2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

Efficient Task Scheduling Approach in Edge-Cloud Continuum based on Flower Pollination and Improved Shuffled Frog Leaping Algorithm

Nasiru Muhammad Dankolo*[1,2], Nor Haizan Mohamed Radzi[1], Noorfa Haszlinna Mustaffa[1], Mohd Shukor Talib[1], Zuriahati Mohd Yunos[1], Danlami Gabi[2]

[1]Department of Computer Science, Faculty of computing, Universiti Teknologi Malaysia, Johor, Malaysia.
[2]Department of Computer Science, Faculty of Physical Sciences, Kebbi State University, Aliero, Nigeria.
*Corresponding Author.
ICAC2023: The 4th International Conference on Applied Computing 2023.

Received 31/10/2023, Revised 10/02/2024, Accepted 12/02/2024, Published 25/02/2024

© 2022 The Author(s). Published by College of Science for Women, University of Baghdad.
This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

The rise of edge-cloud continuum computing is a result of the growing significance of edge computing, which has become a complementary or substitute option for traditional cloud services. The convergence of networking and computers presents a notable challenge due to their distinct historical development. Task scheduling is a major challenge in the context of edge-cloud continuum computing. The selection of the execution location of tasks, is crucial in meeting the quality-of-service (QoS) requirements of applications. An efficient scheduling strategy for distributing workloads among virtual machines in the edge-cloud continuum data center is mandatory to ensure the fulfilment of QoS requirements for both customer and service provider. Existing research used metaheuristic algorithm to solve tak scheduling problem, however, must of the existing metaheuristics used suffers from falling into local mina due to their inefficiency to avoid unfeasible region in the solution search space. Therefore, there is a dire need for an efficient metaheuristic algorithm for task scheduling. This study proposed an FPA-ISFLA task scheduling model using hybrid flower pollination and improved shuffled frog leaping algorithms. The simulation results indicate that the FPA-ISFLA algorithm is superior to the PSO algorithm in terms of makespan time, resource utilization, and execution cost reduction, especially with an increasing number of tasks.

Keywords: Cloud, Continuum, Edge, Metaheuristics Optimization , Task Scheduling,.

Introduction

In edge-cloud continuum, the goal of task scheduling is to create a cohesive computing environment that incorporates a significant number of readily available computing resources[1]. This is necessary in order to successfully fulfil the requirements of end-users regarding the quality of the service. Mobile devices are capitalizing on the benefits that continuum computing has to offer in order to access the diverse range of experiences that the environment offered [2]. The task scheduling in edge-cloud continuum is a strategic process that makes it easier to provide consistent service delivery to the customers of a network[3]. Regardless of whatever strategies adopted in the task scheduling, it must achieve the desired QoS for either customer service provider via facilitating the

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

appropriate distribution of resources to tasks. The problem of task scheduling in such environment is referred to as NP-hard[4]. This is as a result of the fact that the amount of computational time required to solve a specific instance of tasks or applications increases in polynomial time[5].

Dynamic scheduling is an approach used to make resource allocation to task dynamically in heterogeneous environment like edge-cloud continuum[6]. This approach is effective as it can adapt the frequent changes that occur in the availability of various resources[7]. It will allocate resources based on the current knowledge of the scheduler, which allows for more expedient decision-making regarding resource allocation. The dynamic scheduling method has the potential to assign tasks to a variety of computing resources without having prior knowledge of when those resources will become available[8]. The ability of dynamic task scheduling to reassign an active job to another resource is what allows it to demonstrate scalability[9]. However, some task might not be finished within the allotted amount of time[10] due to either delays brought on by non-essential operations or an excessive number of essential operations[11] using the existing dynamic scheduling algorithms. Several algorithms were developed in the literature to handle dynamic task scheduling in heterogenous environment. For example, Gabi et. al, (2020) proposed Cloud customers service selection scheme based on improved conventional cat swarm optimization[12], Abdullahi et. al, (2019) proposed An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment[13], Jayasena and Thisarasinghe (2019) proposed an Optimized task scheduling on fog computing environment using meta heuristic algorithms[5]. Generally the existing dynamic task scheduling algorithms are based on metaheuristics algorithms such as Firefly Algorithm (FA)[6], Cat Swam algorithm (CA) [12], symbiosis organism search (SOS)[13], however, the existing task scheduling techniques experience an undefeatable trap into local mina due to the in efficiency of the algorithms to avoid unfeasible region in the solution search space thereby making them inefficient for task scheduling in the edge-cloud continuum.

Furthermore, the optimization of QoS for task scheduling in the edge-cloud continuum necessitates the careful consideration and balancing of various factors, such as makespan, cost, and resource utilization through efficient constraints handling[14]. The metaheuristics algorithms are designed to handle unconstrained optimization problems while task scheduling is a constrained optimization where the service user will imposed some constrained that need not to be violated such as deadline and budget [13]. For example, when a deadline constrained task is submitted for execution, it's expected that the cloud server executes the task in a time not more than the deadline otherwise the deadline is violated and can lead to unfavorable outcome. Hence, there is a need to develop an efficient task scheduling algorithm in edge-cloud continuum that can efficiently handle dynamic resource allocation for efficient task execution.

To solve the problem of efficient task scheduling, in this study, Flower Pollination Algorithm and improved Shuffled Frog Leaping algorithm for task scheduling in edge-cloud continuum is proposed. The contributions of this research are summarized as follows:

- Modelling and formulation of task scheduling problem in edge-cloud continuum.
- Development of an improved version of shuffled frog leaping algorithm used to improve the FPA algorithm local search strategy for efficient task scheduling.
- Evaluation of the proposed FPA-ISFLA algorithm using makespan, cost and resource evaluation metrics.

Review of Related Work

This section provides a review of works conduct by previous researcher in task scheduling in relation to quality of service (QoS), with a specific focus on the makespan time, execution cost and resource utilization using task scheduling algorithms in the cloud environment.

In the work conducted by[15], a task scheduling scheme called ORFO-TSS, which is

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

based on CloudSim and incorporates IoT capabilities, utilizing the oppositional red fox optimization algorithm is developed. The allocation of cloud-based Internet of Things (IoT) resources posed a significant challenge, which was effectively addressed through the implementation of the ORFO-TSS paradigm. The implementation of optimal Task Scheduling (TS) techniques, considering various features of incoming tasks, enables the achievement of the makespan. The incorporation of oppositional based learning (OBL) into the design of ORFO-TSS serves as an alternative to the conventional RFO technique. Nevertheless, this particular methodology is exclusively applicable in a cloud environment and solely focuses on mitigating the makespan issue. In a recent publication by Attiya et al. (2022), a novel dynamic Jellyfish Search algorithm, DJSD, is developed [16]. In this work simulated annealing operators are integrated into the conventional Jellyfish Search Algorithm during the exploration phase in a competitive manner to improve diversity of the search. to enhance the diversity of candidate solutions, thereby mitigating the risk of converging towards a local optimum. The findings indicated that the DJSD approach exhibited potential, revealed unexplored search domains, and identified previously undiscovered optimal solutions. However, the algorithm's effectiveness in a hybrid edge-cloud environment remains unverified, as its application has been limited to cloud-based scenarios.

In another effort, a hybrid approach for task scheduling in IoT applications by integrating the chimp optimization algorithm (ChOA) with the marine predators algorithm (MPA) and a disruption operator was proposed[17]. The proposed CHMPAD approach is demonstrated through experiments using both synthetic and real workloads sourced from the Parallel Workload Archive. The simulation results indicate that CHMPAD exhibits superior performance compared to other scheduling techniques for HPC2N workloads, with an average improvement ranging from 2.75% to 42.53%. Similarly, for NASA iPSC workloads, CHMPAD demonstrates an average improvement ranging from 1.00% to 43.43%. Additionally, for synthetic workloads, CHMPAD shows an average improvement ranging from 1.12% to 43.20%. Furthermore, there is a lack of empirical evidence demonstrating the feasibility of implementing cloud technology, and it should be noted that the algorithm's processing speed is limited to a maximum of mekspan time.

Another research presented a bi-level multi-follower algorithm that utilizes hybrid metaheuristics to address the multi-objective budget-constrained dynamic Bag-of-Tasks scheduling problem in a heterogeneous multi-cloud environment[18]. The objective function of the model varies based on the level of detail in the scheduling problem being addressed. Each sublevel aims to minimize the execution time and cost of its respective task, taking into consideration the overall budget of the Bag-of-Tasks. On the other hand, the top level endeavors to minimize the makespan of the entire Bag. At a more foundational level, introduced an Efficient NSGA-II (E-NSGA-II), which is an improved variant of the Non-dominated Sorting Genetic Algorithm II (NSGA-II). The algorithm demonstrates exceptional performance in terms of both makespan and execution cost, while adhering to budget constraints, as evidenced by the results obtained from conducting tests on synthetic datasets provided. Nevertheless, it is worth noting that the aforementioned technique lacks memory management capabilities, which poses a significant challenge considering the extensive memory requirements associated with genetic algorithms' computational processes. In addition, the primary objectives being considered, namely makespan time and execution cost, pertain specifically to cloud computing.

The authors of this study[19] proposed a paradigm that aims to enable the deployment of multi-component Internet of Things applications on Fog infrastructures while considering quality of service requirements. The model provides a description of the operational system attributes, such as latency and bandwidth, of the available infrastructure. Additionally, it outlines the interactions between software components, Things, and business policies. Our organization offers algorithms that facilitate the decision-making process regarding the successful deployment of applications in a Fog computing environment. A prototype of FogTorch, a Java tool developed based on the suggested paradigm, has already been created.

The management of the complete life cycle of continuum applications is addressed through the proposition of an alternative architecture grounded

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

in the A3-E paradigm[20]. The methodology utilizes the Functions-as-a-Service paradigm for the deployment of computational resources in the form of microservices across the entire continuum. Furthermore, A3-E determines the appropriate location for executing a specific operation by conducting an analysis of the prevailing circumstances and taking into account the user's requirements. This article also presents a prototype framework for implementing the ideas proposed by A3-E. The findings indicate that the utilization of A3-E enables the dynamic deployment of microservices and the effective routing of application requests. This leads to a significant decrease in latency, up to 90%, when utilizing edge resources instead of cloud resources. Additionally, there is a notable reduction of 74% in battery consumption when offloading computation.

The FSAOS task scheduling algorithm was introduced by Gabi et al. (2022)[6] for application in the mobile edge-cloud continuum, drawing inspiration from the Fruit fly algorithm (FA) and the simulated annealing method (SA). The SA algorithm is incorporated by the authors into the local search of FA in order to mitigate premature convergence and avoid getting trapped in local optima solutions. This integration enables a more effective utilization of resources and reduces task completion times. The simulation results of EdgeCloudSim exhibit a notable enhancement of 95% in resource utilization and a reduction in mekspan when compared to benchmark methods. However, in the standard fruit fly, the distribution of distance values within a global search zone results in an extremely low scent value, denoted as Si. This phenomenon results in the premature convergence of the fitness value, thereby confining it within a local optimum.

Additionally, according to research[7], there is a proposed architecture that enables the flexible coordination of applications throughout the Cloud-Edge continuum. To effectively handle environments with a substantial number of nodes and complex scheduling algorithms, the architecture utilizes a distributed scheduling technique that can be customized for each individual application. In order to ascertain the feasibility of implementing more sophisticated scheduling algorithms that consider application quality of service, the architecture has been deployed on the Kubernetes platform and evaluated.

In order to improve the quality of service (QoS) provided to users in Industrial Internet of Things (IIoT) applications, a study[21] introduces an energy-conscious metaheuristic algorithm known as the Harris Hawks Optimization Algorithm with Local Search Strategy (HHOLS). This algorithm is specifically designed for task scheduling in Fog computing (TSFC). The optimization of HHOLS can be enhanced by incorporating a local search technique. The proposed methodology demonstrates superior performance compared to alternative algorithms in terms of both energy efficiency and mekspan. Nevertheless, the absence of empirical evidence regarding cloud performance is a notable limitation of the study, which exclusively concentrates on edge computing.

Materials and Methods

System Model

This research makes use of a theoretical framework that incorporates a remote cloud environment (RC) with a centralized cloud resource provider to manage distributed cloud resources, mobile edge cloud (MEC) with edge resource provider that works with a centralized cloud resource provider to allocate resources to mobile device (MD) users on the edge. Fig. 1 depicts the overall system architecture.

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

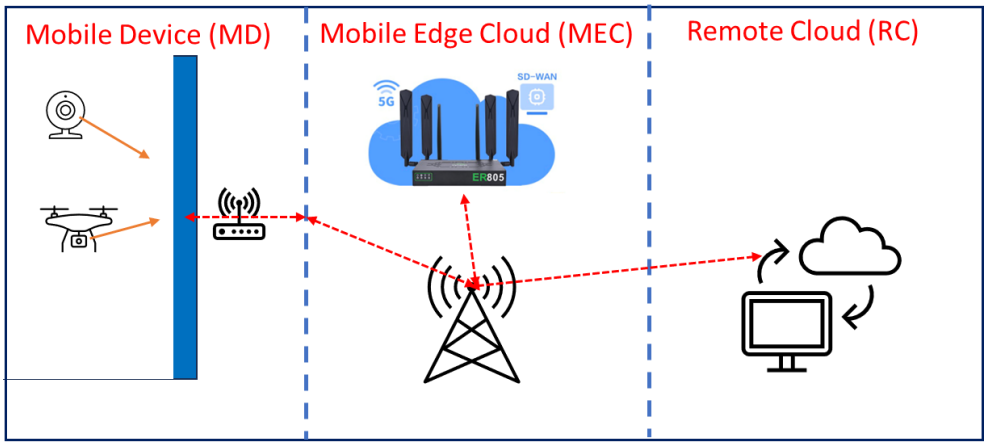

Figure 1. Edge-cloud continuum system model.

An edge resource provider, a distributed cloud resource provider, and a group of mobile device users are the three main players in this scenario. The edge orchestrator uses the FPA-ISFLA algorithm to carry out resource optimization and job allocation. The FPA-ISFLA algorithm will move task to the centralized cloud, which has abundant resources, if an MD needs a large number of computation-intensive resources at the edge. The presence of edge service providers allows for many resource use scenarios to coexist. The efficient distribution of resources is a result of the FPA-ISFLA mechanism, which guarantees that applications are executed efficiently without violating deadlines. It does this by making intelligent allocation of available computational resources to client tasks.

Each flower pollen gamete is initially set up by the swarm as a collection of tasks that have been submitted to the edge resources by MD. Flowers' relative network positions are indicated by the pollen gametes, which can be thought of as a collection of tasks originating from the network's edge. The edge network's resources are essential because they feed the pollinators and make it easier for them to find nectar. Once the service location is established, FPA-ISFLA approach begins the scheduling process. This process is designed to find the optimum trade-offs between virtual resources and other factors like makespan and cost in order to improve service quality. This algorithm guarantees that the selected resource can handle the necessary computations. Each task that uses the allocated computing resources undergoes a series of iterations of the aforementioned method until the desired Quality of Service (QoS) is reached.

Problem Description

This section formulates the edge-cloud continuum task scheduling problem. In edge-cloud computing, task scheduling assigns a set of tasks to edge resources, cloud resources, or both to satisfy customers' QoS needs while minimizing task execution and transmission time. Each application has *k tasks* $T = \{T_1, T_2, \ldots, T_k\}$ with different QoS. (eg, cost, deadline, availability, throughput, etc). Each task Ti may be allocated to an edge VM, remote cloud VM, or both. The jobs are performed by n edge-cloud $VM = \{VM_1, VM_2, \ldots, VM_n\}$ with varied processing capabilities (eg, processing power, memory usage, network, etc). Task scheduling optimizes tasks and resources using a fitness function. This study determines the fitness function according to minimized task execution (Mekspan) time, Task Execution cost and optimized resource usage. Eq.1 represents Makespan time (MT).

$$MT = \max\left(exT_{ij}\right), \quad 1 \le j \le m \qquad 1$$

where MT is the maximum execution time of all edge and cloud server virtual machines; $execTime_{ij}$ is the execution time of task I on virtual machine V_j; hence the execution time (exT_{ij}) of individual virtual machines V_j is computed in Eq.2.

$$exT_{ij} = x_{ij}.\frac{task\ size_i}{v_{npej}}, \quad 1 \le i \le n; 1 \le j \le m \qquad 2$$

X_{ij} is 1 if V_j is assigned to execute task t_i and 0 otherwise; n is the number of tasks t_i and m is the

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

number of virtual machines V_j; and V_{npej} is the computing capacity of V_j.

In this study, the cost associated with executing a task t_i on a virtual machine $v_k$is measured in US dollars ($). This cost is calculated on an hourly basis (/hr). Eq.3 calculates the overall cost of executing task i on all virtual machines V_j.

$$T_{cost} = \sum_{i}^{n} C_{exeij}, \quad 1 \le i \le n \qquad 3$$

Where T_{cost} is the total execution cost of task processing across the continuum and C_{exeij} is the total cost of using a virtual machine V_j by an assigned task t_i and the price of a resource is represented by $C_{j.}$ Hence the total cost (C_{exeij}) of using individual virtual machine V_j is given in Eq.4

$$C_{exeij} = \sum_{i}^{n} X_{ij} . \frac{task\ size_i}{np_e \times v_{jspeed}} . C_{j,} \quad 1 \le i \le n \qquad 4$$

Where C_{exeij} denotes the cost associated with the execution of a resource when a task ti is assigned to virtual machine vj. Meanwhile, Cj represents the unit cost of a resource per second, as specified by the provider of said resource. v_{jspeed} is the performance speed of the virtual machine v_j.

Hence, the objective function is to find an optimal trades-offs between the makespan time and execution cost given as follows:

$$f = min(Makespan, Cost) \qquad 5$$

Eq. 6 is employed for the calculation of resource utilisation in both the edge and cloud environments.

$$Ru = (\sum_{j=1}^{m} \frac{Te_{ij}}{exeTime_{ij}}) / m \qquad 6$$

The variable Te_{ij} represents the total execution time of all virtual machines. The variable $exeTime_{ij}$ represents the execution time of task t_i on virtual machine v_j. The variable m represents the total number of VMs.

Flower Pollination Algorithm

The Flower Pollination Algorithm (FPA) was recently anticipated by yang in (2013)[22], to handle optimization problem. The algorithm was idealized based on the inspiration of the natural biological pollination process of flowering plants. Pollination is characterized by self or cross pollination. The Cross pollination is done when some vectors fuses pollen gamete from one plant of the same type to another or different flowers of the same plant, while self-pollination is done without intervention of any pollen vector, usually the process is aided by some natural process such as wind and diffusion. This is due to the absence of reliable pollen vector that will facilitate the process. Because in cross pollination process the pollen vectors carries the pollen gamete over a long distance with jumps or fly, the distance covered by the vectors agrees to obey the Levey distributions[23], therefore, flower firmness can be used as an increment step using the similarity or uniqueness of two flowers.

The above characteristics of flower pollination process, flower constancy and pollinator behavior was idealized and transform into Flower Pollination Algorithm with the following four updating rules [22,20]:

1. The distance covered by pollinators to perform global pollination is defined as levy distance L and is computed as:

$$L \sim \frac{\lambda\Gamma(\lambda)\sin(\frac{\pi\lambda}{2})}{\pi} \frac{1}{S^{1+\lambda}}, (s \gg s_0 > 0) \qquad 7$$

2. The global pollination is summarized as:

$$x_i^{t+1} = x_i^t + \alpha L(\lambda)(g_* - x_i^t) \qquad 8$$

Where x_i^{t+1} is the i[th] gamete during the t[th] iteration and L and g* are the levy distance and global best during the current iteration respectively.

3. A switch probability function p is required to switch between the global and the local pollination phase.

4. The local pollination is represented as follows:

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \qquad 9$$

Where x_i^{t+1} is the i[th] gamete during the t[th] iteration, e is the probability function for switching from global to local pollination drawn from normal probability distribution p[0,1] and x_j and x_k are any random flowers.

These updating rules are summarized to form the flower pollination algorithm as shown in algorithm1.

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

Algorithm 1: Flower Pollination Algorithm (FPA)

Initialise: Pop Size n, Switch probity p, max iteration maxIter, t.

1. Initialize a population of n pollen gametes at random.
2. Identify the global best solution as g*
3. ***while*** $(\boldsymbol{t < maxIter})$
4. ***for*** $(\boldsymbol{i = 1{:}n})$
5. ***if rand*** $< \boldsymbol{p}$:
6. Global Pollination via:
   $x_i^{t+1} = x_i^t + l(x_i^t - g^*)$
7. ***else*** *draw Q from a uniform distribution in* [0,1]
8. *randomly choose j and k among all the solution*
9. Local pollination via:
10. $x_i^{t+1} = x_i^t + Q\left(x_k^t - x_j^t\right)$
11. ***end if***
12. Evaluate the new solution.
13. If new solution is better, update global best g*
14. ***end for***
15. Record the current global best **g***
16. $t = t + 1$
17. ***end while***

Algorithm 1: Flower Pollination Algorithm.

Although the FPA is often considered an efficient optimization technique due to its limited control parameters and computational load, it is worth noting that the FPA's local search strategy, as demonstrated in Eq.8, does not promote the exchange of information among the superior solutions during the local search process. Consequently, this hinders the convergence speed of the FPA. Hence, this research introduce an information exchange strategy within the local search procedure of FPA in order to enhance the variety of the exploitation capabilities of the FPA using the concept of shuffled Frog Leaping algorithm (SFLA) which incorporates the notion of information exchange among the frogs within a memeplex.

Shuffled Frog Leaping Algorithm (SFLA)

The shuffled frog leaping algorithm (SFLA) is a biological evolutionary algorithm that was introduced by Eusuff and Lansey in 2003[24]. It is founded on the concept of collective intelligence. The algorithm integrates the benefits of a memetic algorithm that utilizes memetic evolution and a particle swarm optimization technique that leverages group behavior. The algorithm exhibits several key attributes, including a reduced number of parameters, a straightforward conceptual framework, efficient computational performance, robust global optimization capabilities, and straightforward implementation [25]. This approach is well-suited for addressing a wide range of combination optimization challenges. The SFLA algorithm consists of four key components namely, population initialization, sub-population partitioning, local search strategy, and sub-population mixing [26].

The first step which is the initial population is formed by randomly generating a set of K solutions within the solution space. The solutions under consideration exhibit a range of characteristics and properties. The dimension of the frog is denoted by S, which is defined as the i^{th} solution of the n-dimensional space $x^i = (x_1^i, x_2^i, \dots, x_s^i)$ where $i = (1,2, \dots, K)$.

In the second stage, within the population, the objective function values of each solution should be computed as $f(x^i)\ where\ i = (1,2, \dots, K)$, and thereafter organized in a descending order. The most favorable solution throughout the population is updated as X_{best}. The frog population as a whole is categorized into distinct memetic groups, with each group consisting of a specific number of frogs. Within this set, the initial frog is allocated to the initial meme group, while the n^{th} frog is assigned

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

to the n^{th} meme group. Additionally, the $(n+1)^{th}$ frog is subsequently reassigned to the initial meme group, and this pattern continues iteratively. Until all of the frogs are allocated. The frogs exhibiting the highest and lowest levels of fitness within each memetic group are referred to as X_b and X_w, respectively.

The local search strategy, which is the third stage of the algorithm used in optimization the algorithm to find the best solution inside a given neighborhood of a current solution. It involves iteratively exploring neighboring solutions and selecting the one that improves the objective function. The local search component is of paramount importance within the overall algorithm. The conventional SFLA algorithm is designed to improve the solution of the least optimal internal frog individual during the entire iterative procedure[25] as presented in Eq. 10 and Eq. 11.

$$D_i = rand(\ \ ) * (X_b - X_w),\ \ i = 1,2,\dots,m \qquad 10$$

$$X_w = X_w + D_i - D_{max},\quad D_i \le D_{max},\quad i = 1,2,\dots,m \qquad 11$$

Where D_i represents the leaping step size of the frog, $rand(\ \ )$is a random number within the range (0,1), and D_{max} represents the maximum allowed step size for the frog. In Eq.10an update is performed on the frog denoted as X_w, which has the poorest objective function value. If the updated solution is obtained, it is used to replace the worst solution X_w. Alternatively, if the updated solution is not obtained, Eq.10 is used to replace X_b with X_{best}, and the updated solution is recalculated using Eq.10 and Eq.11. If the updated solution is superior to X_w, it replaces the worst solution X_w. However, if the updated solution is not superior, a new solution is randomly generated to replace the current worst solution X_w. Perform the procedure for the designated number of iterations, therefore accomplishing a partial search of the population.

Finally, the sub-populations are merged to form a population consisting of a certain number of solutions. Subsequently, the sub-populations are redistributed using the approach, and the subsequent iteration of local search is executed[27]. Continue iterating the procedure until the desired solution is attained or the termination condition is satisfied. These four principles are summarized to form a SFLA algorithm as shown in algorithm2[28].

Algorithm 2: Shuffled Frog Leaping Algorithm (SFLA)

1. Generate random population of K solutions (frogs)
2. **for** each solution $i \in K$ **do**
3. calculate $fitness\ (i)$
4. **End for**
5. Sort the population K in descending order of fitness.
6. Divide K into m memeplex.
7. **for** each memeplex **do**
8. Determine the best and worst frogs.
9. Improve the worst frog position using Eq. ()
10. Repeat
11. **End for**
12. Combine the evolved memeplex.
13. Sort the population K in descending order of their fitness.
14. **If** Max iterations reached
15. Return best solution X_{best}
16. **End if**

Algorithm 2. Shuffled Frog Leaping Algorithm.

Nevertheless, the method has certain drawbacks such as sluggish convergence, susceptibility to local optima, and premature convergence. Consequently, drawing from an extensive examination of the Shuffled Frog Leaping technique (SFLA), this research paper presents an improved version of the technique to address its limitations.

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

Improved SFLA algorithm.

The initial version of the SFLA fails to account for the impact of evolutionary algebra on the magnitude of the frog's leaping step size[25]. The adaptive step factor is incorporated into the local search algorithm in order to modify the magnitude of the leaping step. The utilization of this approach serves to enhance the rate of convergence and the accuracy of optimization in the algorithm. In the context of the algorithm's local search, only the least optimal individuals within each sub-population of frogs undergo updates, while the remaining frogs do not undergo synchronization[28]. In this particular scenario, it is intended that each individual frog inside the sub-population undergoes an updating process. The sub-population of frogs will exhibit enhanced capabilities, resulting in an increased search range and greater search accuracy.

The adaptive step factor.

In the local search procedure of the classical SFLA, as stated in Eq.1, the movement step length is determined by randomly resolving the components in each dimension. Given that $rand()$ is a stochastic variable within the range of $(0,1)$ it is likely that it will either converge to a local optimum throughout the process of evolution or bypass the global optimum altogether. The diminishment of the frog population's evolutionary algebra is resulting in a reduction in the frog's range of movement. However, the conventional step taken in the classical SFLA fails to account for this phenomenon. In this particular situation, an adaptive movement factor was devised to regulate the leaping step size of the frog. The adaptive leaping step size is given in Eq. 12.

$$\beta = \sin\left(\frac{\pi}{2} \cdot \frac{i}{n}\right), i = 1,2,\dots,n \qquad 12$$

where i denotes the current count of local searches conducted within a certain subpopulation, whereas n represents the maximum count of local searches permissible for each subpopulation. Hence, for each frog i in the population, the adaptive leaping step size factor will be computed as follows:

$$\beta_i = \sin\left(\frac{\pi}{2} \cdot \frac{i}{n-1+1}\right), i = 1,2,\dots,n \qquad 13$$

Therefore, the leaping step size D_i of frog i in the population will now be recalculated as follows:

$$D_i = \beta_i * (X_b - X_w), \ i = 1,2,\dots,m \qquad 14$$

Memeplex updating strategy.

In the local search approach employed by the SFLA, only the individuals with the lowest fitness values in a memeplex undergo updates, while the remaining individuals are not updated simultaneously[26]. Hence, our research introduces a new information updating strategy among the frogs in a memeplex as follows:

$$D_k = \beta * (X_{j-1} - X_j), j = 1,2,\dots\ n \qquad 15$$

Where j represents the j^{th} frog in the memeplex and n represents the number of frogs in the memeplex.

Therefore, the updating strategy for each solution in the population is now recalculated as:

$$D_{ki} = \beta * (X_{j-1} - X_j), \qquad i = 1,2,\dots\ n \qquad 16$$

$$X_{ki} = X_{ki} + D_{ki} - D_{max}, \qquad D_{ki} \le D_{max}, \quad i = 1,2,\dots\ n \qquad 17$$

Hence, the Improved SFLA algorithm is summarized in algorithm 3.

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

Algorithm 3: Improved Shuffled Frog Leaping Algorithm (ISFLA)

17. Initialise
18. Generate random population of K solutions (frogs)
19. **for** each solution $i \in K$ **do**
20. calculate $fitness\ (i)$
21. End for
22. Sort the population K in descending order of fitness.
23. Divide K into m memeplex.
24. **for** each memeplex **do**
25. Determine the best and worst frogs.
26. Improve the worst frog position using Eq. ()
27. Repeat
28. End for
29. Combine the evolved memeplex.
30. Sort the population K in descending order of their fitness.
31. **If** Max iterations reached
32. Return best solution X_{best}
33. End if

Algorithm 3. Improved Shuffled Frog Leaping Algorithm.

Flower Pollination based Improved Shuffled Frog Leaping algorithm (FPA-ISFLA)

The Flower Pollination-Based Improved Shuffled Frog Leaping Algorithm for task scheduling (FPA-ISFLA) is to solve task scheduling problem in edge-cloud continuum environment. This algorithm combines two different nature-inspired optimization techniques, namely the Improved version of the Shuffled Frog Leaping Algorithm (ISFLA) and the Flower Pollination Algorithm (FPA), to address the problem of optimizing task scheduling. The FPA-ISFLA for task scheduling takes advantage of both SFLA's shuffling mechanism and FPA's pollination concept to tackle the task scheduling problem.

The algorithm aims to find an optimal or near-optimal task schedule by leveraging the combined strengths of ISFLA and FPA. In the local search of FPA, ISFLA is used to improve the solution exploitation with the aid of enhanced information sharing of the ISFLA withing the local solution thereby enhancing the FPA's convergence rate while avoiding trapping into local optima. The FPA-ISFLA pseudocode is shown in algorithm 4.

Algorithm 4: FPA-ISFLA algorithm

Initialise: Pop Size n, Switch probity p, t, max iteration maxIter.
1. Initialize a population of n pollen gametes at random.
2. Identify the global best solution as g*
3. $\boldsymbol{while\ (t < maxIter)}$
4. $\boldsymbol{for\ (i = 1{:}n)}$
5. $\boldsymbol{if\ rand < p}$:
6. Global Pollination via:
$x_i^{t+1} = x_i^t + l(x_i^t - g^*)$
7. $\boldsymbol{else}$ $draw\ Q\ from\ a\ uniform\ distribution\ in$ [0,1]
8. $randomly\ choose\ j\ and\ k\ among\ all\ the\ solution$
9. Local pollination via:
10. $x_i^{t+1} = x_i^t + Q(x_k^t - x_j^t)$
11. $\boldsymbol{end\ if}$
12. Evaluate the new solution.
13. If new solution is better, update global best g*
14. $\boldsymbol{end\ for}$

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

```
// apply ISFLA algorithm
  Generate random population of K solutions (frogs)
    for each solution i ∈ K do
      calculate fitness (i)
    End for
  Sort the population K in descending order of fitness.
  Divide K into m memeplex.
    for each memeplex do
      Determine the best and worst frogs.
      Improve the worst frog position using Eq. (16)
      Repeat
    End for
  Combine the evolved memeplex.
  Sort the population K in descending order of their fitness.
    If Max iterations reached
      Return best solution X_best
    End if
  Record the current global best g*
  t = t + 1
end while
```

Algorithm 4. FPA-ISFLA algorithm

Experimental Setup and Results Discussion

This section will provide a detailed description of the experiment, including the specific parameter settings that were used as well as the results that were obtained.

Experiment

As was mentioned earlier in the previous section, the FPA-ISFLA is currently being utilised to address the task scheduling problem in edge-cloud continuum environments in order to maximise the use of available resources and reduce task execution time and cost. The experiment was conducted using edgecloudsim simulator with eclipse java IDE environment. The edge-cloudsim was configured with edge data center, one cloud data center, 10 VM on edge and 20 VM on cloud and 10 mobile devices that will transmit task execution request to the edge orchestrator. The task manager in the orchestrator will evaluate the task's requests based on QoS constraints which will serve as the basis for the task scheduler to decide computing resources to allocate to the task. To evaluate the proposed algorithm is effectiveness, a comparison study is carried out between the result of the experiment and the results obtained using the Particle Swarm Optimisation Algorithm (PSO), as described in reference[29]. Re-implementation of the PSO task scheduling algorithm took place in accordance with the parameter settings that were outlined in the research.

During the experiment, a set of ten (10) different task sizes was utilised in the evaluation process. The algorithm is run ten times for each task size, and the results of those runs are recorded for the average makespan time to execute the task on both edge and cloud, the average resource utilization across the edge and cloud data center and the average task execution cost on both edge and cloud for each algorithm.

Results and Discussion

Following the extensive experimental simulation conducted, the result obtained for both makespan time, resource utilization and execution cost were plotted on a graph against each workload size to present the performance of each algorithm against different workload. so that a visual

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

representation of the correlation between the magnitude of the workload and the makespan time, in seconds, required by each algorithm to complete the task execution, the total cost of task execution and the average resource utilisation can be created. Fig. 2 shows the graphic representation of the Makespan time, a graph illustrating how the resources were utilised can be seen in Fig. 3, while the average cost of task execution is presented in Fig. 4.

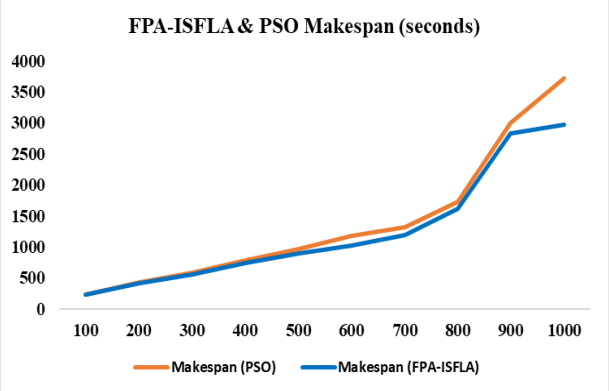

Figure 2. Avaerage Makespan time.

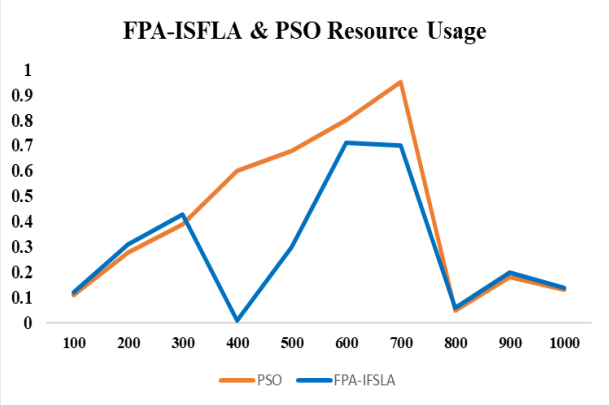

Figure 3. Average Resource Utilisation.

Conclusion

In general, the FPA-ISFLA task scheduling algorithm is efficient in handling task scheduling problem in edge-cloud environment with significant improvement against the particle swam optimization algorithm using makespan time, resource utilization and task execution cost evaluation metrics.

However, it's recommended that further improvement to me is made in the FPA global search strategy to improve its' exploitation effectiveness.

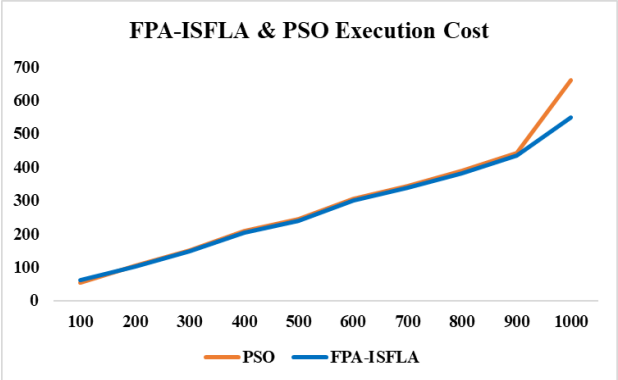

Figure 4. Average Execution Cost.

Discussion

The findings of the experiments show that the FPA-ISFLA algorithm is efficient in task scheduling within the edge-cloud continuum environment, which is comparable to the algorithm's successful performance in other spheres. When dealing with various workload sizes, the algorithm demonstrates a notable improvement in performance in comparison to the PSO algorithm, in terms of the Makespan time. The findings that is presented Fig. 2 lend credence to this observation. Furthermore, the research reveals that the proposed FPA-ISFLA task scheduling algorithm consistently attains the most optimal global solution. This is believed to be due to the incooperation of the shuffled frog leaping algorithm in the FPA local search strategy that improved the FPA exploration capacity. This leads to optimal utilisation of resources and minimises the amount of time required for searching in 20 simulation iterations. This finding is based on the data presented in Fig. 3. Moreover, the FPA-ISFLA algorithm also demonstrates a good performance against the PSO in terms of reducing the task execution cost as presented in Fig. 4.

Acknowledgement

This research was supported by the Minister of Higher Education under the Fundamental Research Grant Scheme (FRGS/1/2021/ICT02/UTM/02/13).

Authors' Declaration

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are ours. Furthermore, any Figures and images, that are not ours, have been included with the necessary permission for re-publication, which is attached to the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Universiti Teknologi Malaysia.

Authors' Contribution Statement

N.M D. contributed to the Literature review and problem formulation and developed the model. N.H.M R. contributed to Problem formulation and review experimental results. N.H M. contributed to the design of the proposed approach and mathematical modelling of makespan and cost equations. T. contributed to the design of the proposed approach and mathematical modelling of resource utilization equation. Z.M. Y. contributed to the report writing and formatting. D .G. contributed to research simulation and result analysis.

References

1. Park J, Chung K. Resource prediction-based edge collaboration scheme for improving qoe. Sensors. 2021;21(24):1–15. https://doi.org/10.3390/s21248500.
2. Mayyahi MAAL, Seno SAH. A Security and Privacy Aware Computing Approach on Data Sharing in Cloud Environment. Baghdad Sci J. 2022;19(6):1572–1580. https://doi.org/10.21123/bsj.2022.7077.
3. Goel G, Tiwari R, Koundal D, Upadhyay S. Analysis ofResource Scheduling algorithms for optimization in IoTFog- Cloud System. CEUR Workshop Proc. 2021;305(8):1–8.
4. Abo-Alsabeh R, Daham HA, Salhi A. A Heuristic Approach to the Consecutive Ones Submatrix Problem. Baghdad Sci J. 2023;20(1):189–95. https://doi.org/10.21123/bsj.2022.6373.
5. Jayasena KPN, Thisarasinghe BS. Optimized task scheduling on fog computing environment using meta heuristic algorithms. In: Proceedings - 4th IEEE International Conference on Smart Cloud, SmartCloud 2019 and 3rd International Symposium on Reinforcement Learning, ISRL 2019. Institute of Electrical and Electronics Engineers Inc.; 2019.23(2): 53–8. https://doi.org/10.1109/SmartCloud.2019.00019.
6. Gabi D, Dankolo NM, Muslim AA, Abraham A, Joda MU, Zainal A, et al. Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme. Neural Comput Appl . 2022; 12(3): 1-21. https://doi.org/10.1007/s00521-022-07260-y
7. Orive A, Agirre A, Truong HL, Sarachaga I, Marcos M. Quality of Service Aware Orchestration for Cloud-Edge Continuum Applications. Sensors. 2022; 23(1): 1-21. https://doi.org/10.3390/s22051755
8. Jiang X, Sha T, Liu D, Chen J, Chen C, Huang K. Flexible and Dynamic Scheduling of Mixed-Criticality Systems. Sensors. 2022; 18(4): 1-18. https://doi.org/10.3390/s22197528
9. Kaur N, Kumar A, Kumar R. A systematic review on task scheduling in Fog computing: Taxonomy, tools, challenges, and future directions. Concurr Comput Pract Exp. 2021;33(21): 1-18. https://doi.org/10.1002/cpe.6432
10. Swarup S, Shakshuki EM, Yasar A. Task scheduling in cloud using deep reinforcement learning. Procedia Comput Sci. 2021;18(4): 42–51. https://doi.org/10.1016/j.procs.2021.03.016
11. Yang X, Rahmani N. Task scheduling mechanisms in fog computing: review, trends, and perspectives. Kybernetes. 2021;50(1):22–38. https://doi.org/10.1108/K-10-2019-0666.
12. Gabi D, Ismail AS, Zainal A, Zakaria Z, Abraham A, Dankolo NM. Cloud customers service selection scheme based on improved conventional cat swarm optimization. Neural Comput Appl. 2020;32(18):17–38. https://doi.org/10.1007/s00521-020-04834-6
13. Abdullahi M, Ngadi MA, Dishing SI, Abdulhamid SM, Ahmad BI eel. An efficient symbiotic organisms search algorithm with chaotic optimization strategy

for multi-objective task scheduling problems in cloud computing environment. J Netw Comput Appl. 2019;133(July 2018):60–74. https://doi.org/10.1016/j.jnca.2019.02.005
14. R SK, Lakshmi J. QoS aware FaaS for Heterogeneous Edge-Cloud continuum; QoS aware FaaS for Heterogeneous Edge-Cloud continuum. 2022 IEEE 15th Int Conf Cloud Comput. 2022; 70-80. https://doi.org/10.1109/CLOUD55607.2022.00023
15. Chellapraba B, Manohari D, Periyakaruppan K, Kavitha MS. Oppositional Red Fox Optimization Based Task Scheduling Scheme for Cloud Environment. CSSE.2023; 45(1): 483-495. https://doi.org/10.32604/csse.2023.029854
16. Attiya I, Abualigah L, Alshathri S, Elsadek D, Elaziz MA. Dynamic Jellyfish Search Algorithm Based on Simulated Annealing and Disruption Operators for Global Optimization with Applications to Cloud Task Scheduling. Mathematics. 2022;10(11): 18-94. https://doi.org/10.3390/math10111894
17. Attiya I, Abualigah L, Elsadek D, Chelloug SA, Abd Elaziz M. An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing. Mathematics. 2022;10(7):1–18. https://doi.org/10.3390/math10071100
18. Karaja M, Chaabani A, Azzouz A, Ben Said L. Efficient bi-level multi objective approach for budget-constrained dynamic Bag-of-Tasks scheduling problem in heterogeneous multi-cloud environment. Appl Intell. 2022; 53: 9009–9037. https://doi.org/10.1007/s10489-022-03942-1
19. Brogi A, Forti S. QoS-aware deployment of IoT applications through the fog. IEEE Internet Things J. 2017;4(5):85–92. https://doi.org/10.1109/JIOT.2017.2701408
20. Baresi L, Mendonça DF, Garriga M, Guinea S, Quattrocchi G. A unified model for the mobile-edge-cloud continuum. ACM Trans Internet Technol. 2019;19(2): 1–21.https://doi.org/10.1145/3226644
21. Abdel-Basset M, El-Shahat D, Elhoseny M, Song H. Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. IEEE Internet Things J. 2021;8(16):12638–49. https://doi.org/10.1109/JIOT.2020.3012617
22. Yang XS. Flower Pollination Algorithm for Global Optimization. 2013; 74(45): 240-249..
23. Pavlyukevich I. Levy flights, non-local search and simulated annealing. J Comput Phys. 2007;226(2):30–44. https://doi.org/10.1016/j.jcp.2007.06.008
24. Tang D, Zhao J, Yang J, Liu Z, Cai Y. An Evolutionary Frog Leaping Algorithm for Global Optimization Problems and Applications. Comput Intell Neurosci. 2021;202(1): 1-21. https://doi.org/10.1155/2021/8928182
25. Bhattacharjee KK, Sarmah SP. Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. Appl Soft Comput J. 2014;19(2):52–63. http://dx.doi.org/10.1016/j.asoc.2014.02.010
26. Wang Z, Zhang D, Wang B, Chen W. Research on improved strategy of shuffled frog leaping algorithm. Proc - 2019 34rd Youth Acad Annu Conf Chinese Assoc Autom YAC 2019. 2019;2(6): 1–8. https://doi.org/10.1109/YAC.2019.8787721
27. Jaballah S, Rouis K, Abdallah F Ben, Tahar JBH. An improved Shuffled Frog Leaping Algorithm with a fast search strategy for optimization problems. Proc - 2014 IEEE 10th Int Conf Intell Comput Commun Process ICCP 2014. 2014;23(7): 23-27. http://dx.doi.org/10.1109/ICCP.2014.6936975
28. Liping Z, Weiwei W, Yi H, Yefeng X, Yixian C. Application of Shuffled Frog Leaping Algorithm to an Uncapacitated SLLS Problem. AASRI Procedia . 2012;1(2): 26–31. http://dx.doi.org/10.1016/j.aasri.2012.06.035
29. Nabi S, Ahmad M, Ibrahim M, Hamam H. AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing. Sensors. 2022;22(3):1–22. https://doi.org/10.3390/s22030920

2024, 21(2 Special Issue): 0740-0754
https://doi.org/10.21123/bsj.2024.10084
P-ISSN: 2078-8665 - E-ISSN: 2411-7986
Baghdad Science Journal

نهج فعال لجدولة المهام في استمرارية السحابة الحافة بناءً على تلقيح الزهور وتحسين خوارزمية قفز الضفدع المختلط

نصيرو محمد دنكولو[1،2]، نور حيزان محمد رادزي[1]، نورفا حصلينا مصطفى[1]، محمد شكر طالب[1]، زريهاتي محمد يونس[1]، دانلمي غابي[2]

[1]قسم علوم الحاسب، كلية الحاسبات، الجامعة التكنولوجية الماليزية، جوهور، ماليزيا.
[2]قسم علوم الحاسب، كلية العلوم الفيزيائية، جامعة ولاية كيبي، أليرو، نيجيريا.

الخلاصة

إن ظهور الحوسبة المستمرة السحابية الحافة هو نتيجة للأهمية المتزايدة للحوسبة الحافة، والتي أصبحت خيارًا مكملاً أو بديلاً للخدمات السحابية التقليدية. يمثل التقارب بين الشبكات وأجهزة الكمبيوتر تحديًا ملحوظًا بسبب تطورها التاريخي المتميز. تمثل جدولة المهام تحديًا كبيرًا في سياق الحوسبة المستمرة السحابية. يعد اختيار موقع تنفيذ المهام أمرًا بالغ الأهمية لتلبية متطلبات جودة الخدمة (QoS) للتطبيقات. تعد استراتيجية الجدولة الفعالة لتوزيع أحمال العمل بين الأجهزة الافتراضية في مركز البيانات المستمر لسحابة الحافة أمرًا إلزاميًا لضمان استيفاء متطلبات جودة الخدمة لكل من العميل وموفر الخدمة. استخدمت الأبحاث الحالية خوارزمية metaheuristic لحل مشكلة جدولة tak، ومع ذلك، يجب أن تعاني خوارزميات metaheuristic المستخدمة من الوقوع في مينا المحلية بسبب عدم كفاءتها لتجنب المنطقة غير المجدية في مساحة بحث الحل. لذلك، هناك حاجة ماسة إلى خوارزمية ميتايورستية فعالة لجدولة المهام. اقترحت هذه الدراسة نموذج جدولة المهام FPA-ISFLA باستخدام تلقيح الزهور الهجينة وتحسين خوارزميات قفز الضفدع المختلط. تشير نتائج المحاكاة إلى أن خوارزمية FPA-ISFLA تتفوق على خوارزمية PSO من حيث زمن التنفيذ، واستخدام الموارد، وتقليل تكلفة التنفيذ، خاصة مع زيادة عدد المهام.

الكلمات المفتاحية: السحابة، الاستمرارية، الحافة، تحسين Metaheuristics ، جدولة المهام.