mathematics

Article

Parallel Meta-Heuristics for Solving Dynamic Offloading in Fog Computing

Samah Ibrahim AlShathri , Samia Allaoua Chelloug * and Dina S. M. Hassan

Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 84428, Saudi Arabia;
sealshathry@pnu.edu.sa (S.I.A.); dshassan@pnu.edu.sa (D.S.M.H.)
* Correspondence: sachelloug@pnu.edu.sa

Abstract: The internet of things (IoT) concept has been extremely investigated in many modern smart applications, which enable a set of sensors to either process the collected data locally or send them to the cloud for remote processing. Unfortunately, cloud datacenters are located far away from IoT devices, and consequently, the transmission of IoT data may be delayed. In this paper, we investigate fog computing, which is a new paradigm that overcomes many issues of cloud computing. More importantly, dynamic task offloading in fog computing is a challenging problem that requires an optimal decision for processing the tasks that are generated in each time slot. Thus, exact optimization methods based on Lyapunov function have been widely used for solving dynamic offloading which represents an NP hard problem. To overcome the scalability issue of exact optimization techniques, we have explored famous population based meta-heuristics for optimizing the offloading process of a set of dynamic tasks using Orthogonal Frequency Division Multiplexing (OFDM) communication. Hence, a parallel multi-threading framework is proposed for generating the optimal offloading solution while selecting the best sub-carrier for each offloaded task. More importantly, our contribution associates a thread for each IoT device and generates a population of random solutions. Next, each population is updated and evaluated according to the proposed fitness function that considers a tradeoff between the delay and energy consumption. Upon the arrival of new tasks at each time slot, an evaluation is performed for maintaining some individuals of the previous population while generating new individuals based on some criteria. Our results have been compared to the results achieved using Lyapunov optimization. They demonstrate the convergence of the fitness function, the scalability of the parallel Particle Swarm Optimization (PSO) approach, and the performance in terms of the offline error and the execution cost.

Keywords: fog computing; dynamic offloading; IoT; cloud computing; meta-heuristics; parallel genetic algorithm; parallel PSO; multi-threading; offline error

MSC: 65E05; 90B18; 90B22

check for updates

Citation: AlShathri, S.I.; Chelloug, S.A.; Hassan, D.S.M. Parallel Meta-Heuristics for Solving Dynamic Offloading in Fog Computing. *Mathematics* **2022**, *10*, 1258. https://doi.org/10.3390/math10081258

Academic Editor: Ioannis G. Tsoulos

Received: 8 February 2022
Accepted: 4 April 2022
Published: 11 April 2022

1. Introduction

The evolution of computing paradigms improves the Information Technology (IT) practices in terms of design, development, and deployment. From this perspective, cloud computing has provided IT service models where customers benefit from hardware and software computing resources on demand through a network [1]. Despite the advantages of cloud computing, there exist some challenges that need to be considered before embracing cloud computing as a viable solution. These challenges include security threats along with the design of the optimal decision rules for moving applications to the cloud [1]. Additionally, the IoT concept allows real world life things to sense, communicate and act remotely through the Internet [2]. However, most IoT applications incorporate different IoT sensors, which generate large volumes of data that usually require a swift analysis; today's cloud models do not handle the size, the variety, or the velocity of IoT generated data [3]. As an example, most critical healthcare applications require the real-time analysis of patients' vital signs, whereas moving data from IoT devices to the cloud for further analysis increases the latency. In addition, IoT devices have limited computing, storage, networking, and energy resources that introduce additional design challenges, which involve an adaptation of IoT architecture [2]. More specifically, the problem of moving data and applications to the cloud is known as offloading. Recently, many research efforts have been dedicated for developing effective mathematical models for deciding on task offloading. One of the ideas consists of investigating fog computing, which has recently emerged as an efficient computing model that manages data and applications and performs computation in fog servers that are close to the end users. In this manner, network latency will be reduced through the allocation of some tasks to the fog layer. The concept of fog computing was first proposed by Cisco in 2012 to improve the performance of IoT applications in terms of a network's congestion, latency, and quality of service (QoS) [4]. The idea of fog computing consists of deploying a set of traditional networking components, e.g., base stations, routers, proxy servers, etc., at the proximity of IoT devices [4]. As reported in [2,4], edge computing, micro-cloudlet, mobile cloud computing (MCC), and mobile edge computing (MEC) are terms related to fog computing that provide services to end users via an extremely distributed layer [2,4]. Despite the advantages of fog computing, some potential concerns have been addressed in some research papers. The integration of Software Defined Networks (SDN) and (Network Functions Virtualization) NFV to fog computing is appropriate for large-scale IoT applications [5], although the integration of NFV and SDN to fog computing includes some design challenges along with the re-design of the south-bound, north-bound, and east–west Application Programming Interfaces (APIs). Another research topic concerns QoS in fog computing, which can be fixed through the dynamic selection of fog nodes, check-pointing, re-scheduling, data placement, and the prediction of future requests [5]. Additionally, fog nodes are vulnerable to potential security attacks [4]. Therefore, authentication and privacy techniques should be implemented while maintaining QoS requirements [4]. Furthermore, fog application management is another research target. Moreover, dynamic offloading that supports the arrival of new requests is a hard problem. The big difficulty for dynamic optimization is how to track the optimum that may change over time. As indicated in [6], the information from the previous environment should be considered after a change has occurred for accelerating the optimization process. We mention also that the authors of [6,7] have comprehensively explained how to apply meta-heuristics including genetic algorithms, PSO, and ant colony optimization (ACO) for dynamic optimization. This paper focuses on designing parallel approaches for dynamic offloading in fog computing using two famous population based meta-heuristics including genetic algorithm and PSO. The major motivation behind selecting genetic algorithm and PSO concerns their utilization for solving many dynamic optimization problems. Compared to the state of the art, our contributions are summarized as follows:

- The solution presented in this paper investigates parallel meta-heuristics for dynamic offloading in fog computing. Most papers presented in the literature are based on Lyapunov optimization. Although this has proved to be an efficient approach, its application to many large scale IoT problems is challenged by the scalability bottleneck.
- This paper discusses the effectiveness of the proposed parallel meta-heuristics for handling the problem of dynamic offloading in fog computing. Our design initializes a set of random solutions that are evaluated using a bi-objective function that considers the delay and energy consumption. The aim of our work is twofold: deciding if the tasks generated in each time slot will be processed locally or offloaded to the suitable fog node for further processing and selecting the best channel of communication in case of task offloading. On one hand, our objective function is appropriate for task offloading in IoT applications that generate delay-sensitive tasks. On the other hand, our objective function is appropriate for energy-constrained IoT devices, as it depends on the processing and

the transmission energy. Furthermore, upon the generation of new tasks in each time slot, our contribution replaces the worst solutions by random ones.

- Our results show the performance in terms of the convergence of the fitness function and the offline error. Additionally, our results demonstrate that the proposed parallel meta-heuristics are applicable for a network comprised of many IoT devices, and the fitness function value decreases as the number of IoT devices increases. Another finding concerns the execution cost of parallel PSO that is lower than the execution cost of Lyapunov technique.

Thus, this paper is organized as follows. Section 2 introduces the related works. Section 3 explains the proposed mathematical model as well as the proposed approaches. Then, Section 4 illustrates our simulation results. Finally, Section 5 concludes this paper and discusses potential future work.

2. Related Works

The problem of static task offloading in fog computing has been studied in [8–11]. The aim of the proposed model consists of offloading multiple tasks to the fog while minimizing the delay and considering the deadline requirements. This idea is appropriate for large-scale applications that are delay-sensitive. Furthermore, the model of [8] assumes that fog nodes are able to process a part of each task and may offload the remaining part to one of the neighbouring fog nodes to satisfy the deadline requirements. Thus, the model presented in [8] aims to find the optimal offloading decision while considering computational resource allocation. A heuristic solution has been proposed in [8] to solve the quadratic constraint-programming problem. The simulation results in [8] show a trade-off between the deadline and the delay violation. The framework presented in [9] incorporates an IoT, a fog and a cloud layer. It allows a domain of IoT devices to communicate to a domain of fog nodes, which is associated with a set of cloud servers. In particular, the framework of [9] supports cooperation between fog nodes of the same domain, such that a fog node can forward a request to a neighbouring fog node that belongs to the same domain. Additionally, the model of [9] distinguishes between light and heavy processing tasks for deciding on local task processing or offloading. More importantly, the authors of [9] have developed an analytical model that minimizes the delay even if some parameters have been changed. Paper [10] has introduced a formal model that enables fog collaboration. Moreover, an algorithm has been developed in [10] to decide on the best opportunity for offloading as well as the data that should be offloaded. The algorithm in [10] determines the congestion level of a fog node based on its queue size and its queue service rate. Therefore, the algorithm in [10] detects the services causing an overload. These latter will be offloaded to another fog node. The evaluation results in [10] have been compared to two benchmarks and they show that the proposed algorithm achieves the lowest latency. The authors of [11] have presented an alternative idea that considers that each fog node includes a set of homogenous servers. The model of [11] has investigated queuing theory for formulating the average waiting time of each request. It enables minimizing energy consumption under the delay constraints. We indicate that the optimization model of [11] is non-convex, and it has been solved through an alternating method of multipliers. The results obtained in [11] demonstrate the impact of the offloading probability and the number of mobile devices on energy consumption. Recent meta-heuristics have been explored for solving challenging problems in fog computing. An improved version of the whale optimization algorithm (WOA) called opposition-based chaotic whale optimization algorithm (OppoCWOA) has been introduced in [12] for task scheduling in fog computing. The proposed meta-heuristic ensures better convergence than other benchmark techniques. Furthermore, the authors of [13] have formulated the problem of offloading in fog computing while optimizing the delay, and they have applied non-dominated sorting genetic algorithm (NSGA-II) and the bees algorithm. Thus, the applied meta-heuristics allow minimizing the delay as well as the consumed energy compared to other methods. Additionally, the authors of [14] have applied PSO and ACO for solving the problem of task offloading in fog computing.

The results obtained in [14] demonstrate the effectiveness of the proposed meta-heuristics for minimizing the delay. Moreover, task scheduling in fog computing has been solved using Harris Hawks optimization algorithm [15], which outperforms other optimization techniques. According to the state-of art on recent meta-heuristics, their application has been limited to static offloading/scheduling in fog computing.

Nevertheless, some research papers [16–27] have tackled the problem of dynamic offloading, which is more complex compared to static offloading in fog computing. Paper [16] has addressed the problem of dynamic offloading while considering a set of energy harvesting devices that may execute computational tasks locally or offload them to the mobile edge-computing server. On one hand, the proposed model in [16] is based on many variables, including the computation mode, channel gain, task arrival indicators, number of CPU cycle frequencies, transmit power, harvested energy, and battery energy level that change over time slots. On the other hand, the objective function of the model in [16] includes a weighted sum of the execution delay and the task dropping cost. In addition, the authors of [16] have defined a set of optimization constraints that comprise the battery discharging, maximum power, and CPU frequency. Moreover, the authors of [16] have investigated Lyapunov optimization technique for developing a low complexity online optimization algorithm that dynamically determines the optimal offloading decision, required energy, transmit power, and CPU frequency. The obtained results in [16] demonstrate that the proposed algorithm achieves the minimal execution cost and reduces the computation failure compared to the benchmark. The contribution presented in [17] is based on multi-user scenario including a set of MEC servers with N core CPUs. Moreover, the channel gain between the users and the MEC servers depends on the path loss and the channel fading. Additionally, the model of [17] assigns a queue to each user and allocates multiple parallel buffers to each MEC server. Further, the objective function of the model proposed in [17] consists of minimizing power consumption, and the constraints concern resource allocation and delay bound violation. We indicate that [17] has explored Lyapunov optimization for proposing an optimization algorithm that minimizes the computation and transmit power under the delay and reliability constraints. According to the results obtained in [17], the proposed method outstands benchmark methods when the task arrival rate is high. Another study that is related to dynamic offloading in fog computing has been explained in [18], which assumes a multi-tiered architecture. The main features of Predictive Offloading and Resource Allocation (PORA) [18] concern offloading between fog tiers and error prediction for optimizing task offloading. Additionally, the model formulated in [18] focuses on minimizing the long-term time average expectation of power consumption while satisfying the stability of all queues. The simulation results presented in [18] are significant and show that PORA achieves a significant trade-off between power and latency. The purpose of the work presented in [19] is to solve the problem of dynamic task offloading for multiple users, while determining the optimal power and radio resources. Thus, paper [19] assumes that each mobile device has an M/M/1 queue and that each edge node has an M/G/1 queue. More specifically, the model introduced in [19] is based on batteries that can collect energy through an energy harvesting technique to power mobile devices. On one hand, the objective function of the model of [19] consists of optimizing the delay and energy consumption. On the other hand, the constraints of the model of [19] include power consumption and channel assignment. The authors of [19] have adopted Lyapunov optimization and they have proposed an algorithm for dynamically assigning the optimal power and communication channels. As reported in [19], the number of sub-carries can affect the cost. As the number of sub-carriers increases, the cost decreases because the mobile devices will have many opportunities for offloading their tasks. The other finding in paper [19] concerns the impact of the number of mobile devices on the average cost that increases if the number of mobile devices increases too. The authors of [20] have proposed an offloading approach for vehicular networks. The aim of the approach proposed in [20] is to minimize the delay while maximizing the throughput and dynamically controlling the offloading threshold. The article [20] has considered delay-sensitive tasks that are generated by vehicle nodes. Hence, the proposed model in [20] is composed of a cloud server, fog nodes, a fog controller, and moving vehicle nodes. As any vehicle moves, it needs to be connected to a fog node. The connection process is based on the position of each vehicle and the communication coverage of each fog node. In addition, each fog node has a waiting queue for storing the arriving tasks. The aim of the model suggested in [20] is to minimize the delay and maximize the throughput while dynamically adjusting the offloading threshold. As reported in [20], the simulation results have been evaluated in terms of the energy latency and the throughput. A comparison has been made with some similar offloading techniques to demonstrate the performance of the method proposed in [20]. The main result obtained in [20] concerns the impact of increasing the number of vehicles as well as the impact of increasing the number of nodes in the neighbourhood. Different from the above models, [21] has considered a model including one IoT device and many fog nodes. Moreover, the IoT device has the ability to perform simultaneous offloading to fog nodes. In addition, the model of [21] assumes that the IoT device can offload a portion of its data in each time slot, and the remaining data will be executed locally. The channel gain remains static in each time slot but may vary in the next time slot. Thus, the proposed model in [21] enables offloading as much data as possible while satisfying the power constraints. Indeed, Lyapunov optimization has been investigated to solve the proposed model using an online optimization algorithm that has been defined. Moreover, Matlab simulation results show the performance of the proposed model in terms of the utility and the average execution delay as well as the impact of the frequency and the length of the task buffer. The model introduced in [22] has considered multiple mobile devices that may offload their tasks to the MEC server. In each time slot, some bits of the generated tasks are executed locally, and the MEC server executes the remaining bits. Similar to Non-orthogonal Multiple Access (NOMA), the model of [22] enables queuing arriving tasks that are not executed. In addition, the model of [18] is based on a frequency flat block-fading channel. The aim of the model of [22] is to minimize the energy consumption under the frequency, power, and delay constraints. Therefore, Lyapunov optimization has been investigated for formulating an online optimization algorithm. The simulation results of [22] demonstrate the ability of the proposed model to balance energy between mobile devices. Another contribution that addresses the problem of dynamic offloading between neighbouring fog nodes that operate using Time Division Duplex (TDD) communication is proposed in paper [23]. The latter considers two scenarios including single and multi-user topologies. In the second scenario, collision can occur when more than one user simultaneously select the same fog node. Moreover, the model of [23] assumes an M/M/K queue for each fog node. Thus, the proposed algorithm in [23] is based on an exploitation and exploration process to determine the best fog node that may execute the offloaded task. The effectiveness of the proposed algorithm proposed in [19] has been verified using the average latency and regret metric. The results of [23] demonstrate that the proposed algorithm can reduce the average latency compared to some other algorithms. Paper [24] has suggested an architecture including the fog and the cloud layers, where vehicles places in the fog layer can offload their tasks to the cloud layer. The architecture of [24] considers static and mobile fog nodes that can collaborate for task offloading. It also considers the impact of mobility on task offloading. Therefore, the mathematical model in [24] represents the coverage as a dynamic variable and allows minimizing the task service time while considering the storage, bandwidth, and deadline constraints. Thus, [24] has proposed a policy for task offloading that rearranges tasks according to their deadline. Then, the suggested policy decides on task offloading according to the capacity and bandwidth constraints. The extensive simulation results in [24] have been obtained using realistic vehicular trajectories. They show the performance of the proposed policy for different scenarios in terms of task delay, service composition, and task completion ratio. The work presented in [25] has considered a network involving a set of end users and a set of fog nodes. The end users generate dynamic tasks that are independent, whereas the CPU cycle to process one bit is identical for all tasks. Additionally, the scenario model in [25] concerns binary offloading, where end users may offload the entire task to a neighbouring fog node. If the selected fog node estimates that its resources will not be available to complete the task while meeting the deadline constraints, the primary fog node will offload the task to another selected fog node. Thus, the model of [25] assumes that each fog node has two types of queues, one for high priority tasks and the other one for low priority tasks. Depending on the type of the task, the model of [21] has specified the rules for deciding on the suitable queue for the generated task. Moreover, if a task is offloaded from a fog node to another fog node, the model of [25] proceeds to send the task to the high-priority queue. However, if a high-priority task is offloaded from the end user, the queuing delay is calculated and compared to the delay deadline requirement of the task. In case that the deadline is satisfied, the task is sent to the high priority queue. Otherwise, the fog node attempts to offload the task to another fog node. The third rule concerns low-priority tasks that are offloaded from end users. If the deadline requirements are satisfied, the tasks are set to the high-priority queue. Otherwise, the low-priority tasks will be offloaded to another fog node. We mention that the model of [26] has also studied task scheduling and has investigated Lyapunov optimization. The simulation results obtained in [26] demonstrate the reliability of the proposed scheme that ensures a high rate of task completion under the deadline requirements. The system architecture proposed in [27] incorporates three tiers. The first tier includes IoT regions, and fog nodes at the fog layer are organized as fog networks. The last tier is composed of cloud data centres. Thus, the objective of the model suggested in [27] is to minimize the delay while satisfying the stability of the discrete time varying queues. We mention that the delay expression introduced in [27] considers the computational, network, and fog-to-cloud delays. Lyapunov optimization has been adopted in [27], and the simulation results show the performance of the proposed approach with respect to the service delay.

It is worth mentioning that the authors of [28] have suggested a new technique that combines Long Short-Term Memory (LSTM) and deep learning for dynamic offloading in fog computing. Thus, the principal feature of the contribution presented in [28] concerns the prediction of the load of the fog servers for optimizing the offloading decision. The results illustrated in [28] reveal that the proposed technique minimizes the average latency. The literature review on offloading in fog computing shows also that new alternative techniques including deep learning and multi-agent systems have been suggested in [29–33]. However, the offloading model presented in [29,30] is static. The works introduced in [31–33] are related to task scheduling.

Thus, Table 1 shows a comparison between related works that have focused on dynamic offloading. Five comments are highlighted:

- The majority of papers indicated in Table 1 are based on Lyapunov optimization that represents an exact method.
- Most papers indicated in Table 1 did not demonstrate the scalability feature that is needed for large-scale problems including many IoT devices and fog nodes.
- Some papers have only considered one optimization parameter.
- Paper [28] has presented an interesting idea for combining LSTM and deep learning. However, the proposed model is based upon a static channel of communication.
- The number of dynamic variables of some of the models summarized in Table 1 is limited.

Consequently, our contribution consists of solving the problem of optimal offloading while considering dynamic tasks and dynamic channel of communication using parallel meta-heuristics that ensure the scalability as opposed to exact optimization techniques. More specifically, our model determines the best offloading decision and the best subcarrier given that the power of transmission, frequency, and channel gain change over time. Additionally, the originality of our work is related to the proposed fitness function and the proposed encoding schemes used by the proposed parallel meta-heuristics.

Table 1. Related works comparison.

Reference	Framework	Dynamic Variables	Optimization Method	Topology	Number of Fog Nodes	Simulation Tool	Metrics
[16]	IoT-Fog	Gain. Frequency. Power. Harvested energy. Battery level.	Lyapunov	One MEC server and one mobile device	One	NA	Average execution time, average execution cost, battery energy level, ratio of completed tasks, completion time, and dropped task.
[17]	IoT-Fog	Frequency. Power.	Lyapunov	Many-user equipment (UE) 36 and MEC network.	36 UE and 4 servers uniformly distributed.	NA	Average end-to-end delay, number of required servers per UE, delay bound, tail distribution of queues, approximated GPD (Generalized Pareto Distribution)
[18]	Fog-Fog. Fog-Cloud	Amount Workload queue. Backlog. Frequency. Gain. Power.	Lyapunov	Hierarchical (two tiers including the fog and cloud).	80 fog nodes. 20 cloud fog nodes (central fog nodes).	Python	Backlog. Power consumption. Workload.
[19]	IoT-Fog	Frequency Power. Workload computation request generation Battery level.	Lyapunov	One edge node. One access point. Mobile devices	6 mobile devices.	NA	Average execution cost.
[21]	IoT-Fog	Power. Frequency. Gain. Power.	Lyapunov	Mobile devices and one MEC server.	N mobile devices and one MEC.	NA	Power consumption.
[22]	IoT-Fog	Power. Frequency. Gain. Power. Power.	Lyapunov	One IoT device and N fog nodes.	5 fog nodes.	Matlab	System utility. Length of task buffer.
[23]	IoT-Fog.	Reward. Total number of selections of fog node. Regret.	Lyapunov	Many fog nodes and many mobile users.	5 fog nodes.	NA	Regret. Average latency. Selection.
[24]	Fog-Cloud	Coverage.	Adaptive task offloading algorithm	Mobile topology	180 mobile vehicles. 6 static fog nodes	NA	NA
[25]	IoT-Fog	Number of tasks queued	Lyapunov	Static topology	10 fog nodes and 20 end users.	NA	Reliability. Backlog.
[26]	IoT-Fog	Gain.	Laypunov	End users and fog nodes.	10 fog nodes. 20 end-users.	NA	NA
[27]	IoT-Fog. Fog-Cloud. Fog-Fog.	Task generation.	Lyapunov	Hierarchical fog architecture.	1 cloud and 3 fog regions.	CloudSim.	Average task delay.
[28]	IoT-Fog.	Size of the task. Computational resources. Maximum tolerated delay.	LSTM and deep learning	IoT devices and edge servers.	50 IoT devices and 5 edge servers.	NA	Time. Cost. Latency.

3. Proposed Method

We consider a system consisting of N IoT devices and M fog nodes such that $M < N$. We also assume that TDD communication is adopted. Each IoT device generates computational tasks at the beginning of every time slot and the generated tasks are stored in an M/M/1 queue. Further, each IoT device i is characterized by its:

- Frequency $f_i(t)$, which depends on CPU cycles for processing the generated tasks. It is expressed in Megahertz (MHz).
- Power of transmission $P_i(t)$, used for offloading a task to a fog node through the optimal sub-carrier. It is expressed in Watt.

We mention that our system allows each IoT device to generate tasks stochastically and independently according to a Poisson process with an average arrival rate $\lambda_i(t)$. More importantly, our generic model is useful for several scenarios including dynamic offloading in 5G Vehicular Adhoc Networks (Vanets) and healthcare applications where a set of IoT devices generate tasks dynamically and accordingly an optimal offloading decision is required.

In this context, each task j is modeled by a tuple $\{\alpha_j, wj\}$, where α_j represents the data size and wj represents the number of CPU cycles that are required to process task j. The communication between IoT devices and fog nodes is carried using OFDM with K channels [15,34]. In particular, our model aims to find the optimal dynamic binary offloading decision, such that IoT devices have the ability to process tasks locally or to offload them to the appropriate fog node.

We define x as the task allocation matrix:

$$x_{ij}(t) = \begin{cases} 1, \text{ if task } j \text{ is allocated to IoT device } i \\ 0, \text{ if task } j \text{ is offloaded} \end{cases} \tag{1}$$

As each IoT device has an M/M/1 queue, the time required for queuing and local execution of task j at time t is obtained using the Little formula [35]. It depends on the average service time and the mean arrival rate:

$$D_{ij}^{Local} = x_{ij}(t) \times \frac{1}{\frac{f_i(t)}{w_j(t)} - \lambda_i(t)} \tag{2}$$

Given that $\frac{f_i(t)}{w_j(t)}$ represents the average service rate at time t. We indicate that we need to ensure that $\frac{f_i(t)}{w_j(t)} - \lambda_i(t) > 0$ for maintaining the stability of the queue.

The energy consumed for local execution at time t is:

$$E_{ij}^{Local} = k \times f_i^3(t) \times x_{ij}(t) \times \frac{1}{\frac{f_i(t)}{w_j(t)} - \lambda_i(t)} \tag{3}$$

where k is defined as the effective switched capacitance, which relies on the chip architecture [16]. More notably, the matrix ρ denotes the channel assignment that is defined as follows:

$$\rho_{ik}(t) = \begin{cases} 1\,, \text{ if channel } k \text{ is allocated to IoT device } i \\ 0\,, \text{ otherwise} \end{cases} \tag{4}$$

Hence, we calculate the uplink rate between the IoT device i on channel k that using Equation (5), which is derived from the Shannon theorem:

$$r_{ik} = \rho_{ik}(t) \times B \times log_2 \left(1 + \frac{P_{ik}(t) \times h_{ik}(t)}{N_0}\right) \tag{5}$$

where B, N_0 and $h_{ik}(t)$ denote the uplink channel bandwidth, white noise, and the uplink channel gain, respectively. We define the computational latency for offloading a task j on channel k at time t as follows:

$$D_{jk}^{Offloading}(t) = \frac{\alpha_j(t)}{r_{ik}(t)} \tag{6}$$

Accordingly, the energy required for offloading a task j on channel k at time t is described through Equation (7):

$$E_{jk}^{Offloading}(t) = P_{ik}(t) \times \frac{\alpha_j(t)}{r_{ik}(t)} \tag{7}$$

The problem of jointly minimizing the delay and energy in dynamic fog computing while selecting the best channel is formulated as a weighted sum that is described by Equation (8):

$$\begin{gathered}\forall t:\ Min\ a \times \left(\sum_i \sum_j x_{ij}(t) \times \frac{1}{\frac{f_i(t)}{w_j(t)} - \lambda_i(t)} + \sum_i \sum_j k \times f_i^3(t) \times x_{ij}(t) \times \right. \\ \left. \frac{1}{\frac{f_i(t)}{w_j(t)} - \lambda_i(t)} \right) + b \times \left(\sum_i \sum_j \sum_k \frac{\alpha_j(t)}{r_{ik}(t)} + \sum_i \sum_j \sum_k P_{ik}(t) \times \frac{\alpha_j(t)}{r_{ik}(t)} \right)\end{gathered} \tag{8}$$

The first weight represented by a is a random number that is generated in each time slot. The second weight represented by b is calculated as $(1-a)$. The aim of introducing the two weights a and b requires a balance between local execution and offloading decisions. The first term of Equation (8) refers to the time and energy for local processing, whereas the second term refers to the delay and energy required for offloading. We mention that, different from some related works that optimize one parameter, our objective function enables minimizing two parameters including the delay and energy consumption based upon the queuing theory.

$$\begin{cases} P(t) \le P_{max} \\ f(t) \le f_{max} \\ x_{ij} \in \{0,1\} \\ \rho_{ij} \in \{0,1\} \end{cases} \tag{9}$$

Equation (9) indicates the power and frequency constraints.

The problem of optimizing the offloading decision has been identified as an NP-hard problem [36–38]. Different from the above related works, our contribution investigates parallel meta-heuristics for solving the proposed model for dynamic offloading in fog computing, which is a complex problem. So far, our solution defines many threads, whereas each thread runs the specified meta-heuristic for a specific IoT device. In other words, we will investigate parallel meta-heuristics for solving the problem of dynamic offloading in fog computing. As stated in the related work section, population-based optimization techniques including genetic algorithms and PSO have been widely investigated for dynamic optimization. The parameters of the suggested algorithms consist of $\alpha(t)$, $w(t)$, $f(t)$, $P(t)$, and $h(t)$. Consequently, our solution runs the proposed genetic algorithm shown in Figure 1 in each time slot. Starting from the second time slot, our proposed genetic algorithm conducts an evaluation of all chromosomes and replaces a certain percentage of the worst chromosomes by new random ones. The remaining chromosomes will be kept for finding the optimal solution F_{best} that represents the best fitness value for the current time slot. This process is repeated in all time slots for ensuring a trade-off between exploitation and exploration. Notably, our proposed genetic algorithm represents each chromosome as one-dimensional array. We have adopted an integer encoding, such that:

$$x_{ij} = 1, \text{if task } j \text{ is offloaded using sub} - \text{carrier } x_{i0} \tag{10}$$

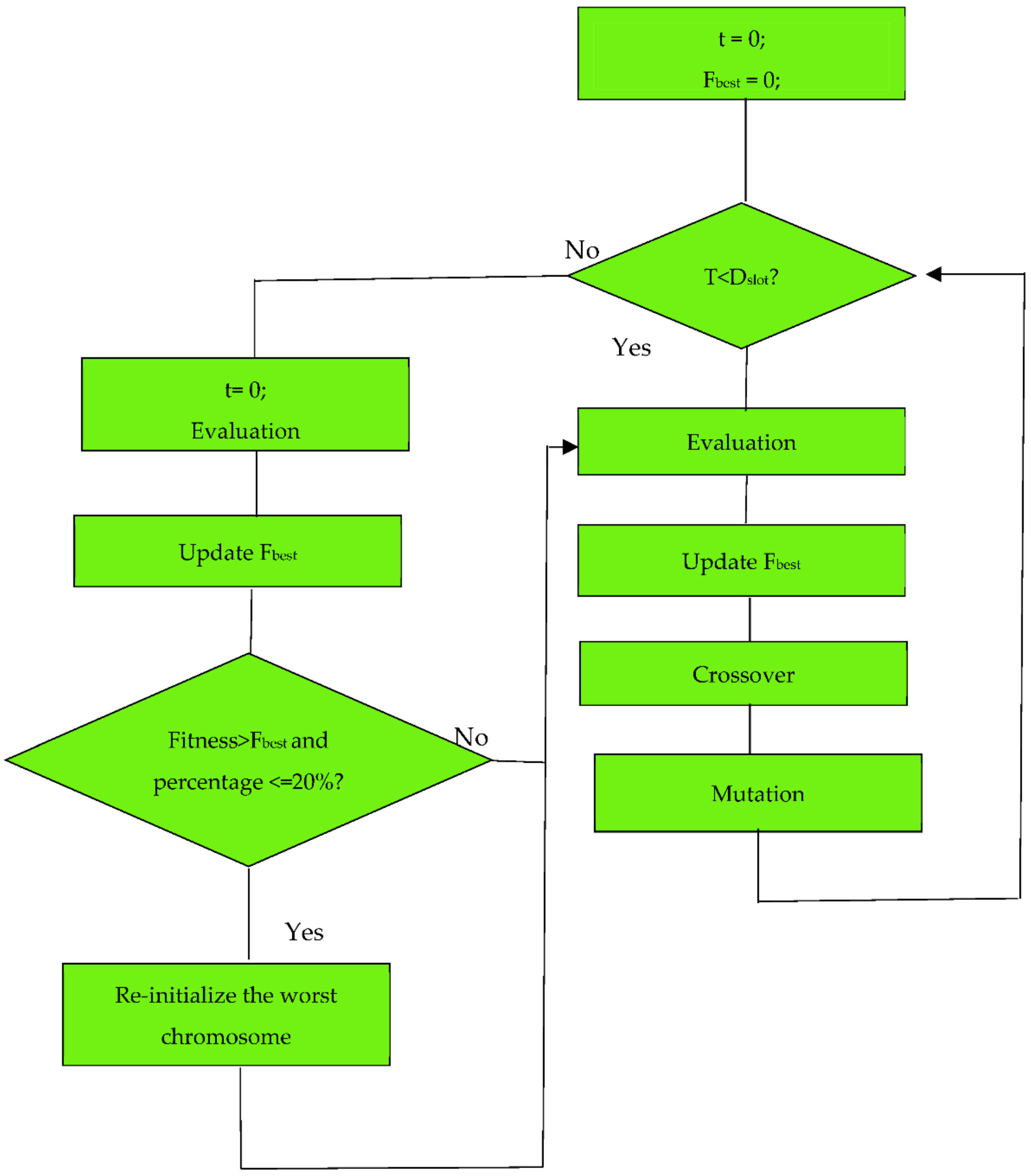

Figure 1. Flowchart of the proposed parallel genetic algorithm.

According to Equation (10), the first element of each chromosome includes the sub-carrier that is randomly initialized, although the other elements represent the offloading decision for the corresponding tasks. At each time slot, the proposed genetic algorithm consists of a set of sub-populations. Each thread generates a random set of initial solutions according to the encoding described in Equation (10). Each sub-population will be handled by a thread and will be generated randomly. The evaluation of the chromosomes of each sub-population is performed using the objective function defined in equation. Then, our genetic algorithm applies the tournament selection, crossover, and two-point mutation operators. The flowchart in Figure 1 depicts the idea of our solution. Assume that the duration of each time slot is Dslot, and random sub-populations are initiated. More specifically, our proposed algorithms rely on a time management module that triggers an event if the duration of a time slot has expired. Next, the chromosomes are evaluated, and the best solution is updated. Then, the crossover and mutation operators are applied, and this process is repeated during the same time slot. Upon the reception of a notification from the time management module, an evaluation of the chromosomes is performed, and 20% of the worst chromosomes are re-initialized before completing the remaining steps of the genetic algorithm. We also indicate that IoT devices may update their parameters at the beginning of each time slot. Assuming that new tasks arrive in the second time slot, our design extends the solutions manipulated by the considered thread by allocating random offloading decision variables for them. The second idea investigated in this paper consists of adapting PSO for solving the problem of dynamic offloading in fog computing. Following the same solution based on a genetic algorithm, we propose to initialize a set of swarms. Each swarm of particles corresponds to an IoT device. The swarms are executed in parallel using a set of threads. More specifically, each particle is represented by a one-dimensional array where the first position corresponds to the sub-carrier, and the other elements correspond the probability of task offloading. During each time slot, the fitness function is used to evaluate the particles based on the probability of offloading of each task. If the probability is high and exceeds 70%, a task offloading is possible; however, when the probability is low, local execution is possible, as shown in Figure 2. After conducting the fitness evaluation, each particle will update its position and velocity based on the following equations that balance between local and global search [39]:

$$v_{k+1}^{i} = W \times v_{k}^{i} + c_1 \times r_1 \times \left(p_{best}^{i} - x_{k}^{i}\right) + c_2 \times r_2 \times \left(g_{best}^{i} - x_{k}^{i}\right) \tag{11}$$

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i} \tag{12}$$

p_{best} represents the personal best solution of the particle, whereas g_{best} represents the global best solution. c_1 and c_2 are random parameters. The proposed parallel PSO is also synchronized to the time management module that provokes an event at the beginning of each time slot. We indicate that we have fixed the problem of boundary exceeding by re-initializing the particles that violate the allowed boundaries of the search space.

Assume the number of chromosomes/particles is n, the number of generated tasks is m, and the number of iterations is g. The complexity of the parallel genetic algorithm that is executed in each time slot is $O(g(n + n \times (m+1) + n \times (m+1) + n))$. The first term of the complexity refers to the evaluation of the solutions, whereas the second and the third terms refer to the crossover and the mutation, respectively. The last term of the complexity refers to the re-initialization process. In the worst case, the last solution of the population represents the best solution, and the mutation may be applied for the last element of the considered solution. Meanwhile, the complexity of the parallel PSO algorithm that is executed in each time slot is $O(g(n + n \times (m+1) + n \times (m+1) + n))$. The first term of the complexity refers to the evaluation of the solutions, whereas the second and the third terms refer to the process of checking the offloading probability and the process of updating the velocity and the position, respectively. The last term of the complexity refers to the re-initialization process. It is obvious that the two parallel meta-heuristics have the same complexity, which is of the order $O(n \times (m+1))$. Hence, the proposed scheme is appropriate for applications with low rate of task generation, where the IoT devices switch between active and idle states. Another parameter that impacts the time complexity concerns the number of time slots.

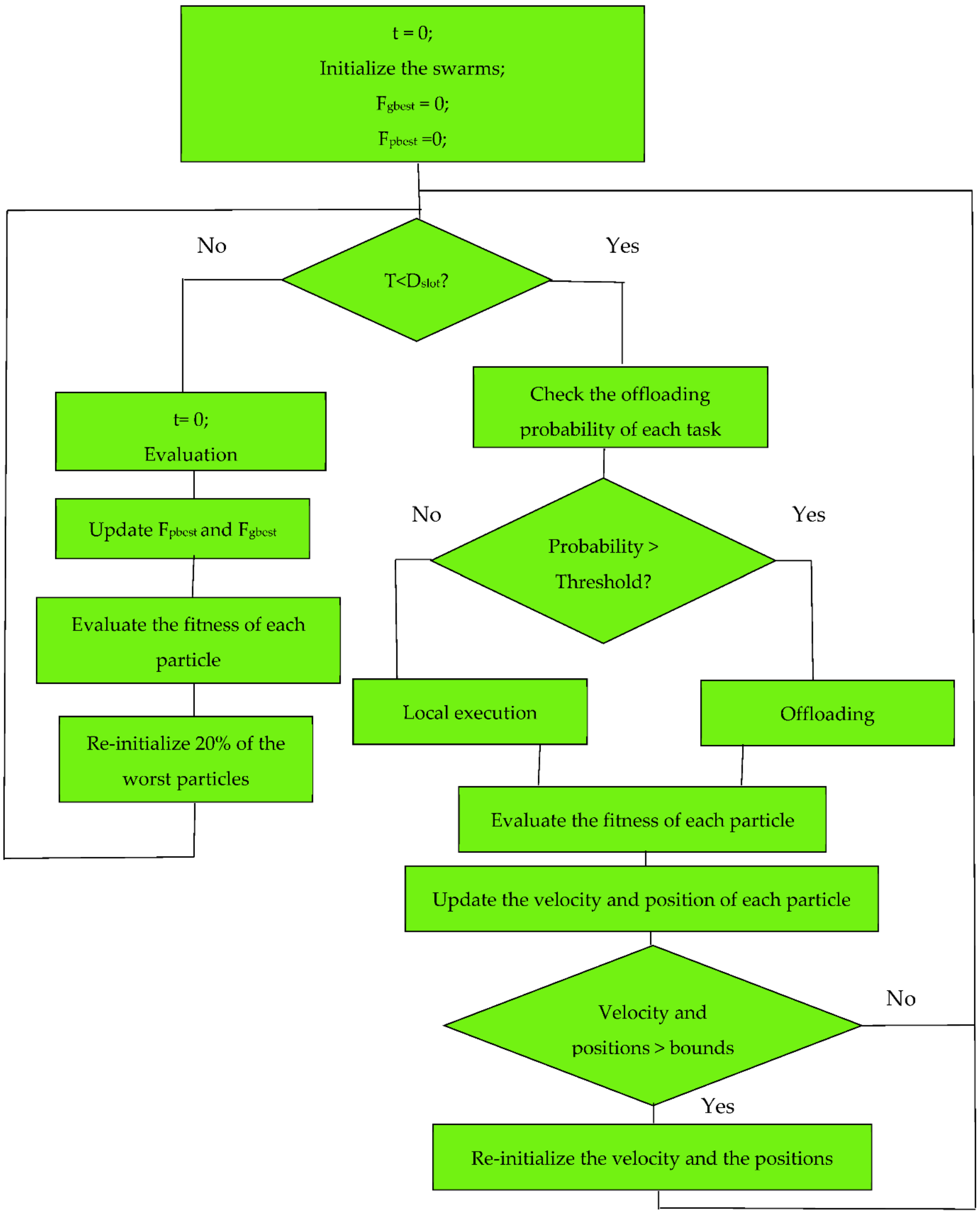
t = 0;
Initialize the swarms;
F_{gbest} = 0;
F_{pbest} =0;
No
Yes
T<D_{slot}?
t= 0;
Evaluation
Check the offloading probability of each task
Update F_{pbest} and F_{gbest}
No
Yes
Probability > Threshold?
Evaluate the fitness of each particle
Local execution
Offloading
Re-initialize 20% of the worst particles
Evaluate the fitness of each particle
Update the velocity and position of each particle
Velocity and positions > bounds
No
Yes
Re-initialize the velocity and the positions

Figure 2. Flowchart of the proposed parallel PSO algorithm.

4. Simulation Results

We have performed extensive simulations using Java Eclipse Oxygen [40] to demonstrate the effectiveness of the proposed meta-heuristics for dynamic offloading in fog computing. Our simulation enables initializing a random number of IoT devices and fog nodes. The tasks are generated according to a Poisson distribution. Table 2 illustrates the parameters of simulation. Most of them are similar to the settings adopted in [16,18,19]. Hence, our aim is to validate the performance of parallel meta-heuristics for dynamic offloading in fog computing. The best and average fitness values when varying the number of IoT devices from three to twelve are obtained by running the proposed approaches under 30 repetitions, and they are listed in Table 3.

It is obvious that the proposed parallel PSO and genetic algorithm generate better results compared to Lyapunov optimization in terms of the cost that is represented by the fitness function. In a similar manner, the proposed PSO outperforms the proposed genetic algorithm as it achieves the smallest best and average fitness value. Another finding concerns the scalability of the proposed genetic algorithm that is demonstrated by the fitness value, which decreases as the number of IoT devices increases too. The second metric used for evaluating dynamic optimization is the offline error that depends on the best fitness value. The results of the offline error for the proposed parallel PSO and the proposed parallel genetic algorithms are, respectively, illustrated in Figure 3. The green bars correspond to the offline error generated by the threads running the proposed parallel PSO algorithm, whereas the blue bars correspond to the offline error generated by the threads running the proposed parallel genetic algorithm. It is clear from Figure 3 that the offline error is very low. Therefore, the obtained offline errors indicate how well the solution meets the objectives of the problem. Additionally, the performance of the proposed parallel PSO in terms of the offline error is better than the performance of the proposed parallel genetic algorithm when the number of IoT devices has been increased to twelve. More specifically, the offline error achieved by all threads of the proposed parallel genetic algorithm is less than the offline error attained by the threads of the proposed parallel PSO when the number of IoT devices has been varied from three to eight. Accordingly, the proposed parallel genetic algorithm has the ability to find the optimal solution faster than the proposed PSO algorithm when the number of IoT devices varies from three to eight. However, the proposed parallel PSO algorithm outstands the proposed parallel genetic algorithm for quickly determining the optimal solution when the number of IoT devices is twelve.

Table 2. Simulation parameters.

Parameter of Simulation	Value
D_{slot}	3 min
Number of sub-carriers	128
W	0.72
Number of particles/chromosomes	30
Initial number of chromosomes	30
k	10^{-28}
N_0	−100 dBm
B	2 MHz
$\alpha(t)$	Gaussian normal distribution. Average 1000 CPU cycles and variance is 200 CPU cycles
$w(t)$	Gaussian normal distribution. Average 0.3 CPU cycles and variance is 1000 CPU cycles
$f(t)$	Random distribution [1–3] MHz
$P(t)$	Random distribution [5–10] Watts
$h(t)$	Random distribution [0.01–0.03]

Table 3. Fitness value comparison.

Number of IoT Devices	Thread Number	Parallel PSO Algorithm		Parallel Genetic Algorithm		Lyapunov Optimization
		Best Fitness	Average Fitness	Best Fitness	Average Fitness	
3	1st thread	0.004	0.005	0.007	0.0075	0.12
	2nd thread	0.0015	0.0016	0.0054	0.0059	
	3rd thread	0.004	0.005	0.0008	0.001	
5	1st thread	0.00015	0.00016	0.006	0.0065	0.2
	2nd thread	0.00015	0.00016	0.0086	0.0090	
	3rd thread	0.0003	0.00004	0.0092	0.0097	
	4th thread	0.0004	0.0005	0.0004	0.00042	
	5th thread	0.0001	0.00015	0.0009	0.002	
8	1st thread	0.00002	0.000015	0.014	0.017	0.35
	2nd thread	0.000023	0.000024	0.086	0.09	
	3rd thread	0.00003	0.00004	0.076	0.081	
	4th thread	0.000045	0.000048	0.17	0.18	
	5th thread	0.00001	0.0000015	0.16	0.19	
	6th thread	0.00002	0.000016	0.024	0.028	
	7th thread	0.000027	0.000029	0.026	0.029	
	8th thread	0.000003	0.000033	0.027	0.03	
12	1st thread	0.000006	0.0000062	0.83	0.84	0.46
	2nd thread	0.0000065	0.0000067	0.94	0.95	
	3rd thread	0.0000055	0.0000059	0.75	0.76	
	4th thread	0.000003	0.00000044	0.56	0.58	
	5th thread	0.0000052	0.0000055	0.07	0.08	
	6th thread	0.0000088	0.000009	0.93	0.95	
	7th thread	0.0000078	0.0000079	0.059	0.062	
	8th thread	0.0000090	0.0000092	0.12	0.16	
	9th thread	0.0000067	0.000007	0.94	0.98	
	10th thread	0.0000045	0.0000049	0.48	0.49	
	11th thread	0.0000075	0.0000076	0.56	0.59	
	12th thread	0.0000018	0.000002	0.24	0.29	

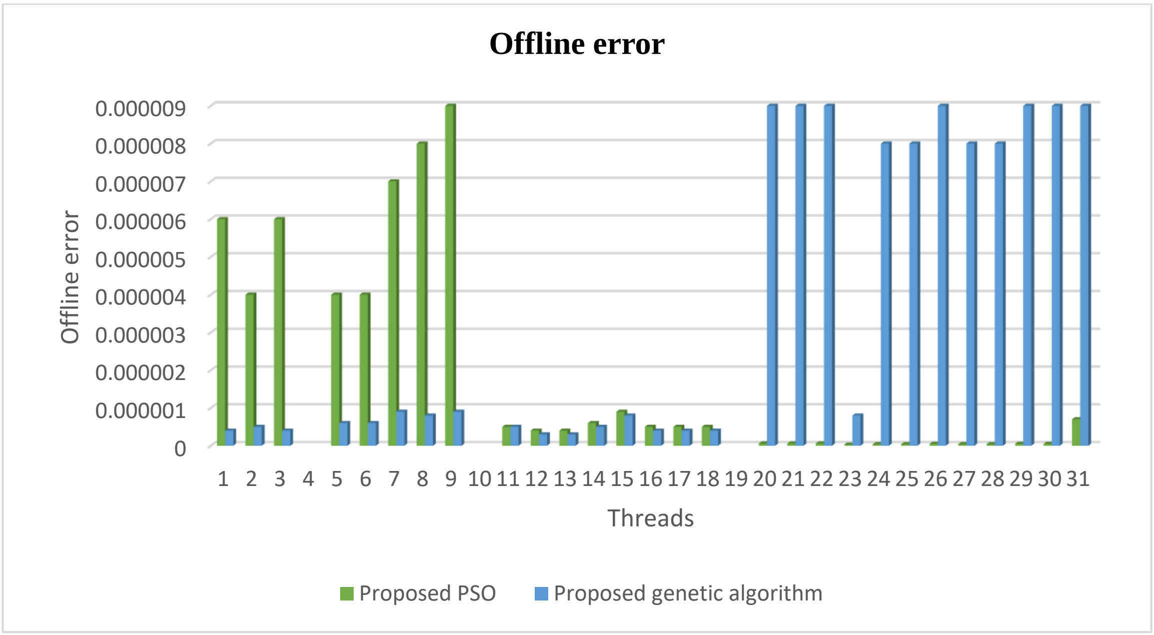

Figure 3. Offline error comparison.

To further analyze the results obtained by the proposed scheme, we studied the impact of varying the number of sub-channels on the execution cost. The number of IoT devices has been fixed to five. It can be observed from Figure 4 that increasing the number of sub-channels increases the probability of offloading and therefore decreases the execution cost for the three considered methods. However, there is a significant difference between the proposed parallel PSO and Lyapunov optimization. The former achieves lower execution cost compared to Laypunov optimization. This indicates that the proposed parallel PSO is more inclined to generate optimal particles in each time slot and the generated tasks have more chance to be allocated to the suitable node.

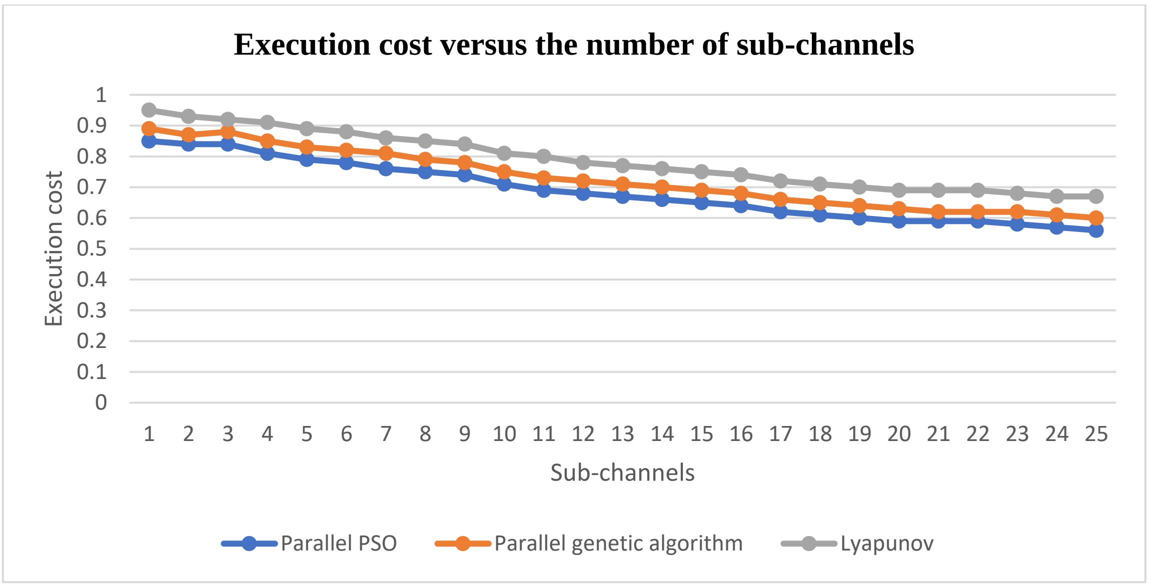

Figure 4. Execution cost comparison.

Despite the advantages of our results, some limitations should be highlighted:

- Our results did not depend upon realistic tasks.
- Our contribution did not handle the security issues in fog computing.
- Our contribution is limited to the offloading process.

5. Conclusions

This paper shows that parallel meta-heuristics are a powerful solution that can be extended for solving dynamic offloading in fog computing. Mainly, our contribution investigates the solution obtained at the end of each time slot for optimizing the solution of the next time slot, while re-initializing a certain percentage of the worst chromosomes/particles for jointly minimizing the delay and energy consumption. According to the extensive simulation results, the average and the best fitness functions converge. Additionally, the proposed parallel PSO approach achieves a minimal offline error, and it is scalable compared to the proposed parallel genetic algorithm. The comparison between the simulation results of the proposed approaches and Lyapunov optimization demonstrates the effectiveness of the proposed parallel PSO when increasing the number of IoT devices and the number of sub-channels. This paper is limited to the evaluation of the proposed generic model. So far, a realistic dataset for a specific application should be used for validating the proposed model. Another direction consists of deploying the simulation on a fog simulator and integrating the offloading technique to the fog scheduler. It is also recommended to consider security issues during task offloading. Potentially, a hybrid recent meta-heuristic will be developed for improving the obtained results by investigating the power of recent meta-heuristics in terms of the convergence and the trade-off between exploitation and exploration process.

Author Contributions: Conceptualization, S.I.A. and S.A.C.; methodology, D.S.M.H.; software, S.A.C.; validation, S.I.A.; formal analysis, S.A.C.; investigation, D.S.M.H.; resources, S.A.C.; data curation, S.I.A.; writing—original draft preparation, S.I.A.; writing—review and editing, S.A.C.; visualization, D.S.M.H.; supervision, S.I.A.; project administration, S.I.A.; funding acquisition, S.I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research project was funded by the Deanship of Scientific Research, Princess Nourah bint Abdulrahman University, through the Program of Research Project Funding After Publication, grant No (PRFA-P-42-10).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Marston, S.; Li, Z.; Bandyopadhyay, S.; Zhang, J.; Ghalsasi, A. Cloud computing—The business perspective. *Decis. Support Syst.* **2011**, *51*, 176–189. [CrossRef]
2. Negash, B.; Rahmani, A.M.; Liljeberg, P.; Jantsch, A. Fog Computing Fundamentals in the Internet-of-Things. In *Fog Computing in the Internet of Things*; Rahmani, A.M., Liljeberg, P., Preden, J.-S., Jantsch, A., Eds.; Springer: Cham, Switzerland, 2018; pp. 3–13.
3. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog Computing and its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile and Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–15.
4. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog Computing: A Taxonomy, Survey and Future Directions. In *Internet of Everything*, 1st ed.; Springer: Berlin, Germany, 2018; pp. 103–130.
5. Yi, S.; Hao, Z.; Qin, Z.; Li, Q. Fog computing: Platform and applications. In Proceedings of the Third IEEE Workshop on Hot Topics in Web Systems and Technologies, Washington, DC, USA, 12–13 November 2015; pp. 73–78.
6. Alba, E.; Nakib, A.; Siarry, P. *Metaheuristics for Dynamic Optimization*; Studies in Computational Intelligence 433; Springer: Berling/Heidelberg, Germany, 2013; ISBN 978-3-642-30664-8.
7. Abdallah, A.F.M.; Essam, D.L.; Sarker, R.A. Genetic Algorithms-based Techniques for Solving Dynamic Optimization Problems with Unknown Active Variables and Boundaries. In *Innovative Computing, Optimization and Its Applications*; Springer: Cham, Switzerland, 2018; pp. 151–166.
8. Mukherjee, M.; Kumar, S.; Zhang, Q.; Matam, R.; Mavromoustakis, C.X.; Lv, Y.; Mastorakis, G. Task Data Offloading and Resource Allocation in Fog Computing with Multi-Task Delay Guarantee. *IEEE Access* **2019**, *7*, 152911–152918. [CrossRef]
9. Yousefpour, A.; Ishigaki, G.; Gour, R.; Jue, J.P. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet Things J.* **2018**, *5*, 998–1010. [CrossRef]
10. Al-Khafajiy, M.; Baker, T.; Al-Libawy, H.; Maamar, Z.; Aloqaily, M.; Jararweh, Y. Improving fog computing performance via Fog-2-Fog collaboration. *Future Gener. Comput. Syst.* **2019**, *100*, 266–280. [CrossRef]
11. Chang, Z.; Zhou, Z.; Ristaniemi, T.; Niu, Z. Energy Efficient Optimization for Computation Offloading in Fog Computing System. In Proceedings of the IEEE Global Communications Conference, GLOBECOM, Singapore, 4–8 December 2017; pp. 1–6.
12. Movahedi, Z.; Defude, B.; Hosseininia, A.M. An efficient population-based multi-objective task scheduling approach in fog computing systems. *J. Cloud Comput. Adv. Syst. Appl.* **2021**, *10*, 53. [CrossRef]
13. Keshavarznejad, M.; Rezvani, M.H.; Adabi, S. Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. *Clust. Comput. J.* **2021**, *24*, 1825–1853. [CrossRef]
14. Hussein, M.K.; Mousa, M.H. Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization. *IEEE Access* **2020**, *8*, 37191–37201. [CrossRef]
15. AL-Amodi, S.; Patra, S.S.; Bhattacharya, S.; Mohanty, J.R.; Kumar, V.; Barik, R.K. Meta-heuristic Algorithm for Energy-Efficient Task Scheduling in Fog Computing. In *Recent Trends in Electronics and Communication*; Lecture Notes in Electrical Engineering; Springer: Singapore, 2022; Volume 777, pp. 905–925.
16. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
17. Liu, C.; Bennis, M.; Poor, H.V. Latency and Reliability-Aware Task Offloading and Resource Allocation for Mobile Edge Computing. In Proceedings of the IEEE Globecom Workshops, Singapore, 4–8 December 2017; pp. 1–7.
18. Gao, X.; Huang, X.; Bian, S.; Shao, Z.; Yang, Y. PORA: Predictive Offloading and Resource Allocation in Dynamic Fog Computing Systems. In Proceedings of the IEEE International Conference on Communications, Changchun, China, 11–13 August 2019; pp. 1–6.
19. Chang, Z.; Liu, L.; Guo, X.; Chen, T.; Ristaniemi, T. Dynamic Resource Allocation and Computation Offloading for Edge Computing. In Proceedings of the AIAI Workshops, Halkidiki, Greece, 5–7 June 2020; pp. 61–73.
20. Alenizi, F.; Rana, O. Dynamically Controlling Offloading Thresholds in Fog Systems. *Sensors* **2021**, *21*, 2512. [CrossRef]

21. Wei, Z.; Jiang, H. Optimal Offloading in Fog Computing Systems with Non-Orthogonal Multiple Access. *IEEE Access* **2018**, *6*, 49767–49778. [CrossRef]
22. Mao, Y.; Zhang, J.; Song, S.H.; Letaief, K.B. Power-Delay Tradeoff in Multi-User Mobile-Edge Computing Systems. In Proceedings of the IEEE Global Communications Conference, Washington, DC, USA, 4–8 December 2016; pp. 1–6.
23. Assila, B.; Kobbane, A.; El Koutbi, M. A Many-to-One Matching Game Approach to Achieve Low-Latency Exploiting Fogs and Caching. In Proceedings of the Ninth IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 26–28 February 2018; pp. 1–2.
24. Yang, M.; Zhu, H.; Wang, H.; Koucheryavy, Y.; Samouylov, K.; Qian, H. An Online Learning Approach to Computation Offloading in Dynamic Fog Networks. *IEEE Internet Things J.* **2021**, *8*, 1572–1584. [CrossRef]
25. Liu, C.; Liu, K.; Guo, S.; Xie, R.; Lee, V.C.S.; Son, S.H. Adaptive Offloading for Time-Critical Tasks in Heterogeneous Internet of Vehicles. *IEEE Internet Things J.* **2020**, *7*, 7999–8011. [CrossRef]
26. Mukherjee, M.; Guo, M.; Lloret, J.; Iqbal, R.; Zhang, Q. Deadline-Aware Fair Scheduling for Offloaded Tasks in Fog Computing with Inter-Fog Dependency. *IEEE Commun. Lett.* **2020**, *24*, 307–311. [CrossRef]
27. Zhao, S.; Yang, Y.; Shao, Z.; Yang, X.; Qian, H.; Wang, C. FEMOS: Fog-Enabled Multitier Operations Scheduling in Dynamic Wireless Networks. *IEEE Internet Things J.* **2018**, *5*, 1169–1183. [CrossRef]
28. Tu, Y.; Chen, H.; Yan, L.; Zhou, X. Task offloading based on LSTM prediction and deep reinforcement learning for efficient edge computing in IoT. *Future Internet* **2022**, *14*, 30. [CrossRef]
29. Lakhan, A.; Mastoi, Q.; Elhoseny, M.; Memon, M.S.; Abed-Mohammed, M. Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using IoT assisted mobile fog cloud. *Entrep. Inf. Syst.* **2021**, 1–23. [CrossRef]
30. Lakhan, A.; Memon, M.S.; Elhoseny, M.; Abed-Mohammed, M.; Qabulio, M.; Abdel-Basset, M. Cost-efficient mobility offloading and task scheduling for microservices iovt applications in container-based fog cloud network. *Clust. Comput.* **2021**, *8*, 1–23. [CrossRef]
31. Lakhan, A.; Mohammed, M.A.; Rashid, A.N.; Kadry, S.; Panityakul, T.; Abdulkareem, K.H.; Thinnukool, O. Smart-Contract Aware Ethereum and Client-Fog-Cloud Healthcare System. *Sensors* **2021**, *21*, 4093. [CrossRef]
32. Mutlag, A.A.; Ghani, M.K.A.; Mohammed, M.A.; Lakhan, A.; Mohd, O.; Abdulkareem, K.H.; Garcia-Zapirain, B. Multi-Agent Systems in Fog–Cloud Computing for Critical Healthcare Task Management Model (CHTM) Used for ECG Monitoring. *Sensors* **2021**, *21*, 6923. [CrossRef]
33. Lakhan, A.; Abed-Mohammed, M.; Ibrahim, D.A.; Abdulkareem, K.H. Bio-inspired robotics enabled schemes in blockchain-fog-cloud assisted IoMT environment. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *30*, 1–12. [CrossRef]
34. Li, L.; Guo, M.; Ma, L.; Mao, H.; Guan, Q. Online Workload Allocation via Fog-Fog-Cloud Cooperation to Reduce IoT Task Service Delay. *Sensors* **2019**, *19*, 3830. [CrossRef] [PubMed]
35. Fan, Q.; Ansari, N. Workload Allocation in Hierarchical Cloudlet Networks. *IEEE Commun. Lett.* **2018**, *22*, 820–823. [CrossRef]
36. Du, J.; Zhao, L.; Feng, J.; Chu, X. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Trans. Commun.* **2018**, *66*, 1594–1608. [CrossRef]
37. Wang, D.; Liu, Z.; Wang, X.; Lan, Y. Mobility-aware task offloading and migration Schemes in Fog Computing Networks. *IEEE Access* **2019**, *7*, 43356–43368. [CrossRef]
38. Swain, C.; Sahoo, M.N.; Satpathy, A. SPATO: A student project allocation based task offloading in IoT-Fog systems. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
39. Huang, S.; Tian, N.; Wang, Y.; Ji, Z. Particle swarm optimization using multi-information characteristics of all personal-best information. *SpringerPlus* **2016**, *5*, 1632. [CrossRef]
40. Eclipse Foundation. Available online: https://www.eclipse.org/oxygen/ (accessed on 21 February 2022).