Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07260-y

ORIGINAL ARTICLE

Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme

Danlami Gabi[1,2] · Nasiru Muhammad Dankolo[2] · Abubakar Atiku Muslim[2] · Ajith Abraham[3] · Muhammad Usman Joda[4] · Anazida Zainal[5] · Zalmiyah Zakaria[5]

Received: 13 February 2021 / Accepted: 29 March 2022
© The Author(s) 2022

Abstract
Achieving sustainable profit advantage, cost reduction and resource utilization are always a bottleneck for resource providers, especially when trying to meet the computing needs of resource hungry applications in mobile edge-cloud (MEC) continuum. Recent research uses metaheuristic techniques to allocate resources to large-scale applications in MECs. However, some challenges attributed to the metaheuristic techniques include entrapment at the local optima caused by premature convergence and imbalance between the local and global searches. These may affect resource allocation in MECs if continually implemented. To address these concerns and ensure efficient resource allocation in MECs, we propose a fruit fly-based simulated annealing optimization scheme (FSAOS) to serve as a potential solution. In the proposed scheme, the simulated annealing is incorporated to balance between the global and local search and to overcome its premature convergence. We also introduce a trade-off factor to allow application owners to select the best service quality that will minimize their execution cost. Implementation of the FSAOS is carried out on EdgeCloudSim Simulator tool. Simulation results show that the FSAOS can schedule resources effectively based on tasks requirement by returning minimum makespan and execution costs, and achieve better resource utilization compared to the conventional fruit fly optimization algorithm and particle swarm optimization. To further unveil how efficient the FSAOSs, a statistical analysis based on 95% confidential interval is carried out. Numerical results show that FSAOS outperforms the benchmark schemes by achieving higher confidence level. This is an indication that the proposed FSAOS can provide efficient resource allocation in MECs while meeting customers' aspirations as well as that of the resource providers.

Keywords Mobile edge clouds · Cloud datacenter · Edge datacenter · Simulated annealing · Fruit fly optimization

1 Introduction

The wireless network has with no doubt reshaped information technology (IT) as it affects positively the way we live. A rudimentary connectivity of the wireless networks is provided with the introduction of packet data services in cellular systems with a number of online users having a great impact on the quality of service produced [1, 2]. On the other hand, the traditional wireless networks are incapable of addressing the exponentially growing demand in both high data rate and high computational capability [3–5]. Consequently, compared to the growing demand for more enhanced user experience, the distance between the mobile hardware and users demand is expected to hinder the mobile devices from providing experience-rich mobile services [6, 7].

Cloud computing can complement the wireless network to provide the possibility for mobile users to access cloud resources (infrastructures, platforms, and software) on demand [8]. The paradigm can provide virtually unlimited resource and service provision to mobile devices [9]. The centralized hosts within the cloud datacenters can act as an agent between the original content providers and mobile devices, enabling resources sharing in remote datacenters [10]. Hence, the integration of cloud computing and wireless mobile environment allows running computation-intensive applications over resource-constrained mobile devices to execute computational intensive tasks [11–13].

Extended author information available on the last page of the article

Published online: 21 April 2022

This is called the mobile cloud computing (MCC). The MCC environment leverages cloud resources to enhance the performance of resource-constrained mobile devices to more powerful computing resources. However, the MCC environment attracts intolerable long latency when accessing centralized cloud resources. Hence, several applications may miss their deadline due to low bandwidth data transmission, causing delays that are unacceptable in certain cases [5, 14]. It is assumed in [6, 15] that a possible way to mitigate the challenge of unacceptable delay is to make sure computational resources are at the proximity of end-users.

Recently, mobile edge clouds (MECs) emerged as a promising solution to offer a service environment that is characterized by proximity, low latency, and high data rate access to mobile subscribers via a radio access network [16, 17]. Although the scale of the edge cloud seems limited in capacity to execute high computational tasks, the remote cloud is used to compliment resource scarcity for mobile devices [15, 18]. One of the challenges faced by MECs is limited resources. This could potentially cause hungry resource applications to miss their deadline. Hence, for MECs to realize its full potentials, the resource allocation challenge needs to be resolved. Dynamic resource allocation is a promising goal that can reduce the quality of service degradation in MEC scenarios if well carried out. Although MEC users often suffer unfair resource allocation, most especially for the resource hungry applications [19–21], the need to provide an ideal resource allocation scheme for MECs is necessary. This can reduce access overhead of the end users' mobile devices in both time consumption and costs of executing their application [22].

In this article, we study how resources allocation meet the computing need of resource-hungry applications in MECs. Makespan (known as the minimum time required for a task to be executed on a virtual machine) and costs minimization and resource utilization are the main constraints of optimization. The literature presents several heuristics and metaheuristic procedures used in solving the challenge of resource allocations in MECs. However, the heuristic techniques are known to be problem dependent and are usually trapped in a local optimum due to their greedy nature, thereby finding it difficult to reach the global optimum solution. This often affects resource allocation if implemented in MECs. On the other hand, the metaheuristic techniques are promising as well as problem independent. They can schedule a large number of resource and tasks while providing a decent approach to run away from local search to the global one, which mostly guarantees the achievability of the global optimality [16]. Therefore, to provide and efficient resource allocation in MECs, a suboptimal solution is then required.

The metaheuristic fruit fly is a swarm optimization algorithm put forward by When Tsao Pan in 2011 [23]. The development was based on food findings of the fruit fly. Since the fruit flies adopt a special smell and vision, they can smell food from a far distance. Two phases are associated with the fruit fly: smell and vision phases. The smell phase is also known as the local search phase where the flies fly through the food by using their smell capability. As they come closer to the source of food, the vision phase starts. This phase is also known as the global search phase, which is repeated until the fruit fly reaches the food. One of the basic challenges of the fruit fly is the convergence speed of its smell capability. When the smell value becomes smaller, entrapment at the local optimum becomes possible, which could lead to poor resource allocation if implemented. The simulated annealing (SA) is used. It is used to increase the diversity of a local search and discover the effect of the parameter settings for the possibility of getting better results [24]. In order to deal with the limitations of the conventional FOA, the SA-based approach is incorporated into the local search of the FOA to not only address the challenge of convergence speed but to also increase its diversity at the early phase of the search process, making it more efficient for resource scheduling in MECs. This led to the proposed fruit fly-based simulated annealing optimization scheme (FSAOS). Simulation results show that the FSAOS outperformed particle swarm optimization (PSO) and FOA in terms of makespan and execution costs, and resource utilization.

The contribution of this article is as follows:

(a) Formulation of a resource-scheduling optimization problem.
(b) Mathematical modelling of the optimization problem for the derivation of makespan and cost models.
(c) Utilization of the proposed FSAOS to solve the models.

The rest of this article is organized as follows: Related work are discussed in Sect. 2. Section 3 provides discussion on the optimization algorithms. Section 4 provides the system model. Problem formulation and proposed approach are discussed in Sect. 5. Section 6 provides discussion on the experiment and the results obtained, while Sect. 7 concludes the article.

2 Related work

This section reviews the literature associated with our research work and pointed out the limitations of the existing techniques.

2.1 Review on resource allocation

Research toward resource allocation in cloud and edge computing has been studied extensively [17] with focus on several resource providers, with each provider contributing a part of the resources (e.g., infrastructure) for service continuity and profit gain. As stated in the existing literature, this is to potentially meet objectives of the consumers and service providers. Therefore, a broad review on the area of economics of MECs with focus on resource pricing and allocation/task scheduling is carried out. As a result, profit maximization and resource utilization are some of the goals of any service provider to achieve. Although lack of pricing standard often affects customer quality-of-service (QoS) expectation in such environment, to offer a promising architecture that will mitigate this concern, [25] put forward an incentive-compatible auction mechanism (ICAM) that allows resource trading among mobile end-users and service providers. A model to incentivize budget balance and truthfulness is introduced to test the efficiency of the ICAM approach. The theoretical analysis and numerical results revealed by the researchers show that ICAM can guarantee efficiency in terms of budget balance and truthfulness for both end-users and service providers. On the other hand, a revenue auction algorithm is introduced to improve locally the end-users experience. Their main objective is to ensure nearby data are processed to reduce transmission delay and ensure minimum processing cost. A reverse auction is used to distribute overloaded data to the edge server as well as balance server load. In another development, [2] exploit the Lagrange function to develop an optimal sales price and bandwidth allocation scheme for internet service providers (ISPs). Their objective is to address the challenge of end users' denial when requesting online service as well as address low connection speeds below their stipulated contract speed limit agreement. As revealed by the researchers, their method can achieve a market equilibrium price and better bandwidth resource allocation compared to the benchmarked scheme.

Homogeneous study on MEC was carried out in [26]. A dual auction framework was developed for heterogeneous MECs. The developed system enabled offload tasks from a large-scale region using both a single and double auction-based model. Simulation results show significant improvement compared with the benchmarked schemes. Lin [1], in their part, studied dynamic resource allocation strategy for mobile edge cloud. The researchers noted that physical resource layer in network model was responsible for providing hardware resources, computing resources and storage resources. According to the researchers, the existing strategies were not efficient enough in providing solution that could potentially address existing concerns. Hence, they considered that deployment of virtual machine monitors at the physical base stations can serve as a potential solution while considering connections between virtual machine stations. Furthermore, study on automated deployment and run-time management of microservice-based applications for cloud and edge computing environments was carried out in [6]. According to the researchers, management of applications and tasks at the cloud-to-edge continuum is far from trivial, due to the lack of robust and production-level solutions. Hence, they put forward an application-level cloud orchestration framework that utilized both edge and fog nodes. Experimental results via simulation using two realistic case studies are used by the researchers to demonstrate the effectiveness of their developed solution. On the virtues of the existing challenges affecting allocation of resource in cloud computing, the need to provide more efficient resource allocation techniques is paramount. These motivated [27], to propose a QoS-based resource allocation scheme using swarm-based ant colony optimization. The effectiveness of their proposed solution was demonstrated via simulation, and results show significant performance than the benchmarked schemes. On their part, [14] put forward both resource allocation algorithm and task scheduling strategy to reduce average completion latency of an environmental monitoring application. According to the researchers, simulation results have shown significant improvement compared with the benchmarked algorithms. Consequently, on virtue of limited performance when allocating better resources across network nodes, a joint optimization algorithm that improved resource utilization, reduced energy consumption, increase network capacity was put forward by [11]. Simulation results show the developed technique can be better compared to the benchmarked schemes. On the other hand, [18] proposed a cloud resource allocation algorithm based on a parallel and improved NSGA-II (RAA-PI-NSGAII) algorithm. The algorithm computes fitness values of individuals and improved on quality of solution set. According to the researchers, simulation results show significant performance when compared to the benchmarked scheme.

Resource allocation multi-player domain is a concern, especially when considering user mobility. These motivates [16] to develop a metaheuristic-based service allocation framework to optimize the trade-off between energy consumption and makespan as well as to deal with resources heterogeneity. In [28], the researchers introduced a market-based framework to provide efficient resource allocation in heterogeneous edge nodes. An equilibrium concept in economics was then explored to generate a market equilibrium solution that maximizes resource utilization and allocates optimal resource bundles to services given their budget constraints. An approach to guarantee

fair resource allocation under constraints' mobile users' transmission rates and throughput maximization is studied in [29]. The researchers first unveiled a mathematical model based on fair Nash bargaining resource allocation game. A near-optimal bargaining resource allocation strategy for mix integer nonlinear programming optimization through time-sharing variable is later developed to solve the model. The results of the simulation as revealed by the researchers shows that their developed scheme can provide a fair allocation of resources that increase the system throughput. In a similar development, [30] stated that to realize spectrum virtualization, multi-service resource allocation needs to be addressed. As a result, a service-centric wireless virtualization model is put forward by the researchers. These then followed with a multi-service resource allocation algorithm derived by decoupling of existing scheduling algorithms to meet the expectations of the end users. According to the researchers, numerical results show that their developed scheduling algorithm can schedule specific services effectively compared to the benchmarked algorithms.

These motivated [31] to develop an auction model that facilitates resource trading between mobile service providers and mobile end-users. Simulation results according to the researchers have shown that their developed scheme can fairly allocate tasks and determine the trading price of the resources compared to the benchmarked schemes. Wang et al. [3] in their part developed a multiplier-based algorithm aiming at addressing in-network caching and computational offloading. The simulation results according to the researchers show their scheme can achieve a promising performance. On the other hand, as services are now moving towards cloud continuum, datacenters are becoming cumbersome with increased in workload, causing growth in datacenters, and therefore required more energy consumption. This motivated [32] to propose a hybrid approach that uses genetic algorithm (GA) and random forest (RF) known as GA-RF to address such concerns. In their approach, a training dataset for random forest model was generated using genetic algorithm. The effectiveness of their proposed approach was verified via simulation, and their proposed GA-RF model was able to improve resource utilization, energy consumption and execution time compared to the benchmarked models. In a similar development, [33] stated that the development of big data and artificial intelligence provide more concern to the cloud resource requests, thereby presenting more complex features like being sudden, arriving in batches and being diverse. According to the researchers, these could potentially cause resource allocation to lag far behind the resource requests and an unbalanced resource utilization that wastes resources. Hence, they proposed a proactive resource allocation method based on the adaptive prediction of the resource requests in cloud computing using nondominated sorting genetic algorithm with the Elite Strategy (NSGA-II). Simulation results show their proposed approach reduced resource allocation. In another development, improving efficiency of a resource allocation on cloud computing is the utmost priorities of several researchers. This led [34] to propose an improved particle swarm optimization (IPSO) algorithm. According to the researcher, his proposed approach compared to conventional PSO shows significant improvement in resource allocation on cloud computing. Nabi et al. [35] in their part put forward an adaptive particle swarm optimization (AdPSO) to reduce task execution time and increased throughput and average resource utilization ratio. The results of the simulation as stated by the researchers show some improvement compared to the benchmarked scheme.

Moreover, a system may decide to allocate both high wireless bandwidth and too many computation resources to a task leading to faster processing, but the system can only accommodate a few tasks with little profits in return. As a concern, a new optimization scheme using the Lyapunov technique was introduced by [31] to address a stochastic resource scheduling problem. The experimental results show that the developed algorithm can reach time average profit close to the optimum while maintaining the strong system stability and low congestion. [36–39], in their part, developed an efficient online multi-resource allocation algorithm that allows deadline-sensitive tasks to be processed at the edge-cloud system. This is to maximize the profit edge-cloud gains from meeting users' service-level agreements (SLAs). The results of the simulations achieved according to the researchers show a better profit in returns with moderate resource augmentation.

2.2 Findings from the literature review

From the related work, resource allocation in MECs is a non-deterministic polynomial-time hard (NP-hard) problem [16, 40, 41]. Inefficient resource allocation will continue affecting the performance of MECs due to its limited capacity and variation in workload. These causes delay in critical application that requires large sum of resources to be processed. Findings from the literature show that an ideal resource allocation scheme is needed for MECs to solve the challenge of resource-hungry applications. Considering QoS constraints such as execution costs, lack of standard pricing model in MECs is a concern, especially where resource providers from both the cloud and edge are self-interested or non-cooperative to agree on a unify resource price. Although few literatures addressed economic factors such as profit, cost, and revenue, majority focused on system performance metrics given system

parameters and constraints. Another vital issue when considering consumers QoS expectations is to bear in mind each consumer has conflicting objectives and different constraints (e.g., budget and technology). Achieving sustainable profit advantage, cost reduction and resource utilization is always a bottleneck when trying to meet the computing needs of resource hungry applications [42–47]. Therefore, an optimal trade-off policy that will allow a resource provider to allocate its resources in such a way to ensure other allocations cannot provide strictly higher efficiency and at the same time be fairer to service consumers [48]. It is therefore imperative to develop an ideal resource allocation scheme that not only allocates resources effectively at the MECs but also meet the expectations of consumers in terms of QoS.

3 Optimization algorithms

3.1 Fruit fly optimization algorithm (FOA)

The fruit fly optimization algorithm (FOA) is a method for finding global optimization based on the food-finding behavior of the fruit fly put forward by When Tsao Pan in 2011 [23]. Because the fruit fly uses special smell and vision to smell food from a far distance, they are superior to other species. Two phases are associated with the fruit fly: smell and vision phases. The smell phase is also known as the local search phase where the flies fly through the direction of the food using their smell capability. As they come closer to the source of food, the vision phase triggers. This phase, which is also known as the global search phase, is repeated until the fruit fly reaches the food. Figure 1 shows the food-finding process of the fruit fly, and the pseudocode for the FOA is shown in Algorithm 1 [23].

In the context of resource scheduling, each fruit fly is encoded as a task from the edge network, which also represents the network position of the fly. The resources (e.g., virtual machines) at the edge network are food search by the flies. Once the service location has been determined, the scheduling process is executed to find the optimal composition of the quality of service (makespan and costs) virtual resource. Each of the position of the fruit flies represents the initial solution of the flies.

Algorithm 1: Fruit Fly Swarm Optimization Algorithm (FOA)

Begin:

1. **Parameters initialization:** Maximum iteration number, size of the population, the random initialization fruit fly swarm Location Range (LR), and the random fly direction and distance zone of fruit fly (FR);
2. Initial position of the fruit fly swarm: $Init\ X_i, Init\ Y_i$;
3. Used smell sensing to update fruit flies location

$$X_i = X_{axis} + Random\ Value\ (LR) \tag{1}$$

$$Y_i = Y_{axis} + Random\ Value\ (FR)$$

4. Compute the fly distance $(Dist)$ from the food location

$$Dist_i = \sqrt{X_i^2 + Y_i^2} \tag{2}$$

5. Compute the smell concentration judgement value (S)

$$S_i = {}^{1}/_{Dist_i} \tag{3}$$

6. Compute the smell concentration $(Smell_i)$ of the individual fruit fly location by inserting the smell concentration judgment value (S_i) into the smell concentration judgment function (also called *objective function*):

$$Smell_i = Objective\ Function(S_i) \tag{4}$$

7. Detect the best fly with maximal smell concentration among the fruit fly swarm:

$$[bestSmell\ bestIndex] = \max(Smell) \tag{5}$$

8. Store the maximal concentration value and location of the best along the x, y coordinate. Then, the fruit fly swarm flies towards that location by using vision:

$$\begin{aligned} Smellbest &= bestSmell, \\ X_{axis} &= X(best\ index)\ , \\ Y_{axis} &= Y(best\ index). \end{aligned} \tag{6}$$

9. Repeat 4-7 until the iteration number reaches the maximum.

End.

3.2 The need to improve FOA

The conventional fruit fly has some advantages such as its fastness in solving an optimization problem, updating strategy and its simplicity in understanding. However, the smell value S_i in the conventional fruit fly is extremely small due to the dispersion of distance value to a global search region. This causes the fitness value to converge early, leading to its entrapment into the local optimum. Similarly, the smell value S is always bigger than zero (0). This means the FOA fitness function value is always positive. When these values become fixed, it is difficult for the smell value S to get a value in a uniform distribution, thereby making it impossible for the FOA to generate a uniform solution. At this point, the ability of the fruit fly toward a global solution search will be lost. Therefore, solving complex optimization problems effectively will be difficult using the fruit fly technique. Another limitation of the fruit fly optimization algorithm is that multidimensional problem cannot be handle as it was suggested for solving only two-dimensional parameter optimization problems. As a result, the two-dimensional random values cause slow convergence at the beginning of the search, while at the later search, it cannot obtain optimal solution [23, 49]. These concerns if not address may cause the FOA to provide inefficient resource allocation that will lead to resource underutilization, poor revenue and high processing time and hence the need to make FOA adaptable for resource allocation in MECs.

3.3 Improved fruit fly optimization algorithm (I-FOA)

In the conventional FOA, $S_i = 1/\text{Dist}_i$. This shows it has a weak exploration. On the other hand, with X_{axis} and Y_{axis} having fixed values, S_i does not follow uniform distribution. Hence, in the improved FOA, S_i is changed according to Eq. 8, where rand() is used to generate a random value in the interval [0,1]. This improvement eliminates the possibility of S_i falling into local optimal point. A control parameter is used by the I-FOA to moderately speed up search efficiency in the earlier stage of searching. The I-FOA is shown in Algorithm 2.

Algorithm 2: Improved-Fruit Fly Swarm Optimization Algorithm (I-FOA)

Begin:

Parameters initialization: *Initialize maximum iteration number (maxgen),* size of the population pop_{size},*searching coefficient* n, *initial weight* w_o, weight coefficient α.
// *Linear generation of a candidate solution*.

1 Initial fruit fly swarm location:

$$X_{axis} = n \times rand(), \quad (7)$$

2 Give the random direction and distance for food finding of an individual fruit fly:

$$X_i = n \times rand() + w \times rand(), \forall w = w_o \times \alpha \quad (8)$$

3 Compute the smell concentration judgement value $(S_i) = X_i$:

$$X_i = n \times rand() + w \times rand(), \quad (9)$$

4 Compute the smell concentration $(Smell_i)$ of the individual fruit fly location by inserting the smell concentration judgment value (X_i) into the smell concentration judgment function (also called *objective function*):

$$Smell_i = Objective\ Function(X_i) \quad (10)$$

5 Detect the best fly with maximal smell concentration among the fruit fly swarm:

$$[bestSmellIndex] = \max(Smell) \quad (11)$$

6 Store the maximal concentration value and location of the best along the X_i coordinate. Then, the fruit fly swarm flies towards that location by using vision:

$$Smellbest = bestSmell,\ X_i = X_i^{best}, \quad i = 1,2,3,\ldots,n \quad (12)$$

7 Repeat 4-7 until the iteration number reaches the maximum.

End.

3.4 Simulated annealing

Simulated annealing (SA) is a local search probabilistic approximation algorithm [41] put forward by Kirkpatrick et al. in 1983. The SA algorithm often begins with an initial solution X according to some neighborhood function N with an updated solution X' created. A neighborhood-based mutation operator strategy can be incorporated into the SA algorithm for enhancing population diversity. Assuming S is the search space, each solution X ∈ S can be represented by its $n(>0)$ components, i.e., $X = (x_1, x_2, \ldots, x_n)$, where $x_i \in X_i, i = 1, 2, \ldots, n$. The neighborhood N of a solution X is defined as:

$$N_X = \{X\prime | X\prime \in S, P_{ro}(X, X\prime, T) \rangle R(0,1)\} \tag{13}$$

where $X \notin N_X$, and $X \in N_{X'}$ iff $X' \in N_X$, P_{ro} is the probability of deciding whether to accept or reject a new solution and R is a random number generated uniformly between 0 and 1. As to how the particle tends to adopt a state, which is an improvement over current one, the algorithm generates a solution when fitness value $f(X')$ becomes lower than $f(X)$. Assume X' has the higher fitness, it will occasionally be accepted if the defined probability shown in Eq. 14 is satisfied [24]:

$$P_{ro}(X, X', T) = \exp\left(-(\Delta f) \times T^{-1}\right) \tag{14}$$

where Δf is the difference between the fitness values and $\Delta f = f(X') - f(X)$, $f(X')$ is the fitness evaluation function and $f(X)$ the current solution of the neighbor accordingly; T represents the temperature parameter. This parameter is determined according to the cooling rate used in [24].

$$T = \sigma^i + T_O + T_{\text{final}} \tag{15}$$

where σ^i = temperature descending rate,$\forall 0 < \sigma < 1$; i = the number of stints, which neighbor solutions have been generated so far; T_O = initial temperature; T_{final} = final temperature. When the initial value of the temperature is low, the algorithm becomes limited in locating global optimal solution as the computation time of the algorithm is believed to be shorter. At each iteration performed by the SA algorithm, the comparison between the currently obtained solution and a solution newly selected is carried out. A solution that shows improvement is always accepted. The non-improving solutions are still accepted since there is a possibility that they may escape being trapped at local optima while searching for a global optimum solution. Based on the defined probability in Eq. 14, the acceptance of the non-improving ones is often determined by the temperature parameter. This makes SA algorithm one of the most powerful optimization mechanism for the improvement of a local search. The basic SA procedure is represented in Algorithm 3.

Algorithm 3: SA pseudocode [24]

INPUT: Initialize Temperature T_o, Final Temperature T_{final}, Temperature change counter $p = 0$, Cooling schedule σ, Number of iteration M_p
OUTPUT: Best Optimum Solution found

1. Generate an initial solution $X \in D$
2. **Repeat**
3. Initialize repetition counter $m \leftarrow 0$
4. Repeat
5. Generate a new solution $X' \in N$, where N according to **Equation (13)** is the neighbourhood of X
6. $\Delta f = f(X') - f(X)$ *// which is the difference between the fitness values*
7. *If* $\Delta f < 0$, take X'as the new best solution X. Otherwise go to step 8
8. Compute $P_{ro} = \exp(-(\Delta f) \times T^{-1}$)according to **Equation (14)**
9. Generate a random number R uniformly distributed in [0,1]
10. *If* $\Delta f \leq 0$ *or* $P_{ro} > R(0,1)$ // decide whether to accept or reject a new solution according to P_{ro}
11. $X \leftarrow X'$
12. Repeat counter $m \leftarrow m + 1$
13. Memorize the optimum solution so far found
14. Until $m = M_p$
15. $p \leftarrow p + 1$
16. **Until** stopping criteria is not exceeded

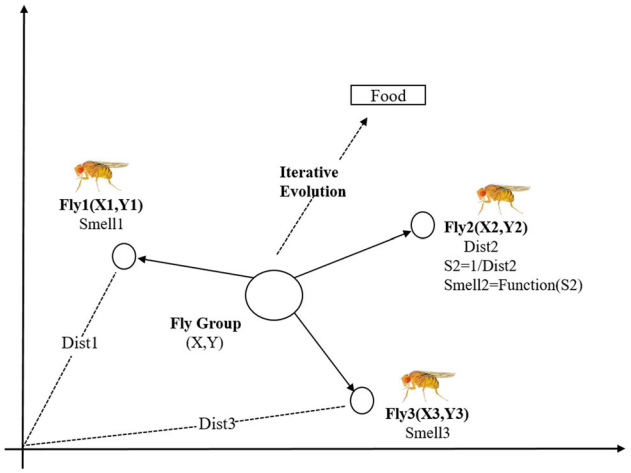

Fig. 1 Fruit fly swarm iteration process of food searching (18)

4 System model

We assume a paradigm with a distributed cloud resource provider (DCRP) that manages resources at the centralized cloud. The edge resource is managed by an edge resource provider, but coexist with DCRP at the centralized cloud to allocate resources to the edge consumers. Figure 2 shows the system framework. Three types of entities are involved in this scenario: resource provider (RP) at the edge, a distributed cloud resource provider (DCRP), and a pool of consumers known as mobile users (MUs). The FSAOS algorithm is situated within the edge orchestrator and does the resource optimization and allocation purposes. If an MU requires high computation-intensive resources at the edge, more resources are requested by the FSAOS algorithm to argument the already existing one at the edge. With the coexistence of the edge service providers, an MU that requests for resource having budgetary constraint need not to concern as the proposed FSAOS dynamically assign resources to customer's application using the right computing resource.

In the resource and task scheduling process, each fruit fly within the swarm is initialized as a set of tasks uploaded by the mobile consumers to the edge resources. The fruit flies are encoded as a set of tasks emanating from the edge network, which also represents the network position of the flies. The resources at the edge network are food search by the flies as indicated in Fig. 1. Once the service location has been determined, the algorithm within the FSAOS executes the scheduling process to find the optimal composition of the quality of service (makespan and costs) virtual resource. The position of the fruit flies represents the initial solution of the flies. Instead of assigning tasks to resources at random, the algorithm within the FSAOS schemes selects the best resource that meets the computing need of each uploaded tasks and assign these resources according to the boundary location. This process is repeated until the QoS is achieved for all tasks scheduled on the computing resources.

5 Problem formulation and proposed approach

This section discusses the resource allocation problem and the proposed approach that is used in solving the model derived from the resource allocation problem.

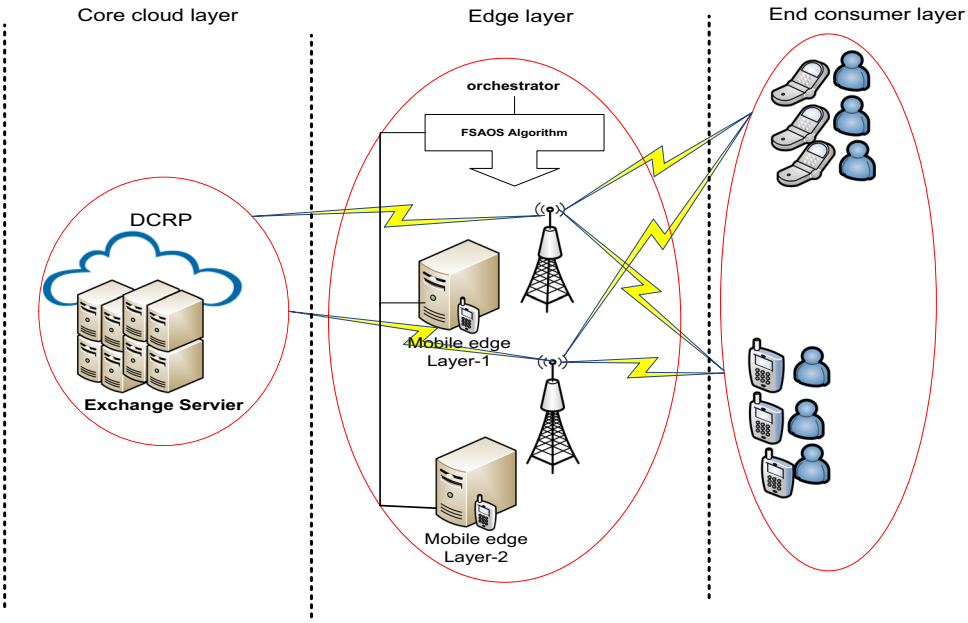

Fig. 2 Framework model for resource scheduling in MECs

5.1 Problem formulation

Due to workload variation and the limited capacity of resources at the MECs, the performance of an application may be affected unless an ideal resource scheduling optimization scheme is designed. We assume the costs of resource usage consist of both the virtual machine and data transfer cost. Consequently, a resource provider cannot be paid less than its resource unit cost, while the allocated providers' resources must fulfill a service consumer's demand. Therefore, the expected time of computation of each resource (e.g. virtual machines) is taken into consideration based on their capacity (ability) to execute tasks. More precisely as it relates to this problem, the resources are considered heterogeneous in nature and the tasks to utilize these resources also vary differently from mobile users based on various factors, like costs, and wireless link performance (makespan). Each mobile consumer submits its bid to the edge provider by selecting the best service that relates to its demand. Our goal is to ensure resources are distributed effectively to execute users' applications without compromising their quality of service while still ensuring better resource utilization. The scheduling model is as follows:

Let $T = \{t_i | 1 \leq i \leq n\}$ represents the tasks groups and n is the overall number of tasks. $V = \{v_j | 1 \leq j \leq m\}$ are the sets of virtual machines and m is the number of virtual machines. The goal is to assign the most needed virtual machine $v_j \forall j = \{1, 2, \ldots, m\}$ to execute users' task $t_i \forall i = \{1, 2, \ldots, n\}$ at the edge taking the following objectives: makespan and cost, and resource utilization. Hence, the makespan time of the tasks executed on a virtual machine is reduced using Eq. 16,

$$\min: \quad \text{Makespan}(M) = \max\left(TE_{ij}\right), \quad 1 \leq j \leq m \tag{16}$$

$$TE_{ij} = \sum_{i,j}^{n,m} \text{exeTime}_{ij}, \quad 1 \leq i \leq n; 1 \leq j \leq m \tag{17}$$

$$\text{exeTime}_{ij} = x_{ij}.\frac{\text{task size}_i}{v_{npej}}, \quad 1 \leq i \leq n; 1 \leq j \leq m \tag{18}$$

$$x_{ij} = \begin{cases} 1, & \text{if virtual machine } v_j \text{ is assigned to execute task } t_i \\ 0, & \text{if virtual machine } v_j \text{ is not assigned to execute task } t_i \end{cases} \tag{19}$$

where $\text{Makespan}(M)$ represents the maximum time of execution among all virtual machines at the edge and cloud servers; TE_{ij} represents the total execution time; exeTime_{ij} represents the execution time of task i on virtual machine v_j; x_{ij} is equal to 1 if v_j is assigned to execute task t_i and zero (0) otherwise; n is the number of tasks t_i and m is the number of virtual machine v_j; v_{npej} represents the computing capacity of the virtual machine v_j.

On the other hand, let $C = \{c_j | m \geq j1\}$ represent the unit cost of virtual machine v_j per time quantum. The unit cost of executing a task t_i on a virtual machine v_k in this research is considered per hour (/hr), which is computed in US dollar ($). Therefore, the amount of time a task spent running on a virtual machine in seconds is converted into hours. We assume the costs of resource usage comprises both the cost of virtual machine and the cost of data transmission. Equation 20 computes the total execution cost of task i on all virtual machine v_j,

$$T_{\text{cost}} = \sum_{i}^{n} C_{\text{exe}_{ij}}, \quad 1 \leq i \leq n \tag{20}$$

$$C_{\text{exe}_{ij}} = \sum_{i}^{n} x_{ij}.\frac{\text{task size}_i}{np_{\text{e}} \times v_{j\text{speed}}}.C_j, \quad 1 \leq i \leq n; 1 \leq j \leq m \tag{21}$$

where T_{cost} represents the total execution cost of processing all tasks; C_{exe} represents the execution cost of a resource usage when a task t_i assigned on virtual machine v_j; C_j is the price of a unit cost of resource per second as specified by the resource provider. Thus, the trade-off factor α for all mobile users can now be introduced to the model for optimizing the trade-off between makespan time and costs based on service preference. This factor is within $0 \leq \alpha \leq 1$ and does help the mobile users to make better realization of the optimization objective. Therefore, the scheduling problem, which is also the fitness function of the scheduling problem, can be presented as shown in Eq. 22.

$$\text{Minimize}: \alpha * \text{Makespan} \ + (1 - \alpha) * T_{\text{cost}}, \tag{22}$$

where α is the trade-off factor, Makespan is the average execution time of the entire tasks processed as achieved by the algorithms, and T_{cost} is the total cost of processing all the tasks. For equation of the fitness function, relevant information concerning the decision variables and a fitness value is encapsulated into the proposed algorithm, which has an indicator of its performance. The best fitness value shows the highest degree of adaptation of the proposed schemes in providing better scheduling solution. It also helps check whether loads were shared equally on the entire system while trying to minimizing the makespan and

increases the processing capacity of a given task set. The resource utilization is the actual amounts of resources consumed at both the edge and the cloud datacenter. One of the main objectives of resource utilization is making sure resources are well utilized to provide better profit to the resource providers in terms of revenue and profits. Equation 23 is used to compute the resource utilization R_{util} on both edge and the cloud environment [41].

$$R_{\text{util}} = \frac{\sum_{j=1}^{m} \frac{TE_{ij}}{\text{exeTime}_{ij}}}{m}. \tag{23}$$

where TE_{ij} is the total execution time of the whole virtual machines, exeTime_{ij} is the execution time of task t_i on virtual machine v_j, and m is the number of virtual machines.

5.2 Proposed approach

An ideal resource and task allocation scheme that can schedule both resource and task to meet the computing demand of mobile cloud users is needed. This led to the developed FSAOS scheduling scheme. The developed FSAOS is a combination of both the FOA and SA algorithms. The local search of the FOA is constrained by premature convergence, causing the FOA to be entrapped at local optima. This can lead to providing poor scheduling performance when implemented in MEC. The procedure of Algorithm 2 and that of Algorithm 3 combined led to the developed Algorithm 4. With the developed FSAOS, moving out of the local optima region is possible for the best resource allocation. To increase the performance parameter estimation of the proposed FSAOS, the local search is enhanced with the SA. The SA also functions in the stabilization of both the local and global search procedures of the FSAOS scheme while ensuring its diversity of solutions. Hence, more powerful optimization search procedures that search for a global optimum solution are made possible using a uniform distribution in the FSAOS scheme.

In the resource scheduling and task allocation process, each fruit fly within the swarm is initialized as a set of tasks to be sent by mobile users to the edge resources and the available resources at the edge are food searched by the flies. In this research, the fruit flies are encoded as a set of tasks emanating from at the edge network. These also represent the network position of the flies. The position of the fruit flies represents the initial solution of the past flies. Instead of assigning tasks to resources randomly, the FSAOS algorithm selects the best resources to compute the need of each task and assign such resources according to the boundary location. The smell sense is then used to update the position of the flies according to Eq. 24,

$$\begin{aligned} x_{i_\text{axis}} &= n \times \text{randValue}() + w_o \times \alpha \times \text{randvalue}(), \\ &\forall i = 1, 2, 3, \ldots, n \end{aligned} \tag{24}$$

where x_i represent the coordinates for the resource services obtained by FSAOS, n is the searching coefficient, w_o is the initial weight and α is the weight coefficient. Distance x_i measure is found for each dimension of the objective space by including the effect of an individual's constraint violation into its objective function. The major steps in calculating the distance measure start with obtaining the minimum and maximum values of each objective function in the population. The objective space is modified to account for the performance and constraint violation of each individual. The objective (fitness) function of the individual fruit flies is computed using Eq. 25. The objective functions are used to facilitate the search of optimal solutions not only in the feasible space but also in the infeasible regions. The cost and the makespan value are used for the computation of the fitness value.

$$\begin{aligned} \text{minimize}(f) = {} & \alpha(\text{makespan of virtual machine}) \\ & + (1 - \alpha)T_{\text{cost}} \end{aligned} \tag{25}$$

where α is the weight coefficient that allows mobile users to select their preferred services. The fruit fly with the best fitness among the fruit fly swarm is selected using Eq. 26.

$$[\text{Smellbest}' \; \text{bestindex}'] = \min(f) \tag{26}$$

For the minimization process, the new fitness is accepted for an update according to the probability function defined in Eq. 27. The new fitness is designed to carry out search both at the feasible and infeasible space to exploit those individuals with better objective values. The number of feasible individuals in the population is used to guide the search process either toward finding more feasible solutions. The number of feasible individuals in the population adaptively controls the emphasis given to objective values or constraint violation in the objective function formulation. If there is no feasible individual in the population, the algorithm selects the optimum from the best solutions.

$$P_{ro}\left(X_i^{'}, X_i, T\right) = \exp\left(-\left(f\left(X_i^{'}\right) - f(X_i)\right) \times T^{-1}\right) \quad (27)$$

where $f\left(X_i^{'}\right)$ and $f(X_i)$ denote fitness functions of the fruit flies and current solutions, and T represents the temperature. The fly with the best fitness is stored at each run of the scheme and is compared with the initial best solution. The probability of accepting a neighbor solution into a new generation of fruit flies using SA is obtained according to Eq. 27. The FSAOS algorithm is described as shown in Algorithm 4.

6 Experiment and results discussion

This section provides detail information about the experiment conducted as well as the results obtained.

6.1 Parameter setup

The EdgeCloudSim simulator tool [41] is used for the experiments. The EdgeCloudSim supports the simulation of multi-tier scenarios where several servers are running in coordination with upper layer cloud solutions. Tasks are

Algorithm 4: FSAOS Algorithm

Begin:
Input: *Initialize maximum iteration number (maxgen), population size* pop_{size}, *searching coefficient* n, *initial weight* w_o, weight coefficient α, *Initialize SA parameters: initial Temperature* T_O, *final Temperature* T_{final}, *cooling rate* α', Number of iteration M_p.
Output: Best fly solution with minimum makespan and cost of resource usage.

1. Randomly generating fruit fly positions $X \in D \forall \boldsymbol{D} = \{dimension\}$
2. **Do**
   {
3. **Repeat**
4. Initialize repetition counter $m \leftarrow 0$
5. ***//Execute the I-FOA based global search***
6. Initial fruit fly swarm location according to **Equation (7)**
7. Assign direction and distance for food finding of an individual fruit fly according to **Equation (8)**
8. Compute smell concentration judgement value $(S_i) = X_i$ according to **Equation (9)**
9. Substitute the smell concentration judgment value (X_i) into the smell concentration judgment function (also called *objective function*) according to **Equation (10)**
10. Detect the best fly with maximal smell concentration among the fruit fly swarm according to **Equation (11)**
11. Store the maximal concentration value and location of the best along the X_i coordinate and use vision search to fly towards the best smell value location according to **Equation (12)**
12. ***// Execute the simulated annealing (SA) based local search***
13. Generate a new solution $X' \in N$, where N according to **Equation (13)** is the neighbourhood of X
14. $\Delta f = f(X_i') - f(X_i)$*// which is the difference between the fitness values*
15. $If\ \Delta f < 0$, take X_i' as the new best solution X_i. Otherwise go to step 20
16. Compute P_{ro} according to **Equation (27)**
17. Generate a uniformly distributed random number R in [0,1]
18. $If\ \Delta f \leq 0\ or\ P_{ro} > R(0,1)$
19. $X_i \leftarrow X_i'$
20. Repeat counter $m \leftarrow m + 1$
21. Memorize the optimum solution so far found
22. Until $m = M_p$
23. $p \leftarrow p + 1$
24. **Until** stopping criteria is not exceeded
25. //**Endif**
26. //**Endif**
27. //**EndDo**
28. **While** ***(gen*** $\neq$ ***maxgen)***
29. Output optimization solution for the execution time and execution cost.

End.

Table 1 Parameter settings for the edge and cloud computing data center

Datacenter	Parameter	Values
Host	No. of edge/cloud	2
	No. of data center per edge/Cloud	1
	Host RAM on edge/cloud	10/20 GB
	No. of hosts per edge datacenter/cloud	1
	Storage on edge/cloud	0.5/ITB
	Bandwidths on edge/cloud	5/10 GB/s
	Accumulated host processing power on edge/cloud	250,000/1000000 MIPS
	Operating systems	Linux
	Virtual machine monitor	Xen
Tasks	No. of tasks	[100–1000]
	Average task length	[100, 1000] MIS
	File size	[200, 400] MB
	Output size	[300]
Virtual machine	Virtual machine number per edge server/cloud	30
	Accumulated Ram on edge/cloud	0.5 GB/2 GB
	Accumulated storage on edge/cloud	5/10 GB
	WAN/WLAN bandwidth	20/300 Mbps
	VMs processing power per edge server/cloud	1000-/20000 MIPS
	Number of processing elements edge/cloud	1–2/3–4
	VM Policy	Time-shared
	Costs per unit VM based on edge/cloud	0.022–0.12/0.051–0.1$/hour

Table 2 The parameter setting for the four task scheduling schemes

Algorithm	Parameter	Value
PSO	Population size	100
	coefficients (c1, c2)	2.0
	Uniform random number ($R1$)	[0,1]
	Minimum iteration number (n_i)	5
	Maximum iteration	1000
	Variable inertia weight (W)	90–40%
FOA	Population size (sizepop)	100
	Minimum iteration number (n_i)	5
	Maximum iteration number (maxgen)	1000
FSAOA	Population size (sizepop)	100
	Minimum iteration number (n_i)	5
	Maximum iteration number (maxgen)	1000
	Searching coefficient	2
	Initial weight (w_o)	1
	Final temperature	0.001
	Weight coefficient	90%
	Initial temperature	10
	Cooling rate	0.9

generated randomly. We extended the edge orchestrator module being the decision layer to implement our developed resource scheduling scheme. The edge/cloud parameter settings for the datacenter (as illustrated in Table 1) were based on [50]. The settings for the scheduling scheme are shown in Table 2.

The parameters of the algorithms are employed by predefining in the initialization stage. Constant value of the coefficients c1 and c2, together with the random vectors r1 and r2, is automatically adjusted for each numeric benchmark function based on greedy search while computing the step size based on greedy search. The parameters are investigated within given ranges and incremental steps. The complexity of a problem is then measured as a function of those parameters. As the coefficients c1 and c2 > 0, each particle finds the best position in its neighborhood by replacing the current best position. On the other hand, the number of iterations n_i is fixed in which it helps the algorithms to reach a good solution. Although too few iterations may terminate the search prematurely, a too large number of iterations has the consequence of unnecessary added computational complexity, provided that the number of iterations is the only stopping condition. As seen in the literature, this paper uses 1000 as its stopping criteria.

6.2 Performance metrics

The performance metrics used to evaluate the proposed FSAOS is makespan, costs and utilization as indicated in Table 3. These metrics are used to determine the stability improvement of the proposed scheme.

Table 3 Evaluation metrics for the scheduling schemes

Metric	Definition	Equation
Makespan time	This is the total time required to process all tasks [12]	$M = \max(\mathrm{ET}_{ij}), (28)$ where M is the makespan; ET_{ij} is the expected time to compute
Cost	The amount payable according to resource usage. This consists of both the bandwidth and virtual machine cost	Using Eq. (20)
Utilization	This is the overall amount of resources that is consumed in the datacenters [31]	Using Eq. (23)

6.3 Results discussion

Ten (10) independent simulation runs were conducted, and an average of the simulation results was reported [51]. To show how efficient is the resource scheduling scheme, makespan and execution cost, and resource utilization were used in the evaluation. The FSAOS resource scheduling scheme is compared against that of the particle swarm optimization algorithm (PSO) and fruit fly optimization algorithm as well as the state-of-the-art approaches. Our scheduling model is used in the evaluation of both the PSO, FOA and the state-of-the-art schemes to show their effectiveness against our developed FSAOS resource scheduling schemes. Tables 4 and 5 show the makespan time obtained simultaneously when optimizing resources and tasks. Tables 6 and 7 show results obtained in terms of the average execution cost, while Tables 8 and 9 show the resource utilization at both the edge and cloud datacenters. Table 10 shows comparison with state-of-the-art approaches.

To demonstrate the effectiveness of the resource scheduling schemes in terms of performance at the edge and cloud computing environments, we show the results of makespan indicated in Tables 4 and 5. As the number of tasks increases, the makespan solutions of the FSAOS, FOA and PSO schemes also increase exponentially on both edge and cloud computing environments. However, with increase in ratio of number of tasks from 100 to 1000 allocated to the virtual machines, the makespan in cloud environment is significantly less than that in the edge environment. These can be attributed to the load variation,

Table 4 Average makespan time achieved by the three scheduling schemes at the edge

TASKS	PSO	FOA	FSAOS
100	8418.54	8757.33	7986.41
200	9458.57	9599.95	8793.48
300	9740.22	9604.81	9863.37
400	9984.06	9772.88	9904.64
500	10,085.53	9752.65	9826.34
600	9865.53	9970.42	8819.14
700	10,089.20	9751.09	9280.14
800	9717.16	9906.16	9902.74
900	9875.22	9866.02	9655.15
1000	9804.01	9838.66	8819.14

Table 5 Average makespan time achieved by the three scheduling schemes at the cloud

TASKS	PSO	FOA	FSAOS
100	222.53	232.68	217.98
200	411.59	428.40	414.29
300	586.65	597.47	599.32
400	785.39	1564.55	824.67
500	963.82	1024.24	975.02
600	1173.99	1200.51	1217.49
700	1296.34	1435.63	1369.22
800	1719.09	1581.66	1547.83
900	2874.87	1773.44	1764.64
1000	3652.53	2005.93	1975.69

Table 6 Average costs of execution achieved by the three scheduling schemes at the edge

Tasks	PSO	FOA	FSAOS
100	1800.94	1974.77	1988.37
200	2185.87	2167.60	2133.12
300	2224.18	2165.92	2196.4
400	2233.47	2203.79	2251.40
500	2215.84	2199.22	2274.24
600	2198.42	2253.19	2224.67
700	2260.64	2198.89	2365.12
800	2233.97	2233.83	2191.22
900	2177.23	2224.79	2226.86
1000	2193.42	2218.61	2190.78

Table 7 Average costs of execution achieved by the three scheduling schemes at the cloud

Tasks	PSO	FOA	FSAOS
100	54.95	52.45	55.86
200	103.98	107.52	103.30
300	150.42	149.96	147.24
400	206.98	205.31	197.13
500	244.70	257.08	241.92
600	305.57	301.32	294.67
700	343.67	360.33	325.38
800	388.50	396.99	384.24
900	442.82	445.13	431.48
1000	660.24	503.48	485.06

Table 8 Resource utilization achieved by the three scheduling schemes at the edge

Tasks	PSO	FOA	FSAOS
100	5.612	5.324	5.838
200	6.305	5.862	6.399
300	6.493	6.575	6.403
400	6.656	6.603	6.515
500	6.723	6.550	6.501
600	6.577	5.879	6.646
700	6.726	6.186	6.500
800	6.478	6.601	6.604
900	6.583	6.436	6.577
1000	6.536	6.486	6.559

Table 9 Resource utilization achieved by the three scheduling schemes at the cloud

Tasks	PSO	FOA	FSAOS
100	0.155	0.145	0.148
200	0.285	0.276	0.274
300	0.398	0.399	0.391
400	1.043	0.549	0.523
500	0.682	0.650	0.642
600	0.800	0.811	0.782
700	0.957	0.912	0.864
800	1.054	1.031	1.916
900	1.182	1.176	1.146
1000	1.337	1.317	2.435

as several tasks are executed using the edge datacenter resources, while few tasks that require high computation resources are executed using the cloud datacenter resources. Consequently, further analysis on the makespan solution demonstrated that the FSAOS has achieved better makespan at the edge and cloud computing environment than the PSO and FOA scheduling schemes. On the other hand, the impact on the makespan solutions achieved by the resource scheduling scheme is revealed on the total execution cost as indicated in Tables 6 and 7. Fluctuation in the simulation results is an indication that all the scheduling schemes try to allocate resources to tasks to guarantee the least execution cost. However, these impact significantly on the execution cost, especially when the scheduling schemes on the other hand maps these resources with low processing time to tasks, thus incurring high execution cost. In a similar development, costs' solutions obtained by PSO and FOA also show some improvement inasmuch as the FSAOS does returned better average execution cost solutions on both edge and cloud computing environments.

In Tables 8 and 9, the performance utilization of the three resource scheduling schemes on both edge and cloud computing environments is provided. In the overall resource utilization, the FSAOS provides better utilization of resources than the benchmarked schemes. It can also be seen from Tables 6 and 7 an average execution costs solution obtained by the FSAOS is better than those of the benchmarked schemes in both the edge and cloud environments under a task scheduling interval of 100–1000. Although FSAOS resource scheduling scheme was able to optimized resources and allocate tasks to the most needed computing resources, its performance can still be improved.

To have a better understanding on the performance of the FSAOS, FOA and PSO schemes, figures are used to show the trend as illustrated in Figs. 3, 4, 5, 6, 7, 8 in terms of makespan, average execution cost and resource utilization. For the convergence graphs in terms of makespan and average execution costs, the trend on the graphs indicates that the FSAOS obtains lowest makespan time and execution cost solutions at the edge and cloud datacenters, respectively, compared to the benchmarked schemes. We

Table 10 Results of the makespan time for the resource allocation algorithms

Tasks Instances	GA-RF	IPSO	Improved NSGA-II	AdPSO	FSAOS
200	3564.32	2974.06	3013.01	2814.76	2485.74
400	4703.01	4603.56	4457. 22	4372.67	4123.35
600	5163.33	5446.43	5291.12	5102.83	5087.45
800	7869.15	7098.34	6739.15	6596.21	6201.11
1000	8743.65	8654.89	8506.13	8379.42	7878.45

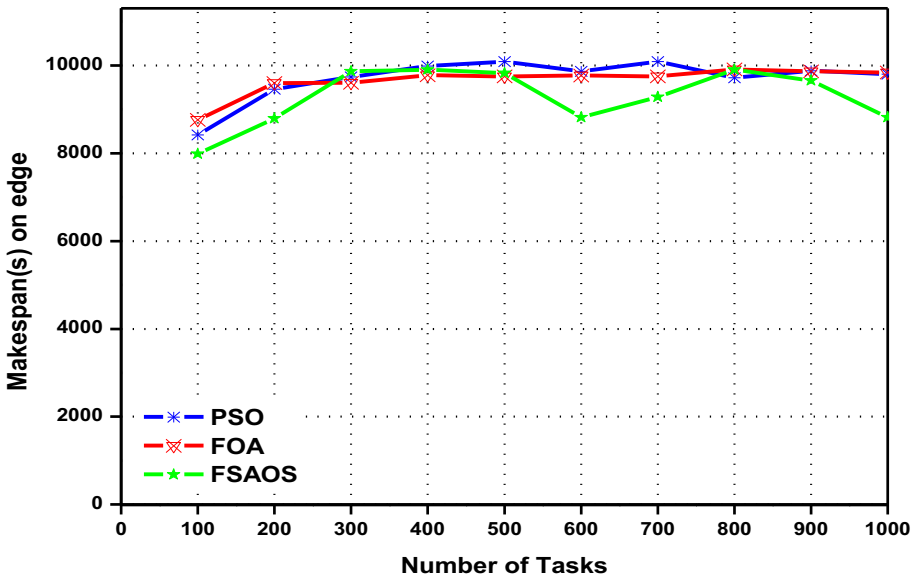

Fig. 3 Performance based on makespan time on edge

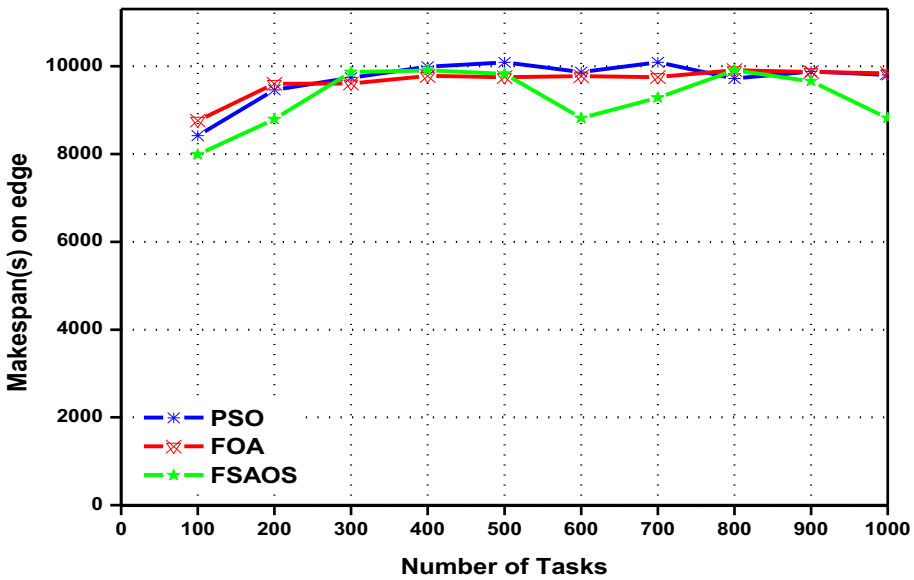

Fig. 4 Performance based on makespan time on cloud

notice that when the tasks scheduling interval increases, the execution time and costs solutions also increase for all the scheduling schemes. The real reason is that the edge datacenter starts allocating resources where the number of tasks to execute these resources increases as well. These increases can be viewed as illustrated on the figures showing the trend of the solutions for both edge and cloud datacenter. We also notice that FSAOS resource scheduling scheme is able to optimize resources and tasks that guarantee an acceptable cost of execution and makespan compared to that of the benchmarked schemes. In a similar development, Figs. 7 and 8 show the resource utilization performances. The resource utilization achieved by FSAOS, FOA and PSO at the edge is significantly higher than at the cloud and thus maintain a relatively better performance. In the overall performance, FSAOS has proven to be more robust in ensuring better utilization of resources than PSO and FOA.

The reason why the FSAOS was able to achieve such performance in terms of makespan, costs and resource utilization is believed to be attributed to the incorporation of SA, which energizes the smell vector within FSAOS that is capable of preventing premature convergence. This also ensures that inactive solutions not needed are eliminated and introducing a more active solutions thus pushing away search processes from local optima. Such incorporation gives local search of the proposed FSAOS exploitative power to traverse the best solution, which makes the FSAOS to obtain near-optimal solution than PSO and FOA.

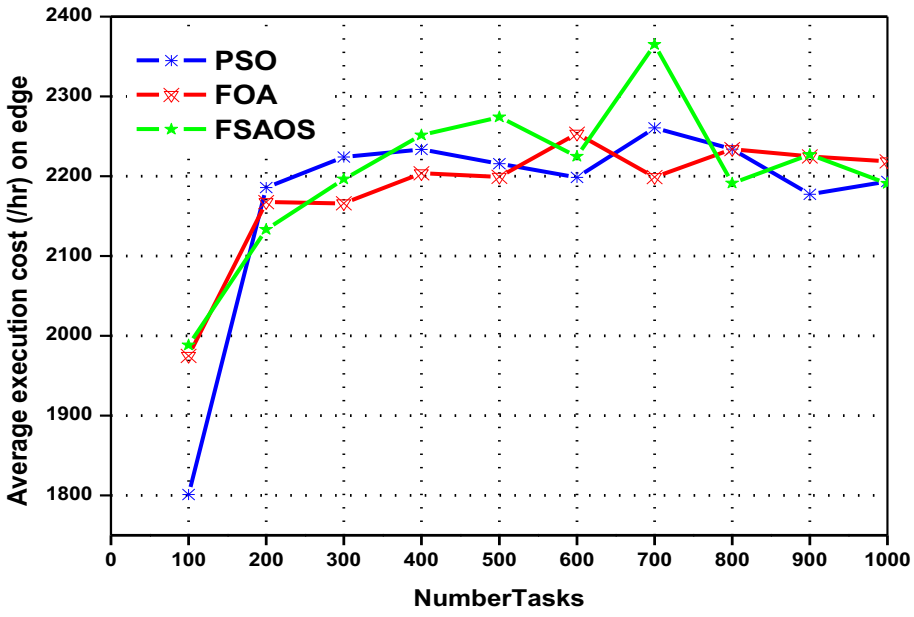

Fig. 5 Average cost of execution on edge

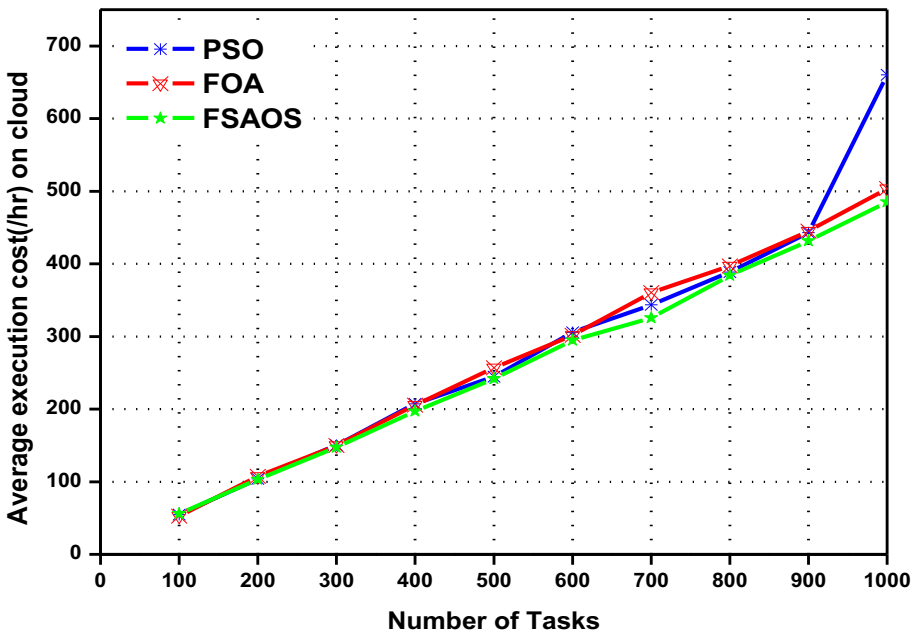

Fig. 6 Average cost of execution on cloud

6.4 Comparison with state-of-the-art Approaches

To compare the proposed FSAOS resource allocation algorithm based on makespan time with that of the state-of-the-art approaches (e.g., in [32–35]), the population size, minimum and maximum iteration parameters used were similar for the benchmarked schemes. Twenty (20) independent simulation runs were conducted on the tasks instances: 200, 400, 600, 800 and 1000, and an average of the results obtained from the clod platform is reported in Table 10.

Moreover, based on the results obtained in Table 10, for every task instance used, study shows that proposed FSAOS resource allocation scheme is able to achieve the best global optimal solution at low searching time on an average of 20 simulation runs and achieve a solution better than GA-RF, IPSO, improved NSGA-II and AdPSO. It can therefore be concluded that the performance of FSAOS was attributed the increase in its convergence speed that improves its local search while maintaining its best global search procedure.

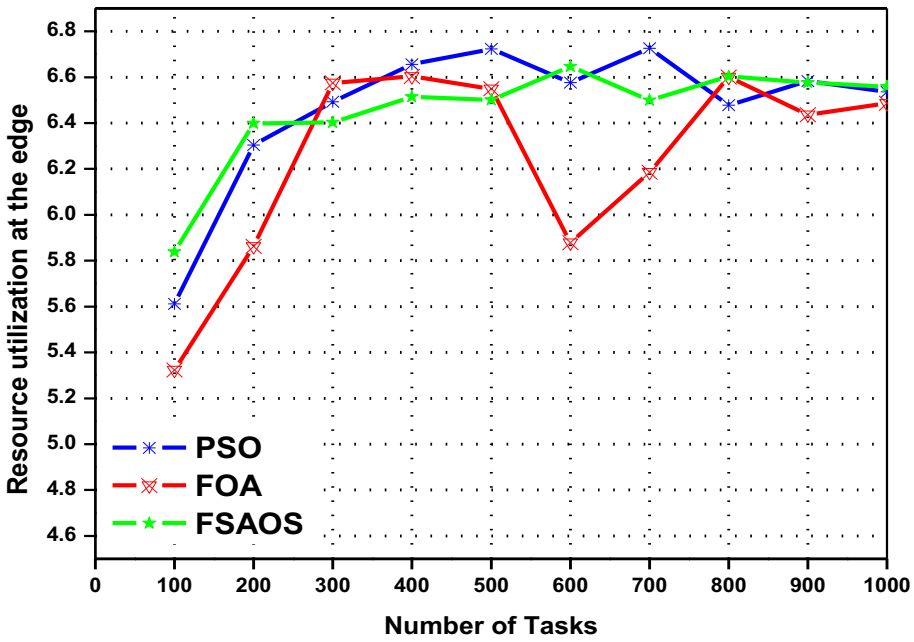

Fig. 7 Resource utilization on edge

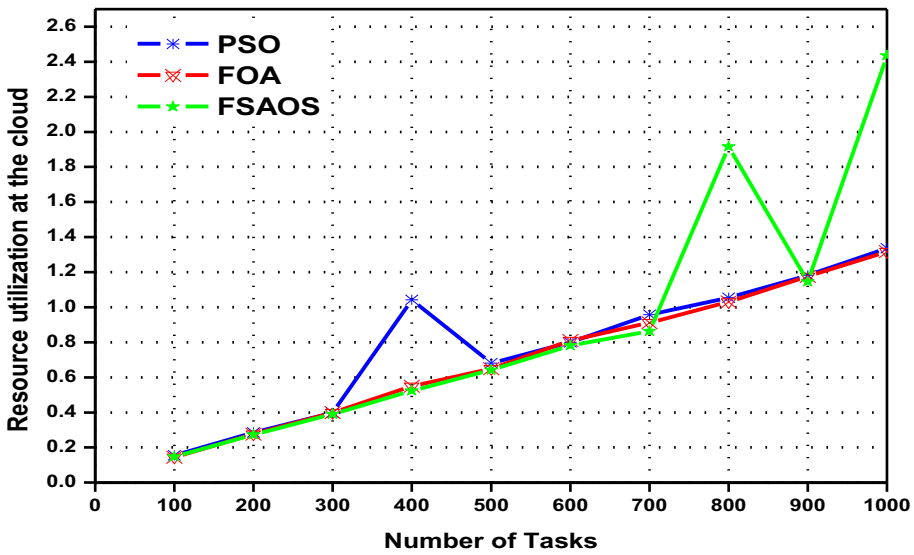

Fig. 8 Resource utilization on cloud

6.5 Statistical analysis on confidence interval

To further elaborate on the performance of our developed scheme, a 95% confidential interval based on execution time for the 10 independent simulation runs is computed. The computed values are derived using Eq. 29 [52], and results are shown in Table 11. These results based on 95% confidence interval are used to examine whether the execution time obtained by FSAOS is significantly less compared to FOA and PSO for all application instances.

$$\text{Confidential Interval (CI)} = \overline{X} \pm Z' S \Big/ \sqrt{N} \tag{29}$$

where $\overline{X}$ is the mean; Z' represents the distribution from the standard normal distribution; S represents the standard deviation of the sample data derived after running task instances on a virtual machine, and N represents the number of generated task sizes.

As can be seen from Table 11, the FSAOS has (9156.17, 9413.95) confidence levels at the edge and (975.87, 1205.37) at the cloud. On the other hand, FOA has achieved (9595.72, 9727.26) confidence levels at the edge, while it achieves (1066.94, 1301.96) at the cloud computing environments. Finally, the PSO has (9607.96, 9799.64) confidential interval at the edge cloud, while it achieved (1152.03, 1585.33). The smaller the value of the

Table 11 The parameter setting for the three resource scheduling schemes

	PSO	FOA	FSAOS
Total population size	1000	1000	1000
Degree of freedom	9	9	9
Confidence level	0.025	0.025	0.025
Z^*-distribution	1.96	1.96	1.96
Mean on edge/cloud	9703.80/1368.68	9961.99/1184.45	9285.06/1090.62
Standard Deviation on edge/cloud	488.98/1105.34	333.03/599.54	657.62/585.45
Lower bound on edge/cloud	9607.96/1152.03	9595.72/1066.94	9156.17/975.87
Upper bound on edge/cloud	9799.64/1585.33	9727.26/1301.96	9413.95/1205.37
95% Confidential Interval on edge/cloud	(9607.96, 9799.64)/(1152.03, 1585.33)	(9595.72, 9727.26)/(1066.94, 1301.96)	(9156.17,9413.95)/(975.87,1205.37)

confidence intervals, the more the acceptable results. The statistical analysis indicated that for 95% confidential intervals, the computed value of the FSAOS is less in both upper and lower bounds on edge and cloud environment, respectively, compared to the values obtained by FOA and PSO. These means that there is a difference between the performance of an FSAOS and the benchmarked schemes in terms of execution time. Although the smaller estimate of the CI is an indication of a significant performance in terms of execution time, this shows that FSAOS can achieve minimum execution time.

Moreover, to further show the statistical significance of the proposed resource scheduling algorithm, an H-test was carried out to examine whether a significant difference in terms of makespan obtained by the three resource scheduling algorithms exists for all task instances. The Kruskal–Wallis H-test procedures were adopted as a non-parametric test for comparing these algorithms, and the H value is computed using Eq. 30 [53].

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \tag{30}$$

where N = number of values obtained from every grouped sample, R_i^2 = summation of ranks taken from a particular sample and n_i = number of values from the equivalent sum of rank. The values for each of the scheduling algorithm (PSO, FOA and FSAOS) in Table 4 were ranked using the rank average function and results shown in Table 12.

The ranks on the statistical analysis of performance of PSO, FOA and FSAOS under different data instances as shown in Table 12 indicate that the sum of rank obtained by FSAOS is significantly less compared to PSO and FOA. This shows that there is a difference between the sum of rank (121) obtained by the proposed FSAOS in terms of achieving minimum makespan time compared to the PSO and FOA, which obtained 183 and 161, respectively. To compute the H value according to Eq. 30, data in Table 13 above were used. The alternative hypothesis (H_a) was set as the statement of the test to determine whether there is no

Table 12 Average rank obtained for the three resources scheduling algorithms

Tasks instances	PSO	FOA	FSAOS
100	2	3	1
200	8	9	4
300	12	10	5.5
400	13	14	5.5
500	17	15	7
600	21	16	11
700	23	19	18
800	28	22	20
900	29	26	24
1000	30	27	25
Sum of Rank	**183**	**161**	**121**

Bold values indicate the sum of the ranks in all the scheduled tasks (100–1000) instances for the resource scheduling algorithms to determine if an obtained rank is significantly less with respect to another

Table 13 Data derived from the resource scheduling algorithms

NI	Count samples in PSO	10
N2	Count samples in FOA	10
N3	Count samples in FSAOS	10
N	Total number of samples	30
K	Number of groups	3
Df	Degree of freedom(K−1)	2
α	Significance	0.05

difference between ranks of the three resource scheduling algorithms, while the null hypothesis (H_0) was set as the complementary statement to determine whether the difference between the ranks of the three resource scheduling algorithms in term of makespan time exists. The minimum $p-$value (also called the significance) was set at 0.05 for the same stopping criteria at 95% confidence level, and the H value obtained was 2.5497 and the calculated $p-$value is 0.27948 based on tasks instances (100–1000). This value is greater than the minimum $p-$value (0.05), which indicates that there is a significant difference between the performance of PSO, FOA and FSAOS for these data instances. This leads to acceptance of H_a for data instances 100 through 1000. This means as the tasks instances keep increasing, makespan obtained by FSAOS is significantly less than that of PSO FOA using same 95% confidence level. It can be concluded that FSAOS outperforms PSO and FOA as well as the state-of-the art approaches (shown in Table 10) as the search space becomes larger.

7 Conclusion

This article focuses on resource allocation in MECs, by considering an edge resource provider that offers heterogeneous resources to the mobile consumers in close proximity based on their computing demand. To improve resource utilization of the provider while meeting mobile consumers' quality-of-service expectations, we have proposed a fruit fly-based simulated annealing optimization scheme (FSAOS) for resource scheduling in MECs. The FSAOS can overcome premature convergence (which is attributed to conventional FOA), to guarantee market-oriented characteristic of mobile edge clouds in terms of minimum makespan, costs and resource utilization. A series of simulation with FSAOS over some benchmarked schemes shows that our proposed FSAOS can ensure profits and revenue gain through the minimization of the makespan and overall execution costs, and efficient resource utilization. Further research is to look at multi-domain resource allocation where resource providers coexist to share resources with individual resource provider having monopoly over his/her resource cost.

Funding Open access funding provided by Umea University.

Declarations

Conflict of interest There is no conflict of interest for this manuscript. All authors agreed to its submission.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visithttp://creativecommons.org/licenses/by/4.0/.

References

1. Lin Q (2021) Dynamic resource allocation strategy in mobile edge cloud computing environment. Hindawi Mobile Inf Syst 2021:10. https://doi.org/10.1155/2021/8381998
2. You P-S, Lee C-C, Hsieh Y-C (2011) Bandwidth allocation and pricing problem for a duopoly market. Yugoslav J Oper Res 21(1):65–78
3. Wang C, Liang C, Yu RF, Chen Q, Tang L (2017) Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans Wirel Commun 16(8):4924–4938
4. Hung PP, Huh N-E (2015) An adaptive procedure for task scheduling optimization in mobile cloud computing. Hindawi Publ Corp Math Probl Eng 2015:1–13
5. Arshad H, Khattak AH, Shah AM, Abbas A, Ameer Z (2018) Evaluation and analysis of bio-inspired optimization techniques for bill estimation in fog computing. Int J Adv Comput Sci Appl 9(7):191–198
6. Ullah A, Dagdeviren H, Ariyattu CR, DesLauriers J, Kiss T, Bowden J (2021) MiCADO-edge: towards an application-level orchestrator for the cloud-to-edge computing continuum. J Grid Comput 19:47. https://doi.org/10.1007/s10723-021-09589-5
7. Johansson K (2007) Cost effective strategies for heterogeneous wireless networks. PhD Thesis. KTH Information and Communication Technology, Stockholm, Sweden.
8. Sardellitti S, Scutari G, Barbarossa S (2015) Joint optimization of radio and computational resources for Multicell mobile-edge computing. IEEE Trans Signal Inf Process Over Netw 1(2):89–103
9. Shabeera TP, Kumar MDS, Salam MS, Krishnan MK (2017) Optimizing VM allocation and data placement for data-intensive application in cloud using ACO metaheuristic algorithm. Eng Sci Technol Int J 20(1):616–628
10. Zhou B, Buyya R (2018) Augmentation techniques for mobile cloud computing: a taxonomy, survey, and future directions. ACM Comput Surv 51(1):1–38
11. Liu J, Zhu L (2021) Joint resource allocation optimization of wireless sensor network based on edge computing. Hindawi Complex 2021:11. https://doi.org/10.1155/2021/5556651
12. Wang X, Sui Y, Wang J, Yuen C, Wu W (2018) A distributed truthful auction mechanism for task allocation in mobile cloud computing. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2018.2818147
13. Jin L-A, Song W, Zhuang W (2018) Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Trans Emerg Top Comput 6(1):45–57
14. Zhao T, Zhou S, Guo X, Niu, Z (2017) Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In: Proceedings of the 2017 IEEE

international conference on communications (ICC). 21–25 May, Paris, France pp 1–7
15. Fang J, Hu J, We J, Liu T, Wang B (2020) An efficient resource allocation strategy for edge-computing based environmental monitoring system. Sensors 20:6125. https://doi.org/10.3390/s20216125
16. Mishra KS, Puthal D, Rodrigues CPJJ, Sahoo B, Dutkiewicz E (2018) Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. IEEE Trans Ind Inf 14(10):4407–4506
17. Chen H, An B, Niyato D, Soh CY, Miao C (2017) Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks. IEEE J Sel Areas Commun 30(3):557–570
18. Chen J, Du T, Xiao G (2021) A multi-objective optimization for resource allocation of emergent demands in cloud computing. J Cloud Comput Adv Syst Appl 10:20. https://doi.org/10.1186/s13677-021-0237-7
19. Zhu Z, Peng J, Gu X, Li H, Liu K, Zhou Z, Liu W (2018) Fair resource allocation for system throughput maximization in mobile edge computing. IEEE Access 6(1):5332–5340
20. Wei Z, Jiang H (2018) Optimal offloading in fog computing systems with non-orthogonal multiple access. IEEE Access 6(1):49767–49778
21. Yang N, Fan X, Puthal D, He X, Nanda P, Guo S (2018) A novel collaborative task offloading scheme for secure and sustainable mobile cloudlet networks. IEEE Access 6(1):44175–44189
22. Tärneberg W, Mehta A, Wadbro E, Tordsson J, Eker J, Kihl M, Elmroth E (2017) Dynamic application placement in the Mobile Cloud Network. Future Gener Comput Syst 70(1):163–177
23. Iscan H, Gunduz M (2015) A survey on fruit fly optimization algorithm. In: Proceedings of the 11th international conference on signal-image technology and internet-based systems. pp 520–527
24. Gabi D, Ismail AS, Zainal A, Zakaria Z, Al-Khasawneh A (2018) Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment. J Inf Commun Technol 17(3):435–467
25. Jin A-L, Song W, Zhuang W (2015) Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Trans Emerg Top Comput 6(1):45–57
26. Lu W, Wu W, Xu J, Zhao P, Yang D, Xu L (2022) Auction design for cross-edge task offloading in heterogeneous mobile edge clouds. Comput Commun 181(2022):90–101
27. Singh H, Bhasin A, Kaveri RP (2020) QRAS: efficient resource allocation for task scheduling in cloud computing. SN Appl Sci 3:474. https://doi.org/10.1007/s42452-021-04489-5
28. Nguyen TD, Le BL, Bhargava V (2018) Price-based resource allocation for edge computing: a market equilibrium approach. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2018.2844379
29. Zhu X, Zhang Z, Wang Y, Wang G (2018) Resource allocation based on reverse auction algorithm in edge computing environment. In: X Sun et al. (Eds.), Proceedings of the 4th international conference. LNCS 11065: Springer Nature Switzerland pp 245–252
30. Li L, Deng N, Ren W, Kou B, Zhou W, Yu S (2018) Multi-service resource allocation in future network with wireless virtualization. IEEE Access 6(1):53854–53868
31. Wang X, Wang K, Wu S, Di S, Jin H, Yang K, Ou S (2018) Dynamic resource scheduling in mobile edge cloud with cloud radio access network. IEEE Trans Parallel Distrib Syst 29(11):2429–2445
32. Madhusudhan HS, Kumar TS, Mustapha SMFD, Gupta P, Tripathi PR (2021) Hybrid approach for resource allocation in cloud infrastructure using random forest and genetic algorithm. Hindawi Sci Program 2021:1–10. https://doi.org/10.1155/2021/4924708
33. Chen J, Wang Y, Liu T (2021) A proactive resource allocation method based on adaptive prediction of resource requests in cloud computing. J Wirel Commun Netw 2021:24. https://doi.org/10.1186/s13638-021-01912-8
34. Yu H (2020) Evaluation of cloud computing resource scheduling based on improved optimization algorithm. Complex Intell Syst 7:1817–1822. https://doi.org/10.1007/s40747-020-00163-2
35. Nabi S, Ahmad M, Ibrahim M, Hamam H (2022) AdPSO: adaptive PSO-based task scheduling approach for cloud computing. Sensors 22:920. https://doi.org/10.3390/s22030920
36. Yin B, Cheng Y, Cai XL, Cao X (2017) Online SLA-aware multi-resource allocation for deadline sensitive jobs in edge-clouds. In: Proceedings of the 2017 IEEE global communications conference. 4–8 December, Singapore, Singapore pp 1–6
37. Gabi D, Ismail AS, Zainal A, Zalmiyah Z (2019) Quality of service (QoS) task scheduling for time-cost trade-off scheduling problem in cloud computing environment. Int J Intell Syst Technol Appl 18(5):448–469
38. Balouek-Thomert D, Renard GE, Zamani RA, Simonet A, Parashar M (2019) Towards a computing continuum: enabling edge-to-cloud integration for data-driven workflows. Int J High Perform Comput Appl 33(6):1–14
39. Gabi D, Ismail AS, Zainal A, Zalmiyah Z (2017) Scalability-aware scheduling optimization algorithm for multi-objective cloud task scheduling problem. In: Proceedings of the 2017 6th ICT-international student project (ICT-ISPC-2017). Faculty of Computing, Universiti Teknologi Malaysia. 23–24 May, Malaysia pp 1–6
40. Bendechache M, Svorobej S, Endo TP, Lynn T (2020) Simulating resource management across the cloud-to-thing continuum: a survey and future directions future internet. 12(95)
41. Gabi D, Ismail AS, Zainal A, Zakaria Z, Abraham A (2018) Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Comput Appl 30(6):1845–1863
42. Ramachandra A, Guruprasad A (2020) Resource provisioning techniques in cloud/edge computing. Int J Ser Multidiscip Res Arch Comput Eng 2(2020):1–11
43. Zahoor S, Javaid S, Javaid N, Ashraf M, Ishmanov F, Afzal KM (2018) Cloud–fog–based smart grid model for efficient resource management. Sustainability 10(2079):1–21
44. Gabi D, Ismail AS, Zainal A, Zakaria Z, Dankolo NM, Abraham A (2020) Cloud customers service selection scheme based on improved conventional cat swarm optimization. Neural Comput Appl 32(18):14817–14838
45. Toczé K, Nadjm-Tehrani S (2018) A taxonomy for management and optimization of multiple resources in edge computing. Wirel Commun Mob Comput 2018:1–23
46. Sediq BA, Gohary HR, Yanikomeroglu H (2012) Optimal tradeoff between efficiency and Jain's fairness index in resource allocation. In: Proceedings of the 2012 IEEE 23rd international symposium on personal, indoor and mobile radio communications (PIMRC). 9–12 September. Sydney, NSW, Australia, pp 577–583
47. Liu M, Liu Y (2018) Price-based distributed offloading for mobile-edge computing with computation capacity constraints. IEEE Wirel Commun Lett 7(3):420–423
48. Deng R, Lu R, Lai C, Luan HT, Liang H (2016) Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J 3(6):1171–1181
49. Shan D, Cao GH, Dong HJ (2013) LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization

problems. Math Probl Eng 2013:1–9. https://doi.org/10.1155/2013/108768
50. Sonmez C, Ozgovde A, Ersoy C (2017) EdgeCloudSim: an environment for performance evaluation of edge computing systems. In: Proceedings of the second international conference on fog and mobile edge computing (FMEC). 8–11 May 2017. Valencia, Spain, pp 39–44
51. Madni HHS, Abd Latiff SM, Coulibaly Y, Abdulhamid MS (2017) Recent advancement in resource allocation techniques for cloud computing environment: a systematic review. Clust Comput 20(1):2489–2533
52. Hosmer DW, Lemeshow S (2015) Confidence interval estimation of interaction. Epidemiology 3(5):452–456
53. Vianee B, Girish J, Lovena N, Krisen P, Veldy MMAE (2021) Project on non-parametric test. Access online at: https://www.researchgate.net/publication/323546900, 2021

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Danlami Gabi[1,2] · Nasiru Muhammad Dankolo[2] · Abubakar Atiku Muslim[2] · Ajith Abraham[3] · Muhammad Usman Joda[4] · Anazida Zainal[5] · Zalmiyah Zakaria[5]

✉ Danlami Gabi
danlami@cs.umu.se

Nasiru Muhammad Dankolo
nasirdankolo@gmail.com

Abubakar Atiku Muslim
alatiku@gmail.com

Ajith Abraham
ajith.abraham@ieee.org

Muhammad Usman Joda
umjoda@gmail.com

Anazida Zainal
anazida@utm.my

Zalmiyah Zakaria
zalmiyah@utm.my

[1] Department of Computing Science, Umeå University, Umeå, Sweden

[2] Department of Computer Science, Kebbi State University of Science and Technology, Aliero, Nigeria

[3] Machine Intelligence Research Labs, Scientific Network for Innovation and Research Excellence, Auburn, WA 98071, USA

[4] Department of Mathematical Sciences, Bauchi State University Gadau, Bauchi 81007, Nigeria

[5] Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia