An efficient population-based multi-objective task scheduling approach in fog computing systems

Zahra Movahedi, Bruno Defude, Amir Mohammad Hosseininia

Movahedi *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:53
https://doi.org/10.1186/s13677-021-00264-4

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

An efficient population-based multi-objective task scheduling approach in fog computing systems

Zahra Movahedi[1*], Bruno Defude[2] and Amir mohammad Hosseinia[1]

Abstract

With the rapid development of Internet of Things (IoT) technologies, fog computing has emerged as an extension to the cloud computing that relies on fog nodes with distributed resources at the edge of network. Fog nodes offer computing and storage resources opportunities to resource-less IoT devices which are not capable to support IoT applications with computation-intensive requirements. Furthermore, the closeness of fog nodes to IoT devices satisfies the low-latency requirements of IoT applications. However, due to the high IoT task offloading requests and fog resource limitations, providing an optimal task scheduling solution that considers a number of quality metrics is essential. In this paper, we address the task scheduling problem with the aim of optimizing the time and energy consumption as two QoS parameters in the fog context. First, we present a fog-based architecture for handling the task scheduling requests to provide the optimal solutions. Second, we formulate the task scheduling problem as an Integer Linear Programming (ILP) optimization model considering both time and fog energy consumption. Finally, we propose an advanced approach called Opposition-based Chaotic Whale Optimization Algorithm (OppoCWOA) to enhance the performance of the original WOA for solving the modelled task scheduling problem in a timely manner. The efficiency of the proposed OppoCWOA is shown by providing extensive simulations and comparisons with the original WOA and some existing meta-heuristic algorithms such as Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA).

Keywords: Fog computing, Task scheduling, Internet of things, Meta-heuristic, Whale optimization algorithm, Opposition-based learning, Chaos theory

Introduction

Over recent years, the Internet of Things (IoT) has been integrated in our daily lives, which make the use of IoT applications (access control or face recognition for instance) in the context of smart city [1, 2], smart healthcare [3], smart home [4], etc., more and more popular. Fog computing paradigm, as an extension to cloud computing, plays a crucial role in executing the IoT applications. in this hierarchical model, the system designer envisages the deployment of fog nodes with a certain computational and storage resources at the edge of network, between the IoT and cloud layer (see Fig. 1).

The fog layer decreases the delay related to transmission of IoT devices' data to/from the cloud layer and thus enabling IoT applications to benefit from the proximity of fog resources as infrastructure for offloading and executing their IoT tasks in real-time.

Besides, IoT tasks possess different computational and storage constraints (e.g., the required cpu, memory, and deadline) for being completed. Moreover, there are different QoS parameters (e.g., delay, energy) that should be taken into account when offloading the IoT tasks to the fog or cloud resources. These parameters are important not only from the end-users IoT applications' point of view,

*Correspondence: zmovahedi@ut.ac.ir
[1]Department of Engineering, College of Farabi, University of Tehran, Tehran, Iran
Full list of author information is available at the end of the article

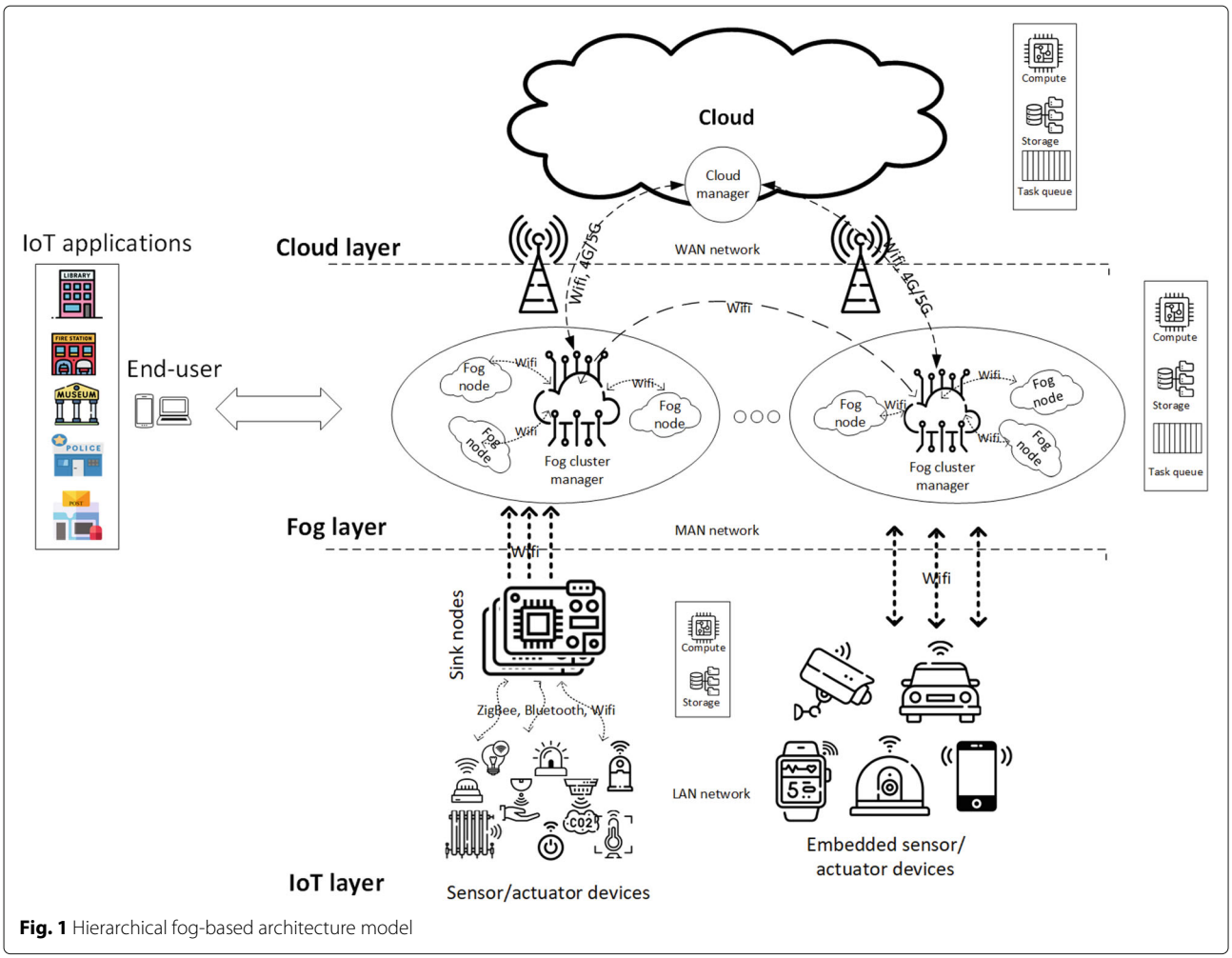


Fig. 1 Hierarchical fog-based architecture model

but also from the view of system designer. For instance, the real-time property is critical as a QoS parameter that concerns the end-users of IoT applications; or, energy consumption of fog nodes is a QoS parameters that matters the system designer.

The process of selecting appropriate fog or cloud resources for allocating IoT tasks under certain constraints corresponds to the optimization problem of task scheduling. The aim is to design a scheduler for the dynamic incoming IoT tasks taking into account multiple objectives for the optimization of QoS parameters for both end-user and system designer. In such context, the task scheduling optimization problem can be formulated as an Integer Linear Programming (ILP) [5] which belongs to the complexity class of NP-hard [6]. It should be noted that in such problems, finding an optimal solution within a polynomial computational time is not possible with conventional mathematical methods.

In order to tackle the task scheduling problem in a timely manner, population-based meta-heuristic algorithms have been demonstrated as effective optimization techniques, especially when facing to NP-hard problems. In these algorithms, an initial population is randomly generated using a uniform distribution which is hopefully converged to the (near) optimal solution in further generations using some defined operators. Exploration (global search) and exploitation (local search) of the search space are the two main phases that should maintain balanced to find the optimal solution. At the beginning, we should consider a diverse population in order to explore all the directions in the search space for determining the promising regions. Next, we should be able to exploit the promising regions in order to find the global optimal solution. However, the transition between exploration and exploitation is not always easy, especially when the population stuck into the local optima at some stage and thus premature convergence may occur.

The task scheduling problem has already been discussed by many researchers in different heterogeneous computing systems [7–11]. However, the literature lacks adequate specialized solutions addressing the task scheduling problem in the context of fog computing when exponential

IoT tasks are generated dynamically. Actually, the search space of IoT and fog environment is more complex than other computing systems due to the different constraints of IoT tasks and fog resources. Although the complexity of task scheduling is significant, the IoT tasks are not always delay-tolerant. Consequently, the task scheduling architecture should enable IoT applications to be handled in short-delay while bypassing the complexity.

In this paper, we tackle the task scheduling problem in fog environment focusing on Whale Optimization Algorithm (WOA) [12], a recently developed population-based meta-heuristic approach that is characterized by its simplicity and well convergence rate. However, randomness aspect in the initial population and within the design parameters affects the effectiveness of this algorithm. In order to cope with this issue and according to the "No free lunch theorem [13]", we enhance the original WOA by hybridizing with opposition-based learning [14, 15] and chaos theory [16]. The former approach maintains the population diversity when exploring the search space by expanding the search in both directions (random and its opposite direction), and the latter approach preserves a proper transition between exploration and exploitation of search space. In principle, the chaos theory is considered in non-linear deterministic systems where pseudo-random behaviour is generated. The basic characteristics of chaos is dynamicity (sensitive dependence on initial conditions), pseudo-randomicity, and ergodicity. Dynamicity results in providing diverse outputs with small differences in the initial values, and ergodicity enhances the speed of convergence. The chaotic behaviour is modelled as a sequence of pseudo-random periodic values given from discrete-time function (so called chaotic map). In the domain of optimization algorithms, the chaotic sequence from chaotic map is used to replace the random sequence from the uniform distribution. In the consequence, the diversity of population is ensured and the stagnation in local optima and premature convergence can be avoided.

The contributions of this paper are summarized below:

1 We investigate a hierarchical cloud and fog-based architecture, which is capable to schedule both real time and computational-intensive IoT tasks. Real-time tasks are processed in the fog layer in order to minimize the data transmission time while computational-intensive tasks are processed in the cloud layer. The task scheduling process is performed in the fog layer closed to the end-user.
2 We use a classical formulation of task scheduling as an Integer Linear Programming (ILP) multi-objective optimization problem considering the selection of appropriate fog nodes under certain constraints with the aim of minimizing both task completion time and energy consumption as two QoS parameters in a fog computing system. To solve this optimization problem, we design a new approach called OppoCWOA, which is a combination of WOA, Opposition-based learning, and Chaos theory. The opposition-based learning is used to inject the diversity into the initialization phase of WOA. In this paper, we introduce four types of opposition-based learning named total, partial, quasi and super opposition-based learning. We employ also the jumping rate to decide the probability of Opposition-based approach occurrence over time that guarantee balance between convergence speed and success rate of opposition-based approach. The chaos theory is integrated to WOA to avoid the impact of randomness in movement towards the optimal solution. This results in improving the convergence speed.
3 We demonstrate the effectiveness of OppoCWOA for our task scheduling problem by providing extensive experiments and comparisons with the original WOA and some other existing population-based meta-heuristic approaches such as Genetic Algorithms, and Artificial bee Colony algorithm (ABC), and Particle Swarm Optimization (PSO).

The reminder of this paper is structured as follows. In "Related work" section, we present some existing works in the context of task scheduling in cloud and fog computing. The system model and the problem formulation is described in "Proposed system model and formulation for the task scheduling problem" section. We illustrate the proposed OppoCWOA approach to solve the formulated problem in "Proposed approach" section. Experimentation and comparison results are provided in "Experiment results" section. Finally, the conclusion and the future works are presented in "Conclusion and future works" section.

Related work

Since the development of cloud/fog model in IoT environment, many efforts have been performed in order to optimally assign the IoT tasks to the fog/cloud resources with minimal computational time complexity. In this context, a number of works surveyed the scheduling and offloading techniques in IoT/fog systems from the perspective of system model architecture and optimal/near optimal solutions for the task scheduling problem [7–9, 17, 18].

Wang et al. [19] proposed an optimal scheduling algorithm for minimizing job completion time (makespan) and load variance of cloud nodes. Their approach is based on GA that applies a greedy initialization and uses a double-fitness adaptive mechanism to update the population in each iteration. Authors in [20] modelled the task scheduling problem considering makespan minimization.

They applied and compared GA, PSO and a modified PSO to solve the modelled problem. The proposed modified PSO (MPSO) enhances the convergence of the original PSO by using the Smallest Job to the Fastest Processor (SJFP) approach to initialize the population. Zaki Hasan et al. [21] described a robust Canonical PSO (CPSO) and fully-informed particle swarm (FIPS) algorithms for task scheduling problem in both heterogeneous and homogeneous cloud environment to optimize throughput and delay. The author compared their proposed CPSO with traditional adaptive GA. Kimpan et al. [22] considered the task scheduling and load balancing among virtual machines in dynamic cloud environments with the aim of task makespan minimizing and load balancing of virtual machines. They improved the ABC algorithm with three basic scheduling algorithms that are Shortest Job First (SJF), First Come First Serve (FCFS), and Longest Job First (LJF) to solve the problem. Chen et al. [23] proposed to model the task scheduling in cloud computing as a multi-objective optimization problem for time cost, load cost, and price resource cost. They enhanced the accuracy and convergence speed of original WOA by describing a self-adaptive mechanism for handling the population size based on the logistic model and a non-linear function for better convergence. Workflow scheduling in cloud environments is studied by Thennarasu et al. [24]. The authors proposed a new framework based on WOA to optimize makespan, deadline and resource utilization.

Recently, with the emergence of fog computing, many studies have been done on the task scheduling problem to adapt cloud based scheduling algorithms to this new paradigm.

Ghobaei-Arani et al. [25] described the task scheduling problem for cyber-physical system (CPS) in the context of fog environment. They considered minimizing the task execution time, transfer time and makespan as QoS metrics and proposed an approach based on the moth-flame optimization algorithm. Their approach eliminated the irrelevant solutions in the early steps of the algorithm, which leads to faster convergence. The scheduling algorithm proposed by Rahbari et al. [26] described a knapsack algorithm optimized by symbiotic organisms search (SOS) in the fog context to optimize execution cost, energy consumption, and network usage. Their proposed algorithm is compared to FCFS and the original knapsack considering two case studies for intelligent monitoring in smart environment and smart home. Wang et al. [27] proposed a task scheduling strategy in the fog environment for a smart production line scenario. A hybrid heuristic (HH) algorithm, which is a combination of Improved PSO (IPSO) and Improved Ant COlony (IACO) is proposed. Simulation is carried out by MATLAB and experimented for various tasks from 50 to 300 on ten fog nodes and proved the proposed HH algorithm outperforms IPSO, IACO, and round-robin (RR) in terms of completion time, energy consumption, and reliability. A cluster-based fog system model is proposed by Sun et al. [28] considering the task scheduling in two-level that is among different fog clusters and among fog nodes within the same fog cluster. They modified the improved NSGA-II by establishing a new crowding distance formula to resolve the task scheduling problem. Service latency and stability are two objectives in this work that are normalized by employing the Simple Additive Weighting technique. Yang et al. [29] formulated the collaborative task scheduling problem to study the balance between energy consumption and service delay in homogeneous fog networks. In accordance with their formulation, they proposed a new algorithm by applying Lyapunov optimization techniques in dynamic fog networks. Random scheduling and Least Busy Scheduling are used as comparison algorithms. Wang et al. [30] proposed an efficient cooperating multi-tasks scheduling algorithm focused on ACO approach in fog environments. The proposed algorithm experimented for various tasks from 40 to 100 on 20 fog nodes. It proved to outperform min-min, improved max-min, and FCFS algorithm in terms of energy consumption, total completion task, and resource consumption. Task scheduling problem in cloud-fog context is described by Nguyen et al. [31] with two objectives: execution time and operating costs. Their proposed algorithm named TCaS is based on GA and is evaluated on 11 datasets of different sizes. Comparison is accomplished with BLA and Modified PSO (MPSO) using iFogSim simulator. Bitam et al. [32] addressed the task scheduling in the context of fog environment considering CPU execution time and allocated memory. The authors proposed a meta-heuristic algorithm called Bees Life Algorithm (BLA) that mimics marriage and food foraging behaviour of bees to resolve the scheduling problem. Simulation are performed by the BLA framework in C++ with 25 tasks on 20 fog nodes and compared with GA and PSO algorithms.

The authors in [33] addressed the task scheduling problem with the objective of minimizing the temperature of Cloud Data Centers (CDC). The authors defined a thermal profile based on different tasks and Cloud host parameters such as cpu, memory, storage, and bandwidth. The optimal value of these parameters is calculated using genetic-based Expectation maximization method of Gaussian Mixture Model (GMM). They considered an IoT-based Smart Home Application called ROUTER in the environment of iFogSim and ThermoSim and evaluated their approach for the temperature reduction of CDC and other QoS parameters such as latency, energy, and bandwidth. However, the details of their model formulation are not presented in this paper. In [34], the authors considered the IoT task scheduling problem in stochastic Edge-Cloud

Computing Environments with dynamic workloads. In each scheduling interval, new arrival tasks and remaining active tasks from the previous interval should be designed by the scheduler to be allocated or migrated to a priority list of hosts based on the state of the system. Their objective is to model a task scheduler that is optimal in terms of loss metric which is a convex combination of average energy consumption, average response time, average migration time, average SLA violations, and cost. The scheduler is based on asynchronous advantage actor critic and residual recurrent neural network for updating model parameters in each interval to quickly adapt to dynamic scenarios. Computation offloading problem is addressed in [35]. The authors formulated the problem as a partially observable Markov decision process considering imperfect knowledge on the dynamicity of channel state and task queue state for offloading decision in each time slot. Their objective is to minimize the average energy consumption of IoT devices along with minimizing the average response delay. They solved the formulated problem using a deep Reinforcement Learning approach (deep Q-learning with a recurrent convolutional neural network) for learning the optimal decision action based on the previous observation-actions. The authors in [36] considered joint Computation Offloading and Scheduling Optimization problem in fog environment. A gateway is considered between the hierarchical IoT layer and the fog/cloud layer that is responsible for the scheduling strategy and offloading decision. Their objective is to minimize the expected time-energy consumption for executing priority-based tasks. To solve the designed problem, they first assigned a priority to the arrival tasks in each time slot. Authors then used the Lyapunov drift-plus-penalty optimization technique which determine queue state and stability for an optimal scheduling policy and task offloading decision.

Mobility is also an important factor of IoT environments and several works recently addressed task-scheduling with mobile IoT devices [37–39]. In fact, IoT devices are frequently mobile (e.g smartphones, embedded IoT devices on vehicles, ...) and it introduces new problems such as disconnections, changes on gateways, task migration. Mobi-IoST (Mobility-aware Internet of Spatial Things) [39] considers time-critical applications such as health care applications. In such applications, disconnections may increase delay in processing and delivering information. Mobi-IoST proposes a Cloud-Fog-Edge based collaborative framework for the processing of IoT data and delivering the result based on user mobility prediction to reduce the delay and the power consumption of the IoT device. Spatio-temporal trajectories of end-users (GPS traces) are stored at the cloud level. These trajectories are semantically enriched using external information such as points of interest, road network. These enriched traces are analyzed using a hidden-markov model to compute real-time predictions of end-user location sequences. Nevertheless, Mobi-IoST does not consider task scheduling but focus on selecting the best communication path between processing nodes and the IoT device. MAGA [37], a mobility-aware offloading decision strategy for distributed cloud computing, uses a mobile access prediction method to estimate cloudlet reliability. The mobile access prediction method is based on the regularity of human mobility. They use a tail matching sub-sequence algorithm to predict the next access point based on the history of mobility. Offloading decision is computed by an integer encoding-based adaptive genetic algorithm (GA) which find the optimal solution taking cloudlet reliability, computation requirements, and mobile device energy consumption into consideration. Experiment results showed that MAGA could improve offloading success rate and decrease energy consumption at mobile devices. In [38], movement of IoT devices is also predicted in order to dynamically change associated IoT base station but also task placement. For that, they propose a novel dynamic mobility-aware partial offloading (DMPO) decision scheme, optimizing both the offloading proportion and the communication path of offloading during IoT device's movement. Simulations show that the algorithm decreases energy consumption while satisfying delay constraint.

Our paper focuses on task scheduling problem and mobility is not our concern at this step. Nevertheless, our optimization method may be used in conjunction with some prediction method to address mobility issues.

Table 1 compares the different works in terms of optimization method and optimization criteria, of architecture (cloud based or cloud and fog based) and mobility. Most of the works address the problem of task scheduling in cloud-fog architectures using different meta heuristics methods and optimizing different objectives (makespan, delay and energy consumption are generally considered). All these works show that task scheduling is a very complex problem that is addressed using many different optimization methods. Recent works addresses the problem of mobility and its impact on energy consumption and delay. They focus on the definition of accurate prediction model but not specifically on task scheduling. In our work, we want to address the problem of task scheduling of IoT tasks in a fog and cloud hierarchical architecture. We focus on the exchange of data between the different layers of the architecture and the computing resources. Due to the space complexity of IoT tasks, the optimization algorithms should control the exploration and exploitation steps to guarantee good efficiency in terms of premature convergence and population diversity. For that, we propose some improvements to the WOA optimization algorithm which has recently caught many attention in the context

Table 1 Task scheduling in fog and cloud computing

Ref.	Cloud/ Fog	Optimization method	Optimization criteria	Mobility
[19]	Cloud	GA	Makespan and load balancing	No
[20]	Cloud	GA and PSO + (SJFP)	Makespan	No
[21]	Cloud	Canonical PSO and fully-informed PSO	Throughput and delay	No
[22]	Cloud	ABC + (FCFS, SJF, LJF)	Makespan and load balancing	No
[23]	Cloud	IWC (Improved WOA)	Execution time, system load, and price cost	No
[24]	Cloud	WOA	Makespan, deadline, and resource utilization	No
[25]	Fog	Moth-flame	Execution time, transfer time, and makespan	No
[26]	Fog	Knapsack + symbiotic organisms search	Energy consumption, execution cost, total network usage, and sensor lifetime	No
[27]	Fog	Hybrid heuristic (IACO + IPSO)	Completion time, energy consumption, and reliability	No
[28]	Fog	NSGA-II	Service latency and stability	No
[29]	Fog	Lyapunov optimization	Service delay and energy consumption	No
[30]	Fog	Ant Colony	Energy consumption, resource consumption, and total completion task	No
[31]	Fog	GA	Execution time and operating cost	No
[32]	Fog	Bees Life	Execution time and allocated memory	No
[33]	Fog	GA + Gaussian Mixture Model	Temperature of CDC, latency, energy, and bandwidth	No
[34]	Fog	Deep Reinforcement learning	Energy consumption, response time, migration time, SLA violations and cost	No
[35]	Fog	Deep Reinforcement learning	Average energy consumption of IoT device and response delay	No
[36]	Fog	Lyapunov optimization + drift-plus-penalty	Time-average and energy consumption	No
[37]	Fog	Adaptive GA	Offloading success rate and energy consumption	Yes
[38]	Fog	-	Energy consumption	Yes
[39]	Fog	-	Delay and power consumption	Yes
Our work	Fog	WOA+OB learning + chaos	Response delay and energy consumption	No

of optimization problem [40, 41] due to its simplicity of use.

Proposed system model and formulation for the task scheduling problem

In this section, we first provide a use case in the domain of smart city. We then specify the hierarchical architecture model for IoT task scheduling problem. The architecture is composed of three layers, namely IoT, fog and cloud layer, as illustrated in Fig. 1. Based on this architecture, we describe the process of handling IoT tasks' requests for scheduling purpose. Next, we describe the modelling of time and energy consumption as QoS parameters to be optimized in our system. Finally, we formulate the task scheduling problem using ILP model considering a set of constraints to minimize the time and energy consumption for task scheduling problem.

Use-case scenario

As a potential use case, one can consider a variety of scenarios in a smart city. For efficiency and network coverage issues, a city is usually divided into different regions (clusters). A great number of IoT devices such as RFID devices, motion detection devices, temperature devices, video surveillance, etc. are deployed in these regions. Based on these IoT devices, a smart city provides a set of smart services as web or mobile applications for citizen (e.g., Smart parking, smart video surveillance, etc.). These services are deployed on servers (fog nodes) that are distributed in the city. The servers that are located in a region are under the control of a specific node (called cluster manager). A citizen is connected to the nearest cluster manager via WLAN. Cluster managers are responsible to handle the citizens' requests under their coverage.

A first type of smart service is smart parking, allowing car drivers to get newly available parking slots in a specific area.

For that, different types of sensors and actuators such as RFID and motion detection devices are distributed within the parking lots to discover the information on the number and location of available places. These information are sent periodically to a server node for being stored. A smartphone application enables the citizens to discover the available parking places around their current location.

This IoT application is obviously sensitive in delay (the available parking places should be detected immediately after the departure of vehicles), but not in computation requirements (the task requires only simple instructions for detecting the occupied/available places via RFID readers). In this case, the fog cluster manager should assign the smart parking task to a fog node that is located in the vicinity of the citizen and is closed to the needed data.

Another type of smart service is a video surveillance system. Based on a specific demand (including location and time criteria), this system analyses videos acquired from several video cameras deployed within different parking slots in the region. The application requires an object recognition task which is very intensive in computation requirements, but is not sensitive in delay. In this case, the fog cluster manager should allocate the resources of a cloud node for executing the object recognition task. This implies the transfer of video data from video devices or fog nodes to the cloud.

Fog-based architecture for task scheduling problem

Figure 1 illustrates our proposed hierarchical fog-based architecture for the problem of task scheduling.

The *IoT layer* is composed of a massive number of IoT devices that are widely distributed within areas such as cities, factories, hospitals, homes, libraries, museums, etc... IoT devices comprise sensors and actuators (e.g., environmental sensors, switching actuators) that generate continuous raw data. This latter are either independent devices without computing or storage resources that only sense/act the surrounding environment or embedded into the mobile or fixed end devices with limited computing and storage resources (e.g., smartphones, smart cameras). In the first case, the data are forwarded to the fixed sink nodes (e.g., micro-controllers) through wired/wireless protocols (e.g., zigbee, bluetooth, wifi) in LAN network for data storing and further analysis. In the second case, end devices play the role of both IoT devices and sink nodes. Sink nodes are connected to the upper layer (fog or cloud layer) through wired/wireless channel (e.g., wifi) for sending the IoT device data related to the execution of IoT applications.

The *fog layer* is composed of fog nodes that are distributed in different areas close to the IoT devices and possess certain computing and storage resources as well as a task queue. Fog nodes that are located within an area are grouped and construct a cluster. Each cluster disposes a fog cluster manager that is aware of fog nodes within the cluster (fog clusters) and the IoT devices/sink nodes that are in proximity of the cluster. Moreover, the fog cluster managers know each others and can communicate together via wireless MAN network (e.g., wifi). The fog cluster manager is responsible for handling the optimal scheduling of IoT tasks. The process should ensure the optimal use of system resources in terms of time and energy consumption using the proposed optimization algorithm. We suppose that the grouping of fog nodes into clusters, the selection of one fog node as fog cluster managers, and the maintain of the related information about the clusters (such as the IP address of the different fog managers) are supported by appropriate distributed protocols. Furthermore, the fog cluster manager is assumed to dispose enough computational resources for executing the process of task scheduling.

The advantage of this layer is the deployment of fog resources in the vicinity of IoT devices that allows simple, low-latency and real-time responses regarding the delay-sensitive IoT applications.

The *cloud layer* encompasses high-performance computing and storage equipment (known as cloud nodes) such as cloud servers. The communication between this layer and the lower layers is controlled by a cloud manager. Cloud manager gathers all information on IoT and fog layer and handles occasionally the IoT tasks execution when the fog layer is not able to do so (because of large-scale or complex processing requirements, for instance). This layer cannot provide delay-sensitive services as fog layer due to its remote deployment from the IoT devices. However, it can provide high-quality services for IoT applications such as storage and analysing massive data for computing.

IoT applications are installed on the end-users devices and generate a set of independent IoT tasks, each one with specific computing and storage requirements. We consider two types of IoT task: 1) IoT tasks that are delay-sensitive and should be processed in real-time and, 2) IoT tasks that require intensive resource and should be processed by resource-rich devices.

Figure 2 describes the IoT tasks scheduling process. The end-users send their requests for executing the IoT applications to the nearest fog cluster manager (called initial fog cluster manager). Each request contains input (data size, data source type, location), output (result type), type (delay-sensitive with deadline or computation-intensive), the desired requirements (e.g., RAM: 1 MB, CPU: 100 MIPS, storage: 1 MB, bandwidth: 1 Mbps). It is worth mentioning that the requests are heterogeneous in terms of inputs, outputs, and requirements. If the data source for executing the IoT task is under the control of another fog cluster manager, the initial fog cluster manager will forward the end-user' requests to that one (called scheduler fog cluster manager). The received IoT tasks are then enqueued into the task queue of fog cluster manager for being scheduled then after.

The task scheduling process is based on a time division into equal time slots represented as $S = \{s_1, s_2, ..., s_q\}$. The time interval between two consecutive time slots is δs.

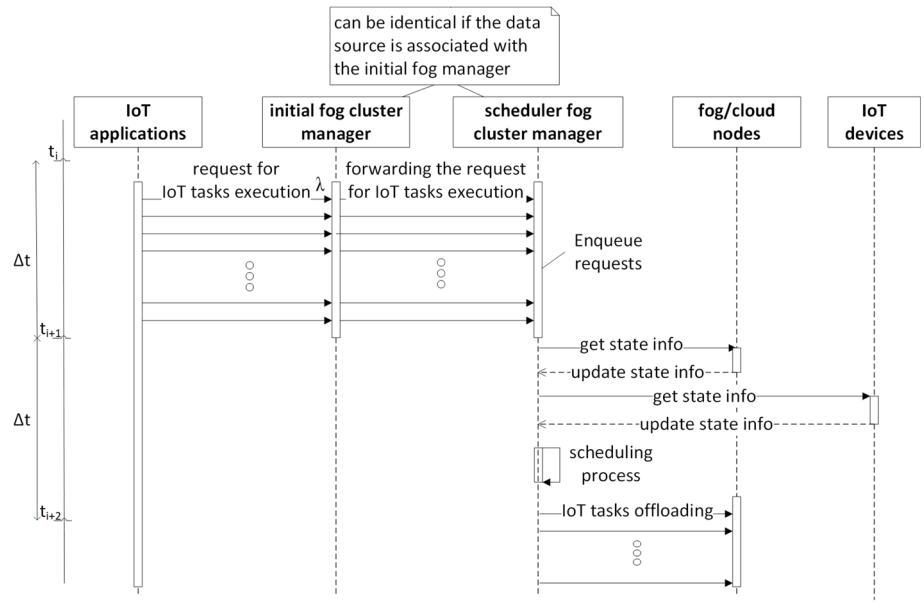


Fig. 2 The process of handling the IoT task requests for task scheduling

IoT tasks are arriving into the queue of fog nodes with an arrival rate λ that follows the Poisson distribution process [35]. The accumulated arriving tasks during δs are taken into account using a priority queue, i.e., the delay-sensitive tasks are set at the front of the queue.

At each time slot, the fog cluster manager is updated on the status of the present IoT devices and sink nodes as well as the fog clusters (i.e., remaining computation and storage capacity, remaining energy). Once the update is complete, the task scheduling algorithm starts over the existing IoT tasks in the fog cluster manager' queue to compute the optimal scheduling.

Actually, the decision for selecting and allocating appropriate fog or cloud resources is based on the type of each IoT task requirements (delay-sensitive or resource-intensive) and available resources on fog or cloud nodes and should optimize the time and energy consumption.

According to the result of scheduling, the IoT tasks are offloaded on the task queue of fog/cloud nodes for being executed. Sink nodes are also communicated to forward the necessary data to the fog/cloud nodes. If there are no available fog resources to execute the delay-sensitive tasks within the given deadline, the IoT tasks will be dropped. As soon as the execution of IoT tasks is completed, the output is sent to the initial fog cluster manager for being forward to the end-user. If a computation-intensive task can not be handle at the fog layer, it is forwarded to the cloud manager.

The steps for executing one IoT task, after the completion of task scheduling by the fog cluster manager, are outlined as below (see Fig. 3):

Step 1 The scheduler fog cluster manager offloads the IoT task to the selected fog/cloud node.

Step 2 The scheduler fog cluster manager requests from corresponding IoT devices to send the data to the selected fog/cloud node.

Step 3 IoT devices send the data to the selected fog/cloud node.

Step 4 The selected fog/cloud node executes the task.

Step 5 The results will send back to the fog cluster manager to be forwarded to the initial fog manager and to the end-user.

For simplicity's sake, the following assumption is considered within each time interval:

- The clusters are created automatically and remain unchangeable.
- The state of network is static.
- The programming codes for the execution of IoT tasks are deployed in all fog and cloud nodes.
- The data required for the execution of an IoT task are associated to a single fog cluster manager

Furthermore, the end-user may be mobile and her/his location may change during the time, moving to the coverage area of another fog cluster manager. This latter means that the initial fog cluster manager may not be the one that returns the output result to the end-user. In such a situation, based on the data locations of end-user and his/her moving pattern, we can predict her/his location at the next time slots by a predictor module and if needed, the results will be sent to the predicted fog

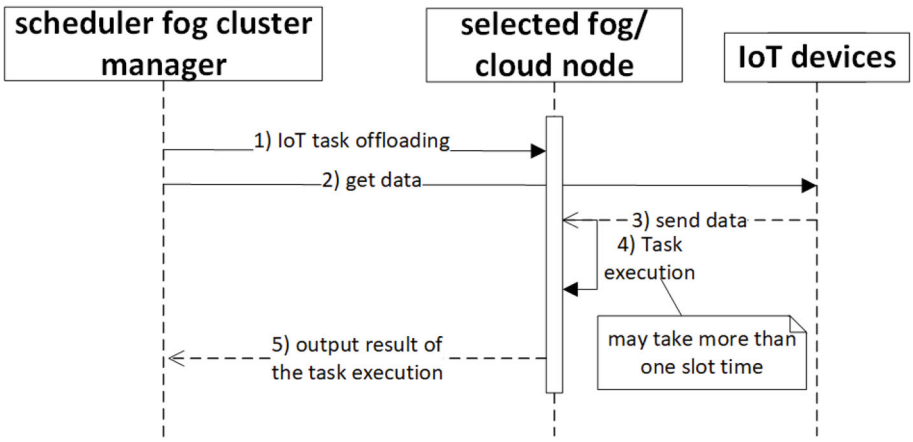


Fig. 3 The process of task execution

cluster manager that is in vicinity of the mobile end-user. It should be noted that the mobility of end-user is not considered in the rest of the paper.

Proposed qoS parameters modelling for task scheduling problem

The problem of task scheduling in the context of fog computing consists in assigning IoT tasks to appropriate fog nodes among a set of candidate fog nodes aiming at optimizing the overall QoS. We consider in this paper the time and energy consumption as QoS parameters.

IoT tasks are represented as a set $T = \{t_1, t_2, ..., t_i, ..., t_n\}$ in which each element t_i is described using a set of attributes $t_i = \{D_i, I_i, type, DE_i\}$, where D_i, I_i, type, and DE_i stand for input data size (in bits) to be transmitted toward the fog node, required computing density (in CPU cycles/bit) to execute the task, the type of the IoT task that is either delay-sensitive, or computation-intensive, and deadline (in second) of the IoT task to be respected for completing the task, respectively.

Fog nodes are shown as a set $F = \{f_1, f_2, ..., f_j, ..., f_m\}$ in which each element f_j is characterized using a set of attributes as $f_j = \{S_j^{max}, C_j^{max}, E_j^{max}\}$, where S_j^{max}, C_j^{max}, and E_j^{max} refer to storage capacity (in bits), computing capacity (in CPU cycles/bit), and the total battery capacity of the fog f_j (in watt), respectively.

With the above notations, the problem of task scheduling is to assign n IoT tasks to m fog nodes in such a way that the QoS parameters are optimized. We denote the assignment of an IoT task t_i to a fog node f_j as a_{ij}. In the following, we formulate the total time and energy consumption as two QoS parameters to be optimized.

The total time consumption for assigning IoT task t_i to fog node f_j is mathematically described as follows:

$$T_{total}(a_{ij}) = T_{up}(a_{ij}) + T_{execute}(a_{ij}) + T_{down}(a_{ij}) \quad (1)$$

Where, T_{up} denotes the time that is taken to transmit and offload task t_i to fog node f_j. $T_{execute}$ refers to the time that is related to the execution of task t_i running on fog node f_j. T_{down} indicates the time required for transmitting the result of the task execution from fog node f_j to the end-user. It is worth mentioning that T_{down} has minor effect on total time consummation T_{total} because of the small size of the output data comparing to the input data size. For that reason, T_{down} can be ignored and the total time is thus simplified as follows:

$$T_{total}(a_{ij}) = T_{up}(a_{ij}) + T_{execute}(a_{ij}) \quad (2)$$

In principle, the majority of T_{up} is usually related to the amount of data that should be sent from the IoT device to the selected fog/cloud node. In the above equation, T_{up} is defined as the ratio of task data size (D_i) to the transmission capacity of the channel (R_{ij}) between IoT device t_i and fog node f_j and is expressed as below:

$$T_{up}(a_{ij}) = \frac{D_i}{R_{ij}} \quad (3)$$

where,

$$R_{ij} = B \times log_2(1 + \frac{P_i \times h_{ij}}{N_0 \times B}) \quad (4)$$

R_{ij} is calculated based on the Shannon's capacity formula [42], where P_i denotes the transmission power of IoT device t_i, h_{ij} refers to the channel gain between IoT device t_i and fog node f_j, and N_0 denotes gauss noise power of the channel. The execution time $T_{execute}$ of task t_i on fog node f_j is defined as the required amount of computation for IoT task t_i divided by the computing capability of fog node f_j, as below:

$$T_{execute}(a_{ij}) = \frac{D_i \times I_i}{C_j^{max}} \tag{5}$$

Similar to the time modelling, the total energy consumption for execution a task t_i that is scheduled to be assigned to fog node f_j is denoted as the energy consumption for the transmission of task t_i to fog f_j (E_{up}) and the energy consumption for the execution of task t_i on fog f_j ($E_{execute}$). We mathematically formulated the total energy as below:

$$E_{total}(a_{ij}) = E_{up}(a_{ij}) + E_{execute}(a_{ij}) \tag{6}$$

where,

$$E_{up}(a_{ij}) = T_{up}(a_{ij}) \times P_i \tag{7}$$

$$E_{execute}(a_{ij}) = T_{execute}(a_{ij}) \times P_j \tag{8}$$

Where, P_j is the power consumption (in watt) of fog when executing task t_i.

Proposed formulation for task scheduling problem

The purpose of scheduling is to optimally assign IoT tasks to the resources of fog nodes for minimizing the time and energy consumption. The fog cluster manager may find a solution, denoted as a boolean matrix $A_{assign} = [a_{ij}]_{n*m}$ in which an element a_{ij} is 1 if the j^{th} fog is selected as fog f_j for executing task t_i and 0, otherwise.

The problem of assigning the IoT tasks to appropriate fog nodes can be modelled using ILP [5], as follows:

$$\underset{A_{assign}}{\text{argmin}}\, U(A_{assign}) \tag{9}$$

where

$$U(A_{assign}) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} \times Q(i,j) \tag{10}$$

subject to

$$\sum_{j=1}^{m} a_{ij} = 1 \qquad \forall i \in \{1, \cdots, n\}, \tag{11}$$

$$\sum_{i=1}^{n} a_{ij} \times I_i \leq C_j^{max} \qquad \forall j \in \{1, \cdots, m\}, \tag{12}$$

$$\sum_{i=1}^{n} a_{ij} \times D_i \leq S_j^{max} \qquad \forall j \in \{1, \cdots, m\}, \tag{13}$$

$$\sum_{i=1}^{n} a_{ij} \times E_{execute}(a_{ij}) \leq E_j^{max} \qquad \forall j \in \{1, \cdots, m\}, \tag{14}$$

$$T_{total}(a_{ij}) \leq DE_i \qquad \forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, m\}, \tag{15}$$

$$a_{ij} \in \{0, 1\} \qquad \forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, m\} \tag{16}$$

where $U(A_{assign})$ denotes the overall utility of a given task scheduling solution determined by A_{assign} matrix and calculated using Eq. 10. In this measure, $Q(i,j)$ indicates the QoS score of a_{ij}. The constraint 11 ensures not assigning a task to more than one fog node. Constraint 12 indicates that the required computing intensity of a set of IoT tasks that are assigned to fog node f_j could not be exceeded the computing capacity of the fog node. Constraint 13 specifies that the data size required for executing a set of IoT task given to fog node f_j could not be over the storage capacity of the fog node. Constraint 14 ensures that the required energy consumption of fog node f_j for executing a set of IoT tasks is less than the reminder battery capacity of the fog node. Constraint 15 ensures that the total time required for executing IoT task t_i by fog node f_j is not being exceeded the deadline of the IoT task. Constraint 16 defines our binary decision variables.

The QoS score given to a_{ij} is calculated as the weighted sum of time and energy QoS parameters which is formulated as follows:

$$Q(i,j) = w_t * T_{total}(a_{ij}) + w_e * E_{total}(a_{ij}) \tag{17}$$

where, $w_t, w_e \in [0, 1]$ denote the weight factors related to the importance impact of time and energy QoS parameters in a selected solution, respectively.

All the notations used are listed in Table 2.

Proposed approach

In this section, we review the concepts of WOA algorithm, Opposition-based learning, jumping rate, and Chaos theory. We then describe the new Opposition-based chaotic whale optimization algorithm named OppoCWOA for the problem of task scheduling in the context of fog computing environments.

Whale optimization algorithm (WOA)

In 2016, Mirjalili et al. [12] introduced a new nature-based meta-heuristic algorithm which is motivated by the particular hunting method of humpback whales. The procedure to find the (sub)optimal solution is similar to the other meta-heuristic algorithms: First, a population is initialized randomly that is the position of Whales in WOA. Second, the population seeks to be updated based on the best found solution so far using specific operators that is the hunting behaviours of humpback whales: looking for prey, encircling prey, and the spiral bubble-net feeding. The mathematically formulation of these behaviours is given below.

Encircling prey

The humpback whales move towards the preys and encircle them for feeding purpose. In WOA, the prey is considered to be the current best solution that represents the target towards which the other whales (search agents) move. The model for position updating is formulated as follows:

Table 2 Description of used notations

Symbol	Description
T	Set of IoT tasks in the task scheduling requests
n	Number of tasks in the task scheduling requests
t_i	The i^{th} task in the task scheduling request
F	Set of fog nodes in the system
m	Number of fog nodes in the system
f_j	The j^{th} fog node in the system
D_i	Input data size of task t_i
I_i	Required computing density of task t_i
DE_i	Deadline of task t_i
C_j^{max}	Computing capacity of fog node f_j
S_j^{max}	Storage capacity of fog node f_j
E_j^{max}	Battery capacity of fog node f_j
a_{ij}	A binary variable determining whether t_i is assigned to f_j
A_{assign}	assign matrix of size $n * m$
$U(A_{assign})$	Overall utility of a task scheduling solution
$Q(i,j)$	QoS score related to t_i that is assigned to fog node f_j
w_t	Weight for time QoS parameter
w_e	Weight for energy QoS parameter
$T_{total}(a_{ij})$	Total time consumption for assigning IoT task t_i to fog node f_j
$T_{up}(a_{ij})$	Time consumption for transmitting IoT data for executing the task t_i on fog node f_j
$T_{execute}(a_{ij})$	Time consumption for executing IoT task t_i in fog node f_j
$E_{total}(a_{ij})$	Total energy consumption for assigning IoT task t_i to fog node f_j
$E_{up}(a_{ij})$	Energy consumption for transmitting IoT data for executing the task t_i on fog node f_j
$E_{execute}(a_{ij})$	Energy consumption for executing IoT task t_i in fog node f_j
R_{ij}	Transmission capacity of the channel between IoT device and fog node f_j

$$\vec{D} = |\vec{C}\,\overrightarrow{X_{best}}(i) - \vec{X}(i)| \tag{18}$$

$$\vec{X}(i+1) = \overrightarrow{X_{best}}(i) - \vec{A} \cdot \vec{D} \tag{19}$$

Where, i specifies the current iteration, $\vec{X}(i)$ denotes position vector, $\overrightarrow{X_{best}}(i)$ denotes position vector of the current best solution. $\vec{A}$, and $\vec{C}$ are coefficient vectors defined as follows, respectively:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{20}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{21}$$

Where, $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations and $\vec{r}$ is a random vector within interval [0, 1].

Bubble-net feeding (exploitation)

The movement of humpback whales around the prey is simultaneously in a circular or spiral-shaped direction. This cooperating feeding behaviour is known as Bubble-net feeding. To model that, an equal probability (0.5) is assumed for selecting each of these two movements during the position update. The formulation is given below:

$$\vec{D'} = |\overrightarrow{X_{best}}(i) - \vec{X}(i)| \tag{22}$$

$$\vec{X}(i+1) = \begin{cases} \overrightarrow{X_{best}} - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\Pi l) + \overrightarrow{X_{best}}(i) & \text{otherwise.} \end{cases} \tag{23}$$

Where, $\overrightarrow{D'} = |\overrightarrow{X_{best}}(i) - \overrightarrow{X}_{(i)}|$ is the distance between the i^{th} whale and the prey, b is the constant for defining the logarithmic spiral shape, and l is the random number within $[-1, 1]$.

Search for prey (exploration)

The search for prey of humpback whales is performed randomly according to the position of each other. To force a search agent (whale) to move in the other directions than a reference whale, a random whale is selected from the population and the position of the other ones is updated towards it. This behaviour gives occasion to obtain a global search. This model is as follows:

$$\vec{D} = |\vec{C}\,\overrightarrow{X_{rand}} - \vec{X}| \tag{24}$$

$$\vec{X}(i+1) = \overrightarrow{X_{rand}} - \vec{A} \cdot \vec{D} \tag{25}$$

Where, $\overrightarrow{X_{rand}}$ denotes a random whale chosen from the current population. Equation 25 is executed in case $|\vec{A}| > 1$.

Limitations of WOA

Meta-heuristic algorithms such as WOA, often converge prematurely to the point that is not globally optimal.

Generally, premature convergence takes place due to (1) The population has converged towards the local optimum. (2) The diversity is excluded from the population. (3) The search algorithm is progressing slowly or not at all. In the case of the WOA, the random vector A (Eq. 20) controls exploration and exploitation; in cases with $|A| \geq 1$ and $p < 0.5$, iteration is applied to exploration; otherwise, iterations are devoted to exploitation.

However, sometimes search agents become inactive in local optimum after some exploitative movements. In addition, agents update their positions toward the elite vector of the population in half of the iteration. Therefore, the population diversity decreases rapidly (no completely new solutions will be created any more) which consequently increase the local optimum possibility. To prevent this, we should find a trade-off between global exploration and local exploitation, which is difficult, since these two phases behave contradictory; i.e. a better exploration

results in a worse exploitation and vice-versa. In WOA, the exploitation-exploration trade-off is set in favour of local exploitation, which may cause the premature convergence [43]. Given these limitations, we propose OppoC-WOA in order to enhance the performance of WOA.

Opposition-based learning

Opposition-based Learning (OBL) is proposed by Tizhoosh [14] to improve solutions by taking into account the current population and its opposite simultaneously. According to probability theory, there is a 50% chance that an opposite guess is closer to the solution than the guess itself. Furthermore, without prior knowledge, it is difficult to make the best initial guess that is close to a possible solution, so in order to find it, we should search in all directions at the same time or in a more concrete term in the opposite direction. The process can be improved with a better (i.e., fitter) initial guess. In the worst-case scenario, our random guess is in the neighbourhood of opposite location of the possible solution, which by utilizing the opposition-based approach we will start with a better initial guess. This approach can be applied to the current and initial population. Every estimate solution $X(i)$ within the interval $[U + L]$ has an opposite solution $\widehat{X(i)}$ which is calculated as follows:

$$\widehat{X(i)} = U + L - X(i) \tag{26}$$

Figure 4 illustrates the opposite solution, where n tasks are assigned to 5 fog nodes.

Quasi and super opposition-based learning

According to different definitions and theorems, it can be seen that the center point is critical. Rahnamayan and Wang [44] have indicated that the possibility of the proximity of the center point to an unknown optimal solution is higher than other points. Because of the importance of center point, many opposition-based techniques employed center point as a reference. Tizhoosh [15] proposed Quasi and Super opposition-based learning by employing center point as a reference to improve optimization problems. Every estimate solution $X(i)$ within the interval $[U + L]$ has Super-opposite points and Quasi-opposite points $\widehat{X(i)^s}$, $\widehat{X(i)^q}$ which are defined as follows:

	t_1	t_2	...	t_h	t_i	t_j	t_k	...	t_{n-1}	t_n
Solution	1	4	...	2	1	3	1	...	2	3
Opposite Solution	5	2	...	4	5	3	5	...	4	3

Fig. 4 Opposite-solution generation

$$\widehat{X(i)^s} = \begin{cases} rand\,[L, U + L - X(i)) & \text{if } X(i) > {}^{(U+L)}/_2 \\ [\text{L,U}]\text{-}\{\text{X(i)}\} & \text{if } X(i) = {}^{(U+L)}/_2 \\ rand\,(U + L - X(i), U) & \text{if } X(i) < {}^{(U+L)}/_2 \end{cases} \tag{27}$$

$$\widehat{X(i)^q} = \begin{cases} rand\left[U + L - X(i), \frac{(U+L)}{2}\right) & \text{if } X(i) > {}^{(U+L)}/_2 \\ \emptyset & \text{if } X(i) = {}^{(U+L)}/_2 \\ rand\left(\frac{U+L}{2}, U + L - X(i), U\right) & \text{if } X(i) < {}^{(U+L)}/_2 \end{cases} \tag{28}$$

Figure 5 shows the relationship between these opposition-based techniques.

Partial-opposition-based learning

By producing opposite solutions, we assume that all dimensions should change to their opposites; this strategy seems to be persuasive in the algorithm's initial iterations to find a better initial guess. Nevertheless, in later iterations, only a few dimensions need to be changed into their opposite to approach an optimal solution.

Owing to this rationale explanation, several different partial-opposition-based learning schemes have been designed in the literature. For example, Hu et. al [45], proposed a partial-opposition-based scheme to optimize the differential evaluation algorithm (DE) in which a set of candidate partial-opposition solutions is created for each candidate solution.

In the candidate partial-opposite solutions, only one dimension maintains its original value while others change to their opposites. some of these partial-opposite solutions are then selected randomly to compete for the original candidate solution substitution.

The proposed partial-opposition-based learning

To maintain population diversity and convergence speed, we propose a partial-opposition-based learning scheme by combining the 2 point crossover operation in Genetic Algorithm with OBL techniques reported in "Opposition-Based learning" section as follows: 1) The candidate solution (whale position) and its opposite solution will be chosen as parents, 2) two crossover points will be selected at random, 3) The candidate solution exchanges the middle segment with the opposite solution, 4) and the remaining segments will remain unchanged, producing new offsprings (partial solution candidates). These offspring are chosen to compete for the substitution of the original candidate solution. This method will enhance the exploration ability of the algorithm due to the crossover

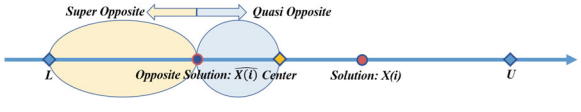


Fig. 5 The relationship between opposition-based, super opposition and quasi opposition techniques

operation and OBL characteristics. Figure 6 illustrates the proposed partial-opposite solution, where n tasks are assigned to 5 fog nodes.

Jumping rate

The convergence speed can be increased by OBL techniques; i.e. more area is probably visited in search space. This may prevent the convergence toward an optimum solution. In addition, excessive use of this technique could lead the population to stick in local optimum instead of converging toward the global optimum. To prevent this problem, we employed the jumping Rate factor, which decides the probability of Opposition-based technique occurrence over time. The higher jumping rate results in faster convergence to the global optimum at a lower success rate and vise versa. In other words, a higher jumping rate reduces the exploitation ability of the algorithm, which results in a success rate decrease, while a lower jumping rate often fails to a local optimum. In this work, we need to find a suitable jumping rate to balance between convergence speed and success rate.

Chaos theory

Chaos theory is one of the most appropriate approaches to deal with the premature convergence problem due to its properties of non-repetition, ergodicity, and dynamicity [46]. Chaos is a stochastic process in the non-linear deterministic system, which ultimately makes the numerical sequences of two closed initial values irrelevant after a certain number of iterative operation performed by the same chaotic function.

The operation of WOA is simple and easy to achieve optimal solution; however, the search process only relies on the randomness of parameters: the vector $\vec{r}$ and the probability p. $\vec{r}$ (Eqs. 20 and 21) has random value between [0, 1] which is designed for moving towards any position close to the current best solution and the probability p (Eq. 23) within the interval [0, 1] is defined for changing between the circular or spiral-shaped position update toward the current best solution. We employ a chaotic map to define $\vec{r}$ and p in every iteration. Since the value of the other parameters depends directly on the value of $\vec{r}$ and p, we can increase simultaneously algorithm convergence speed and prevent premature convergence by using chaotic sequence.

In this paper, eleven types of chaotic maps are used that is shown in Table 6 (see Appendix). These maps have different behaviours with the initial point 0.7. It is also possible to select any number as the initial point between $[0, 1]$ or $[-1, 1]$ based on the range of the chaotic map. However, the fluctuation pattern in some of these maps depend significantly on the initial value.

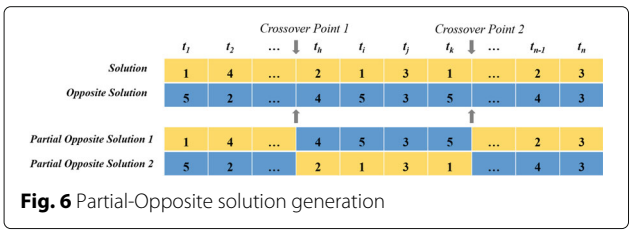


Fig. 6 Partial-Opposite solution generation

Non-linear functions

As mentioned in "Limitations of WOA" subsection, the random vector $\vec{A}$ controls the exploration and exploitation phase of WOA. In accordance to Eq. 20, the value of $|A|$ directly depends on the value of $\vec{r}$, and $\vec{a}$, where $\vec{a}$ linearly decreased from 2 to 0. In this case, a large $\vec{a}$ will encourage a global search in the early iterations, because with a larger $\vec{a}$, the chances of $|A| \geq 1$ will be higher, which leads WOA to reach the global exploration phase according to Eq. 25 and accelerate the convergence speed. while a small $\vec{a}$ in the late iterations leads WOA to reach the exploitation phase to find local optimum. This linear decrease, though, may have two potential problems: (1) a premature convergence could occur due to low exploration capability, and (2) convergence speed at local exploitation could be slow because of small step size. In order to change the global exploration and the local exploitation behaviours of WOA, we propose to apply four non-linear functions to replace the linear function $\vec{a}$. These functions and their possible effects on search capabilities are presented in Table 5 (see Appendix) where t denotes current iteration, and t_{max} denotes number of iterations.

Fog task scheduling using oppoCWOA

In this part, we transform the task scheduling problem formulated in "Proposed formulation for task scheduling problem" section in the form of whale foraging design considering the proposed approaches (opposition-based learning, Chaos theory, non-linear functions) combined with WOA to enhance the performance of our task scheduling algorithm in terms of convergence speed and accuracy.

Solution encoding

A task scheduling solution in our design refers to an individual that is the position of whales in WOA. Whale positions are represented as an m-dimensional array. Each index of the array represent a whale and the value of that index is the whale position. In our design, each index denotes the IoT tasks and the value of that index is the fog node j (within the interval $[1, n]$) that would be mapped to the corresponding task. For instance $W = \{1, 3, 3, 1, 2\}$ means tasks 1 and 4 are assigned to fog node 1, tasks 2 and 3 are assigned to fog node 3, and task 5 is assigned to fog node 2.

Population initialization

The initial population (denoted as NP: Number of Population) is the set of NP whales that will be participate in

WOA to reach the optimal solution. In our design, this is the different possible task scheduling solutions (different possible mapping between IoT tasks and fog nodes). *NP* whale positions will be initialized randomly to ensure population diversity. Then, the opposite whale positions will be initialized and added to the population according to "Opposition-based learning" section.

Fitness evaluation

The fitness function is the metric that determines the quality of whale positions within the population. In our design, that is the quality of the mapping between IoT tasks and fog nodes. solutions with low fitness value represent high-quality mapping, thus considered effective. The fitness value of each solution (whale positions) is evaluated by our formulated objective function using Eq. 9, which depends on time completion of tasks and energy consumption of fog nodes. Based on the fitness value, the solutions can be updated.

Whale position updating

After the calculation of fitness value for all whale positions (i.e. task scheduling solutions), the best whale position will be chosen to be used by other whales to update their positions. Whale positions updating depends on two essential factors $|\vec{A}|$, and probability p. $|\vec{A}|$ value is determined by the value of $\vec{r}$ and $\vec{a}$ (Eq. 20). We use chaotic sequence to initial the value of $\vec{r}$ as mentioned in "Chaos theory" section and the value of $\vec{a}$ is determined by one of the non-linear functions defined in "Non-linear functions" section. p is initialized by chaotic sequence as discussed in "Chaos theory" section. Based on these calculations, the whale positions will be updated as follows: WOA will fall into the exploitation phase if $|\vec{A}| < 1$ and depending on the value of p, the positions will be updated. If $p < 0.5$, we have the Eq. 19 and if $p > 0.5$, the positions are updated using the Eq. 23. In the case of $|\vec{A}| \geq 1$, WOA will fall into the exploration phase and the positions are updated according to the Eq. 25 after the position updating of all whales, this new population will replace the old one.

Opposite-population

After whales have updated their positions, population diversity will drop rapidly, and we therefore need to inject diversity into the population. To this end, opposite whale positions will be initialized and added to the population according to "The proposed partial-opposition-based learning" section. Then, to control population size, the fittest *NP* whale positions will survive to create a new population while others will perish. As we explained in "Jumping rate" section, to prevent premature convergence, we have used a jumping rate, which decides the probability of opposition-based technique occurrence over time. This means each candidate solution (whale position) has a chance to bring its partial-opposites to the new population. To find the optimal jumping rate, we have conducted several experiments that is reported in "Experiment results" section.

Algorithm 1 Pseudocode for OppoCWOA

Input vector of n tasks, vector of m fog nodes, NP: number of population, t: number of iteration
Output $\vec{X}_{gbest}$: vector of optimal task-fog mapping solution

function TASK SCHEDULING
  Initialize randomly the whales population ($i = 1, 2, 3, ..., NP$)
  Initialize parameters (p and r) initializing the sequences of the chaotic maps in "Chaos theory" section
  Initialize parameter (a) using non-linear functions in "Non-linear functions" section
  Initialize other WOA parameters ($l, A,, C,$ and jumpring rate (jr))
  Calculate partial-opposite whales for the population and add it to the population
  Calculate fitness value of each search agent using **Eq.** 9
  Select NP fittest agents as the population
  Determine $\vec{X}_{gbest}$ as the current best agent
  while ($t <$ *number of iterations*) **do**
    for each search agent **do**
      if ($p < 0.5$) **then**
        if ($|A| < 1$) **then**
          Update the position of the current search agent by the **Eq.**19
        else if ($|A| \geq 1$) **then**
          Select a random search agent
          Update the position of the current search agent by the **Eq.**25
        end if
      else {$p \geq 0.5$}
        Update the position of the current search agent by the **Eq.**23
      end if
      Check if any search agent goes beyond the search space and amend it
      if ($jr < 0.5$) **then**
        Create partial-opposite-solution and add them to the population
      end if
      Calculate fitness value of each search agent using **Eq.** 9
      Select NP fittest agents as the current population
      Update $\vec{X}_{gbest}$ if there is a better search agent
      Update the chaotic numbers using chaotic map equations in Table 6 for parameters (p and r)
      Update parameter (a) using non-linear functions in "Non-linear functions" section
      Update parameters (l, A, C, and jr)
    end for
    $t = t + 1$
  end while
  Return $\vec{X}_{gbest}$
end function

OppoCWOA algorithm

Based on the aforementioned principles, the main steps of our OppoCWOA approach for task scheduling problem in the context of fog computing is given as below:

Step 1 [lines 2 to 9] This step comprises the solution encoding ("Solution encoding" section), population generating based on the *NP* fittest agents from the combination of random population and opposite-population and determining the current best agent based on the fitness value, initialization of chaotic sequence in chaotic map for the parameters p and r, initialization of parameter a using non-linear functions, initialization of other WOA parameters (l, A, C, and jumping rate (jr)) and some other initial parameters such as the number of iterations, and population size *NP*.

Step 2 [lines 10 to 21] In this step, the WOA-based searching process to find optimal solution begins. The exploration and exploitation behaviours of whales is based on the value of the current A (Eq. 20) and p. In the case of $p < 0.5$ and $|A| \geq 1$, the exploration phase is executed using Eq. 25. For $p < 0.5$ and $|A| < 1$, positions are updated using the Eq. 23 and if $p > 05$, we update the positions by the Eq. 19.

Step 3 [lines 22 to 32] after all whales (or agents) updated their positions, partial-opposition whales will be initialized according to "Opposite-population" subsection depending on the random value of jumping rate. Then the fittest partial-opposite whales will be added to the population to create a new population. Then half of this new population with fittest values will be selected as the population to prevent population size growth. One iteration will end by updating the parameters (p, r, a, l, A, C, and jr).

Step 4 [line 33 to 35] By reaching the maximum number of iterations, the process will end; otherwise, a new search will be performed by returning to step 2. The algorithm return the best search agent as output.

The pseudocode of OppoCWOA algorithm is provided in Algorithm 1.

Computational complexity

The computational complexity is a key element to present the scalability analysis of an algorithm. According to the instruction steps and the structure of WOA, we can calculate the complexity of WOA as $O(t(D*NP)+Obj*NP)$, where NP is the total number of whales, t is the number of iteration, D is the dimension of the problem, and Obj is the cost of calculating the objective function.

In our OppoCWOA algorithm, we consider the opposition-based calculation for generating initial population initialization. In addition, we use chaos theory during the iteration process for updating the population. Each of these two methods add obviously an overhead to the original WOA. For the opposition-based calculation, the time complexity is $O(D * NP + Obj * NP)$ and for the chaos calculation for all iteration, we have $O(t(D(NP + Ch)) + Obj * NP)$, where Ch indicates the iteration number of Chaos calculation. The overall computational complexity is therefore calculated as below:

$$\mathrm{O(t\ (D\ (NP+Ch))+Obj*NP)} \tag{29}$$

It can be seen that the overhead added by the opposition-based and Chaos theory is negligible considering the efficiency of the proposed OppoCWOA.

Experiment results

To validate the efficiency of the proposed OppoCWOA from different perspectives, extensive experimentation studies are conducted considering the evaluation metrics (Time, energy and time-energy balance) presented in "Proposed system model and formulation for the task scheduling problem" section. The fog environment is composed of twenty fog nodes with different characteristics such as computing capacity, bandwidth, distance from the manager within the cluster, computing energy. Furthermore, six datasets of different number of tasks (from 50 to 300) are created to participate in the experimentations. Table 3 shows the parameter values considered in our settings. In order to provide different scenarios, many types of tasks are created with various amounts of processing power, bandwidth, and data usage. Since our proposed algorithm is based on the WOA, we have compared our method with the original WOA, and other popular meta-heuristic optimization algorithms such as PSO, ABC, and GA.

All algorithms used in the paper have population size of 50 (bees, particles, and whales) and the maximum number of iterations is 200. The results are averaged over 25 independent runs. As presented in "Proposed approach" section, the opposition-based learning and Chaos theory enhance the search ability and accelerate the convergence

Table 3 Parameter values of the simulation

Parameter	Value	Description
n	50 — 300	Number of tasks
m	20	Number of fog nodes
D_i	10 — 50 Mb	The data size of task t_i
I_i	300 — 500 Cycles/bit	Computing density of task t_i
C_j^{max}	1 — 2 Giga cycles/bit	The computing capacity of fog node f_j
S_j^{max}	500 — 2000 Mb	Storage capacity of fog node f_j
B	100 — 200 MHz	Bandwidth between IoT device and fog node
L_j	1.5 — 2 Km	Distance of fog node f_j from the fog node manager
p_j	1 — 5 W	Power consumption of fog node f_j
p_i	0.1 W	Transmission power of IoT task device t_i

of WOA algorithm. To characterize the specific effect of each component (opposition-based learning and Chaos theory) on WOA, we report the archived fitness values for each applied components separately. The weight factor values for time and energy consumption ($w1$ and $w2$) in objective function being set up to 0.5, which means having equal priority in optimization process. The settings of the experimental environment are *Intel(R) Core (TM) i7-8550U CPU @ 1.80 GHz, 16GB Memory on Windows 10 professional OS*, the simulation was developed in Python with PyCharm editor. It is worth mentioning that our simulation testbed is at the early stage and not completely pre-validated. It is designed and implemented according to our system model described in "Proposed system model and formulation for the task scheduling problem" section. The evaluation metrics include time, energy, and time-energy balance. In the following subsections, we have reported both mean (or average) and median of experiment results. We have used the mean in our diagrams to study the convergence speed of different algorithms as a typical case. However, the mean is particularly susceptible to the influence of outliers, while the median is less affected by outliers and skewed data. Therefore, we have used the median to compare the performance of the algorithms, which is more suitable than the mean.

Chaotic maps effect

In this part, we analyse the effect of chaotic maps (Table 6) to initialize $\vec{r}$, and p values in WOA (CWOA). The results are compared to the original WOA (NoChaos).

Chaos maps for $\vec{r}$

In Fig. 7(a) it can be observed that in term of energy cost optimization, all the chaotic maps except Circle map show worse results as compared to WOA algorithm. In other words, Chebyshev, Iterative, Logistic, Cubic, Sine, Sinusoidal, Singer, Tent, Piecewise, and Gauss/Mouse chaotic maps can not enhance the efficiency and convergence speed of WOA algorithm. For the time cost objective, the Circle and Singer chaotic maps can enhance the performance and convergence speed of the WOA as shown in Fig. 7(b). Regarding the Time-energy balance objective, it can be seen that Logistic, Sine, Gauss, Piecewise, iterative, Circle, and Chebyshev chaotic maps can slightly enhance the performance of the WOA as presented in Fig. 7(c). On the contrary, other chaotic maps results are worse than WOA while achieving a time-energy balance. It can be considered that Circle maps have faster convergence in comparison to others for all three metrics.

Table 7 (see Appendix) shows the results of 11 chaotic maps on all evaluation metrics in CWOA for $\vec{r}$ value. As depicted, in terms of achieving energy costs and time cost, random choice of r in $[0, 1]$ outperforms all chaotic maps. While in terms of pursuing a balance between time and energy costs, Circle map can slightly enhance the performance of WOA.

Chaos maps for p

Figure 8 represents the results of applying the 11 chaotic maps for the p value. In regard to the energy cost (Fig. 8(a)), all chaotic algorithms except Gauss and Iterative show better results as compared to WOA algorithm. Otherwise speaking, Chebyshev, Circle, Logistic, Cubic, Sine, Sinusoidal, Singer, Tent, and Piecewise chaotic maps can enhance the performance and convergence speed of the WOA algorithm. In regard to the time cost (Fig. 8(b)), Chebyshev, Logistic, Cubic, Sine, Sinusoidal, Singer, Tent, and Piecewise chaotic maps can enhance the performance and convergence speed of the WOA algorithm. Also, Gauss and Iterative chaotic maps show worse results than the WOA algorithm. In regards to the time-energy balance performance (Fig. 8(c)), all chaotic maps except Gauss and Iterative can enhance the performance of the WOA algorithm while achieving a time-energy balance. Also, it can be seen that the Sinusoidal map yields the best result in pursuing a time-energy balance.

Table 8 (see Appendix) shows the results of 11 chaotic maps on all evaluation metrics on CWOA for p value. It can be seen that in terms of archiving lower energy costs, the Sinusoidal map outperforms others. Also, Piecewise, Iterative, Gauss, and Cubic yields worse results in comparison to random choice. In terms of achieving lower time cost, Sinusoidal map yields the best result in comparison to others. While Piecewise, Gauss, Iterative, Circle, and Chebyshev show worse results than random choice. In terms of pursuing a balance between time and energy costs, Sinusoidal shows better results than others. While Tent, Sine, Iterative, and Gauss yields worse results than random choice.

Opposition-based techniques effect

We employed four different opposition-based techniques named Opposition (O), Partial-Opposition (PO), Quasi-Opposition (QO), and Super-Opposition (SO), as explained in "Proposed approach" section to improve the performance of WOA. Therefore we proposed OWOA, POWOA, QOWOA, and SOWOA to compare their performance with the original WOA to find out which technique is more suitable for our scheduling problem. Also, we set up the jumping Rate to 0.5. As seen in Fig. 9(a), comparing to the original WOA, with increasing the number of iterations, the fitness value of the opposition-based algorithms to achieve less energy consumption decreases, showing the efficiency and effectiveness of opposition-based algorithms on task scheduling in fog computing. The energy cost of the proposed POWOA is smaller than other opposition-based algorithms. This means that

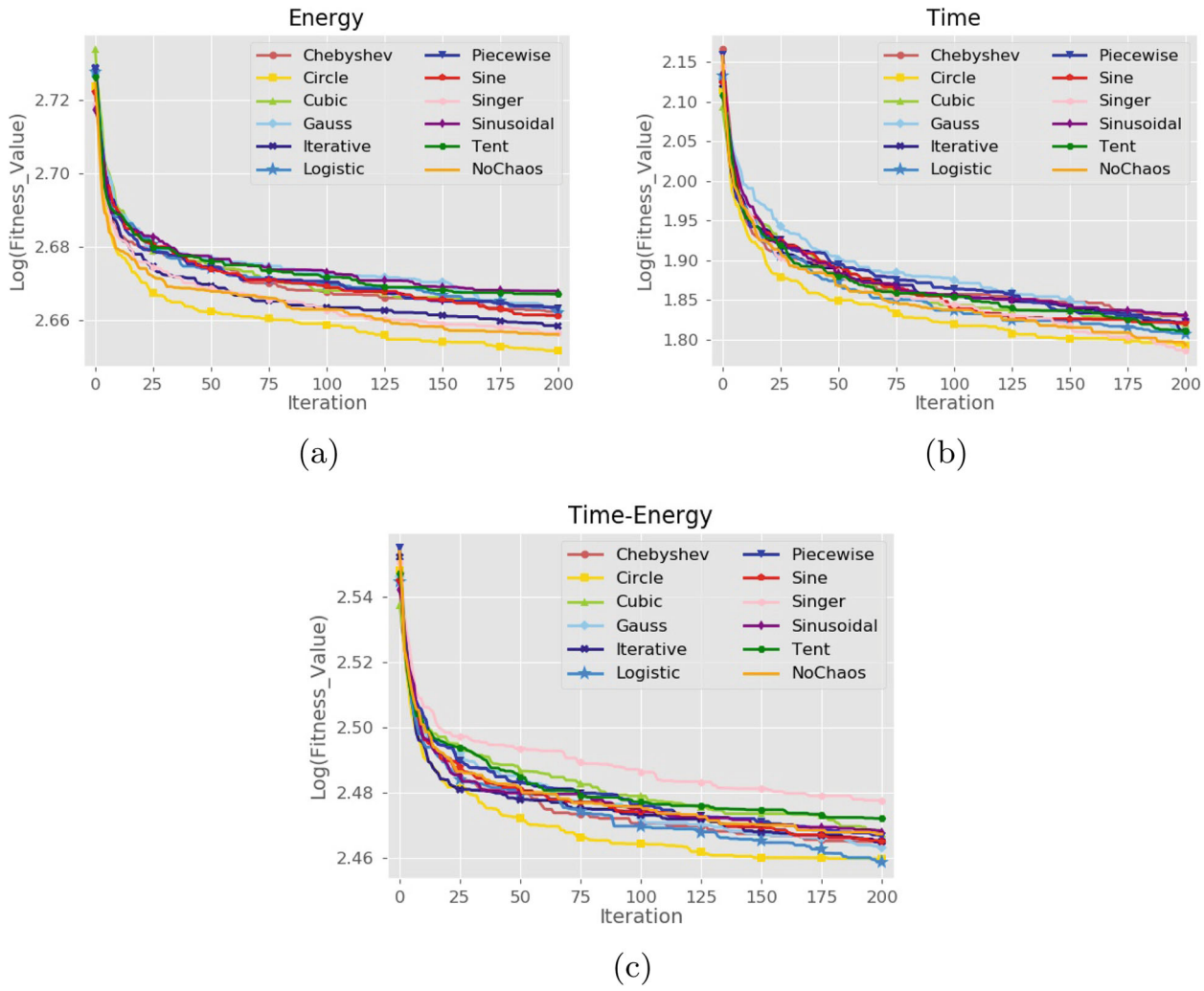


Fig. 7 Performance comparison of 11 chaotic maps for $\vec{r}$

POWOA has greater search capabilities than others. In addition, with an increase in the number of iterations, POWOA can get its optimal solution faster than WOA, which means that our proposed optimization techniques will greatly boost the WOA algorithm's convergence speed. The results of the fitness values of the time and time-energy balance are depicted in Figs. 9(b) and 9(c), respectively. Same as the energy cost metric, POWOA outperforms the other algorithms in terms of convergence speed and accuracy in achieving a time-energy balance. In particular, in comparison with WOA, OWOA, QOWOA, and SOWOA, the cost of time-energy balance is reduced significantly. However, It cannot produce much better results in time costs.

Table 9 (see Appendix) shows the results of the 4 opposition-based techniques applied to WOA for energy, cost, and energy-cost balance metrics. It can be seen that in terms of achieving lower Energy cost, POWOA outperforms other algorithms. In other words, Partial-opposition-based technique can significantly enhance the performance of WOA. Moreover, SOWOA algorithm shows slightly better results, while OWOA and QOWOA present worse results than WOA. In terms of achieving lower Time costs, POWOA and SOWOA yield better results than the WOA algorithm. In other words, the Partial and Super opposition-based technique can enhance results of WOA, while Opposition and Quasi opposition-based technique can not enhance the performance of the WOA algorithm. As shown in the table, to achieve a balance between time and energy costs, the partial-opposition technique can significantly enhance the performance of WOA. Opposition and Super opposition

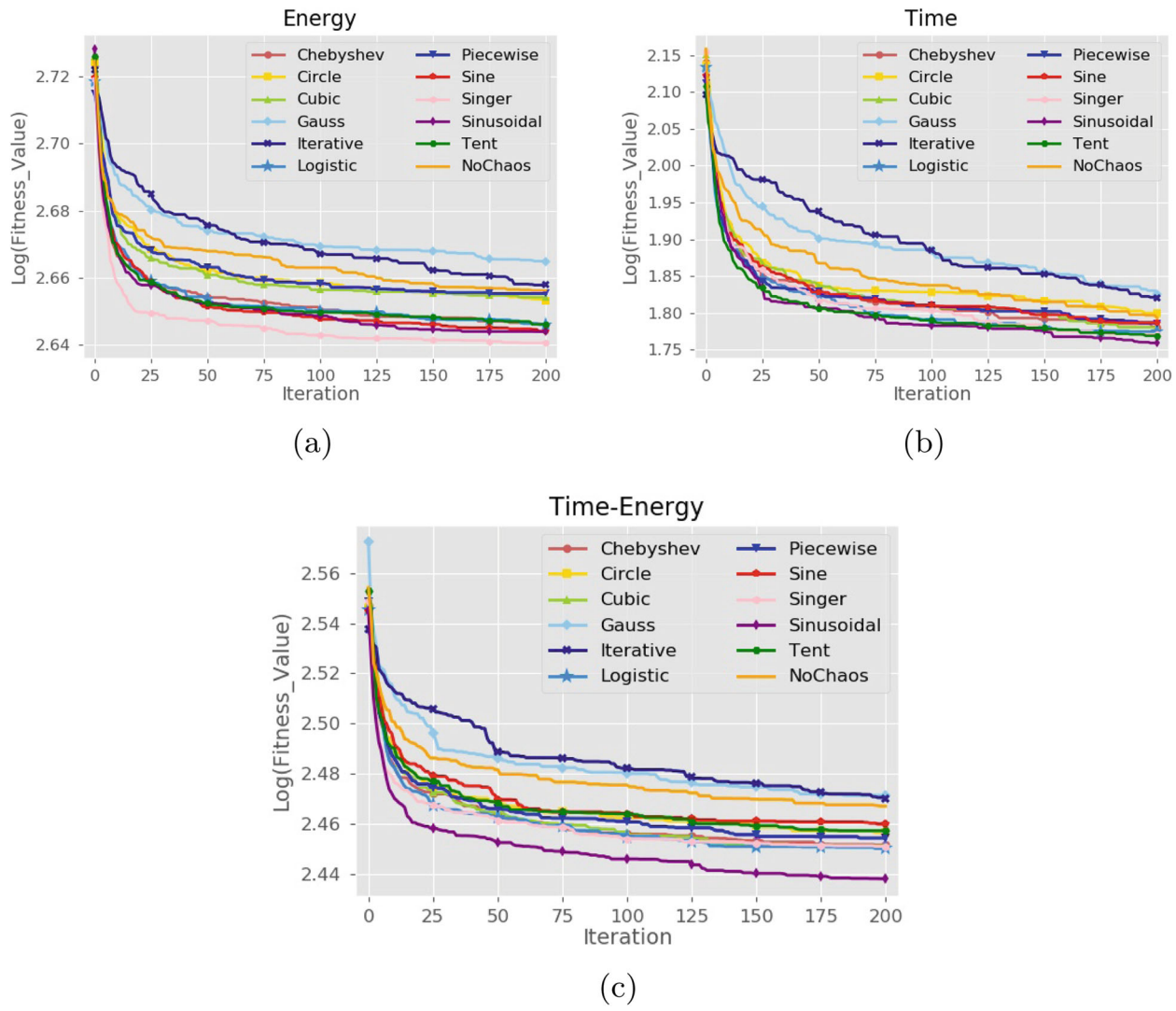


Fig. 8 Performance comparison of 11 chaotic maps for *p* value

can slightly enhance WOA performance. Table 9 shows that partial-opposition-based technique can enhance the performance of WOA over our evaluation metrics.

Jumping rate effect

As we discussed in "Proposed approach" section, to prevent the premature convergence of POWOA, we employed the Jumping Rate (JR) factor. We increase JR form 0.1 to 0.9 to study its effect on the behavior of POWOA. In Fig. 10(a), it can be observed that JR over 0.6 can cause premature convergence, which prevents the algorithm from yielding better results in terms of achieving energy cost. In Figs. 10(b) and 10(c), it can be seen that JW over 0.8 and 0.7 can cause premature convergence in terms of achieving time cost and time-energy balance, respectively. This study shows that any value within the interval of $[0.6, 0.8]$ is appropriate for JR to prevent premature convergence of the POWOA algorithm.

Non-linear functions

As presented in "Proposed approach" section, to improve the exploration phase of the WOA algorithm, we employed four non-linear functions namely Beta1, Beta2, Beta3, and Beta4 and replace them with the linear function that changes the vector $\vec{a}$ (Eq. 20). As seen in Fig. 11(a), Beta3 shows slightly better results while other Beta functions provide worse results comparing to the linear function in terms of achieving lower energy cost. In Fig. 11(b), it can be seen that regarding the time cost, Beta2 and Beta3 yield slightly better results while Beta1 and Beta4 results are worse than the linear function. As depicted in Fig. 11(c), all proposed Beta functions yield

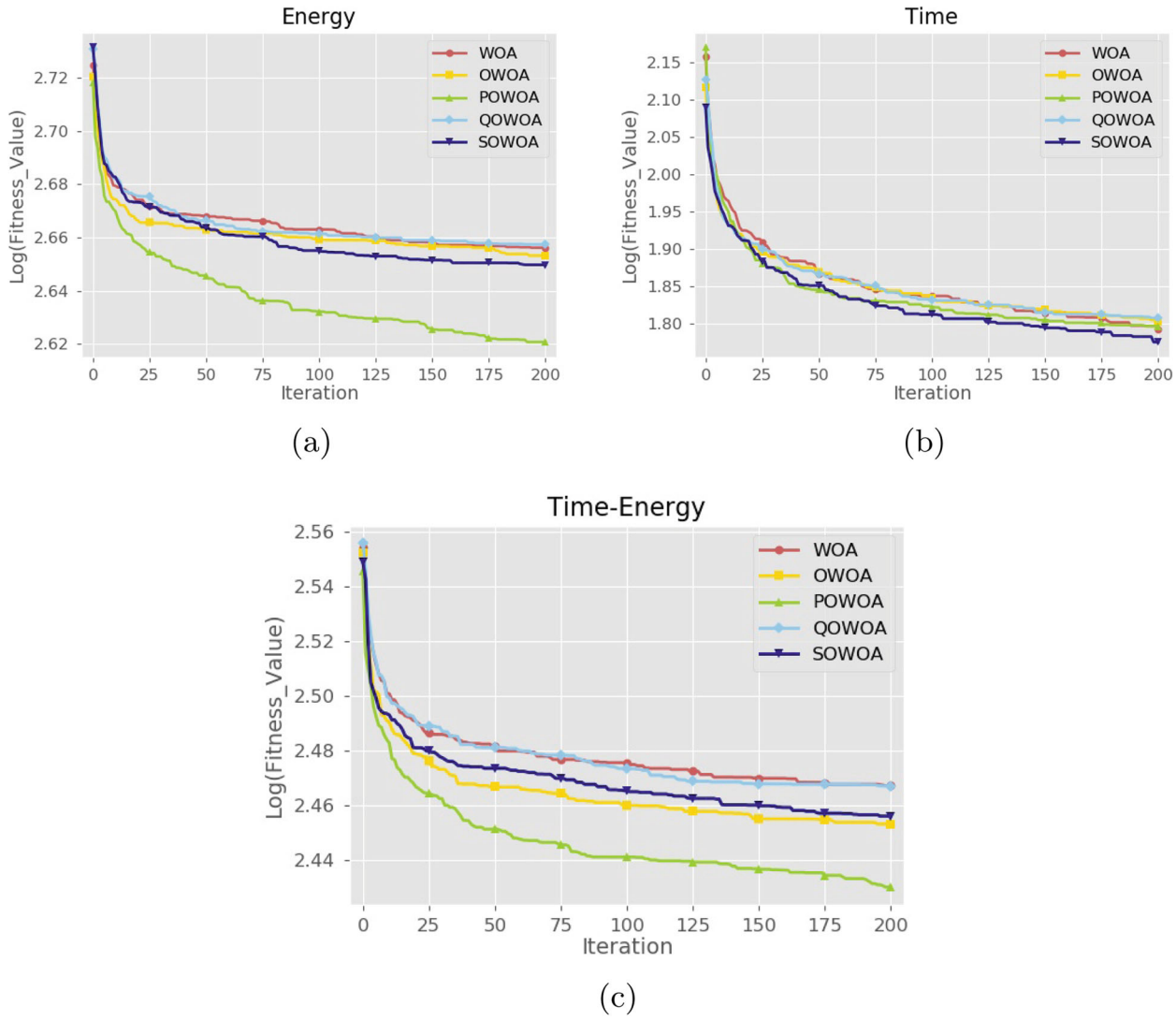


Fig. 9 Performance comparison of 4 OBL techniques with three evaluation metrics

better results than the linear function; also, Beta2 shows the best results out of other functions while achieving a balance between time and energy costs.

Table 10 (see Appendix) presents the effects of 4 non-linear functions on WOA for the three evaluation metrics. It can be observed that in terms of achieving lower energy cost, linear function outperforms other non-linear functions, and Beta3 yields better results than other non-linear ones. Concerning the lower time cost, Beta3 has the best performance among all non-linear functions and slightly better than the linear function. Besides, regarding the balance between time and energy costs, Beta1, Beta2, and Beta3 achieve better results than the linear function.

OppoCWOA evaluation

As presented in "Proposed approach" section, to enhance the WOA algorithm, we employed many different techniques that are partial-opposition learning, chaotic maps, non-linear functions, and jumping rate. In this section, we report the experiment results from the combination of these techniques to illustrate the efficiency of our approach. The parameter setting of these simulations is depicted in Table 4. We record the achieved values for time, energy, and time-energy balance with two separate scenarios that are varying the number of iterations (scenario 1) and the number of tasks (scenario 2) during each experiment.

Scenario 1: In this scenario, the number of tasks is fixed to 50, and the number of iterations is increased gradually from 1 to 200. Figure 12 depicts the comparison analysis of the five algorithms execution. As illustrated in Fig. 12(a), the fitness value for energy cost keep decreasing gradually by incrementing the number of iterations, that indicates the effectiveness of all algorithms. The figure shows that

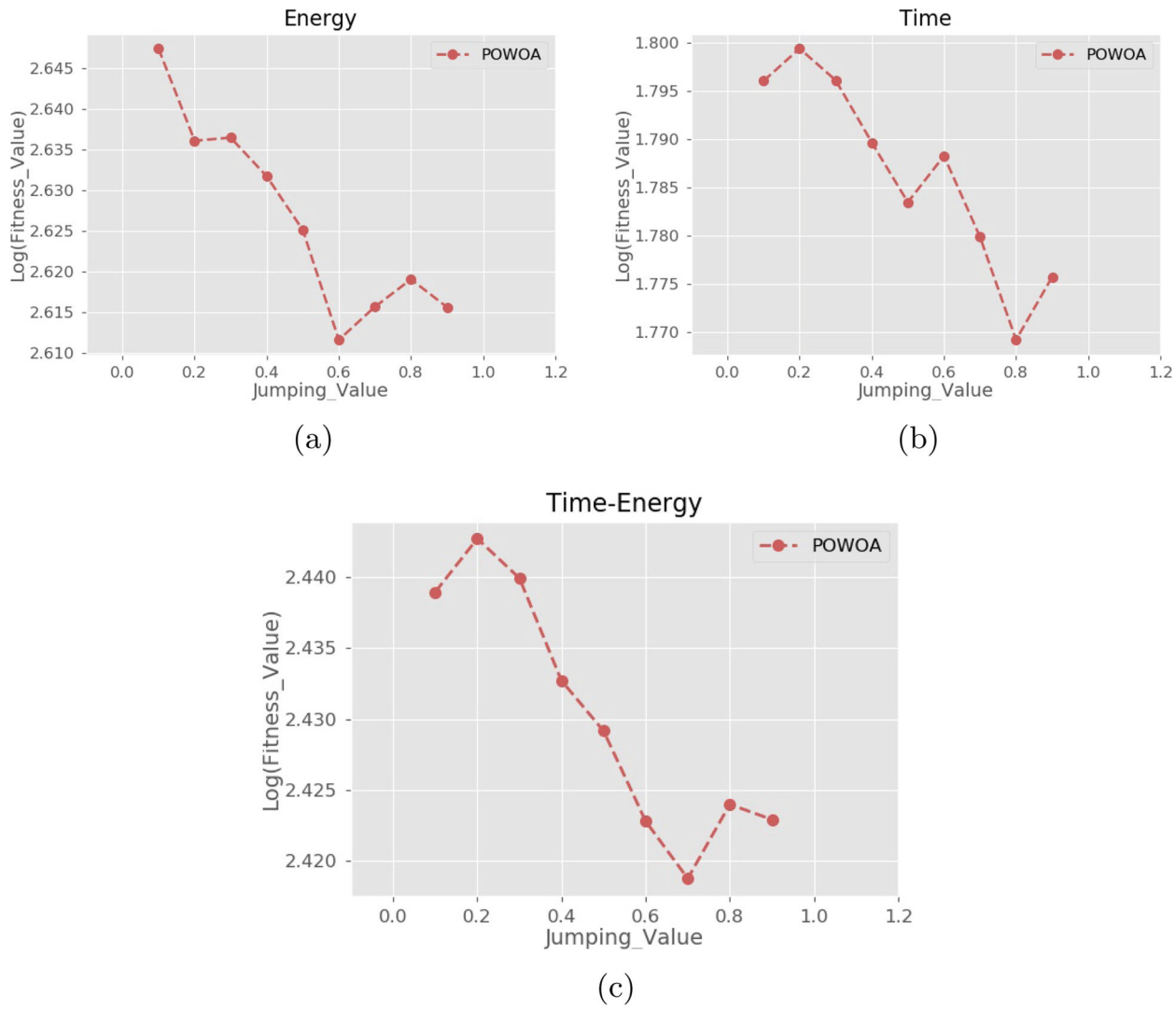


Fig. 10 Effect of jumping rate on the behavior of POWOA

the energy cost in OppoCWOA is significantly lower that the others. Furthermore, it can been seen that the convergence speed and solution precision of OppoCWOA is higher that the other compared algorithms. The results of the fitness values of the time and time-energy balance is shown in Figs. 12(b) and 12(c), respectively. Similar to the energy cost, the value of fitness decreases when incrementing the number of iterations. In terms of time cost, GA outperforms OppoCWOA.The reason for this could be that Beta2 decreases the global exploration capability of our proposed algorithm that causes our algorithm stuck in a local optimum. In terms of achieving a time-energy balance, it is noticeable that OppoCWOA outperforms other algorithms, which means our algorithm acts more accurately than other algorithms. Also, with the incrementing of iterations, OppoCWOA can get its optimal solution faster comparing to the other algorithms, which means that our proposed algorithm convergence speed is faster than other compared algorithms.

Table 11 (see Appendix) illustrates the results given from the compared algorithms for the three formulated evaluation metrics. It can be observed that regarding the energy cost, OppoCWOA yields much better results than other algorithms. In terms of time costs, OppoCWOA outperforms WOA, PSO, ABC, but cannot outperform GA. As shown in the table, for balanced time-energy costs, OppoCWOA provides significantly better results and better solution precision comparing to the other algorithms. It can be seen that GA is ranked first for the time metric and second for energy and time-energy cost metrics.

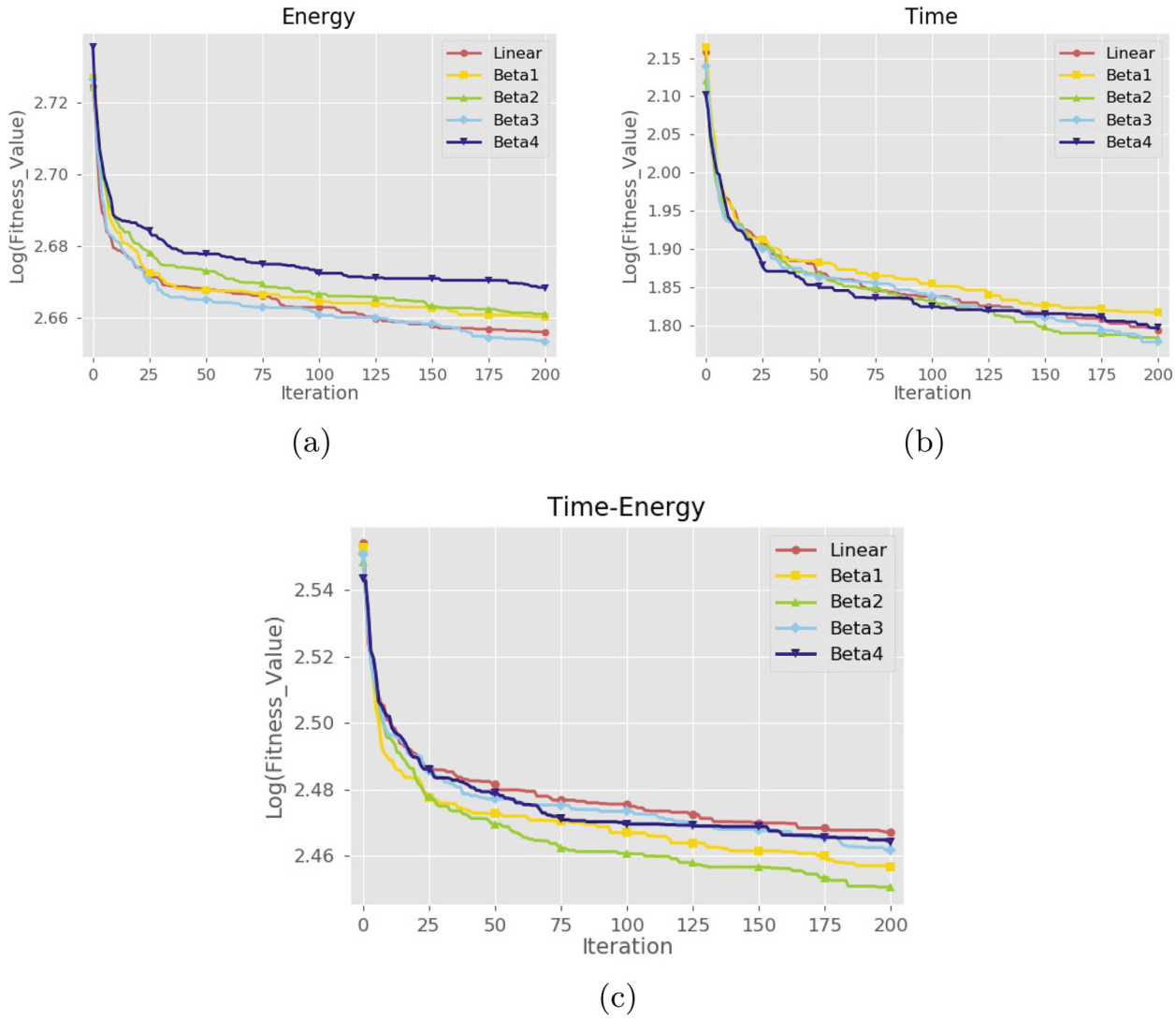


Fig. 11 Performance comparison of 4 non-linear functions with three evaluation metrics

Also, the GA algorithm performance is better compared to WOA, PSO, and ABC.

Scenario 2: In this scenario, the number of tasks are gradually increased from 50 to 300. Figure 13 presents the performance analysis of the five algorithms for this scenario. In regards to the energy cost, OppoCWOA provides better results comparing to the other algorithms (Fig. 13(a)). This implies that the proposed algorithm keeps the energy consumption low under the high load of tasks. Also, the ABC algorithm has a worse performance among others in a high load of tasks while in the contrary case, its performance is ranked second after OppoCWOA. This means the ABC algorithm is not suitable for our fog environments where a high number of tasks should be scheduled. On the other hand, GA performance ranks on the second position for a high load of tasks, indicating that GA is more suitable than ABC, PSO, WOA for our fog environment. In Fig. 13(b), we can see that OppoCWOA outperforms others in a high load of tasks regarding the time cost. In accordance with the results, GA acts well only for low number of tasks, meaning that GA loses its advantage over OppoCOWA when increasing the number of tasks. In view of time-energy balance cost, Fig. 13(c) illustrates the superiority of the proposed OppoCWOA comparing to the other algorithms for task scheduling in the fog computing environment.

Conclusion and future works

In this paper, we proposed a WOA-based task scheduling algorithm for fog computing environments named OppoCWOA by using opposition-based learning with

Table 4 OppoCOWA setting

Method	Value
Chaotic map for $\vec{r}$	Circle
Chaotic map for p-value	Sinusoidal
Opposition-based learning type	Partial
Non-linear function for $\vec{a}$	Beta2
Jumping rate	0.7

jumping rate, and chaos theory to achieve time-energy efficiency. In the proposed method, we used partial-opposition instead of full-opposition that increases the population diversity. This latter improves the performance of task scheduling while achieving faster convergence. We studied the effects of 11 different chaotic maps on two different parameters ($\vec{r}$, and p) of WOA. We presented four different non-linear functions to replace the linear function in WOA and compared their effects on the algorithm. Also, to prevent premature convergence that can be caused by utilizing opposition-based techniques, we employed jumping rate and studied its effects on the algorithm. We have reported the effects of different jumping rate values within the interval $[0.1, 0.9]$ and demonstrated that any jumping rate value within the interval $[0.6, 0.8]$ is suitable for our OppoCWOA. The simulation results indicated that the proposed algorithm considerably outperformed the original Whale Optimization Algorithm (WOA), Artificial Bees Colony (ABC), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) in terms of achieving time-energy efficient task scheduling.

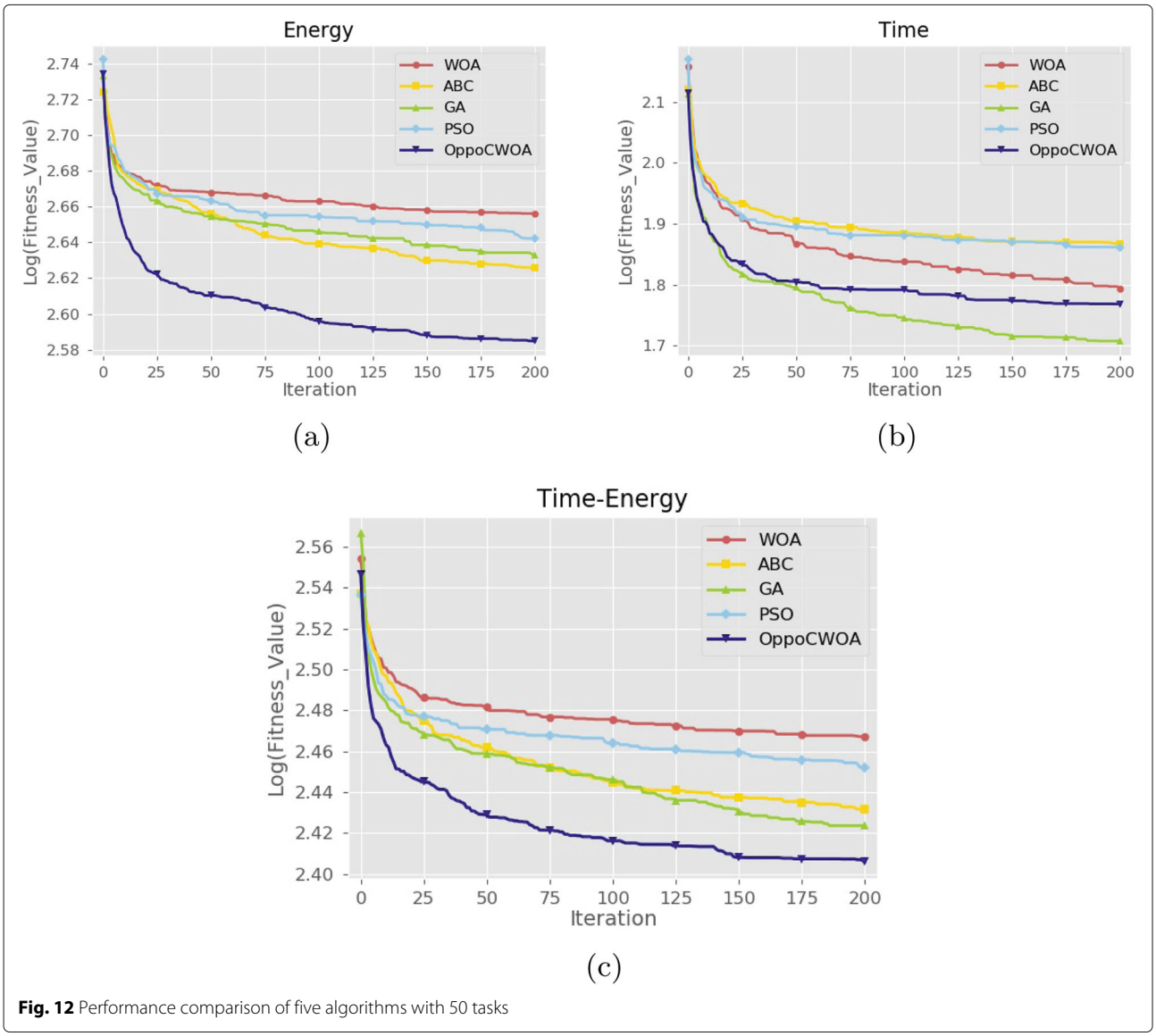


Fig. 12 Performance comparison of five algorithms with 50 tasks

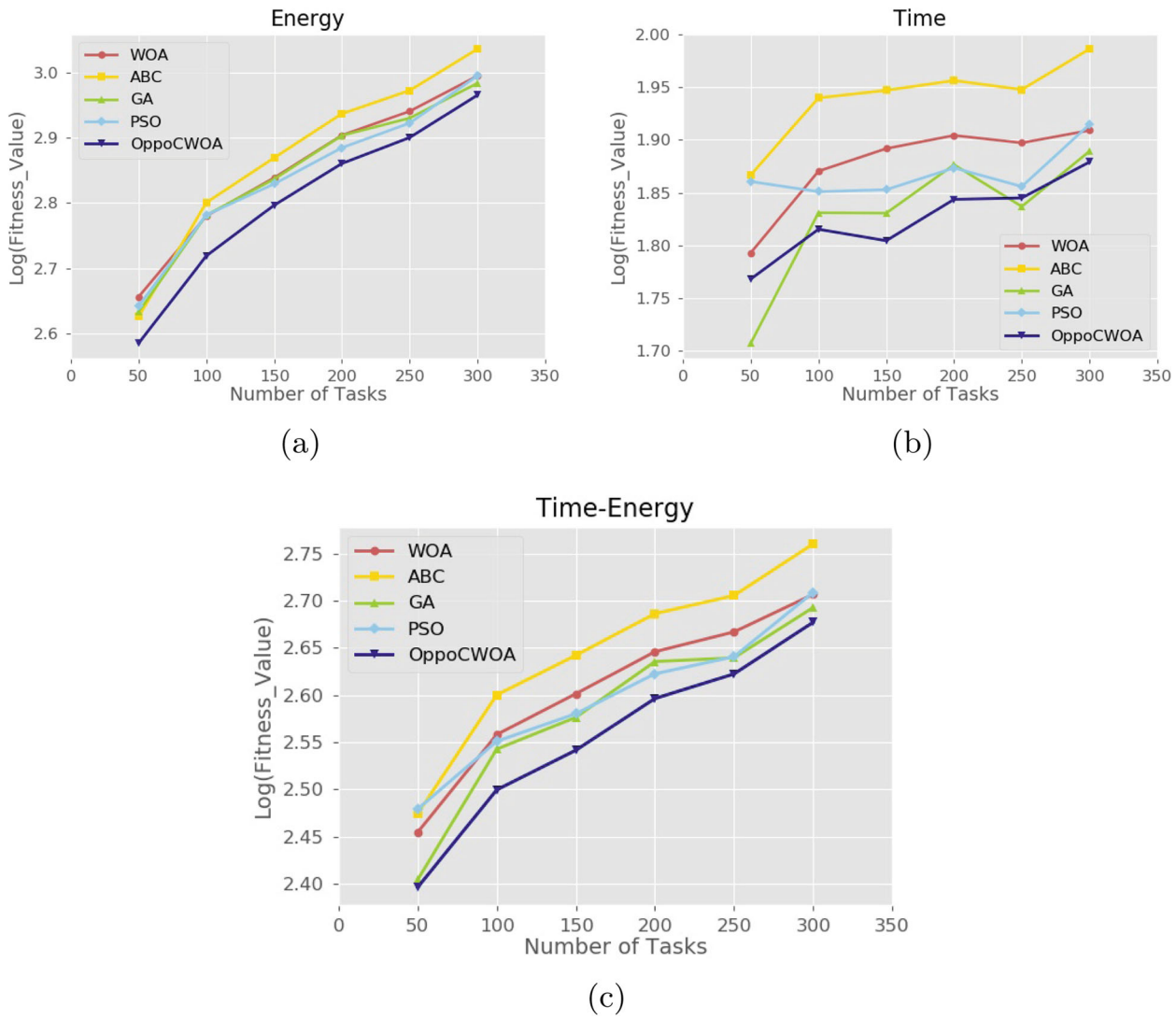


Fig. 13 Performance comparison of five algorithms from 50 to 300 tasks

This work opens the way to multiple extensions. Here we cite some of the most promising. First, we plan to overcome some limitations in the current version of the simulator. For instance, we did not consider the situation in which executing a task is failed on a fog node (due to the damage of the CPU, for instance). One solution to overcome this limitation is to allocate another fog node at the next time slot. However, this solution may exceed the deadline of the task. Another limitation is that tasks are implemented as simple objects which are independent to each other. We aim to consider complex tasks that generally structured as a graph of subtasks (or jobs) deployed and executed on different fog nodes (either in the fog or the cloud layer, depending on their type). This latter requires data synchronization of complex tasks at some stage. Another limitation is the static aspect of the network, i.e., IoT devices or/and fog nodes may be mobile. This latter requires implementing the task migration techniques between fog/cloud nodes. Second, we desire to simulate the proposed system model and OppoCWOA using open-source simulators such as Cloudsim (or alike) to align with the existing well-known and popular platforms. This latter is possible using the container and scheduler interfaces in cloudsim to provide the implementation of our model. Furthermore, we plan to investigate data privacy in the future using the blockchain-based FogBus framework. Finally, to cope with the stochastic feature of the fog environment, we intend to apply AI-based approaches such as deep reinforcement learning to quickly learn and update the policy weights based on the fog nodes and task workload behaviors.

Appendix

Table 5 Linear and non-linear functions

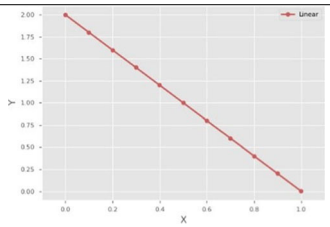
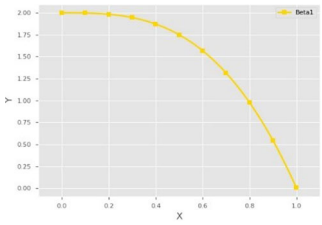
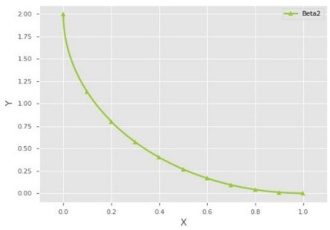
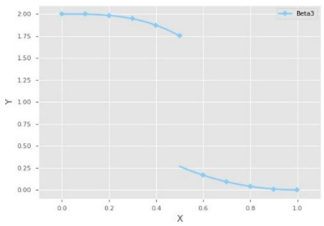
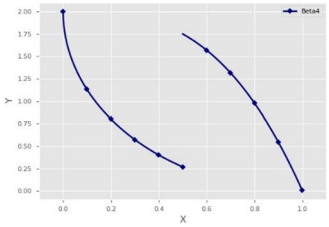
Name	Function	Possible Effect	Diagram
Linear	$y = 2 - 2x$	In the first half of the iterations, WOA is mostly devoted to global exploration, while in the other half iterations, it is mostly devoted to local exploitation.	
Beta1	$y = 2 - 2x^3$	In more than half of the iterations, WOA is devoted to global exploration (about 4/5 of the iterations), while in the other iterations, it is devoted to local exploitation. Better global exploration capability.	
Beta2	$y = 2 - \sqrt{4 - 4(x-1)^2}$	In more than half of the iterations, WOA is devoted to local exploitation(more than 4/5 of the iterations), while in the other iterations, it is devoted to global exploration. Better local exploitation capability.	
Beta3	$y = \begin{cases} Beta1 & \text{if } x \leq 0.5 \\ Beta2 & \text{Otherwise} \end{cases}$	In the first half of the iterations, WOA is strictly devoted to global exploration, while in the other iterations, it is strictly devoted to local exploitation.	
Beta4	$y = \begin{cases} Beta2 & \text{if } x \leq 0.5 \\ Beta1 & \text{Otherwise} \end{cases}$	In the first quarter of the iterations, WOA is devoted to global exploration, in the second quarter of the iterations, it is devoted to local exploitation. In the third quarter of the iteration, it is devoted to global exploration, while in the last quarter of iterations, it is devoted to local exploitation.	

Table 6 Chaotic maps

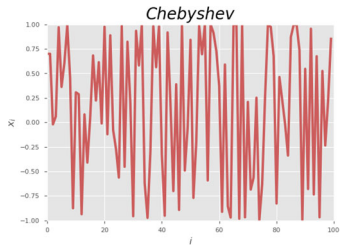
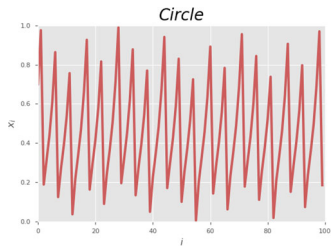
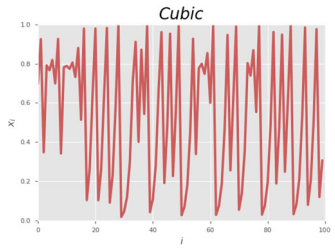
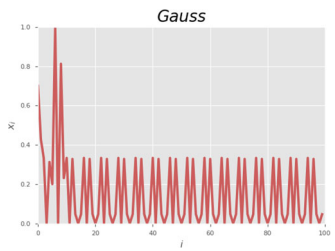
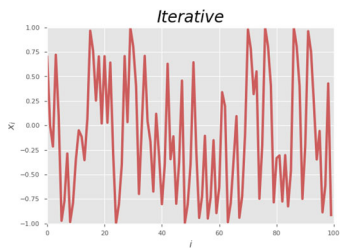
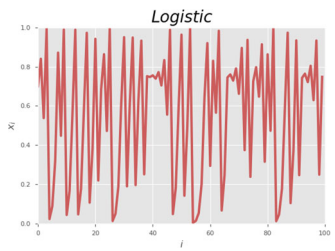
Number	Name	Chaotic Map	Diagram
1	Chebyshev [47]	$x_{(i+1)} = \cos(i cos^{-1} x_i)$ $x_0 = 0.7$	
2	Circle [48]	$x_{(i+1)} = (x_i + b - (\frac{a}{2\pi}) \sin(2\pi x_i)) \mathrm{mod}(1)$ $a = 0.5, b = 0.2, x_0 = 0.7$	
3	Cubic [49]	$x_{(i+1)} = 2.59 x_i (1 - x_i^2)$ $x_0 = 0.7$	
4	Guass [50]	$x_{(i+1)} = \begin{cases} 0 & \text{if } x_i = 0 \\ \frac{1}{x_i} \mathrm{mod}(1) & \text{if } x_i \neq 0 \end{cases}$ $x_0 = 0.7$	
5	Iterative [51]	$x_{(i+1)} = \sin(\frac{a\pi}{x_i})$ $a = 0.7, x_0 = 0.7$	
6	Logistic [51]	$x_{(i+1)} = a x_i (1 - x_i)$ $a = 4, x_0 = 0.7$	

Table 6 Chaotic maps (*Continued*)

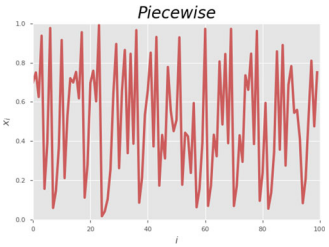
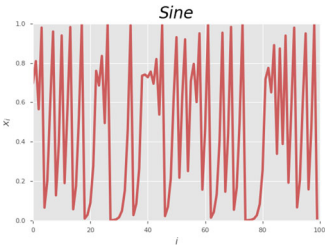
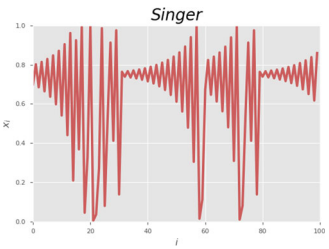
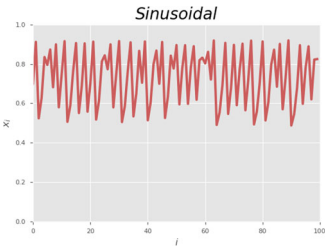
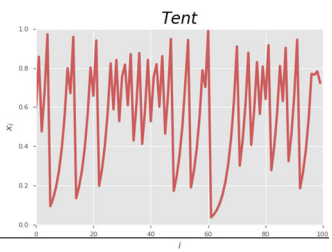
Number	Name	Chaotic Map	Diagram
7	Piecewise [52]	$x_{(i+1)} = \begin{cases} x_i/p & \text{if } x_i < p \\ x_i-p/0.5-p & \text{if } p \le x_i \le 0.5 \\ 1-p-x_i/0.5-p & \text{if } 0.5 \le x_i \le 1-p \\ 1-x_i/p & \text{if } 1-p < x_i \end{cases}$ $p = 0.4, x_0 = 0.7$	
8	Sine [53]	$x_{(i+1)} = \frac{a}{4}\sin(\pi x_i)$ $a = 0.4, x_0 = 0.7$	
9	Singer [54]	$x_{(i+1)} = \mu(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4)$ $\mu = 1.073, x_0 = 0.7$	
10	Sinusoidal [55]	$x_{(i+1)} = ax_i 2\sin(\pi x_i)$ $a = 2.3, x_0 = 0.7$	
11	Tent [56]	$x_{(i+1)} = \begin{cases} x_i/0.7 & \text{if } x_i < 0.7 \\ 10/3(1-x_i) & \text{otherwise} \end{cases}$	

Table 7 Results of 11 chaotic maps on all evaluation metrics on CWOA for $\vec{r}$ value

Evaluation metric	Statistics	Chebyshev	Circle	Cubic	Gauss	Iterative	Logistic	Piecewise	Sine	Singer	Sinusoidal	Tent	No Chaos
Energy	*Max*	510.06	509.23	511.9	498.18	514.31	493.91	502.92	502.42	492.55	519.93	511.47	502.14
	Min	386.96	380.17	392.38	424.6	407.76	321.38	396.5	417.73	411.97	425.55	398.06	383.84
	Mean	459.65	448.47	460.6	460.02	455.45	459.49	460.52	458.4	453.51	464.95	464.71	453.07
	Median	457.38	453.35	462.38	463.46	458.07	460.96	470.33	458.74	455.46	463.81	469.1	449.4
Time	*Max*	85.9	68.72	86.31	79.24	84.6	81.01	79.06	78.59	74.4	93.38	75.28	81.29
	Min	48.53	47.61	54.44	49.76	51.93	48.27	50.11	47.63	42.06	49.61	50.14	47.02
	Mean	66.79	62.08	65.97	64.52	64.54	64.08	66.34	66.09	61.09	67.78	64.66	62.1
	Median	66.35	64.12	64.51	62.4	64.13	66.78	67.91	68.77	63.15	66.42	65.89	61.07
Time – Energy	*Max*	320.87	319.92	324.16	322.13	327.1	317.75	322.5	318.69	325.55	321.24	322.17	328.25
	Min	260.07	255.78	245.8	258.96	255.31	240.15	257.41	255.94	264.26	265.66	268.85	250.68
	Mean	291.66	288.13	293.37	290.49	291.61	287.43	292.8	291.62	300.25	293.47	296.47	293.23
	Median	292.5	288.05	291.81	292.97	293.3	291.67	297.06	293.12	301.66	290.02	296.9	289.83

Table 8 Results of 11 chaotic maps on all evaluation metrics on CWOA for *p* value

Evaluation metric	Statistics	Chebyshev	Circle	Cubic	Gauss	Iterative	Logistic	Piecewise	Sine	Singer	Sinusoidal	Tent	No Chaos
Energy	*Max*	482.41	478.04	486.95	508.69	505.02	482.45	495.37	484.4	481.73	490.62	484.81	502.14
	Min	406.34	406.41	403.43	402.02	393.84	373.87	402.71	406.95	301.87	389.03	390.01	383.84
	Mean	442.69	449.83	451.01	462.3	454.78	442.55	452.13	440.58	437.02	440.43	442.63	453.07
	Median	443.7	448.62	450.81	457.02	452.55	447.64	456.14	438.3	449.24	437.93	446.14	449.4
Time	*Max*	70.35	74.05	72.23	85.28	84.33	68.08	70.13	74.35	68.23	68.79	69.07	81.29
	Min	49.2	52.87	49.63	49.07	44.32	47.6	46.56	46.33	39.88	44.19	49.77	47.02
	Mean	60.39	63.09	60.35	66.97	66.2	59.51	60.98	61.25	59.06	57.46	58.76	62.1
	Median	61.24	61.48	58.93	64.48	65.65	60.85	62.53	60.39	60.29	58.2	58.51	61.07
Time – Energy	*Max*	320.7	331.87	316.66	317.69	328.24	314.08	306.96	320.77	311.32	311.54	316.26	328.25
	Min	256.13	238.4	257.08	266.88	273.77	231.86	256.23	244.74	245.91	242.62	248.11	250.68
	Mean	283.03	286.28	282.8	296.14	295.26	282.1	284.8	288.49	282.63	274.38	286.73	293.23
	Median	283.06	285.69	278.54	298.03	293.63	278.79	284.72	291.91	288.64	276.44	292.96	289.83

Table 9 Results of 4 OBL techniques on all evaluation metrics on WOA

Evaluation metric	Statistics	WOA	OWOA	POWOA	QOWOA	SOWOA
Energy	*Max*	502.15	491.83	480.75	502.31	501.41
	Min	383.84	405.04	355.72	378.69	403.88
	Mean	453.08	450.01	417.64	454.37	446.44
	Median	449.4	450.98	412.11	468.51	446.3
Time	*Max*	81.3	70.77	79.84	79.9	77.27
	Min	47.02	54.31	56.93	50.39	43.26
	Mean	62.1	63.47	62.67	64.23	59.68
	Median	61.07	63.38	60.67	62.69	60.83
Time – Energy	*Max*	328.26	319.08	310.84	340.33	313.31
	Min	250.68	233.09	232.06	223.64	264.67
	Mean	293.23	283.87	269.32	293.15	285.84
	Median	289.83	281.56	267.41	297.39	285.54

Table 10 Results of 4 non-linear functions on all evaluation metrics

Evaluation metric	Statistics	Linear	Beta1	Beta2	Beta3	Beta4
Energy	*Max*	502.15	504.37	484.95	485.05	514.8
	Min	383.84	413.63	407.31	412.51	421.54
	Mean	453.08	457.48	458.31	452.52	466.08
	Median	449.4	457.95	458.16	454.71	478.06
Time	*Max*	81.3	80.07	75.23	74.73	74.72
	Min	47.02	47.73	44.96	45.07	50.95
	Mean	62.01	65.59	60.75	60.49	62.6
	Median	61.07	66.34	62.28	60.71	63.43
Time – Energy	*Max*	328.26	315.33	303.52	321.92	316.23
	Min	250.68	252.55	265.21	258.88	241.04
	Mean	293.23	286.33	282.26	289.41	291.28
	Median	289.83	288.22	280.92	287.15	292.28

Table 11 Results of 5 task scheduling algorithms on all evaluation metrics

Evaluation metric	Statistics	WOA	PSO	ABC	GA	OppoCWOA
	Max	502.15	467.75	444.95	447.1	441.12
	Min	383.84	368.91	400.89	408.66	309.87
Energy	*Mean*	453.08	438.39	422.34	429.58	384.48
	Median	449.4	441.16	417.74	432.66	386.25
	Max	81.3	87.09	81.97	62.2	72.94
	Min	47.02	56.28	62.32	41.55	40.23
Time	*Mean*	62.01	72.56	73.6	50.97	58.64
	Median	61.07	72.2	74.11	51.22	59.54
	Max	328.26	303.09	287.19	281.64	306.91
	Min	250.68	255.13	244.42	249.17	214.58
Time – Energy	*Mean*	293.23	282.45	270.2	265.25	254.7
	Median	289.83	286.18	271.14	268.46	251.82

Acknowledgments
Not applicable.

Authors' contributions
This research paper has been done through a concerted effort by three authors. Hence, any author has participated in conducting every single part of the paper. But each author's basic role has been summarizing in the following: Z.M. is the corresponding author, the coordinator of the group and the designer of the proposed model and method. B.D. is the main reviewer of the paper and assisted Z.M for the model design. A.M.H. is the responsible for experimentation of the proposed method with support from Z.M and B.D. All authors have read and agreed to the published version of the manuscript.

Authors' information
Bruno Defude is a Professor in the Computer Science Department and member of the ACMES group of SAMOVAR in Telecom SudParis, France. He is also Deputy Dean of Research and Doctoral Studies of TELECOM SudParis, school of institut Mines-Telecom and component of Institut Polytechnique de Paris. His current research themes are semantic web (ontologies for provenance management, personnalised access to documents), distribution (P2P information retrieval systems, data management for VANET), data and service management at large scale (cloud computing, NoSQL database, big data). He is a member of the editorial board of two French journals, I3 (http://revue-i3.org) and RSTI/ISI (Ingéniérie des Systèmes d'Informations). Zahra Movahedi received her M.S. and Ph.D. degrees in Artificial Intelligence from University of Pierre and Marie Curie (UPMC-Sorbonne Universite) in 2011 and University of Paris Saclay in 2015, Paris, France, respectively. She worked as a post-doc researcher at university of Marnle-la-Vallee in 2016. Since 2017, she is a faculty member at University of Tehran, Tehran, Iran. She served as the head or collaborator for a large number of international research and industrial projects, including French Zodianet company representative for FP7 European project of VITRO (Virtualized dIstributed plaTfoRms of smart Objects) and LIGM (Laboratoire d'Informatique Gaspard-Monge University of Marne-la-vallee) representative for ITEA3 European project of SITAC (Social Internet of things: Apps by and for the Crowd). Her research interests include IoT, Fog and cloud computing, Multi-agents system, business process and distributed algorithms. Amir Mohammad Hosseininia is a M.Sc. student at the university of Tehran. He received the B.Sc. degree from the College of Farabi, University of Tehran, Tehran, Iran, in 2019. His current research interests are in the field of evolutionary computation, particularly in the areas of multi-criterion and real-parameter evolutionary algorithms, machine learning, neural networks, cloud and fog computing.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Competing interests
Not applicable.

Author details
[1]Department of Engineering, College of Farabi, University of Tehran, Tehran, Iran. [2]SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France.

Received: 12 January 2021 Accepted: 30 August 2021
Published online: 07 October 2021

References

1. Mehmood Y, Ahmad F, Yaqoob I, Adnane A, Imran M, Guizani S (2017) Internet-of-things-based smart cities: Recent advances and challenges. IEEE Commun Mag 55(9):16–24. https://doi.org/10.1109/MCOM.2017.1600514
2. Hosseini Bidi A, Movahedi Z, Movahedi Z A fog-based fault-tolerant and qoe-aware service composition in smart cities. Trans Emerg Telecommun Technol. https://doi.org/10.1002/ett.4326. https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.4326
3. Islam S. M. R, Kwak D, Kabir MH, Hossain M, Kwak K (2015) The internet of things for health care: A comprehensive survey. IEEE Access 3:678–708. https://doi.org/10.1109/ACCESS.2015.2437951
4. Stojkoska BLR, Trivodaliev KV (2017) A review of internet of things for smart home: Challenges and solutions. J Clean Prod 140:1454–1464. https://doi.org/10.1016/j.jclepro.2016.10.006
5. Pochet Y, Wolsey LA Production Planning by Mixed Integer Programming. Springer Series in Operations Research and Financial Engineering. Springer, New York
6. Ullman JD (1975) Np-complete scheduling problems. J Comput Syst Sci 10(3):384–393. https://doi.org/10.1016/S0022-0000(75)80008-0
7. Hosseinioun P, Kheirabadi M, Kamel Tabbakh SR, Ghaemi R atask scheduling approaches in fog computing: A survey. Trans Emerg Telecommun Technol n/a(n/a):3792. https://doi.org/10.1002/ett.3792. e3792 ETT-19-0285.R1. https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3792
8. Hong C-H, Varghese B (2019) Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. ACM Comput Surv 52(5). https://doi.org/10.1145/3326066
9. Mach P, Becvar Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun Surv Tutor 19(3):1628–1656. https://doi.org/10.1109/COMST.2017.2682318
10. Kumar M, Sharma SC, Goel A, Singh SP (2019) A comprehensive survey for scheduling techniques in cloud computing. J Netw Comput Appl 143:1–33. https://doi.org/10.1016/j.jnca.2019.06.006
11. Kalra M, Singh S (2015) A review of metaheuristic scheduling techniques in cloud computing. Egypt Inform J 16(3):275–295. https://doi.org/10.1016/j.eij.2015.07.001
12. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
13. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
14. Tizhoosh HR (2005) Opposition-based learning: A new scheme for machine intelligence. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), vol. 1. pp 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
15. Tizhoosh HR, Ventresca M, Rahnamayan S (2008). In: Tizhoosh HR, Ventresca M (eds). Opposition-Based Computing. Springer, Berlin, Heidelberg. pp 11–28. https://doi.org/10.1007/978-3-540-70829-2_2
16. Ikeguchi T, Hasegawa M, Kimura T, Matsuura T, Aihara K (2011). In: Nedjah N., dos Santos Coelho L, Mariani VC, de Macedo Mourelle L (eds). Theory and Applications of Chaotic Optimization Methods. Springer, Berlin, Heidelberg. pp 131–161. https://doi.org/10.1007/978-3-642-20958-1_8
17. Barros C, Rocio V, Sousa A, Paredes H (2020) Survey on job scheduling in cloud-fog architecture. In: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI). pp 1–7. https://doi.org/10.23919/CISTI49556.2020.9141156
18. Matrouk K, Alatoun K (2021) Scheduling algorithms in fog computing: A survey. Int J Netw Distrib Comput 9:59–74. https://doi.org/10.2991/ijndc.k.210111.001
19. Wang T, Liu Z, Chen Y, Xu Y, Dai X (2014) Load balancing task scheduling based on genetic algorithm in cloud computing. In: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. pp 146–152. https://doi.org/10.1109/DASC.2014.35
20. Abdi S, Motamedi SA, Sharifian S, et al. (2014) Task scheduling using modified PSO algorithm in cloud computing environment. In: International conference on machine learning, electrical and mechanical engineering, vol 4, issue 1. pp 8–12
21. Hasan MZ, Al-Rizzo H, Al-Turjman F, Rodriguez J, Radwan A (2018) Internet of things task scheduling in cloud environment using particle swarm optimization. In: 2018 IEEE Global Communications Conference (GLOBECOM). pp 1–6. https://doi.org/10.1109/GLOCOM.2018.8647917
22. Kimpan W, Kruekaew B (2016) Heuristic task scheduling with artificial bee colony algorithm for virtual machines. In: 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS). pp 281–286. https://doi.org/10.1109/SCIS-ISIS.2016.0067
23. Chen X, Cheng L, Liu C, Liu Q, Liu J, Mao Y, Murphy J (2020) A woa-based optimization approach for task scheduling in cloud computing systems. IEEE Syst J:1–12. https://doi.org/10.1109/JSYST.2019.2960088

24. Thennarasu SR, Selvam M, Srihari K (2020) A new whale optimizer for workflow scheduling in cloud computing environment. J Ambient Intell Humanized Comput. https://doi.org/10.1007/s12652-020-01678-9
25. Ghobaei-Arani M, Souri A, Safara F, Norouzi M (2020) An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. Trans Emerg Telecommun Technol 31(2):3770. https://doi.org/10.1002/ett.3770. e3770 ETT-18-0545.R2
26. Rahbari D, Nickray M (2017) Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In: 2017 21st Conference of Open Innovations Association (FRUCT). pp 278–283. https://doi.org/10.23919/FRUCT.2017.8250193
27. Wang J, Li D (2019) Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. Sensors 19(5). https://doi.org/10.3390/s19051023
28. Sun Y, Lin F, Xu H (2018) Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii. Wirel Pers Commun 102(2):1369–1385. https://doi.org/10.1007/s11277-017-5200-5
29. Yang Y, Zhao S, Zhang W, Chen Y, Luo X, Wang J (2018) Debts: Delay energy balanced task scheduling in homogeneous fog networks. IEEE Internet Things J 5(3):2094–2106. https://doi.org/10.1109/JIOT.2018.2823000
30. Wang T, Wei X, Tang C, Fan J (2018) Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. Peer-to-Peer Netw Appl 11:793–807
31. Nguyen BM, Thi Thanh Binh H, The Anh T, Bao Son D (2019) Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment. Appl Sci 9(9). https://doi.org/10.3390/app9091730
32. Bitam S, Zeadally S, Mellouk A (2018) Fog computing job scheduling optimization based on bees swarm. Enterp Inf Syst 12(4):373–397. https://doi.org/10.1080/17517575.2017.1304579
33. Tuli S, Gill S, Casale G, Jennings N (2020) ithermofog: Iot-fog based automatic thermal profile creation for cloud data centers using artificial intelligence techniques. Internet Technol Lett. https://doi.org/10.1002/itl2.198
34. Tuli S, Ilager S, Ramamohanarao K, Buyya R (2020) Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. IEEE Trans Mob Comput:1–1. https://doi.org/10.1109/tmc.2020.3017079
35. Xie R, Tang Q, Liang C, Yu FR, Huang T (2021) Dynamic computation offloading in iot fog systems with imperfect channel-state information: A pomdp approach. IEEE Internet Things J 8(1):345–356. https://doi.org/10.1109/JIOT.2020.3004223
36. Hazra A, Adhikari M, Amgoth T, Srirama SN (2020) Joint computation offloading and scheduling optimization of iot applications in fog networks. IEEE Trans Netw Sci Eng 7(4):3266–3278. https://doi.org/10.1109/TNSE.2020.3021792
37. Shi Y, Chen S, Xu X (2018) Maga: A mobility-aware computation offloading decision for distributed mobile cloud computing. IEEE Internet Things J 5(1):164–174. https://doi.org/10.1109/JIOT.2017.2776252
38. Yu F, Chen H, Xu J (2018) Dynamic mobility-aware partial offloading in mobile edge computing. Futur Gener Comput Syst 89:722–735. https://doi.org/10.1016/j.future.2018.07.032
39. Ghosh S, Mukherjee A, Ghosh SK, Buyya R (2020) Mobi-iost: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. IEEE Trans Netw Sci Eng 7(4):2271–2285
40. Pham Q, Mirjalili S, Kumar N, Alazab M, Hwang W (2020) Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Trans Veh Technol 69(4):4285–4297. https://doi.org/10.1109/TVT.2020.2973294
41. Gharehchopogh FS, Gholizadeh H (2019) A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm Evol Comput 48:1–24. https://doi.org/10.1016/j.swevo.2019.03.004
42. Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423
43. Strumberger I, Bacanin N, Tuba M, Tuba E (2019) Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. Appl Sci 9(22). https://doi.org/10.3390/app9224893
44. Rahnamayan S, Wang GG (2009) Center-based sampling for population-based algorithms. In: 2009 IEEE Congress on Evolutionary Computation. pp 933–938. https://doi.org/10.1109/CEC.2009.4983045
45. Hu Z, Bao Y, Xiong T (2014) Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp 2259–2265. https://doi.org/10.1109/CEC.2014.6900489
46. Tang R, Fong S, Dey N (2018) Metaheuristics and Chaos Theory. https://doi.org/10.5772/intechopen.72103
47. Wang N, Liu L, Liu L (2001) Genetic algorithm in chaos. OR Trans 5:1–10
48. Zheng W-M (1994) Kneading plane of the circle map. Chaos, Solitons Fractals 4(7):1221–1233
49. Rogers TD, Whitley DC (1983) Chaos in the cubic mapping. Math Model 4(1):9–25
50. Jothiprakash V, Arunkumar R (2013) Optimization of hydropower reservoir using evolutionary algorithms coupled with chaos. Water Resour Manag 27(7):1963–1979
51. He D, He C, Jiang L-G, Zhu H-w, Hu G-r (2001) Chaotic characteristics of a one-dimensional iterative map with infinite collapses. IEEE Trans Circ Syst I: Fundam Theory Appl 48(7):900–906
52. Saremi S, Mirjalili SM, Mirjalili S (2014) Chaotic krill herd optimization algorithm. Procedia Technol 12(1):180–185
53. Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H (2014) Chaotic krill herd algorithm. Inf Sci 274:17–34
54. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
55. Barton R (1990) Chaos and fractals. Math Teach 83(7):524–529
56. Bhattacharya A, Chattopadhyay PK (2010) Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch. IEEE Trans Power Syst 25(4):1955–1964

Publisher's Note