Nature inspired computational offloading in fog-cloud of things ecosystem for smart city applications.

Adam A. Alli

Islamic University of Technology

Magombe Yasin

Islamic University of Technology

Muhammad Mahbub Alam

Islamic University of Technology

Abstract

Studies leading to optimization of resources and applications in the fog-cloud of things ecosystems have gained importance. This is because these studies form the basis upon which improved performance of Internet of Things(IoT) infrastructure can be realized. In this study, we explore heuristic approach that permits offloading to optimal offsite fog by developing modified dynamic PSO(mDyPSO) mechanism. We compared our results with the traditional simple PSO(SiPSO). Our simulation results show that mDyPSO out performs SiPSO in terms of application latency, network usage and energy utilization. We note that our mDyPSO offloading mechanism improves network performance up to one third. We conclude that mDyPSO mechanism performs well in fluctuating topology. This further proves that considering multiple computational parameters to modify PSO yield better offloading Results.

Keywords: Computational offloading, fog computing, particle swarm optimization, fog-cloud of things.

1. Introduction

The Internet of Things (IoT) industry has quickly expanded and revealed more opportunities to improve Quality of Service (QoS) in industries, governments, and businesses through a connection paradigm that allows anything, anywhere to exploit connectivity to attain desired services[1, 2]. The IoT ecosystems provide powerful platforms that influence intelligent systems in the automation of factories, education, military, medical care, surveillance, transportation, etc. Embedding the Internet of Things alongside Artificial Intelligence, data analytics, and large language models has enabled the development of high-performance systems that have upgraded future prospects of applications in cities[3], social media[4], education [5], manufacturing, the environment,space exploration, etc. [6, 7]. Even if there is tremendous performance improvement in many systems developed today as a result of adopting IoT, the performance of most of the systems developed is affected by i) the nature of connectivity, ii) the characteristics of applications that run on them, iii) the features of the platform that host the applications, and iv) the configurations of the IoT systems. Undistinguishably, the limitations such as uncertainty of IoT device's behavior, nature of operations as influenced by changing physical world, and finding optimal solutions that map applications to the optimal remote platform for processing are largely unresolved[8, 9].

IoT systems performance can be improved by extending the service of latency and security-sensitive applications to a remote location through fogging [10]. To effectively exploit fogging the challenge of selecting optimal location in the network to which tasks may be mapped in a constantly dynamic environment to achieve minimized latency, improve inter fog node communication and achieve better load balancing is still a challenge, moreover, minimized resource utilization in fog-cloud of things ecosystems is another important problem to study[11, 12].

Among the important problems to be resolved in computational offloading solutions is finding an appropriate offsite infrastructure to which resource-constrained user terminals in the fog-cloud of things ecosystems may offload complex applications so as to improve the general performance of the IoT ecosystems. Offloading process improves performance by reducing program turnaround and increasing system throughput [13]. This is done by minimizing communication overheads and maximizing resources utilization across the fog nodes. Inopportunely, there exist a trade-off in achieving an appropriate offsite infrastructure with sufficient resources to offload task(s) and minimizing intra-infrastructure communication that achieves sufficient turnaround required by IoT applications. This trade-off makes this problem NP-complete [14, 15]. NP-complete problems have no optimal solution in polynomial time; therefore, they may yield better solutions when heuristic techniques are used.

From Solutions proposed in the literature, a number of optimization techniques have been used to accomplish computational offloading in the IoT-fog ecosystems. For instance, Zhou et al. in [16] explored a contract-matching approach to task assignment and resource allocation in Vehicular Fog Computing (VFC). In their study, they proposed an efficient incentive mechanism based on contracts. They further transformed the task assignment into a two-sided matching between vehicles and user equipment. From their numerical result, the matching algorithm proposed was observed to improve performance. De Jong et. al. [17] proposed a deterministic delay constrained task partitioning as a mechanism to solve offloading decision. In their study, they highlighted the previous studies in which algorithms based on integer linear programming and stochastic analysis formulation were seen to under perform. They observe these mechanisms didn't guarantee polynomial-time convergence. To improve the performance of the system, they proposed a "deterministic approach", a proposal that guarantees polynomial convergence.

Recently, a number of heuristic mechanisms have been proposed for example ant colony mechanism was proposed in [18], they formulate multi-tasks scheduling as an optimization . Their optimization objective was to maximize profit and constraints. Their proposals performed better than previous explored mechanisms based on deterministic methods. In our previous work in [8] we proposed a pipeline of machine learning mechanisms to perform computational offloading. Amongst the mechanisms that was used in the pipeline to facilitate selection of suitable fog node for computational offloading was Particle Swarm Optimization (PSO).

We assumed during the offloading process the network conditions and resources utilization remained unchanged. This study considered a Simple PSO(SiPSO) mechanism in which for every offloading scenario in a cluster there arises one peak fog that provides maximum processing power and other resources. This peak node is the one considered the most suitable candidate to execute an offloading process. The results of this study enumerated considerable improvement in resource, throughput, and energy utilization. Further there was considerable improvement in response time. In the same study, dynamic nature of IoT environment that resulted in many IoT devices that may initiate offloading at the same time was considered, but clustered nature of Fog-cloud of things which could result in multiple topology, different underlying topological functions and batch offloading at diverse points in the network was not considered. Again, for simplicity we did not consider multiple fogs that may arise due to resource fluctuation in the network. Naturally, multiple offloading and varying availability of task processing resources at the offloading points on fog-cloud of things network may provide numerous optimum fog that may offer better offloading performance, or make the offloading positions actively change. This is a shifting fog-peak .

The shifting fog-peak results in the peak fogs to gain or lose resources required to process offloading dynamically, this activity makes the algorithm fail to converge, hence requiring the algorithm to diverge and re-converge so as to find the optimum fog. And, most often when using traditional SiPSO the personal best and global best may change resulting in memory loss. Looking at the application of PSO proposed in our previous studies in [8], we intent to extend the proposed mechanism to include a dynamic selection considering multiple clusters. Our proposal takes care of changing dynamic that may offer moving maxima fogs during the offloading processes. This study continues the effort to explore PSO mechanisms and its variations for different offloading conditions in dynamic IoT-fog environment.

The paper therefore, aimed at designing and developing a modified dynamic particle swarm optimisation algorithm for computational offloading in the fog-cloud of things ecosystem. During the study an evaluation of our algorithm in terms of latency, network utilization and energy consumption was done. Secondly, a comparison of the results obtained through simulation of mDyPSO and SiPSO is completed and conclusion drawn. To achieve the aim of the study we

i) performed a classification of computational offloading strategies. This enabled us to observe how experts have applied them to solve offloading and other related solutions,
ii) developed and designed modified Dynamic Particle Swarm Optimisation (mDyPSO) algorithm for computational offloading in a clustered Fog of things ecosystem,
iii) performed an evaluation of the developed mDyPSO algorithm and tested its performance against Simple Particle Swarm Optimisation.

The remainder of the paper is organized as follows: Section 2 presents review of related literature in terms of offloading technologies, application of nature inspired algorithm for computational offloading and classes of computational offloading strategies as they appear in literature. Section 3 and 4, presents the proposed system, the network model, formulation of the optimisation framework and modified Dynamic Particle Swarm Optimisation offloading Mechanism. In section 5, the simulation results are presented and compared to results of SiPSO. Conclusion of the study is drawn in section 6.

2. Review of related works

2.1. Offloading in the fog computing ecosystem

Fog computing paradigm provides data, compute, application and services to the user at the edge [19]. Fog computing is promoted by scenario that require fast and reliable computing closer to the source of data such as in autonomous systems, smart city applications, smart health care, smart infrastructure and disaster management systems. Fog computing is characterised by low latency computing closer to the edge of the network. In addition, they use reliable protocols, provide easy and affordable installations. Also they consist of slightly powerful devices that contain programmable function to allow accommodation of multiple applications. The Fog-cloud of things infrastructure that supports offloading processes run in versatile operating environment that calls for processing big data using powerful artificial intelligence applications. This phenomena may result in the device at the edge to fail handling the application. Besides, IoT devices at the edge are designed to handle very powerful application but come with small battery capacity, this therefore, calls for a mechanism that should allow devices at the edge to conserve energy. Fog technologies are often designed to conserve resources(processing power, energy and memory) by developing mechanisms that accept the edge devices to process active tasks generated by IoT and provide both trust and security.

Fog-cloud of things paradigm provides effective solutions to eliminate latency caused by physical distance between the cloud and devices in request of service coupled with huge volumes of data along the service path. Additionally, they enable organisations to save bandwidth and mitigate network congestion given that essential storage and computing can be provided along the edge[3, 20, 21]. Though the fog is viewed as Superior technology that will increase the efficiency of a network through mechanism such as computational offloading, they are deterred by their complexity and expensiveness due to their distributed nature. Their implementation calls for well defined scope in addition to equipment, applications and resources to meet the objective of adoption[22, 23]. Along with their location at the edge, their mobility make them vulnerable to security concerns. Lastly, their processing capacity may require data reduction which may result in partial data processing. Partial data processing at the edge limits their capacity to function in a similar way such as the cloud infrastructure in an intensive big data environment.

To achieve computational offloading at the fog the middle ware system should partition the offloadable tasks, present the task for processing at the host fog device and make an offloading decision. The computational offloading is based on if the ecosystem; (i) support offloading (ii) benefits from the offloading process, and (iii) if the resources are not available to process the application tasks locally [24].

Authors in [25] presented three classes of archetypes that defines computational offloading strategies in IoT-cloud ecosystem. They include homogeneous, heterogeneous and neutral models. To allow for offloading using homogeneous model, the run-time environment must be implemented on both the IoT device and the Fog/cloud infrastructure. This configuration enables the IoT to execute its task independent of network connectivity and Fog/cloud in situations when offloading is not necessary. The results of execution of tasks in this model are not compromised Offloading only happens when it is suitable and unavoidable. On the other hand, heterogeneous offloading models require that the run-time environment implementation is simpler and lighter for the IoT devices and complete for the fog/cloud environment. This enables the IoT devices to execute their own task but the result may not be as good as the one produced by the fog/cloud run-time environment. During offloading, input data is transmitted to the server and result of the computation received. Whereas, the neutral models does not require the run-time environment to be installed on the IoT-device during task outsourcing. Therefore, the IoT device must always consult with the fog/cloud to execute offloading. In this model IoT can not execute offloading independent of network connectivity and fog/cloud connection. Generally speaking implementation of computational offloading solution take one of the above three forms.

2.1.1. offloading Decision

The ability of the IoT devices or smart gateway to initiate a decision is regarded as an offloading decision. The decision occurs after an evaluation of application needs to warrant offloading. The evaluation is done in terms of data type, data size, power of devices, status of activities at the initiation point, intermediate nodes, and the end node. Furthermore, the need to improve performance of the IoT ecosystem is at the heart of offloading decision strategy. An offloading decision is passed subject to whether the application can benefit from offloading in terms of the overall performance of the system. Also, an evaluation to determine if data, its code or its application require to be offloaded. for every offloading strategy the destination to where an offloading request shall be processed is critical for a rational offload decision [24]. sometimes it is important to consider the portion of data that should be offloaded. Finally, offloading strategy used to perform offloading is critical for offloading. Most often offloading decisions are taken by middleware installed over the top of smart device.

The decision is held based on whether the application at hand i) needs extra computational resources in excess of the hosting device, ii) if it is latency sensitive, iii) if it is security or privacy sensitive, iv) does it require its data to be stored on storage space in excess to what the hosting device has, v) if the application at hand is demanding such that it's execution may degrade the performance of the system during operation, and, vi) does offloading improve the general quality of service of the whole IoT system. Other means through which an offloading decision may be necessary is if offloading is supposed to improve infrastructure utilization as observed in load balancing, parallel processing and distributed computing.

2.1.2. Application task partitioning

Application task partitioning is another serious activity performed in preparation for offloading. It involves dividing an application tasks into subsequent chunks that can be executed either on a client device or the server. In a number of studies authors have shown that a good design strategy towards an optimum partitioning solution can affect resources utilization at run time. Additionally, levels of granularity affect compatibility, offloading, and performance in general [26]. Partitioning mechanisms can be static, dynamic or hybrid. Hybrid mechanism the bridges between adaptability to offloading conditions and balancing cost of performance.

Li et al. in [27] presented a partition scheme at a procedure call level. Their solution is based on a cost graph. they modelled the behaviour of the task assignment during offloading. In this study, the authors explored the branch and bound mechanism for task partitioning and assignment. Their mechanism implemented pruned heuristic component that improved its performance of expensive branch and bound mechanism considerably. Gao at al. in [28] proposed a layered computational strategy that performed partitioning of tasks based on deep neural networks. In their study joint optimisation design was achieved. This design strategy minimised latency by optimising task allocation hence improving the performance of offloading mechanism. In another study by Jianhui et al.[29], they illustrated that when portioning is done at both the mobile device and on the remote device, processing latency is minimised. The above studies confirms that partitioning of task during or before the offloading process is an important factor. In our study, we consider the first partitioning occurs at the IoT device. Here the task is partitioned such that a portion that is processed on the local device is considered for offloading and further partitioning may occur at the smart gateway so that the task may be processed by multiple Fogs.

2.1.3. Preparation

During the preparation stage actions that are necessary to support successful optimal offloading performance are finalized. Three events are important here i.e i) selection of destination offsite location where offloading will occur, ii) Transfer and installation of code, iii) and transfer of data to and from the offload site [30]. Preparation is done in an effort to initialize the offloaded process on the offsite environment. [31].

Mitsis et al. in [32] showed the of importance well designed selection mechanism. In their simulation study they proposed a two component data offloading and MEC server selection algorithm based on stochastic learning automata. Their mechanism scored sufficient performance improvement in realizing optimal offloading and pricing. Therefore,in our study we opt to develop mDyPSO in a Fog-cloud of things environment to improve performance by using PSO.

2.2. Nature inspired algorithm for computational offloading and related problems

The dynamic nature of applications, computational and data transactions that are supported by IoT-Fog-cloud ecosystems inspire the use of evolutionary algorithms based on either genetic algorithms or swarm intelligence. Yang et al. in [33] performed an analysis of nature inspired algorithms and their applications. In their study, they elaborated the two broad categories of nature inspired algorithms falling in procedure based and equation based. They further provided examples of such algorithms to include Deferential Evolutionary(DE), Particle Swarm Optimisation(PSO), fire-fry algorithm, Bat algorithm, Cuckoo search algorithm etc.

Also studies in [34] performed a comparative study of nature inspired algorithm on travelling salesman problem to show how they provide platform to solving combinatorial optimization. These algorithms provide diverse opportunities to solving many problems that arise in the IoT-Fog ecosystems except for lack of well-formed frameworks to enable a proper consideration of their efficiency, effectiveness and their robustness in the new computing environment. In this work we are in efforts to explore PSO for offloading mechanism in IoT environment.

PSO is principally based on social behavior of animals [35]. In a PSO system multiple solutions co-exist and collaborate by iteratively changing their current position until an optimum solution is achieved. Each set of candidate solutions are known as particles. In a bid to find an optimal solution in a search space of D dimension, the particles fly around adjusting their position in accordance with his personal experience (P_b) and neighbor particles' experience (P_g). Each particle preserves a memory of its own experience and best experience of the neighbourhood. PSO combines particle dynamics and information sharing to derive a powerful heuristic optimization tool. The canonical PSO achieves optimisation through the following two equation 1 and 2

$$V_{ij}(t+1) = \omega V_{ij}(t) + C_1 \times r_1 \times (P_{b(i,j)} - X_{ij}(t)) + C_2 \times r_2 \times (P_g - X_{ij}(t)) \quad (1)$$

$$X_{ij}(t+1) = X_{ij}(t)) + V_{ij}(t+1) \quad (2)$$

where $V_{ij}(t+1)$ and $V_{ij}(t)$ are current and previous velocity respectively, whereas $X_{ij}(t+1)$ and $X_{ij}(t)$ are current and the previous particle positions, c_1

and c_2 are cognitive and acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1, ω is the initial coefficients, t are the number of iterations [36] and lastly, $P_{b(i,j)}$ and P_g are personal and global best respectively [37].

Originally, PSO has been thought to solve optimization problems that were continuous in nature . Recently, PSO has attracted interest in solving both discrete and combinatorial problems with small modification. Resendo et al. in [38] presented PSO with path rethinking for combinatorial optimization problems. In their work, they presented a PSO algorithm in which the particle was viewed to be guided by three components. These components include i) the component that is guided by it's own way k_1, ii) the other component that allows it to get to its previous best solution k_2 and iii) the component that allows it to align with the global solution of the whole swarm k_3. Component k_1 facilitates the local search, where as k_2 and k_3 helps move the particle to the new position. grouping the components into functional partitions allows them to create the local search and the path rethinking dimension of the velocity equation in 1. From their formulation the PSO equations can be formed as in eqn. 3 and 4 as follows

$$V_{ij}(t+1) = \omega k_1(t) + C_1 \times r_1 \times (K_2(t)) + C_2 \times r_2 \times (k_3(t)) \tag{3}$$

$$X_{ij}(t+1) = X_{ij}(t)) + V_{ij}(t+1) \tag{4}$$

in the eqn. 3 $\omega k_1(t)$ represents the local search and the rest form the path rethinking. They Performed experiment and showed that their result compared well on averages with other solution presented in literature. This study gives evidence that hybridization is one of the ways of attaining a competitive discrete PSO.

Another study by Gupta et al. [39], proposed a hybrid Genetic Algorithm-Particle swarm optimization (GA-PSO) to solve travelling salesman problem. In their algorithm they take advantage of fast convergence rate of PSO and robustness of GA. Their result show that hybrid GA-PSO achieves better computational average mean time and low mean error, thus attaining superior performance. Mohammed et al. [40] investigated the application of PSO to solve the shortest path problem. They experimented their work of different network topology. In there study good success of discovering the shortest path was realized as compared to GA.

Authors in [41] and [42] presented PSO for transport problem and assignment problem respectively. In both studies they presented tractable solutions to the problems in questions. [41] noted in there study that the PSO traditional updating rule does not hold for the constrains that result from formation of transport problem. Therefore, they presented an alternative updating rule that suites transport problem as stated in equation 5 and equation 6

$$V_{ij}(t+1) = \begin{cases} r_1 \times (P_b(t) - X(t)) + r_2 \times (P_g(t) - X(t)), t = 0 \\ r_3 \times Vij(t) + r_4(P_b(t) - X_{ij}(t)) + r_2 \times (P_g(t) - X(t)), t > 0 \end{cases} \tag{5}$$

$$X_{ij}(t+1) = Vij_(t+1) + X_{ij}(t) \tag{6}$$

They considered the following conditions to overcome the shortcoming of the traditional PSO to solve TP problems

i) $X(t)$ and V_{ij} is viewed in $n \times m$ dimension

ii) r_1, r_2, r_3, r_4 are random number between 0 and 1

iii) r_1, r_2 $r_1 = U(0, 1), r_2 = 1 - r_1$

iv) r_3, r_4 $r_1 = U[0.8, 1), r_4 = 1 - r_3$

v) if $P_b(t) = X(t)$ AND $P_g(t)! = X(t)$ then $r_1 = 1$

vi) if $P_b(t)! = X(t)$ AND $P_g(t) = X(t)$ then $r_2 = 1$

vii) if $P_b(t) = X(t)$ AND $P_g(t) = X(t)$ then $r_3 = 1$.

Our study is based on the assignment of tasks to a fog that has capacity. This problem is viewed as a version of the TP problem.

Lastly, Rafique et al. presented Bio-inspired hybrid strategy based on PSO to schedule tasks and Cat swarm optimisation (CSO) algorithm to manage resources. Results of their study show with slight modification of the PSO and

CSO NBIHA [43] balances load well amongst Fog nodes, hence presenting an efficient resource allocation strategy. we deduce from this study that further tuning of PSO would yield improved performance in the Fog-cloud of things environment. Drawing from the experiences of the above authors, we build confidence that our proposal provides a better platform to solve offloading problems, which consist of the selection of optimal fog, placement of tasks for optimisation, and balancing tasks amongst appropriate fogs.

2.3. Classes of computational offloading strategies

Sno.	Proposed offloading strategy	Principle	Limitation	Edge technology	Reference
1	Offloading for Energy minimizing in mobile-edge cloud computing	Formulated as joint energy and Task completion minimization problem for mobile users. mobile users are constrained by both time and energy consumption. Formed as a convex optimisation problem	formation results in complex system that difficult to solve for distributed Fog ecosystem	mobile Edge, cloud	[44, 45, 46]
2	Task offloading based on integer programming	Formulated as mixed integer non-linear programming problem. the problem is often transformed sub problems to tune the results of the system	Difficulty in representing high dimensions of distributed system environment created by the fog system	Software-defined mobile edge	[47, 48, 45]
3	Bio-inspired task scheduling and resource allocation.	Formulated as optimization problems to be solved using bio-inspired algorithms. Approaches include modification of PSO, CSO, Bee swarms, etc. These methods are used to schedule tasks, resources, and demands	difficulty in matching resources for computation	Fog computing	[49, 50, 51]

Sno.	Proposed offloading strategy	Principle	Limitation	Edge technology	Reference
4	Machine learning-based offloading	Formulated based on machine learning approaches. They include supervised, semi-supervised and supervised Approaches. Deep learning and reinforcement learning have been widely proposed in literature. These methods have shown that reliable offloading that maximizes network performance and improves system utility is possible	They suffer from heavy training requirements, lack of explainability, and intelligence. Often the underlying systems are complex to debug	Fog MEC, Cloud, Mist	[52, 53, 54]
5	Stochastic offloading	Formulated with association of randomness in the system that generates the tasks, or the systems that processes the offload. Communication between the nodes may also be random. The success of stochastic systems are mainly dependant on task uploads and download possibilities	Stochastic models help in learning the behaviour of offloading and complexities involves but, suffer from convergence issues when the systems under study become larger, dynamic and complex	Fog, Mec	[55, 56, 57]

Table 1: Table showing state of art strategies for computational offloading

In the table 1, we present five state of art proposed classes for computational offloading. Each of the classes are based on principle that has been stated against them. In addition, the limitation of each of the proposal have been illustrated. We note that each of the strategies provide an advantage over another.

3. Formal problem formulation

3.1. Network model

Figure 1 shows the network model proposed in this study. We consider a moderately large IoT network of N devices. The devices in the network are grouped into smaller C clusters. C_i is the i^{th} cluster and i is a positive number. This model consists of p IoT devices at the lower layer, q Fog nodes organized in the multi-hierarchical middle layer and r clouds devices in the upper layer. In the model, the number of IoT devices, fog devices and the cloud devices are such that $p >> q > r$. All IoT devices in a single cluster are connected to the upper layers through the smart gateways. The fogs are in turn connected to the cloud infrastructure consisting of private clouds, public clouds or hybrid clouds systems. All packets shall enter or exit the cluster through the smart gateway (G_i). Each IoT device I belongs to some cluster C_i.

Application tasks($A\tau$) arrives to the IoT randomly and periodically. Each of the $A\tau$ generated is forwarded to G_i when offloading is required. These tasks are associated with resource requirements. If the generated application tasks cannot be completed in the required sums of time slot (t) at I, an offload decision is initiated or the task is dropped [8]. The task generated at this point may be delay, resource constrained or demand driven. That means the tasks must be completed within specified time bound or the tasks demands processing requirements that are not available at I to complete the task. To this end, the offloading decision may be to execute the tasks in whole on a local device or offload the task(s) to fog device(s) in the next layer. Moreover, the offload decision might support further offloading to the cloud.

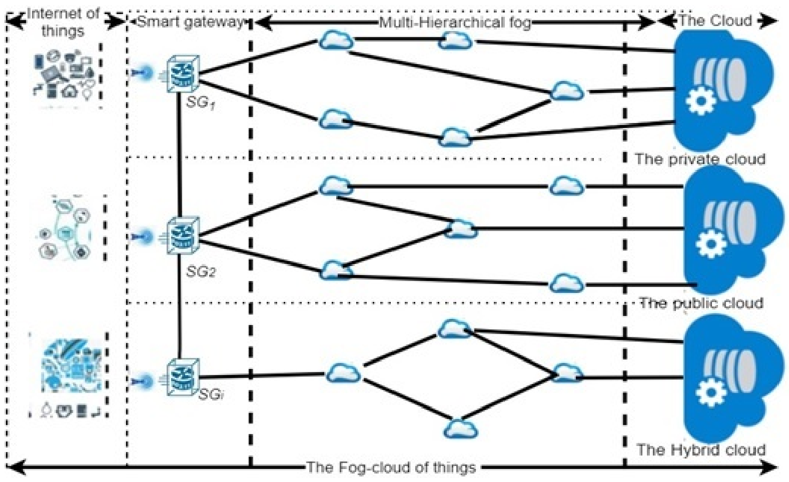


Figure 1: showing Network Model

Let us consider $0 \leq \lambda \leq 1$ to be a fraction of the application tasks that shall be offloaded during offloading process. If $\lambda = 0$ then the task is executed in totality at point of generation that is the local device I else if $\lambda = 1$ then the task is offloaded in whole to the fog except for the none offloadable parts of the task, otherwise a portion λ of the task is offloaded to the fog and $1 - \lambda$ will be executed on the local device [8].

Each Task $(A\tau)$ generated is characterized by input component (α), output component (β) and computational component (γ). Thus $A\tau$ is represented as a tuple $A\tau = \{\alpha, \beta, \gamma\}$. To offload $A\tau_i$ to a fog j a considerable delay bound (δ_i) is required. δ is the sum of transmission delays (δ_{tr}), queuing delay (δ_{qd}), fixed processing delay (δ_{fd}), packetization delay (δ_{pd}) and depacketization delays (δ_{dd}). Based on the the portion of data that may be offloaded a decision models presented in table 2 may be chosen.

Let us denote f_k to be the k^{th} fog node to which an offload occurs at time slot (t_s); where, $k = \{1, 2, 3, \ldots, m\}$ and m, is the maximum number of fog nodes in the network organized in multi-hierarchy. Finally, the lower levels of the network are connected to the fog through smart gateway G_i. Fog nodes may be connected to the cloud directly or indirectly through multiple sub-layers of Fogs devices.

The cloud has abundant resources but high latency requirement, therefore, if any task is offloaded to the cloud it shall be executed at negligible execution time. If a task is not executed by any fog, then the fog may perform additional offload to the cloud for complex part of the workload. In general, this framework adopts that there exists Fog(s) that are capable of solving the computational offload, therefore, offloading to the cloud may often be immaterial.

The computational offloading process happens in time slots. The time slots contain three (3) time cycles as follows a) The assignment cycle (T_{as}), the time through which the tasks shall be received by the smart gateway of the cluster b) The operational cycle (T_{op}), the time cycle at which the PSO mechanism is run to select the optimal fog platform to execute offloaded task... c) The dispatch cycle (T_{di}), the time cycle through which the results shall be received by the IoT devices.

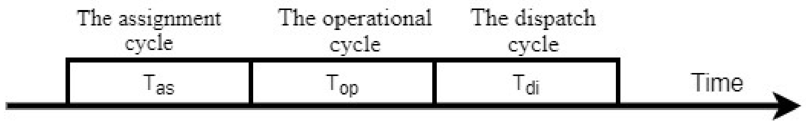
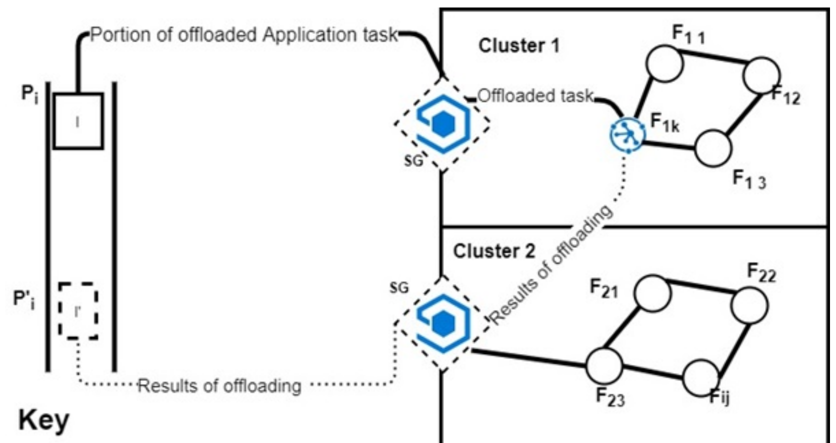


Figure 2: showing offloading Time slots

3.2. Problem statement

An IoT device $I_{i,j,k}$ randomly generates application task($A\tau_{i,j}$). $A\tau_{i,j}$ consist of offloadable part. If offloadable part does exist then $I_{i,j,k}$ issues an offloading request (Or_k) through the smart gateway G_j. G_j forwards the request to the selected Fog node f_k which is considered optimal at the time of request. The selection process is driven by heuristic algorithm such that the selected Fog node to which an offload process ($op_{i,k}$) minimizes task allocation, latency and response time. The offload process happens at the fog and uses fog resources which include memory and processing power in terms of CPU cycles. In case an offload process can not happen completely in one time slot, the process is postponed to next time slot as longer the latency constrain is maintained. Since



Key

I: Internet of things device, The device may have mobility function.

P_i : Position from which an offloading decision has been made

P'_i : Position from which an offloading results are recieved

F_{ij} : A set of Fog notes from which one is choosen to erform offloading process

SG: smart gateway

Op: selected node with offloading process

Figure 3: showing offloading process

the tasks are forwarded to the fog through the gateway, we assume at a certain time(t) the gateway has at least one offloading requests. In addition, all the offload requests are assembled at G_i. Our problem assumes that all the task from the IoTs surrogate at the smart gateway. Again it is assumed that the cloud has unlimited resources but far from the IoT devices hence latency is high. Our solution will only forward a task to cloud only when there is no fog to solve the task. The main concern is to find a fog that executes offloaded tasks available at the gateways at minimum latency and enables maximization reservation of energy at the IoT devices.

Offloading decision Model	Formula	Def. of symbols	Description
Local processing model	$\tau_{lp} = (1 - \lambda) \times \frac{\Omega_{iot}}{f_{iot}}$	i) λ is the fraction of work load that is offloaded, ii) Ω_{iot} is work load generated at the IoT, iii) f_{iot} is the processing power of the IoT device	Time required to process a fraction of workload generated at IoT device this includes User interfaces, peripheral management programs etc. [8, 58, 59]
Processing offloaded work load at the Fog	$\tau_{lp} = \lambda \times \frac{\Omega_{iot}}{f_{fog}}$	iv) f_{fog} is processing power of the fog processor	Time required to process a fraction of a workload generated at IoT and offloaded to the Fog for processing. [8, 58, 59]
Transmission Time to selected fog	$\tau_{tf} = \lambda \times \frac{\Omega_{iot}}{\beta_{tf}}$	β_{tf} = is the transmission rate on the link(s) between the selected fog node and the IoT device	Amount of time required to transmit the offloaded workload to the fog device [8, 58, 59]
Transmission Time to the cloud	$\tau_{tc} = \lambda \times \frac{\Omega_{iot}}{\beta_{tc}}$	β_{tc} = is the transmission rate on the link(s) between the selected cloud and the IoT device	Amount of time required to transmit the offloaded workload to the cloud [8, 58, 59]

Table 2: showing Time constructs required during offloading process

3.3. Formation of optimization framework

An optimization framework for computational offloading in clustered fog-cloud of things is presented in this sub section. The optimisation framework determines the optimal computational offloading strategy.

Optimisation related entities

1. **Gateway:** From our model presented in figure 1, all IoT devices are associated with a smart Gateway forming a cluster, therefore all the tasks generated from the lower levels of the model are surrogated by the gateway. In our optimisation framework, IoT devices are encapsulate by the gateway.
2. **The Fog:** This middle tier computing devices at the edge that is responsible for computational offloading. In case none of the fogs in the clusters can perform computation offloading, then the tasks are forwarded to the upper tier of the network model.
3. **The cloud:** Form the upper tie computing environment that is responsible for computational offloading if and only if there exists no fog from the computing environment to perform computation offloading.

Overall objective function

$$\text{Maximize Task allocation: TA} = \sum_{j=1}^{G} \sum_{i=0}^{T_j} \sum_{k=1}^{F} x_{i,j,k} \tag{7}$$

subject to:

Symbols	Description
g_j	The j^{th} Gateway $j = 1, 2...G$
$T_{i,j}$	The i^{th} Task at j^{th} Gateway $i = 1, 2...T_j$
f_k	The k^{th} fog to which offloading is performed $k = 1, 2...F$
$D_{i,j}$	Maximum delay tolerance between the task $T_{i,j}$
$d_{i,j}$	Delay experienced during the processing of task $T_{i,j}$
t_d	Round trip time
t_c	Computational time at the fog
$x_{i,j,k} = \{0, 1\}$	Boolean indicator if $x_{i,j,k} = 1$ then that fog is chosen.
	this indicator is used to generate the assignment table
h_{jk}	Number of hope counts between g_i and the selected fog
l_d	the average link delay
m_{ij}	the memory requirement for tij
M_k	the maximum available memory at the f_k
$c_{i,j}$	the required processing power required for a task ti, j
C_k	the maximum available processing power available at the Fog f_k

Table 3: showing symbols used in formation of optimization framework

$$\sum_{k=1}^{F} x_{i,j,k} \leq 1$$

$i = 1, 2, ...T_j; \quad j = 1, 2, ...G$

$$d_{i,j,k} * x_{i,j,k} \leq D_{i,j,k}, i = 1, 2, ...T_j; \quad j = 1, 2, ...G$$

$$m_k \leq M_k, k = 1...F$$

$$c_k \leq C_k, k = 1...F$$

$$x_{i,j,k} = \{0, 1\}$$

$$\begin{aligned} d_{i,j} &= t_d + t_c \\ &= t_d \text{ is twice the delay between the selected fog and the gateway} \\ t_d &= 2 \times h_{j,k} \times l_d \end{aligned} \tag{8}$$

$$\begin{aligned} d_{i,j,k} &= t_c + 2 \times h_{j,k} \times l_d \\ &= \text{delay assigned if } t_{i,j} \text{ is assigned to a fog } f_k \\ &= x_{i,j,k} \times d_{i,j,k} \leq D_{i,j} \end{aligned} \tag{9}$$

Resource requirement

i) Central Processing Unit (CPU) requirement for task $t_{i,j}$ is measured in CPU cycles as follows

$$c_k = \sum_{j=1}^{G} \sum_{i=0}^{T_j} x_{i,j,k} \times c_{ik} \leq C_k \tag{10}$$

ii) Memory requirement for task $t_{i,j}$ is measured in bits as follows

$$m_k = \sum_{j=1}^{G} \sum_{i=0}^{T_j} x_{i,j,k} \times m_{ik} \leq M_k \tag{11}$$

4. The modified Dynamic PSO for computational offloading

In this section, we present the mDyPSO A PSO motivated computational offloading algorithm applied in multiple cluster topology of fog-cloud of things ecosystem. This algorithm is based on dynamic particle swarm optimization mechanism, which forms a basis of many selection and scheduling problems[60, 61]. Studies in [62, 63] proposed PSO mechanism to solve multi-objective problems. Since this study involves Fog-cloud of things clustered in multiple domains whose topology and search space may vary. It becomes central to treat the problem as a multi-objective particle swarm optimization problem.

4.1. Traditional Dynamic Particle Swarm Optimisation algorithm

The traditional particle swarm optimisation is best suited for problems that are represented in n-dimension space[8]. Using a particle with defined velocity, acceleration and communication channels between them, they are made to drift towards the best suited solutions known as the best fit among the other potential solutions [42]. The dynamic variation of Particle swarm optimisation considers that the swarm size may not be the same or the space consist of varying topology. Therefore in such a case acceleration is weighted by random term and average of the fitness may be considered [64]. in algorithm 1 a dynamic particle swarm optimisation is presented.

```
Method DyPSO():
    $noParticles \leftarrow p = (1, 2, ...)$
    $avaragefit \leftarrow 0$
    $currentfit \leftarrow 0$
    $fitness$
    $personalBest \leftarrow 0$
    $bestFit \leftarrow max(personalBest_p)$
    $globalBest \leftarrow bestFit$
    for all the particles do
        if $currentFit \leq avarageFit$ then
            $currentFit \leftarrow avaragaFit$
            compute $fitness$
        end
        if $personalBest \leq fitness$ then
            $personalBest \leftarrow fitness$
        end
    end
    select a particle with bestFit as globalBest
    for All the particles do
        Compute particleVelocity using equation 5
        compute particlePosition using equation 6
    end
End DyPSO
```

Algorithm 1: TradDyPSO

4.2. Dynamic Task allocation algorithm

Algorithm 3 presents the dynamic task allocation in the fog-cloud of thing ecosystem. Application tasks arrive at the gateway for offloading to the optimal-fog that provides sufficient resources to execute the task with minimum latency.For each of the cluster mDyPSO is initiated to determine an optimal fog node for a cluster. The best fit fog amongst all the clusters is assigned the function of a global best fit to whom the tasks are allocated. If all the fogs are busy and can not execute the tasks allocated in the threshold allocated the task is forwarded for processing to the cloud.

```
Method Dynamic-task-allocation():
    $resource - bank \leftarrow 0$
    $resource - required \leftarrow 0$
    $min - reserve\ sum - of - resources\ fog \leftarrow fog_1, fog_2, ...fog_j$
    $clustersize \leftarrow m$
    $cloud \leftarrow cloud_1, cloud_2...$

    for all incoming application tasks $A\tau_1, A\tau_2, A\tau_3...$ do
        for each of the clusters $K$ do
            $local - optimal - fog \leftarrow DyPSO(fog)$
            if local-optimal-fog is better than global-optimal-fog then
                $global - optimal - fog \leftarrow local - optimal - fog$
            end
            allocate-task(global-optimal-fog)
        end
        if all fogs are busy then
            allocate-task(cloud)
        end
    end
End Dynamic-allocation
```

Algorithm 2: Dynamic Task allocation algorithm

```
Method mDyPSO():
    $local - optimal - fog \leftarrow 0$
    $global - optimal - fog \leftarrow 0$
    $fog \leftarrow fog_1, fog_2, ...fog_j$
    $gateway \leftarrow gw_1, gw_2, ...gw_m$
    $clustersize \leftarrow m$
    $cloud \leftarrow cloud_1, cloud_2...$
    $required_memory \leftarrow m_{i,j}$
    $available_memory_fog \leftarrow M_k$
    $required_cpu_cycle \leftarrow c_{ij}$
    $available_cpu_cycles_fog \leftarrow M_k$

    for each of the clusters $i$ do
        for all incoming application tasks $A\tau_{i,1}, A\tau_{i,2}, A\tau_{i,3}...$ do
            for each fog do
                if $m_{i,j} - M_k \leq 0$ then
                    if $c_{i,j} - C_k \leq 0$ then
                        $f_k \leftarrow DyPSO(fog)$
                        assign $A\tau_{ij} to f_k$
                    end
                end
            end
        end
    end
mDyPSO
```

Algorithm 3: Modified dynamic Particle Swarm Optimisation

5. Comparative study between SiPSO and mDyPSO

5.1. Experimental environment and setting

5.1.1. System configuration

Name	Description
Simulation tool	IFogsim
Operating System	Windows 10
Development Platform	Eclipse IDE for Java Devlopers (2021-03)
Processor	IntelCore i_3 8^{th} Generation 2.1GHz 2.3GHz
Installed Memory	16GB

Table 4: Showing the system configuration

Table 4 shows the system configuration and development environment. IFogSim was used to develop and simulate our proposal. In addition, we run our simulation on a machine installed with windows 10, IntelCore i_3 8^{th} Generation 2.1GHz and 16GB installed RAM Operating system and Eclipse IDE development platform.

The choice of using IFogSim in this study is motivated by simplicity underlying the it's architecture in terms of application placement, load balancing, resource application and network utilization[10]. Secondly, IFogSim is free and popular event driven simulation tool used modeling the IoT-fog environment. the underlying architecture enables creation of physical, logical and management components that constitute the fog-cloud of things ecosystem. IFogSim gives us the capacity to evaluate resources management policies based on network usage, energy consumption and other operational costs[65].

5.1.2. simulation parameters

In our simulation, we consider number of IoT devices ranging from 60 to 70 devices per cluster. Each cluster is bound by a smart gateway. Further a hybrid cloud and fog devices ranging from 2 to 5 was considered. Lastly, a simulation area of 10000 and simulation area of 1200×200. Other parameters that included

Item	Description/number
Number of IoT nodes	[60-70] nodes
Number of Fog nodes	[2,3,4,5]
Number of cloud	1 hybrid cloud
No of application tasks	[1,4,10,20,30,40,50,60,70,80]
Bandwidth	10000
simulation Area	1200 x 200

Table 5: Showing the simulation parameters

the task arrival rates, simulation time, traffic types, traffic arrival rates are set to default setting.

5.2. Evaluation parameter

i) Application latency: - in this study we define application latency as the total round trip taken by a data packet to travel to and from the IoT device. The lower the application latency the better the offloading strategy. Higher application latency can strangle network reducing the performance.

ii) Network usage: - we define network usage as the amount of data that travel back and forth across a network due IoT applications, the fog devices and network users.

iii) Energy consumed during application execution: This parameter refers to total amount of energy consumed when an application is launched to execute on an IoT device. We note that IoT devices are often constrained by battery life. The less the energy consumed the better the mechanism for running the applications that may require intensive application.

5.3. Application latency for mDyPSO as compared SiPSO

The graph in Fig 4 shows the application latency achieved by IoT devices attached on a network that uses SiPSO mechanism and mDyPSO. From the data collected through simulation the application latency scored by SiPSO is 31.61 ms and mDyPSO scored 21.61 ms. Application latency is concealed through

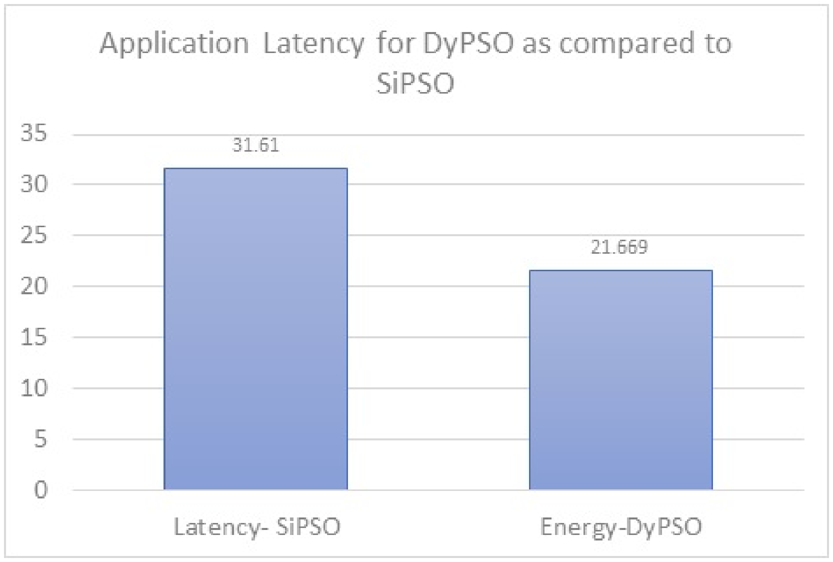


Figure 4: showing application latency for mDyPSO and SiPSO

offloading mechanism, multi-tasking techniques. Consequently, well-designed offloading algorithms for offloading can improve performance drastically. From the above, therefore, we conclude that cautious design of mDyPSO improves offloading performance by approximately one third.

5.4. Network Usage for mDyPSO as compared SiPSO

The Figure 5shows the network usage for both SiPSO and mDyPSO. From the graph, when numbers of application tasks are low, the network usage foe both mechanisms does not differ significantly. As the number of application tasks increase mDyPSO achieves better network usage than SiPSO. Since network usage determines the amount of data that move back and forth across the network due to application and related devices on the network, we conclude that mDyPSO is a better mechanism for computational offloading than SiPSO.

5.5. Energy consumption for mDyPSO as compared to SiPSO

The graph 6 above shows the energy consumption for offloading strategy using SiPSO and mDyPSO. We observe that energy consumed during offload-

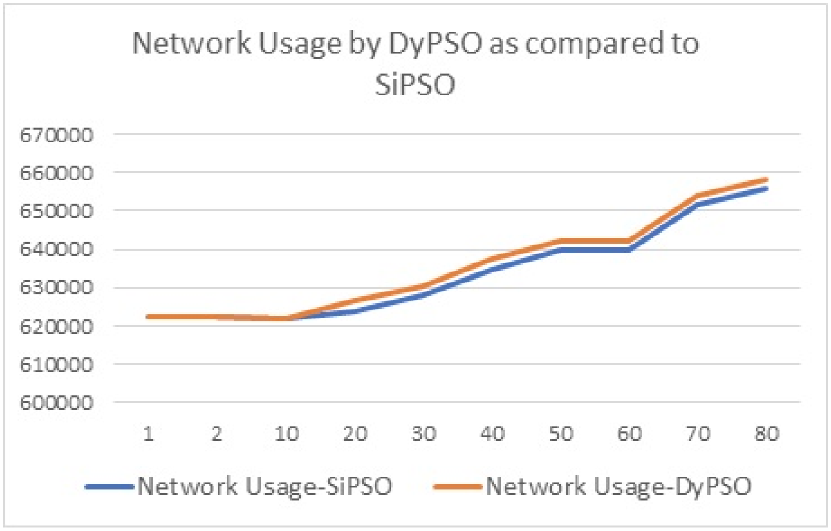


Figure 5: showing network usage for both mDyPSO and SiPSO

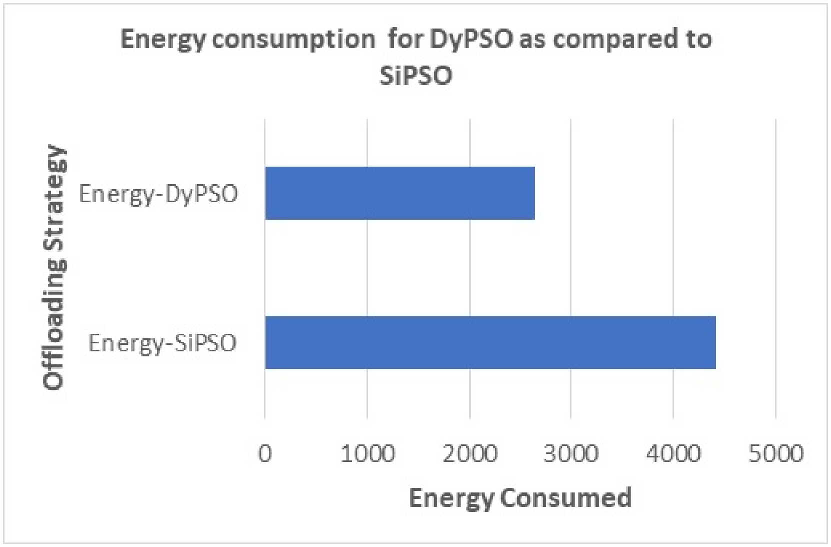


Figure 6: showing energy consumption for both mDyPSO and SiPSO

ing while using mDyPSO strategy is less that energy consumed while using SiPSO. Since the amount of energy relates to energy conserved, we conclude that mDyPSO enable better energy conservation at the IoT device hence a better offloading strategy.

6. Conclusion

Fog computing aims at improving responsiveness of real time application in the fog-cloud of things. In our study we conclude that Application latency can be reduced through adoption of well-designed mechanism in the fog-cloud of things infrastructure. Secondly, we achieve lower response by adapting computational offloading at the edge through proper resource utilization and load balancing. Our Nature inspired computational offloading in clustered fog of things ecosystem exhibits better network utilization as the network grows, a typical characteristics of IoT networks. Further our study show that network performance is reduced by one third, energy consumption is reserved and application latency is moderately lower than earlier proposed mechanism in SiPSO. In future, we hope to explore models for computational offloading with hybrid cloud in IoT ecosystem, 5G-enabled services for task offloading in fog-cloud of things ecosystems and future perspectives for fog-cloud of things computing cooperation.

References

[1] C. Tang, S. Xia, Q. Li, W. Chen, W. Fang, Resource pooling in vehicular fog computing, Journal of Cloud Computing 10 (1) (2021) 1–14.

[2] S. H. Asaba, A. A. Alli, S. A. Olawale, Y. Umar, A. Kasule, R. R. Bwambale, Review of cloud computing framework for the implementation of elearning systems, Uganda National Council of Higher Education Jounal.

[3] A. A. Alli, K. Kassim, N. Mutwalibi, H. Hamid, L. Ibrahim, Secure fog-cloud of things: Architectures, opportunities and challenges, in: Secure Edge Computing, CRC Press, 2021, pp. 3–20.

[4] J. Kasadha, A. A. Alli, A. K. Basuuta, A. Mpoza, Social media taxation and its impact on africa's economic growth, Journal of Public Affairs (2019) e2004.

[5] M. Yasin, A. A. Alli, N. Mutwalibi, J. Kasadha, Video conferencing as a teaching mode in higher educational institutions in uganda: teacher perception, International Journal of Smart Technology and Learning 3 (1) (2022) 46–66.

[6] A. A. Alli, I. LWEMBAWO, F. MUGIGAYI, J. KASADHA, Teaching as a service: An exploration of educational framework for technology driven teaching for higher education institutions, in: 2024 IST-Africa Conference (IST-Africa), IEEE, 2024, pp. 1–11.

[7] K. Kalinaki, W. Shafik, M. Masha, A. A. Alli, A review of artificial intelligence techniques for improved cloud and iot security, Emerging Technologies for Securing the Cloud and IoT (2024) 38–68.

[8] A. A. Alli, M. M. Alam, Secoff-fciot: Machine learning based secure offloading in fog-cloud of things for smart city applications, Internet of Things 7 (2019) 100070.

[9] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660.

[10] A. A. Alli, M. M. Alam, The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications, Internet of Things (2020) 100177.

[11] P. G. López, M. Sánchez-Artigas, G. París, D. B. Pons, Á. R. Ollobarren, D. A. Pinto, Comparison of faas orchestration systems, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), IEEE, 2018, pp. 148–153.

[12] D. Van Le, C.-K. Tham, A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2018, pp. 760–765.

[13] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocessors and Microsystems 26 (8) (2002) 363–371.

[14] K. A. De Jong, W. M. Spears, Using genetic algorithms to solve np-complete problems., in: ICGA, 1989, pp. 124–132.

[15] T. Q. Dinh, J. Tang, Q. D. La, T. Q. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Transactions on Communications 65 (8) (2017) 3571–3584.

[16] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez, Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach, IEEE Transactions on Vehicular Technology 68 (4) (2019) 3113–3125.

[17] Y.-H. Kao, B. Krishnamachari, Optimizing mobile computational offloading with delay constraints, in: 2014 IEEE Global Communications Conference, IEEE, 2014, pp. 2289–2294.

[18] T. Wang, X. Wei, C. Tang, J. Fan, Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints, Peer-to-Peer Networking and Applications 11 (4) (2018) 793–807.

[19] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, 2012, pp. 13–16.

[20] S. P. Ahuja, N. Wheeler, Architecture of fog-enabled and cloud-enhanced internet of things applications, International Journal of Cloud Applications and Computing (IJCAC) 10 (1) (2020) 1–10.

[21] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, L. F. Bittencourt, Mobfogsim: Simulation of mobility and migration for fog computing, Simulation Modelling Practice and Theory 101 (2020) 102062.

[22] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, Z. Han, Architectural design alternatives based on cloud/edge/fog computing for connected vehicles, IEEE Communications Surveys & Tutorials 22 (4) (2020) 2349–2377.

[23] Z. Javed, W. Mahmood, A survey based study on fog computing awareness, International Journal of Information Technology and Computer Science (IJITCS) 13 (2) (2021) 49–62.

[24] G. Carvalho, B. Cabral, V. Pereira, J. Bernardino, Computation offloading in edge computing environments using artificial intelligence techniques, Engineering Applications of Artificial Intelligence 95 (2020) 103840.

[25] H. Flores, X. Su, V. Kostakos, A. Y. Ding, P. Nurmi, S. Tarkoma, P. Hui, Y. Li, Large-scale offloading in the internet of things, in: Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on, IEEE, 2017, pp. 479–484.

[26] F. Gu, J. Niu, Z. Qi, M. Atiquzzaman, Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions, Journal of Network and Computer Applications 119 (2018) 83–96.

[27] Z. Li, C. Wang, R. Xu, Computation offloading to save energy on handheld devices: a partition scheme, in: Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems, 2001, pp. 238–246.

[28] M. Gao, R. Shen, L. Shi, W. Qi, J. Li, Y. Li, Task partitioning and offload-

ing in dnn-task enabled mobile edge computing networks, IEEE Transactions on Mobile Computing.

[29] J. Liu, Q. Zhang, Adaptive task partitioning at local device or remote edge server for offloading in mec, in: 2020 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2020, pp. 1–6.

[30] S. Singh, Optimize cloud computations using edge computing, in: 2017 International Conference on Big Data, IoT and Data Science (BID), IEEE, 2017, pp. 49–53.

[31] K. Akherfi, M. Gerndt, H. Harroud, Mobile cloud computing for computation offloading: Issues and challenges, Applied computing and informatics 14 (1) (2018) 1–16.

[32] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, S. Papavassiliou, Intelligent dynamic data offloading in a competitive mobile edge computing market, Future Internet 11 (5) (2019) 118.

[33] X.-S. Yang, Nature-inspired optimization algorithms: challenges and open problems, Journal of Computational Science (2020) 101104.

[34] K. Chaudhari, A. Thakkar, Travelling salesman problem: An empirical comparison between aco, pso, abc, fa and ga, in: Emerging Research in Computing, Information, Communication and Applications, Springer, 2019, pp. 397–405.

[35] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[36] P. K. Tripathi, S. Bandyopadhyay, S. K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Information sciences 177 (22) (2007) 5033–5049.

[37] W. Bin, P. Qinke, Z. Jing, C. Xiao, A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior, European Journal of Operational Research 219 (2) (2012) 224–233.

[38] M. Rosendo, A. Pozo, Applying a discrete particle swarm optimization algorithm to combinatorial problems, in: 2010 Eleventh Brazilian Symposium on Neural Networks, IEEE, 2010, pp. 235–240.

[39] I. K. Gupta, S. Shakil, S. Shakil, A hybrid ga-pso algorithm to solve traveling salesman problem, in: Computational Intelligence: Theories, Applications and Future Directions-Volume I, Springer, 2019, pp. 453–462.

[40] A. W. Mohemmed, N. C. Sahoo, T. K. Geok, Solving shortest path problem using particle swarm optimization, Applied Soft Computing 8 (4) (2008) 1643–1653.

[41] H. Huang, Z. Hao, Particle swarm optimization algorithm for transportation problems, Particle swarm optimization, ed. Aleksandar Lazinica, In-Tech (2009) 275–290.

[42] J. L. Pierobom, M. R. Delgado, C. A. A. Kaestner, Particle swarm optimization applied to task assignment problem, ChemBioChem (2016) 1–8.

[43] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, C. Maple, A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing, IEEE Access 7 (2019) 115760–115773.

[44] U. Saleem, Y. Liu, S. Jangsher, Y. Li, Performance guaranteed partial offloading for mobile edge computing, in: 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, 2018, pp. 1–6.

[45] L. Li, X. Zhang, K. Liu, F. Jiang, J. Peng, An energy-aware task offloading mechanism in multiuser mobile-edge cloud computing, Mobile Information Systems 2018.

[46] P. Zhao, H. Tian, C. Qin, G. Nie, Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing, IEEE Access 5 (2017) 11255–11268.

[47] S. Misra, N. Saha, Detour: Dynamic task offloading in software-defined fog for iot applications, IEEE Journal on Selected Areas in Communications 37 (5) (2019) 1159–1166.

[48] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, R. Buyya, An online algorithm for task offloading in heterogeneous mobile clouds, ACM Transactions on Internet Technology (TOIT) 18 (2) (2018) 1–25.

[49] A. Kishor, C. Chakarbarty, Task offloading in fog computing for using smart ant colony optimization, Wireless Personal Communications (2021) 1–22.

[50] M. Keshavarznejad, M. H. Rezvani, S. Adabi, Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms, Cluster Computing (2021) 1–29.

[51] J. Wang, J. Hu, G. Min, A. Y. Zomaya, N. Georgalas, Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Transactions on Parallel and Distributed Systems 32 (1) (2020) 242–253.

[52] L. Huang, X. Feng, C. Zhang, L. Qian, Y. Wu, Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing, Digital Communications and Networks 5 (1) (2019) 10–17.

[53] D. S. Rani, M. Pounambal, Deep learning based dynamic task offloading in mobile cloudlet environments, Evolutionary Intelligence (2019) 1–9.

[54] J. Liu, X. Wang, S. Shen, G. Yue, S. Yu, M. Li, A bayesian q-learning game for dependable task offloading against ddos attacks in sensor edge cloud, IEEE Internet of Things Journal 8 (9) (2020) 7546–7561.

[55] J. Wang, W. Wu, Z. Liao, R. S. Sherratt, G.-J. Kim, O. Alfarraj, A. Alzubi, A. Tolba, A probability preferred priori offloading mechanism in mobile edge computing, IEEE Access 8 (2020) 39758–39767.

[56] N. Zhang, S. Guo, Y. Dong, D. Liu, Joint task offloading and data caching in mobile edge computing networks, Computer Networks 182 (2020) 107446.

[57] X. Zhang, R. Zhou, Z. Zhou, J. C. Lui, Z. Li, An online learning-based task offloading framework for 5g small cell networks, in: 49th International Conference on Parallel Processing-ICPP, 2020, pp. 1–11.

[58] M. Min, D. Xu, L. Xiao, Y. Tang, D. Wu, Learning-based computation offloading for iot devices with energy harvesting, arXiv preprint arXiv:1712.08768.

[59] L. Belem Pacheco, E. Pelinson Alchieri, P. Mendez Barreto, Device-based security to improve user privacy in the internet of things, Sensors 18 (8) (2018) 2664.

[60] Z. Wang, J. Zhang, S. Yang, An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals, Swarm and Evolutionary Computation 51 (2019) 100594.

[61] M. A. A. Aziz, M. N. Taib, N. M. Hussin, An improved event selection technique in a modified pso algorithm to solve class scheduling problems, in: 2009 IEEE Symposium on Industrial Electronics & Applications, Vol. 1, IEEE, 2009, pp. 203–208.

[62] N. Delgarm, B. Sajadi, F. Kowsary, S. Delgarm, Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (pso), Applied energy 170 (2016) 293–303.

[63] L.-b. Zhang, C.-g. Zhou, M. Ma, X.-h. Liu, Solutions of multi-objective optimization problems based on particle swarm optimization, Journal of computer research and development 7 (41) (2004) 7.

[64] H. S. Urade, R. Patel, Dynamic particle swarm optimization to solve multi-objective optimization problem, Procedia Technology 6 (2012) 283–290.

[65] S. V. Margariti, V. V. Dimakopoulos, G. Tsoumanis, Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective, Future Internet 12 (5) (2020) 89.