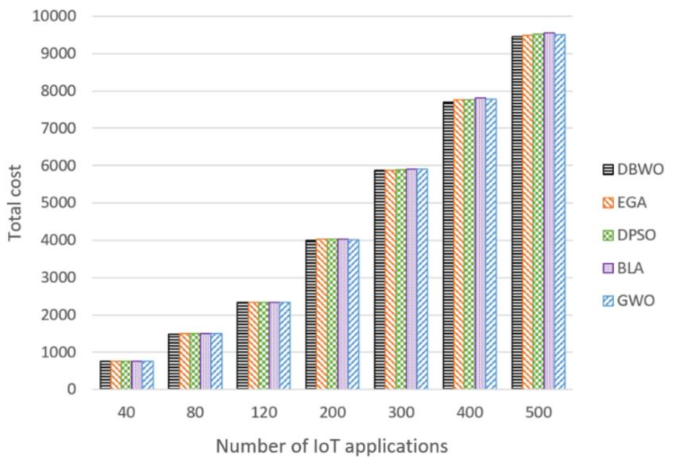


Figure 10. Applications placement total cost in the first scenario

In Figure 10, the results for the total cost of the five algorithms are shown for the fog-only scenario. The DBWO algorithm had the lowest total cost, but as the topology consisted only of fog nodes, the processing, memory, and bandwidth costs were similar for all nodes. However, even in this scenario, the DBWO algorithm outperformed the other four algorithms (GWO, EGA, DPSO, and BLA) and achieved better overall performance, reducing the total cost in all data sets by 0.45%, 0.50%, 0.62%, and 0.74% respectively. This comparison shows that the DBWO algorithm can be an effective tool for optimizing fog-based IoT networks in terms of both time efficiency and cost reduction compared to other algorithms.

6.6 Second scenario: Application placement in fog and cloud topology

In the second scenario, we defined a more complex topology of 27 nodes composed of 20 fog nodes and 7 cloud nodes, using the characteristics mentioned above. This allowed us to utilize the advantages of both fog and cloud nodes, as the fog nodes had better connectivity and responsiveness but limited resources, whereas the cloud nodes provided higher processing power and memory while also requiring a greater monetary cost. In this scenario, we varied the number of requested applications from 100 to 1000, and, according to Table 3, the application requirement was generated at random. We performed simulations to determine the performance of both physical topologies, exploring their respective impacts on application execution time, fitness value, and cost.

The fitness value of the five algorithms is depicted in Figure 11. The results showed that using both fog and cloud nodes allowed us to balance the performance between execution time and cost. Figure 11, clearly shows that the proposed DBWO outperforms all of the compared algorithms across all applications simples by 8%, 12.16%, 13%, and 15.16% for DPSO, EGA, GWO, and BLA respectively. With an average fitness value of 0.63.

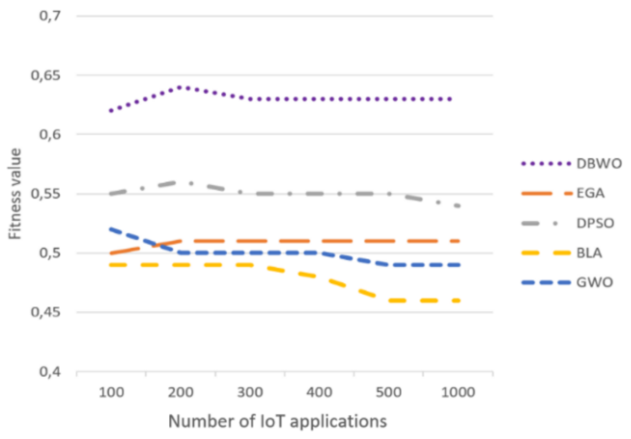


Figure 11. Fitness value of DBWO, GWO, EGA, DPSO and BLA algorithms in the second scenario

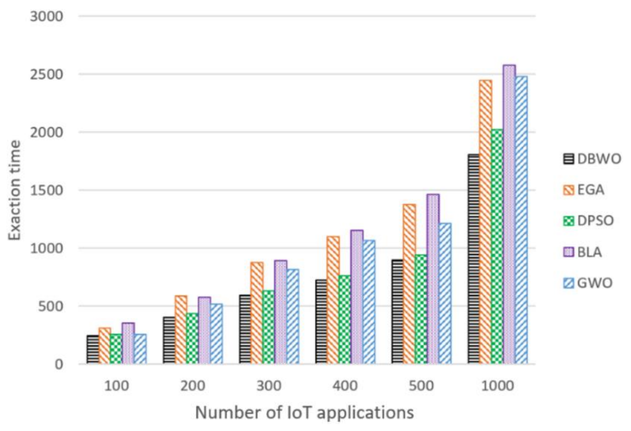


Figure 12. Applications placement exaction time in the second scenario

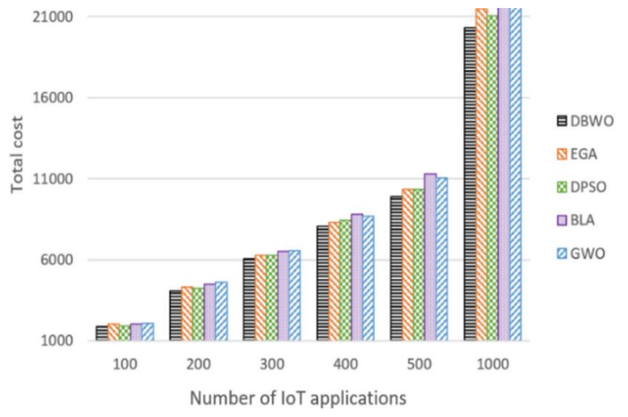


Figure 13. Applications placement total cost in the second scenario

In all of the experiment, the DBWO had the lowest execution time and total cost compared to other algorithms as illustrated in Figure 12. In terms of execution time, DBWO outperformed DPSO, EGA, GWO, and BLA by 1.95%, 9.16%, 7.25%, and 10.11% respectively. In terms of cost, as illustrated in Figure 13. It outperformed them by 2.9%, 4.19%, 7.70%, and 8.0% respectively.

6.7 Analysis of experimental results

According to the analysis of the results in the two scenarios in which DBWO was tested and compared with EGA, DPSO, GWO, and BLA. In both scenarios, DBWO outperformed the compared algorithms in terms of fitness value, in fog only topology with improvement range from 4.30% to 9.87%. Similarly, in the fog cloud topology showed notable performance improvement, with fitness value range from 8% to 15.16%. Notably, DBWO proved to be particularly effective in the more complex topology, where its performance improvement was most pronounced.

It is clear that DBWO surpassed the other algorithms in terms of efficiency in both execution time and total cost in both scenarios, where the first scenario was a simple fog only scenario and the second scenario was a more complex scenario consisting of both fog and cloud nodes. In simple environments, GWO performed better than EGA, DPSO, and BLA. When the environment was more complex, DPSO provided better results than GWO, EGA, and BLA However, DBWO provided better results than all the algorithms in both scenarios, confirming its versatility and adaptability in optimizing resource utilization and enhancing the scalability of fog environments.

7. CONCLUSION

The execution performance constraints have a major influence on the accommodation of IoT applications running in fog scenarios, which require a best management of the applications in the network. In this work, we present a novel metaheuristic called DBWO for optimizing IoT application deployment in a fog computing system that consider the applications constraints and the network characteristics. Our approach is a multi-objective discrete black widow optimization algorithm that aims to reduce the application's processing time and overall cost in a fog computing system. Experimental results on the Ifogsim simulator demonstrate that in both simple and complicated contexts, the proposed method surpasses four existing meta-heuristic approaches in terms of cost and execution time.

Our current work primarily focuses on static scenarios, and there is a growing need to address real-time dynamic environments in fog computing. Future improvements could involve developing adaptive algorithms capable of dynamically adjusting application placements based on changing network conditions and workload demands. Additionally, the scalability of the proposed method for larger and more complex systems warrants further investigation to ensure its effectiveness in practical deployment scenarios. Additionally, future research could investigate the scalability of the proposed method for larger and more complex systems. In addition, our plan involves expanding the proposed algorithm to encompass additional functionalities to other types of services and analyze the impact of mobility and resource requirements in a fog-cloud architecture.

REFERENCES

[1] Alavi, A.H., Jiao, P., Buttlar, W.G., Lajnef, N. (2018). Internet of Things-enabled smart cities: State-of-the-art and future trends. Measurement, 129: 589-606. https://doi.org/10.1016/j.measurement.2018.07.067

[2] Ray, P.P. (2018). A survey on Internet of Things architectures. Journal of King Saud University-Computer and Information Sciences, 30(3): 291-319. https://doi.org/10.1016/j.jksuci.2016.10.003

[3] Bonomi, F., Milito, R., Zhu, J., Addepalli, S. (2012). Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, pp. 13-16. https://doi.org/10.1145/2342509.2342513

[4] Azizi, S., Khosroabadi, F., Shojafar, M. (2019). A priority-based service placement policy for fog-cloud computing systems. Computational Methods for Differential Equations, 7(4S): 521-534.

[5] Lera, I., Guerrero, C., Juiz, C. (2018). Availability-aware service placement policy in fog computing based on graph partitions. IEEE Internet of Things Journal, 6(2): 3641-3651. https://doi.org/10.1109/JIOT.2018.2889511

[6] Salaht, F.A., Desprez, F., Lebre, A. (2020). An overview of service placement problem in fog and edge computing. ACM Computing Surveys (CSUR), 53(3): 65. https://doi.org/10.1145/3391196

[7] Brogi, A., Forti, S., Guerrero, C., Lera, I. (2020). How to place your apps in the fog: State of the art and open challenges. Software: Practice and Experience, 50(5): 719-740. https://doi.org/10.1002/spe.2766

[8] Nayeri, Z.M., Ghafarian, T., Javadi, B. (2021). Application placement in Fog computing with AI approach: Taxonomy and a state of the art survey. Journal of Network and Computer Applications, 185: 103078. https://doi.org/10.1016/j.jnca.2021.103078

[9] Hayyolalam, V., Kazem, A.A.P. (2020). Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 87: 103249. https://doi.org/10.1016/j.engappai.2019.103249

[10] Al-Rahlawee, A.T.H., Rahebi, J. (2021). Multilevel thresholding of images with improved Otsu thresholding by black widow optimization algorithm. Multimedia Tools and Applications, 80(18): 28217-28243. https://doi.org/10.1007/s11042-021-10860-w

[11] Houssein, E.H., Helmy, B.E.D., Oliva, D., Elngar, A.A., Shaban, H. (2021). A novel black widow optimization algorithm for multilevel thresholding image segmentation. Expert Systems with Applications, 167: 114159. https://doi.org/10.1016/j.eswa.2020.114159

[12] Salaht, F.A., Desprez, F., Lebre, A., Prud'Homme, C., Abderrahim, M. (2019). Service placement in fog computing using constraint programming. In 2019 IEEE International Conference on Services Computing (SCC), Milan, Italy, pp. 19-27. https://doi.org/10.1109/SCC.2019.00017

[13] Skarlat, O., Nardelli, M., Schulte, S., Dustdar, S. (2017). Towards QoS-aware fog service placement. In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, pp. 89-96. https://doi.org/10.1109/ICFEC.2017.12

[14] Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience, 47(9): 1275-1296.

https://doi.org/10.1002/spe.2509

[15] Minh, Q.T., Nguyen, D.T., Van Le, A., Nguyen, H.D., Truong, A. (2017). Toward service placement on fog computing landscape. In 2017 4th NAFOSTED Conference on Information and Computer Science, Hanoi, Vietnam, pp. 291-296. https://doi.org/10.1109/NAFOSTED.2017.8108080

[16] Arkian, H.R., Diyanat, A., Pourkhalili, A. (2017). MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. Journal of Network and Computer Applications, 82: 152-165. https://doi.org/10.1016/j.jnca.2017.01.012

[17] Tran, M.Q., Nguyen, D.T., Le, V.A., Nguyen, D.H., Pham, T.V. (2019). Task placement on fog computing made efficient for IoT application provision. Wireless Communications and Mobile Computing, 2019(1): 6215454. https://doi.org/10.1155/2019/6215454

[18] Brogi, A., Forti, S. (2017). QoS-aware deployment of IoT applications through the fog. IEEE Internet of Things Journal, 4(5): 1185-1192. https://doi.org/10.1109/JIOT.2017.2701408

[19] Brogi, A., Forti, S., Ibrahim, A. (2017). How to best deploy your fog applications, probably. In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, pp. 105-114. https://doi.org/10.1109/ICFEC.2017.8

[20] Xia, Y., Etchevers, X., Letondeur, L., Lebre, A., Coupaye, T., Desprez, F. (2018). Combining heuristics to optimize and scale the placement of IoT applications in the fog. In 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), Zurich, Switzerland, pp. 153-163. https://doi.org/10.1109/UCC.2018.00024

[21] Benamer, A.R., Teyeb, H., Ben Hadj-Alouane, N. (2018). Latency-aware placement heuristic in fog computing environment. In On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, pp. 241-257. https://doi.org/10.1007/978-3-030-02671-4_14

[22] Agrawal, P., Abutarboush, H.F., Ganesh, T., Mohamed, A.W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). IEEE Access, 9: 26766-26791. https://doi.org/10.1109/ACCESS.2021.3056407

[23] Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y. (2019). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Computing and Applications, 31(11): 7665-7683. https://doi.org/10.1007/s00521-018-3592-0

[24] Bitam, S., Zeadally, S., Mellouk, A. (2018). Fog computing job scheduling optimization based on bees swarm. Enterprise Information Systems, 12(4): 373-397. https://doi.org/10.1080/17517575.2017.1304579

[25] Ayoubi, M., Ramezanpour, M., Khorsand, R. (2021). An autonomous IoT service placement methodology in fog computing. Software: Practice and Experience, 51(5): 1097-1120. https://doi.org/10.1002/spe.2939

[26] Canali, C., Lancellotti, R. (2019). GASP: genetic algorithms for service placement in fog computing systems. Algorithms, 12(10): 201. https://doi.org/10.3390/a12100201

[27] Djemai, T., Stolf, P., Monteil, T., Pierson, J.M. (2019). A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures. In 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), Amsterdam, Netherlands, pp. 32-40. https://doi.org/10.1109/ISPDC.2019.00020

[28] Guerrero, C., Lera, I., Juiz, C. (2019). Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. Future Generation Computer Systems, 97: 131-144. https://doi.org/10.1016/j.future.2019.02.056

[29] Salimian, M., Ghobaei-Arani, M., Shahidinejad, A. (2021). Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment. Software: Practice and Experience, 51(8): 1745-1772. https://doi.org/10.1002/spe.2986

[30] Yadav, V., Natesha, B.V., Guddeti, R.M.R. (2019). GA-PSO: Service allocation in fog computing environment using hybrid bio-inspired algorithm. In TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, pp. 1280-1285. https://doi.org/10.1109/TENCON.2019.8929234

[31] Natesha, B.V., Guddeti, R.M.R. (2021). Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment. Journal of Network and Computer Applications, 178: 102972. https://doi.org/10.1016/j.jnca.2020.102972

[32] Maarouk, C., Haouassi, H., Malik, M.M., Saidi, K. (2024). Hybrid grey wolf optimizer and elitism genetic algorithm for multi-objective IoT service placement in fog computing environment. AIJR Abstracts, pp. 45-46. https://doi.org/10.21467/abstracts.163

[33] Mahmud, R., Ramamohanarao, K., Buyya, R. (2020). Application management in fog computing environments: A taxonomy, review and future directions. ACM Computing Surveys (CSUR), 53(4): 88. https://doi.org/10.1145/3403955