Improvement of Task Offloading for Latency Sensitive Tasks in Fog Environment

Parmeet Kaur and Shikha Mehta

Abstract Fog computing is gaining rapid acceptance as a distributed computing paradigm that brings cloud-like services near the end devices. It enhances the computation capabilities of mobile nodes and IoT (Internet of Things) devices by providing compute and storage capabilities similar to the cloud but at a lower latency and using lesser bandwidth. Additional advantages of fog computing include its support for node mobility, context awareness, reliability and scalability. Due to its multiple benefits, fog computing is used for offloading tasks from applications executing on end devices. This allows faster execution of applications using the capabilities of fog nodes. However, the task offloading problem in the fog environment is challenging due to the dynamic nature of fog environment and multiple QoS (Quality of Service) parameters dependent on the application being executed. Therefore, the chapter proposes a QoS-aware task offloading strategy using a novel nature-inspired optimization algorithm, known as the Smart Flower Optimization Algorithm (SFOA). The proposed strategy takes into account the QoS parameters such as the task deadlines and budget constraints in selection of appropriate fog nodes where computation tasks can be offloaded. The proposed strategy has been simulated and the results have verified the efficacy of the strategy.

Keywords Task offloading · Fog computing · IoT · Smart Flower Optimization Algorithm · Particle Swarm Optimization · Shuffled Frog Leaping Algorithm

P. Kaur (✉) · S. Mehta
Department of Computer Science & Enginnering and Information Technology, Jaypee Institute of Information Technology, Noida, India
e-mail: parmeet.kaur@jiit.ac.in

S. Mehta
e-mail: shikha.mehta@jiit.ac.in

R. Tiwari et al. (eds.), *Energy Conservation Solutions for Fog-Edge Computing Paradigms*, Lecture Notes on Data Engineering and Communications Technologies 74,
https://doi.org/10.1007/978-981-16-3448-2_3

1 Introduction

Mobile nodes and sensor devices are finding a number of applications in today's smart world. Apart from collecting rich data, many sophisticated tasks are also being executed on such devices. These tasks or applications are usually data-driven and require computational power to process this data. Since mobile devices and sensors may face resource constraints in terms of limited storage and compute powers, cloud computing has been used to support these devices. Tasks requiring computation power or data exceeding the storage available at mobile devices or sensors are transferred to cloud servers for successful execution. The transfer of data and related tasks from a resource-constrained device to external servers is referred to as offloading [1, 2].

Cloud computing provides multiple benefits in context of task offloading in resource-constrained environments. The most important of these benefits is the availability of a vast pool of heterogeneous resources that can be leveraged for computation of the required tasks [3]. The obtained results are the transferred to the destination nodes. However, use of cloud computing is not preferable for latency-sensitive tasks. This is due to the centralized nature of cloud computing as a result of which transfer of tasks and subsequently the results involves latency. The amount of latency depends on the bandwidth availability for communication between the nodes (mobile or sensor) and the cloud servers. This delay makes cloud-based processing infeasible for many tasks. Further, larger size tasks may take a longer time in transferring the cloud. Therefore, cloud may not be a viable solution for tasks that need immediate execution and cannot afford latency. Instead, there is a need for computing resource availability closer to the nodes so that results can be obtained in desired time [4].

Fog computing is emerging as a possible solution for the offloading problem for latency-sensitive tasks. It is a distributed computing paradigm that acts as an intermediate layer between offshore Cloud data centres and IoT devices or sensors/mobile nodes [5]. Fog devices are placed at the edge of network and closer to the IoT devices as compared to cloud servers. Fog computing provides computation, networking and storage facilities to resource-constrained devices so that cloud-like services can be extended closer to the IoT devices, sensors or mobile nodes. The concept of Fog computing was first introduced by Cisco in 2012 to address the challenges of IoT applications in conventional Cloud computing. It is related to the concept of Edge computing. If fog devices are used for processing the tasks, it will reduce the latency in processing significantly. Further, this also helps in alleviating the network congestion that can occur when a large number of IoT devices transfer data and tasks simultaneously to the cloud servers. In comparison, a judicious selection of fog nodes for task offloading by specific devices can prevent network congestion.

This chapter presents a Smart Flower Optimization Algorithm (SFOA) [6] based task offloading approach for latency-sensitive tasks in fog environment. Tasks that need quick processing are offloaded to selected Fog nodes in the system. The selection of fog nodes for specific tasks is done by using SFOA which is a new and

robust nature-inspired algorithm for solving optimization problems. Nature-inspired algorithms have been found to provide near-optimal solutions in multiple complex problems such as resource allocation, scheduling of workflows, search optimizations, etc. Implementation of SFOA in task offloading problem was postulated to give effective results and this was validated by the simulation results. Its performance has been compared with that of two other well-known nature-inspired algorithms, namely, Particle Swarm Optimization (PSO) and Shuffled Frog Leaping Algorithm (SFLA).

The chapter makes the following contributions:

- Presents task offloading problem as an optimization problem in the IoT-fog environment
- Presents a discrete version of a new nature-inspired algorithm (NIA), the Smart Flower Optimization Algorithm and maps the task offloading problem to it.
- Evaluates the efficacy of SFOA under varying workloads for the considered problem and compares with two other well-known nature-inspired algorithms
- Determines which NIA performs well for a given workload.

The chapter is organized as follows: Sect. 2 provides an overview of the paradigms of cloud, fog and edge computing. Literature review is presented in Sect. 3. Section 4 discusses the Smart Flower Optimization algorithm. The use of WOA for task offloading problem has been described in Sect. 5. Results of simulation experiments are put forth in Sect. 6. Concluding remarks are listed in Sect. 7.

2 Cloud, Edge and Fog Computing

With the growth in technology, computing has become possible on a variety of devices, such as mobile nodes and Internet of Things (IoT) devices. Of these, IoT connects multiple, ranging in hundreds to billions, of heterogeneous nodes through a network for gathering data from the underlying system. This data is analyzed and processed to gain insights about the system. IoT has led to development of applications such as smart transportation, smart cities, smart homes, smart agriculture, waste management, etc. [7, 8]. Similarly, many applications such as gaming, video processing, etc. are being commonly executed on mobile devices. However, despite the advancements in technology, IoT and mobile devices are still resource-constrained. These devices possess limited storage, computing and networking capabilities and may not be able to independently process vast amounts of data or execute compute-intensive tasks within a given time frame. If size of data or complexity of applications increases, these devices are unable to execute these applications within a given time.

Cloud computing has been used to perform computing and storage tasks for these devices. Cloud-based servers can support data storage and computations by providing resources on a pay-per-use over the Internet. However, computing on cloud, it being a centralized infrastructure, causes a delay in execution. Applications such as sprinkling water in response to fire detection, changing settings of devices, opening a

valve if pressure reading exceeds a threshold, sending an alert to a health worker in case an anomaly is detected, etc. cannot involve such delays in execution. Further, the growth in number of IoT devices also increases the chances of network congestion as the size of data being transferred increases. Therefore, there is a need to minimize the processing time for latency-sensitive applications while also utilizing the computing resources efficiently.

One method that has emerged to timely process latency-sensitive applications is the use of special nodes, known as fog nodes placed in the local network near the IoT devices or the mobile nodes. Tasks can be offloaded to a suitable fog node that has the adequate processing and storage resources. Though a fog node cannot match the resource availability of a cloud, yet a number of fog nodes collectively can be employed to meet the computing requirements of a given set of IoT devices/mobile nodes. These fog components are equipped with higher computing, storage or networking capabilities and are able to support execution of latency-sensitive tasks not possible on the IoT devices themselves. Network switches, routers, Raspberry Pi, etc. can act as fog nodes. Fog computing is related closely to the concept of edge computing. An edge network comprises of end devices, such as mobile phones, smart devices, etc. or the edge devices such as set-top boxes, bridges, base stations, wireless access points, etc. and the edge servers [9]. Edge computing refers to computing at the device itself, while fog computing causes computing to take place in the local network of the device. The use of fog nodes is beneficial, especially, for latency-sensitive applications as the round-trip time to cloud is eliminated. Cloud can be used to store summaries and statistics periodically from multiple fog nodes and also for executing applications that can take longer times to execute.

A typical fog computing system is illustrated in Fig. 1 and used in the current work. It is a three-layered architecture, where the IoT devices such as the sensors, actuators or the mobile devices form the first layer. The fog nodes comprise of the second or intermediate layer which takes input (in the form of tasks or data) from the IoT devices and subsequently returns results or values to them. Cloud computing resources are the third layer in the architecture. Each layer is connected to the other through some communication medium.

Fog computing nodes are located closer to the IoT, they act as an intermediate layer between the IoT nodes and cloud computing. We assume a flat organization of fog nodes in the current work. However, they can form dynamic or static Clusters based on their location or the application requirements. Similarly, master-slave organization may also be used where a master Fog node can coordinate the functions, workload distribution, data flow, etc. of the slave nodes. In the current problem, the IoT devices should make a judicious selection of fog nodes for task offloading. This involves checking the availability of required resources along with the latency incurred in the offloading process.

There can be several QoS parameters that need to be met while offloading tasks. Sine latency-sensitive tasks are being considered here, latency of response from the fog node is the most important parameter. It is imperative that the results are delivered from the fog nodes to IoT devices within an acceptable time interval that is defined as the maximum tolerable latency of a task. Another QoS parameter relates to the

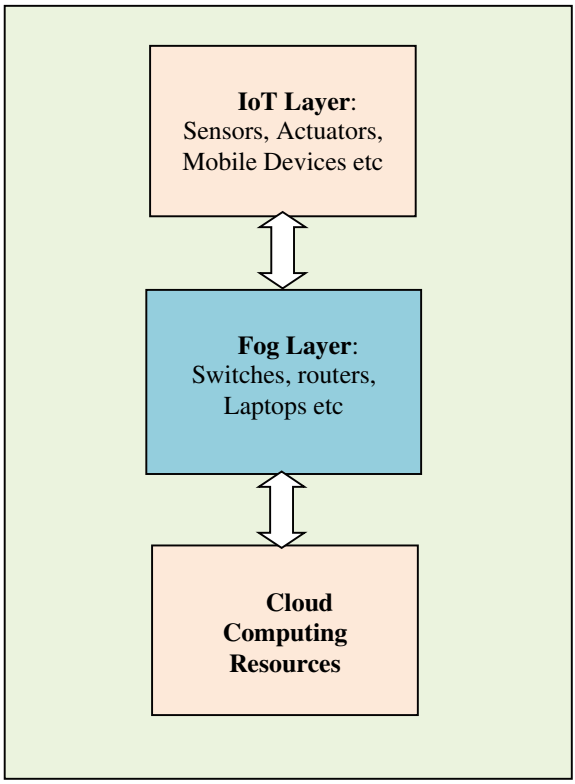

Fig. 1 System model

cost involved in use of fog nodes. The cost is incurred when fog nodes are deployed in the network as well as during their operation. Hence, fog nodes can be used only if the gains exceed these costs. Similarly, there can be QoS parameters related to user requirements of security, privacy, etc.

Since the selection of fog nodes for task offloading is guided by multiple objectives along with the QoS parameters, it is a complex and challenging problem. Nature-inspired algorithms have often been used for solving such problems. Hence, this work explores the Smart Flower optimization algorithm for finding a near-optimal solution to the task offloading problem. Selection of a suitable computing device for each delay-sensitive task is a challenge addressed by the proposed solution using Smart Flower optimization algorithm (SFO) technique. The proposed SFO based strategy finds an optimal computing device (i.e., fog device) for each real-time task using multiple Quality-of-Service (QoS) parameters, namely cost and resource utilization.

3 Literature Review

The approach to offload tasks from resource-constrained devices to fog devices has garnered significant research attention in last few years [5, 10]. Different works have focussed on optimization of various goals with respect to offloading. A method to offload IoT applications with an objective to minimize the execution time along with the energy consumption of the resources has been proposed in [11]. The offloading strategy in [12] focussed on reducing the latency of applications and energy usage of the fog devices. A few other offloading approaches, such as [13, 14] have also been developed for minimizing the energy consumption of resources and the execution time of the applications. An offloading approach specific to latency-sensitive applications has been presented in [15] with the aim of decreasing the time required for computation by the applications. The framework proposed in [16] endeavours to minimize the service delay of IoT applications executing in fog-cloud environment support. Based on the size of a task, the framework ensures that the delay that will be caused by its execution on a fog node is less than a threshold of time. Otherwise, it finds another fog node that will complete the execution of the task within the stipulated time. If no such fog node is found, the task is transferred to a cloud server. Another offloading approach for delay-sensitive applications has been proposed in [17].

Workload from IoT nodes is divided among fog nodes and cloud servers by the work in [18] in a way that reduces the consumption of power and the delay incurred in service delivery. In [18], an analysis of the workload allocation between the fog nodes and the cloud nodes is investigated to minimize the power consumption under service delay constraints. This approach assumes connection and cooperation between fog nodes and cloud servers.

The authors of [19] considered balancing the load distribution among fog nodes while offloading tasks to these nodes. This is expected to give an efficient application execution. In comparison, the work in [20] is based on tasks' priority in a system that assumes the presence of fog nodes as well as the cloud services. By prioritizing the tasks, waiting time and delay of tasks that need to meet a deadline are reduced.

Due to the complexity of this problem, several heuristic and meta-heuristic-based solutions have been presented in literature. The work in [21] presents a task offloading strategy for IoT applications using task classification and calculating cost of execution. However, the authors have not accounted for the cost incurred in communication amongst tasks. In comparison, a genetic algorithm-based single objective data offloading approach has been proposed in [22]. The objective is to minimize the overall cost of data offloading in a fog-based system by reducing the costs of execution as well as communication between tasks. In comparison, the work in [23] aims to minimize the overall time and cost of application execution. A few other approaches of task offloading [24, 25] have also paid attention to efficient resource utilization in the fog environment. The work in [26] utilizes Bee Swarm algorithm for optimizing the execution time along with efficient resource usage. Similar objectives are achieved using a genetic algorithm (GA) based offloading approach in [27].

Further, the approach of Particle Swarm Optimization (PSO) is utilized in designing an offloading strategy that optimizes the energy consumption in the fog environment along with the latency in [28]. Authors of [29] have presented a deadline-aware offloading approach that uses both GA and PSO to optimize the delay and execution time of applications.

Another swarm optimization algorithm, Ant Colony Optimization (ACO) is employed in [30] for scheduling tasks of IoT applications on the fog nodes. The proposed approach aims to reduce the delay of task execution using profiling the IoT tasks. A strategy to optimize operational time and cost using a genetic algorithm is presented in [31]. Another genetic algorithm based offloading strategy in fog systems is put forth in [32]. The work in [33] employs a variation of the PSO algorithm to select an appropriate fog for each real-time IoT task. The work considers the parameters of cost and resource utilization. An Ant Colony Optimization based task distribution over the fog nodes is proposed in [34] to enhance quality of service and reduce the task response time. The present work makes use of a recent nature-inspired algorithm for the considered problem.

4 Smart Flower Optimization Algorithm

The Smart Flower Optimization Algorithm (SFOA) is a meta-heuristic technique inspired by the growth behaviour of immature sunflowers. In this algorithm, the stem length of the sunflower represents the individual solution of the population. The growth of stem length in sunflowers is determined by the internal mechanisms like direction of the sun in the daytime and biological clock all through the night. It is well known that direction of the immature sunflowers changes according to the sun such that during the day, immature sunflowers align themselves in the east and gradually turn towards the west with the change in direction of the sun. After the sun set, they swing themselves again towards the east waiting for the sunrise. These heliotropic movements are only shown by the immature sunflowers. For simulation, the growth of immature sunflower is modelled mathematically along two dimensions: Sunny day vs cloudy or rainy day. As the growth pattern of immature sunflower is different on cloudy or rainy days as compared to sunny days.

In SFOA, stem length of the immature sunflower represents the potential solution to the problem to be solved. Since SFOA is used to solve optimization problems, objective of the problem represents the fitness function that indicates the stem length of the sunflower. Longer the stem length better is the solution. The evolved stems length or novel individuals are created according to the internal mechanism that is heliotropic movements, which are responsible for complete growth of the sunflower during the day in the decision space decision space. The detailed steps of SFOA are shown in Fig. 2. As SFOA is swarm intelligent technique which begins with generation of random population of initial potential solution depending upon the problem to be solved. In the next step fitness of all potential solutions is computed and best

Input: population size(P), maximum number of iterations(Itr_{max}), number of dimensions(dim), sun parameter(sun)

Output: Best solution(L_{best}), Fitness of the best solution(f_{best})

//Main Algorithm
Generate initial population of size P randomly.
Compute fitness of all solutions.
Identify and Store fitness of best solution or best length as $L_{best.}$
For Itr=1 to Itr_{max}
$d = damping_{max} - Itr * (\frac{(damping_{max} - damping_{min})}{Itr_{max}})$ **//Generate the damping factor(d)**
For i=1 to P
Generate parameter ↻ where ↻ is the angle of sine function $\in [0, 160^o]$
For j =1 to dim
If sun= =1 //sunny day
Aux $\in [0, 1]$ **//growth harmone**
Hrs $= \in [0, 100]$ **// biological clock**
If Hrs<=24
$L_{new\,SF}^{Itr+1} = L_{old\,SF}^{Itr+1} + d * sin(↻) * [Aux * L_{best\,SF}^{Itr+1} - L_{old\,SF}^{Itr+1}]$ **(1) //sunny day**
Else
$L_{new\,SF}^{Itr+1} = L_{old\,SF}^{Itr+1} + d * sin(↻) * [L_{best\,SF}^{Itr+1} - L_{old\,SF}^{Itr+1}]$ **(2) // cloudy or rainy day**
$\varphi \in [0, 0.01]$
$\omega^{j+1} = \omega^{j} + \varphi$
end j
end i
Compute fitness of newly generated population and find $L_{best\ new.}$
If $L_{best\ new} < L_{best.}$
$L_{best\ new} = L_{best.}$
End Itr

Fig. 2 Description of Smart Flower Optimization Algorithm

solution of the population is identified as Lbest. The algorithm is executed for predefined number of iterations. In every iteration, evolution of the candidate solutions is performed according to the growth mechanism of the immature sunflowers.

For simulating the heliotropic movements, parameter "Sun" is used for switching among the sunny and cloudy modes. Value of "Sun" parameter is assigned '1' for sunny day and it is assigned value of '0' for the cloudy/rainy day's mode. The default mode is Sunny day. In nature, heliotropic movements of the immature sunflower decrease gradually with the change in direction of the sun from sunrise to sunset and stop completely at the mature stage. This behaviour is simulated with the help of a damping factor 'd' and is computed using Eq. 1 as shown in the algorithm. Heliotropic movements occurring due to 'Auxin' hormone are represented using the sine function. 'Auxin' is a growth hormone responsible for development of sunflower and stem elongation along with inducing the heliotropic movements during a 24 h day and night cycle. In the SFOA, 'Aux' variable represents 'Auxin' and its value is chosen in the range of [0, 1]. Since heliotropic movements happen only during the day, updates in the stem length are represented using Eq. 1 whenever hours of the day are less than 24. Equation 1 incorporates the effects of Auxin hormone using

'Aux' parameter. The angles made by the heliotropic movements are defined using parameter 'ω'. To depict the random movements of sunflower in odd hours, value of 'ω' is updated using phase angle 'ϕ'. The value of 'ϕ' is decided at random in the range [0, 0.01]. After evolution, fitness of new stem length is compared with the current best, if it's better, it is updated else old fitness value is retained. This whole process is carried out for predefined number of iterations. In the presented work SFOA is adapted to solve task offloading problem in fog environment. For the same, fitness or length of the flower is computed using Eq. 2.

5 Application of SFOA to Task Offloading Problem

The present work has utilized SFOA finding an optimal solution to the task offloading problem in the fog environment. SFOA is a meta-heuristic-based algorithm in which population comprises of immature sunflowers each of which is a n-dimensional search agent. The algorithm needs to be mapped to the task offloading problem for implementation.

In SFOA, every immature sunflower in the population is assumed to possess a stem length which represents the Dim-dimensional search space. We have developed a discrete version of the basic SFOA to obtain a solution to the considered problem. Therefore, we have used n-dimensional sunflowers in the population where *n* indicates the number of tasks for each workload that need to be offloaded to *m* fog devices. The value of each dimension of the immature sunflower is an integer between 1 and *m*, denoting the fog device on which the corresponding task is offloaded.

Consider the example of Fig. 3, if 9 tasks are required to be offloaded on 3 fog nodes, then each search agent, i.e., an immature sunflower, will be 9-dimensional where each dimension takes a value of 1, 2, or 3. Now, if the value of third dimension in a search agent is 1, it shows that the third task in the workload is offloaded to the fog node, 1 in a given solution. The fitness of each solution is computed in SFOA by using the fitness function described next.

Every nature-inspired algorithm possesses a fitness or objective function that is used to evaluate the goodness of the solution. In SFOA, the stem length of every immature sunflower denotes a potential solution to the optimization problem. Every flower is associated with the objective function to compute its fitness. A sunflower with a longer stem implies better fitness value and therefore, a better solution. Since presented work aims to reduce the execution cost of a task workload offloaded to the fog devices, the considered problem is formulated as a minimization problem. The

Structure of Search Agent									
Task No	1	2	3	4	5	6	7	8	9
Fog Device ID	2	3	2	1	3	2	1	2	1

Fig. 3 Mapping of task to fog devices

fitness value of a sunflower, F_{sf} is evaluated by computing the total cost of workload execution. This is, in turn, the sum of execution cost of each task in the workload on the corresponding fog node in the solution. The aim is to select those fog nodes for executing tasks which result in the minimum execution cost. Therefore,

$$F_{\text{sf}} = \sum_{i=1}^{n} \text{TEC}_i \tag{1}$$

where TEC_i represents the task execution cost of task, i on the fog node as indicated by the sunflower. Further, we have also computed the total execution time (TET) of each solution for compliance with the QoS parameter of execution deadline. Any solution, regardless of its execution cost is discarded if the TET exceeds the provided deadline.

$$\text{TET} = \text{FT}_i \tag{2}$$

where FT_i denotes the finish time of a task, i in the workload. Thus, the designed fitness function will provide only those solutions that adhere to the deadline, as specified for the application.

6 Simulations and Results

The efficacy of the proposed task offloading strategy using SFOA is evaluated by simulation experiments. The simulation experiments have considered different workload-Fog scenarios for evaluation. The performance of proposed approach has been studied for both small as well as large size workloads. Depending on the size of workloads, number of fog devices has been varied too. The performance of SFOA has been compared with two well-established nature-inspired algorithms, Particle Swarm Optimization (PSO) algorithm and Shuffled Frog Leaping (SFLA) algorithm [35]. All the algorithms have been executed on a Java simulator developed by the authors. Considering the random or non-deterministic nature of the nature-inspired algorithms, each simulation was performed 25 times and the results reported in the chapter are an average of the obtained results.

Particle swarm optimization (PSO) [36] is a well-established population-based nature-inspired algorithm used as a stochastic optimization technique. The algorithm was designed taking an inspiration from the flocking behaviour of birds and a similar schooling nature of fish in search of food. PSO is an evolutionary algorithm like the Genetic Algorithms (GA). The initial population of the system represents a set of random solutions and this is evolved iteratively in search of the optima. Each potential solution is called a particle and moves through the problem space by following the particles currently nearest to the optimal solution (or the food). In this manner, all particles eventually converge to the optimal solution. PSO is an easy-to-implement

algorithm that has been frequently used in several optimization problems in diverse areas, such as resource scheduling, clustering, control system design, search engines, etc.

Similar to PSO, the Shuffled Frog Leaping Algorithm (SFLA) [37] is also a nature-inspired algorithm based on the behaviour of frogs searching for food. SFLA is based on Metaheuristic memetics and has also been employed for solving multiple optimization problems in literature. It is a population-based algorithm that utilizes a local search space exploration method within a memeplex and a global exploration using communication between frogs belonging to different memeplexes. The initial population of frogs is set randomly and subsequently sorted in descending order based on their respective fitness. Frogs are then divided into memeplexes. The population evolves iteratively in SFLA by directing the worst solutions towards best local/global solutions.

The simulations were performed to evaluate the execution cost when tasks are offloaded to fog nodes using the considered algorithms. In order to assess the scalability of algorithms, the size of workloads was varied. The first experiment evaluated the SFOA for workloads with a large number of tasks. The number of tasks in the workload has been varied from 250 to 1000. Execution cost for each workload was computed of the three algorithms. It can be observed from Fig. 4 that SFOA resulted in least execution cost for each workload and outperformed both PSO and SFLA. Percentage improvement in performance for each algorithm is depicted in Table 1.

Fig. 4 Comparison of SFOA, PSO and SFLA for large workloads

Table 1 Percentage improvement due to SFOA with respect to PSO and SFLA for large workloads

	Percentage improvement of SFOA with respect to	
Number of tasks	PSO	SFLA
250	7.8	75.8
300	92.9	75.1
500	94.9	68.3
1000	95	55.6

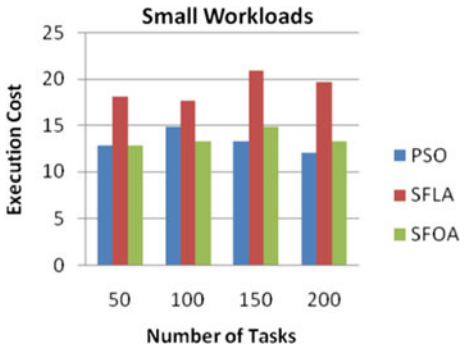

Fig. 5 Comparison of SFOA, PSO and SFLA for small workloads

Table 2 Percentage improvement due to SFOA with respect to PSO and SFLA for small workloads

	Percentage improvement of SFOA with respect	
Tasks	PSO	SFLA
50	0	28.9
100	10.8	25
150	−12.1	28.8
200	−10	32.7

The second experiment computed the execution costs for small-sized workloads using SFOA, PSO and SFLA. Similar to large-sized workloads, SFOA was observed to result in smaller execution cost than SFLA (Fig. 5). However, PSO and SFOA depicted equivalent performance as both performed comparably. Table 2 depicts the percentage improvement due to SFOA as compared to both PSO and SFLA.

From the experiments, it can be inferred that as the size of workload increases, the performance of SFOA based task offloading algorithm improves and it outperforms other algorithms.

7 Conclusion

The chapter has presented the use of a nature-inspired algorithm, namely the Smart Flower Optimization Algorithm (SFOA) for finding a solution to the task offloading problem in IoT-Fog environment. The focus of the work is upon offloading of delay-sensitive tasks that need to be executed in the IoT. Executing them on centralized and off-shore cloud computing servers is not feasible due to the latency incurred. The solution presented in this work finds a mapping between tasks and available fog nodes. The obtained mapping is based on the criterion of reducing the execution cost as well as execution cost such that the provided deadlines for task execution are not

exceeded. The performance of the presented algorithm is compared to that of two other nature-inspired algorithms, Particle Swarm Optimization (PSO) and Shuffled Frog Leaping Algorithm (SFLA). Simulation results show that performance of SFOA is comparable to PSO and better than SFLA for small-sized workloads. Moreover, SFOA outperforms SFLA and PSO both for large workloads.

References

1. Zhu Q, Si B, Yang F, Ma Y (2017) Task offloading decision in fog computing system. China Commun 14(11):59–68
2. Sun H, Huiqun Yu, Fan G, Chen L (2020) Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture. Peer-to-Peer Networking Appl 13(2):548–563
3. Xu X, Liu Q, Luo Y, Peng K, Zhang X, Meng S, Qi L (2019) A computation offloading method over big data for IoT-enabled cloud-edge computing. Futur Gener Comput Syst 95:522–533
4. Aazam M, Islam SU, Lone ST, Abbas A (2020) Cloud of things (CoT): cloud-Fog-IoT task offloading for sustainable internet of things. IEEE Trans Sustain Comput
5. Mukherjee M, Shu L, Wang Di (2018) Survey of fog computing: fundamental, network applications, and research challenges. IEEE Commun Surv Tutorials 20(3):1826–1857
6. Sattar D, Salim R (2020) A smart metaheuristic algorithm for solving engineering problems. Eng Comput, 1–29
7. Stoyanova M, Nikoloudakis Y, Panagiotakis S, Pallis E, Markakis EK (2020) A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. IEEE Commun Surv Tutorials 22(2):1191–1221
8. Puri V, Kaur P, Sachdeva S (2020) Data anonymization for privacy protection in fog-enhanced smart homes. In: 2020 6th international conference on signal processing and communication (ICSC). IEEE, pp 201–205
9. Shi W, Dustdar S (2016) The promise of edge computing. Computer 49(5):78–81
10. Aazam M, Zeadally S, Harras KA (2018) Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. Futur Gener Comput Syst 87:278–289
11. Wang Y, Wang K, Huang H, Miyazaki T, Guo S (2018) Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. IEEE Trans Industr Inf 15(2):976–986
12. Liang K, Zhao L, Chu X, Chen H-H (2017) An integrated architecture for software defined and virtualized radio access networks with fog computing. IEEE Network 31(1):80–87
13. Zhao X, Zhao L, Liang K (2016) An energy consumption oriented offloading algorithm for fog computing. In: International conference on heterogeneous networking for quality, reliability, security and robustness. Springer, pp 293–301
14. Zhang K, Mao Y, Leng S, Zhao Q, Li L, Peng X, Pan L, Maharjan S, Zhang Y (2016) Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. IEEE access 4:5896–5907
15. Hasan R, Hossain M, Khan R (2018) Aura: an incentive-driven adhoc iot cloud framework for proximal mobile computation offloading. Futur Gener Comput Syst 86:821–835
16. Wang Y, Sheng M, Wang X, Wang L, Li J (2016) Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. on Commun 64(10):4268–4282
17. Tan H, Han Z, Li X-Y, Lau FC (2017) Online job dispatching and scheduling in edge-clouds. In: INFOCOM 2017-IEEE conference on computer communications. IEEE, pp 1–9
18. Yousefpour A, Ishigaki G, Jue JP (2017) Fog computing: towards minimizing delay in the internet of things. In: Proceedings of IEEE international conference edge computing (EDGE). pp 17–24

19. He J, Wei J, Chen K, Tang Z, Zhou Y, Zhang Y (2018) Multitier fog computing with large-scale iot data analytics for smart cities. IEEE Internet Things J 5(2):677–686
20. Deng R, Lu R, Lai C, Luan TH, Liang H (2016) Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption. IEEE Internet Things J 3(6):1171–1181
21. Fricker C, Guillemin F, Robert P, Thompson G (2016) Analysis of an offloading scheme for data centers in the framework of fog computing. ACM Trans Model Perform Eval Comput Syst (TOMPECS) 1(4):16
22. Adhikari M, Mukherjee M, Srirama SN (2019) DPTO: a deadline and priority-aware task offloading in fog computing framework leveraging multi-level feedback queueing. IEEE Internet of Things J, 1–1
23. Chen Y-C, Chang Y-C, Chen C-H, Lin Y-S, Chen J-L, Chang Y-Y (May 2017) Cloud-fog computing for information-centric Internetof-Things applications. In: Proceedings of international conference applied system innovation (ICASI). pp 637–640
24. Brogi A, Forti S, Guerrero C, Lera I (2019) Meet genetic algorithms in monte carlo: optimised placement of multi-service applications in the fog. In: International Conference on Edge Computing (EDGE). IEEE, pp 13–17
25. Jian C, Li M, Kuang X (2018) Edge cloud computing service composition based on modified bird swarm optimization in the internet of things. Cluster Comput 1–9
26. Bitam S, Zeadally S, Mellouk A (2018) Fog computing job scheduling optimization based on bees swarm. Enterp Inf Syst 12(4):373–397
27. Aryal RG, Altmann J (2018) Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization ai algorithm. In: 2018 third international conference on fog and mobile edge computing (FMEC). IEEE, pp 147–154
28. Wan J, Chen B, Wang S, Xia M, Li D, Liu C (2018) Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Trans. on Industrial Informatics 14(10):4548–4556
29. Ezhilarasie R, Reddy MS, Umamakeswari A (2019) A new hybrid adaptive ga-pso computation offloading algorithm for iot and cps context application. J Intell Fuzzy Syst (Preprint), pp 1–9
30. Naqvi SAA, Javaid N, Butt H, Kamal MB, Hamza A, Kashif M (2019) Metaheuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid. In: Barolli L, Kryvinska N, Enokido T, Takizawa M (eds) Advances in network-based information systems. Springer, Cham, pp 700–711
31. Binh HTT, Anh TT, Son DB, Duc PA, Nguyen BM (2018) An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment. In Proceedings of 9th International Symposium on Information and Communication Technology (SoICT). New York, pp 397–404. https://doi.org/10.1145/3287921.3287984
32. Canali C, Lancellotti R (2019) GASP: genetic algorithms for service placement in fog computing systems, algorithms 12(10):201. [Online]. Available: https://www.mdpi.com/1999-4893/12/10/201
33. Adhikari M, Srirama SN, Amgoth T (2019) Application offloading strategy for hierarchical fog environment through swarm optimization. IEEE Internet Things J 7(5):4317–4328
34. Hussein MK, Mousa MH (2020) Efficient task offloading for iot-based applications in fog computing using ant colony optimization. IEEE Access 8:37191–37201
35. Dai Lu, Wang B, Yang LT, Deng X, Yi L (2019) A nature-inspired node deployment strategy for connected confident information coverage in industrial Internet of Things. IEEE Internet Things J 6(6):9217–9225
36. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4. IEEE, pp. 1942–1948
37. Eusuff M, Lansey K, Pasha F (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optim 38(2):129–154

38. Sharif MU, Javaid N, Ali MJ, Gilani WA, Sadam A, Ashraf MH (2018) Optimized resource allocation in fog-cloud environment using insert select. In: International conference on network-based information systems. Springer, pp 611–623
39. Kaur P, Mehta S (2017) Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm. J Parallel Distrib Comput 101:41–50