QoS-aware Resource Scheduling using Whale Optimization Algorithm for Microservice Applications

Mohit Kumar[1], Jitendra Kumar Samriya[2], Kalka Dubey[3], and Sukhpal Singh Gill[4]
[1]Department of Information Technology, Dr B R Ambedkar NIT Jalandhar, India
[2]Department of Computer Science and Engineering, Graphic Era University, Dehradun, India
[3]Department of Computer Science and Engineering, RGIPT Amethi, India
[4]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

Abstract: Microservices is a structural approach, where multiple small set of services are composed and processed independently with lightweight communication mechanism. To accomplish the end-user demand in minimum delay and cost without violating the service level agreement (SLA) constraints, and overhead is a challenging issue in cloud computing. In addition, existing framework tries to deploy the microservice over the best computing resource for latency-sensitive applications, but long boot-time, and low resource utilization still remains a challenging task. To find the solution for aforementioned issues, we propose a Quality of Service (QoS) aware resource allocation model based on a Fine-tuned Sunflower Whale Optimization Algorithm (FSWOA) that find the best resources for microservice deployment and fulfill the objectives of users as well as service provider. The proposed technique deploys the container-based services over the physical machine based upon the capacity, to execute the micro services by utilizing the CPU and memory maximally. The proposed work aims is to distribute the workload in efficient manner and avoid the wastage of resources that leads to optimize the QoS parameters. The experimental results conducted in simulation environment demonstrates that proposed approach perform superior over baseline approaches and reduces the time, memory consumption, CPU consumption, and service cost up to 4.26%, 11.29%, 17.07% and 24.22 % compared to SFWAO, GA, PSO and ACO.

Keywords: Microservices, Resource allocation, Quality of Service, Fine-tuned Sunflower Whale Optimization Algorithm

1 INTRODUCTION

We live in a data-driven world where data pervades and run our lives. The amount of data generated is exponentially popping up from innumerable applications such as the Enterprise Resource Planning (ERPs), Customer Relationship Management (CRM), social networks, online streaming and significantly from IoT sensors. This increasing data size demands large data centers to store, process and retrieve it for analytics and further usage. Traditional system comprises of a huge datacenter which involves huge Return on Investment (ROI), which is not feasible for small enterprises. Moreover, datacenters comprise of massive number of components that enhances complexity of the system [1]. As a consequence of which the recent years have observed affluence of cloud-oriented techniques including Docker, Swarm and Spring Native which aggrandize the emergence of microservices architecture. It is an architectural pattern which bifurcates a complex application into small and independent processes which communicate via language agnostic APIs. In contrast to Service Oriented Architecture (SOA) which enables location and platform independent services to reside on/off premises of cloud services user, it decomposes the application into set of potential services which can be leveraged by other applications [2] [3]. The microservices possess the capability of reusability, which means that once created, the services can be easily incorporated in other application modules without building them from scratch [4].

The significant benefit of employing the resource allocation technique in deploying microservices in cloud computing is that it maximizes resource utilization while lowering operating costs [5]. Hardware virtualization, such as memory, network, storage, and CPU, contributes to the virtualization technology and versatility. Furthermore, in a cloud computing situation, the principal purpose of allocating resources is to efficiently enhance physical machines (PMs) and harmonize workload in operating physical machines in a dispersed way to minimize congestion and low-loaded or overburdened resource utilization. The practice of distributing resources to cloud apps across the internet in a systematic manner is referred to as resource allocation. When cloud resources fail to allocate resources to microservices in an on-demand way, the utilities shall be short-lived. Allowing cloud service providers to allocate resources independently at every component to handle such challenges. As a result, allocation of resources is emphasized as a component of the resource maintenance scheme, demonstrating that it is a significant aspect in allocating resources efficiently and effectively. It is utilized in the cloud framework to improve customer happiness while reducing processing time. Minimizing resource usage guarantees cloud service quality, satisfaction for the service provider, and increased throughput.

Microservices are considered a new software-based application for creating and deploying using the cloud computing model. It is a combination of smaller services that can be operated indivisibly by deploying in the cloud

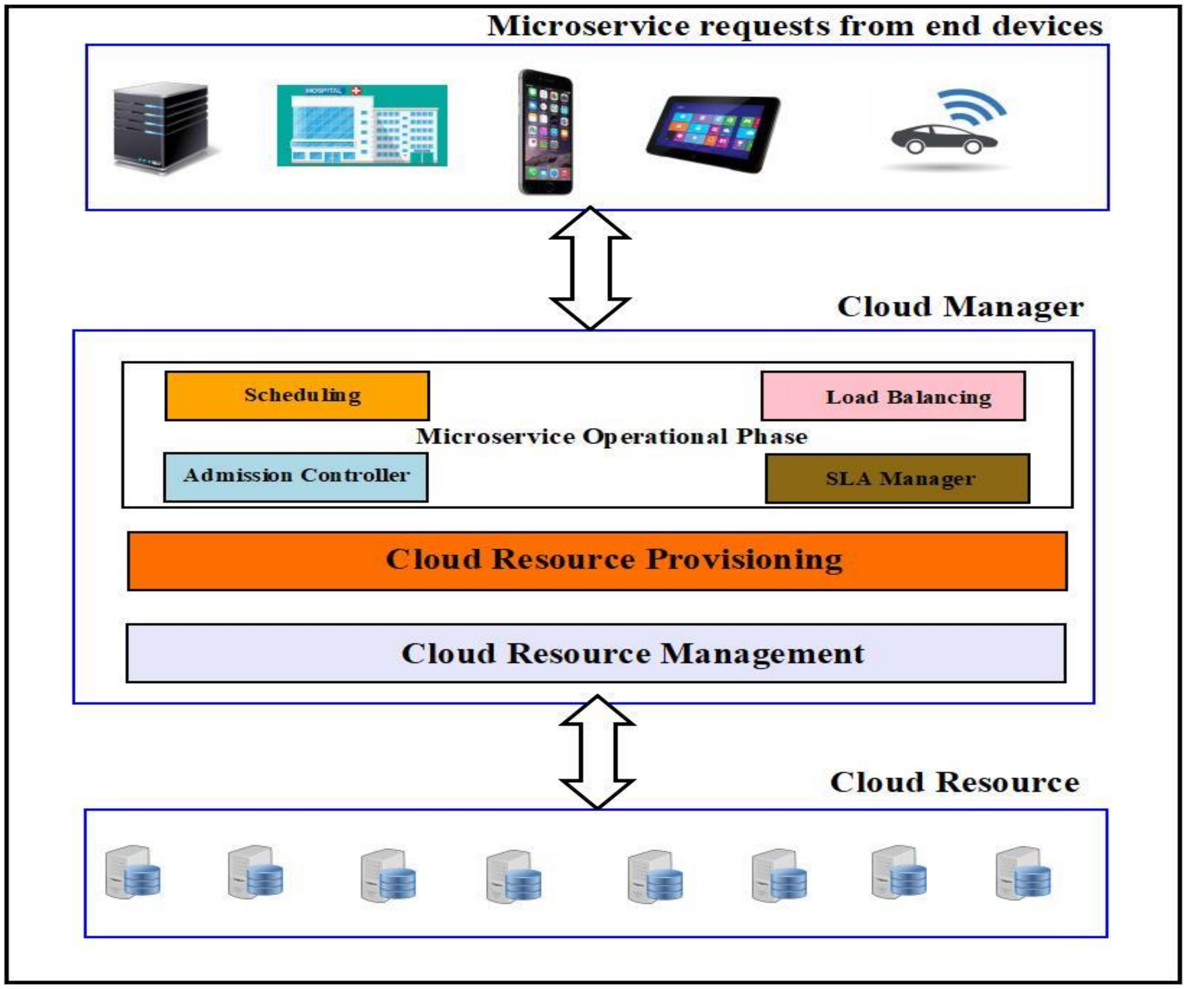
Microservice requests from end devices
HOSPITAL
Cloud Manager
Scheduling
Load Balancing
Microservice Operational Phase
Admission Controller
SLA Manager
Cloud Resource Provisioning
Cloud Resource Management
Cloud Resource

Figure 1: Architecture for deploying microservice-based applications on the Cloud Computing model

with different end devices. Figure 1 shows the architecture for deploying the microservices-based application on the cloud computing model. Various microservices are received from the various end-user devices for deployment in the cloud environment. The cloud manager is responsible for scheduling, load balancing, admission control, and SLA management of the microservices application. The admission controller manages the flow of the microservices application whereas the SLA manager is responsible for checking the various Quality of Standard (QoS) parameters like CPU, Memory, Cost, Energy, etc. for the microservices application. Resource provisioning and resource management is required for the scaling down and up of cloud resources according to the microservices application. Finally, the actual computing resources is allocated from the cloud resource pool. But, for effective microservices deployment on cloud computing platform, efficacious management and utilization of underlying resources is a must. For the same, Task scheduling and resource management allow providers to maximize income and resource utilization to their maximum capacity.

Motivation for the article: With the microservices gaining prominence due to their characteristics of reusability and flexibility, it is now being considered as optimal architectural solution for implementing complex applications such as OTT streaming. As far as its real implication is concerned, containers are utilized to encapsulate microservices, wherein each microservice is run in the form of multiple instances or copies. But at the same time, many challenges are encountered such as dynamism amongst underlying resources, dependencies between various instances, etc. [26-27] Although many previous studies have been conducted over containerized solutions to microservices, but as per the author's knowledge none has discussed the significance of resource utilization via optimal resource scheduling and hence are at precarious stage. Hence, in order to truly reap the benefits of this leading-edge technology, our work proposes a dynamic resource allocation model which holds potential to run multiple instances of microservices efficiently. The proposed work outlays improved QoS parameters along with minimizing SLAVs.

The **main contributions** of this research work are:

- The mathematical model is formulated with constraints to execute the microservices over the cloud resources.
- Designed a resource allocation model based on a Fine-tuned Sunflower Whale Optimization Algorithm for microservice application.
- The proposed approach improves the time, cost, container deployment time by utilizing the resource maximally and improve the system performance without violating the SLA.
- Finally, the performance of the proposed approach is tested and analyzed over significant QoS parameters and compared with well-known baseline algorithms.

Hence, this paper proposes a new QoS aware resource allocation system for microservices based cloud computing framework. We propose a Fine-tuned Sunflower Whale Optimization Algorithm (FSWOA) to optimize resource scheduling for smooth execution of microservices for real time applications which require to have impactful QoS parameters in terms of resource consumption (memory and CPU) and skewness [6].

The remaining section are organized as follows: Section 2 explains the literary works associated with the microservices application in the cloud environment. Section 3 discusses the problem statement and formulation in mathematical form, after that proposed framework is elaborated in section 3.2 with details discussion. whereas section 4 elaborates the proposed methodology. Section 5 analyzes and compares the behavior of the proposed algorithm and performance evaluation over the exiting state-of-art techniques. And finally, Section 6 concludes the overall idea of our proposed system, highlighting that our technique in resource optimization brings significant results and future research direction.

2 RELATED WORK

There are various lightweight scheduling techniques has been proposed for allocating the microservices over the cloud resources, here we will discuss few recent research works related to the cloud resource management. A heuristic technique has been proposed by C. Joseph and K. Chandrasekaran to deploy the microservice over the cloud resources in robust manner to enhance the throughput and response time [7]. The proposed approach performance is evaluated over google cloud platform for diverse applications and computational results proved that it performs superior over baseline approaches. The allocation of resources in a cloud environment has been addressed by In Subalakshmi and Jeyakarthic using a WOA-TRA [8]. The WOA is hybridized with the tumbling action that has a superior exploration capability, and generates a WOA-TRA model to optimize the objective functions like energy efficiency and other QoS parameters. The experimentation found that the WOA-TRA methodology provided the desired quality of service and improved energy efficiency over the comparison methods. QoS-aware resource provisioning (QBP) was depicted based on ACO strategy by Sharma et al. [9]. AHP uses k-means clustering to find tasks with similar QoS constraints, and afterwards AHP uses QoS requirements to order the tasks. ACO is used to search the optimal resource for the service. QBP could be developed in the future by detecting task dependencies.

An autonomic framework has been proposed by M. Kumar et al, for cloud resource provision and scheduling to offer the most suitable services to end users [10]. The proposed framework finds the trade-off solution for multi-objective problem by decision making capability and integrated with spider monkey algorithm to enhance the performance. The container-aware scheduling approach has been proposed by S. N Srirama et al, for microservice deployment over cloud resources to improve the service cost and cloud resource utilization with auto-scaling [11]. The best fit containers are used to deploy the microservice applications and bin packing approach is used to minimize the cloud physical resources. S. Taherizadeha and M. Grobelnika describes the mechanism about the cloud based microservice applications to enhance the time with reactive autoscaling [12]. The extensive experiments performed over mixed workload to test the scalable feature for computationally intensive microservice applications. X. He et al, presents self-adaptive microservice based framework for edge cloud environment to offer the services at network edge and proposed approach has the capability to monitor the services as well as resources, analyze the data and planning to process over the best suitable edge resource for latency sensitive microservice applications [13]. The proposed framework is open sourced and fruitful to offer the real time services using two-phase strategy and accomplished the demand of end users.

Table 1: Comparison of Related work along with Objective and Limitations

year	Objective addressed	Techniques	Limitation
2020 [7]	Microservices allocated to Cloud in an interaction aware policy using IntMA and IntRR	Interaction graph and (0/1) Quadratic Programming Problem	IntMA doesn't perform efficiently for smaller datasets; network congestion not considered
2020 [8]	Energy efficient resource allocation in Cloud	Whale optimization algorithm with tumbling effect (WOA-TRA)	Deadline not considered
2021 [10]	Schedule jobs in Cloud in most efficient resources within deadline and optimize energy, execution cost and time	Spider Monkey optimization algorithm	Requests are assumed to be non-preemptive
2021 [13]	Make the microservice capable adapt to changes in user needs	NSGA-II, WSGA, MOEA/D	Deadline among user needs isn't considered
2020 [14]	Resource allocation in Cloud for energy efficiency and performance	Multi objective optimization-based allocation policy (MOOA)	Static requests, deadline not considered
2020 [21]	Transmit packet coding	Cuckoo Search Optimization	Static data
2021 [23]	Enhance performance of inverter based microgrid; select parameters of proportional integral controller	Sunflower optimization algorithm	Simulated for small dataset
2022 [24]	Multi objective optimization of microservice applications	Pareto Optimal Front selected	Static microservice allocation
2022 [25]	Resource management of cloud microservice resources in real time	Linear programming model and quadratic optimization model implemented	Proposed models are reactive in nature

Authors have developed and tested a multi-objective optimization based-allocation policy (MOOA) in cloud computing including power usage and SLA concerns [14]. A multi-objective optimization-driven further generalized VM consolidation approach may be developed in future study. Mohit et. al, have depicted about the need of resource provisioning and represented the taxonomy of scheduling techniques including QoS parameter for better services in virtualization environment [15]. Further, authors have discussed about the real environment and simulation tool to implement the algorithm. In Zhou et al. (2018) [16], they offered a comprehensive system containing a resource-efficient computation offloading methodology for clients and a joint communication and computation (JCC) allocation method for network operators. To deal with system dynamics, we shall build effective online resource allocation methods. The prediction of upcoming workload along with resource allocation is a challenging issue in cloud computing. Authors have developed genetic algorithm (GA) based multi-objective technique to forecast the workload based upon historical data [17]. Further, authors have considered cpu and memory as parameters to predict the next time slot workload so that they can easily place the virtual machine (VM) over the physical machine (PM) for better services. Particle swarm optimization (PSO) based dynamic power-saving resource allocation technique has been proposed by the authors to cut down the power of VMs, PMs and air conditioner [18]. In addition, authors have predicted the resource utilization using least squares regression approach to eliminate the unnecessary migration of VMs. The computational results show that proposed approach improved the performance metrics in more efficient way as compared to baseline approaches. M. Kumar and S C Sharma have proposed PSO-COGENT approach that offer better exploration and exploitation compared to basic PSO and improved the significant parameters time, cost and energy consumption for the services [19]. Authors also depicted the solution of multi-objective problem using pareto optimal principle.

In Hasan et.al [20], the DLBS load balancing paradigm was presented for appropriate load balancing and resource optimization on public clouds. A load balancing model has been created utilizing EC2 Instances. Every server has a load balancer that detects the load and updates the management. Less loaded servers got greater responses whereas overloaded ones got none. In Hammood et.al [21], for effective and stable multicasting in VANET, they provide a TPC network coding method. In order to determine the effectiveness of network gadgets, network coding reduces packet transmission. The Cuckoo search method is used to select the safe relay nodes in this trust-based graph optimization case. M. Vrbaski et al., form multi-objective optimization model for microservices and improve the service cost, resource utilization considering deadline as constraint [24]. In addition, authors have presented pareto optimal solution for multi-objective optimization. L. Qassem et al have proposed a reactive resource manager based novel technique to allocate the best possible resources for cloud microservices and improve the QoS metric [25]. The proposed optimal technique results shows that proposed approach improve the deployment cost. Several other approaches also exist in cloud computing for resource provisioning, scheduling, microservices, but most of approaches failed to find the optimal solution and hard to achieved the defined objective. In this article, authors have addressed all the issues and try to find the optimal solution between significant objective using whale optimization approach by deploying the microservice applications over cloud resources.

3. Problem formulation and Proposed Methodology

Initially, this section discusses about objective of our research work and formulate the problem mathematically to optimize the QoS parameters. After that, we will discuss the proposed framework along with its components and working methodology for microservice application in cloud platform.

Table 2 Symbols notation and their description

Notation	Description
$PM_1, PM_2, PM_3 \ldots\ldots\ldots PM_m$	Heterogeneous m Physical Machines
$C_1, C_2\ C_3 \ldots\ldots C_k$	Set of containers
$MS_1, MS_2, MS_3 \ldots\ldots\ldots MS_n$	Request submitted in the form of microservices
η and $\boldsymbol{\rho}$	Number of core and processing power of each core
NB_{MM}	Denotes required main memory blocks
ET_{MS_i}	Execution time of microservice
SCO_{MS_i}	Service cost for the microservice
$DCO_{C_i}^{PM}$	Deployment cost of container over PM
$Workload_j^{max}(CPU, MM)$	Denotes the maximum utilization of cloud resources

γ	Constant with range [0, 1]
$x_{i,n}$	Binary variable with value either 0 or 1

3.1 Problem Statement and formulation

The aim of this section is to improve the significant QoS parameters that directly impact the performance of the system. The problem is formulated mathematically to optimize the objective functions that accomplished the demand of end user as well as service provider without violating the sla. Let us consider a cloud datacenter that contains m number of heterogeneous physical machines, denoted as PM = $\{PM_1, PM_2, PM_3 \ldots\ldots\ldots\ldots PM_m\}$, where k number of container can be deployed over each physical machine depends upon the availability of resources, represented as C=$\{C_1, C_2\ C_3 \ldots\ldots\ldots C_k\}$. Various types of services and library function can be installed over the PM to execute the microservices within defined time and cost. End users submitted n request in the form of microservices, denoted as MS= $\{MS_1, MS_2, MS_3 \ldots\ldots\ldots\ldots MS_n\}$, where a microservice can communicate with other one and work independently to process the application. Each microservice MS_i required the resources from the PM for the completion of services, and represented in the form of tuples as $\{R^{CPU}, R^{MM}, R^{BC}, R^{Bw}\}$, Where R^{CPU} is required CPU, R^{MM} is main memory required, R^{BC} is budget constraint and R^{Bw} is required bandwidth. The proposed approach searches the best container to deploy the microservice and fulfill the resource requirement. Resource capacity of container is defined in equation 1-2

$$R_i^{CPU} = \eta * \boldsymbol{\rho} \qquad (1)$$

Where η denotes the number of core and $\boldsymbol{\rho}$ is size of or processing power of each core.

$$R_i^{MM} = NB_{MM} * \text{sizeof}(NB_{MM}) \qquad (2)$$

Where NB_{MM} represents the required main memory blocks and sizeof (NB_{MM}) is each block size. The time required to execute each microservices over cloud resources is defined in equation 3

$$ET_{MS_i} = \frac{SMS_i}{R_i^{CPU}} \qquad (3)$$

Where ET_{MS_i} is the execution time of microservice, SMS_i is the size of microservice. The service cost for the microservice (SCO_{MS_i}) based upon CPU and memory usage is calculated by equation 4 & 5

$$SCO_{MS_i}^{CPU} = R_i^{CPU} * \frac{ET_{MS_i} * SCO_{CPU_{usage}}^{MSi}}{\mathcal{L}_1} \qquad (4)$$

$$SCO_{MS_i}^{MM} = R_i^{MM} * \frac{ET_{MS_i} * SCO_{MM_{usage}}^{MSi}}{\mathcal{L}_2} \qquad (5)$$

Where $SCO_{CPU_{usage}}^{MSi}$ denotes the container cost in per unit time interval ($\mathcal{L}_1$) based upon the utilization of CPU, and $SCO_{MM_{usage}}^{MSi}$ denotes the container cost based upon the utilization of main memory in per unit time interval ($\mathcal{L}_2$). Total service cost is represented by the equation 6

$$SCO_{Total} = SCO_{MS_i}^{CPU} + SCO_{MS_i}^{MM} \qquad (6)$$

After calculating the execution time of microservices, we find the deployment time of each container (DT_{C_i}) over physical machine for the services with the help of applications and its components (microservices) by equation 7. Suppose an application (A_{app}) consists of α microservices and it required k container for executing in a time period t, deployment cost of container over PM is represented by $DCO_{C_i}^{PM}$.

$$DCO_{C_i} = \frac{DT_{C_i} * DCO_{C_i}^{PM}}{\mathcal{L}_3} \qquad (7)$$

If DT_{C_i} =0, then total time taken to execute the applications is calculated by the equation 8

$$COST_{A_{app}} = \sum_{i=1}^{\alpha} SCO_i \ + \ \sum_{i=1}^{k} DCO_{C_i} \qquad (8)$$

After deploying the container over the PMs, need to monitor the workload continuously to avoid the condition of over and underutilization. Hence, workload cannot be allocated more than the capacity of physical machines resources especially CPU and memory. Current workload over the physical machine can be calculated using the equation 9

$$Workload_j(CPU, MM) = \gamma * \sum_{i=1}^{k} R_i^{CPU}(t) + (1-\gamma) * \sum_{i=1}^{k} R_i^{MM}(t) \qquad (9)$$

Where range of CPU and memory utilization are between 0 to 100 and γ is constant with range [0, 1]. The utilization of cloud resources can be found with the help of equation 9.

$$ResU_j = \frac{Workload_j(CPU,MM)}{Workload_j^{max}(CPU,MM)} * 100 \qquad (10)$$

Where $Workload_j^{max}(CPU, MM)$ represents the maximum utilization of cloud resources. Our objective is to minimize the time and cost ($COST_{A_{app}}$), while maximizing the utilization of cloud resources ($ResU_j$) without any sla violation like deadline, budget etc. Considering the equation no. 8 & 10 to fulfill the objectives,

$$\text{Min } COST_{A_{app}} \qquad (11)$$

$$\text{Max } ResU_j \qquad (12)$$

Subject to:

$$COST_{A_{app}} <= R_{A_{app}}^{BC} \qquad (13)$$

$$ResU_j <= 100 \qquad (14)$$

$$\sum_{i=1}^{k} ResU_i <= ResU_j \qquad (15)$$

$$Workload_j(CPU, MM) <= Workload_j^{max}(CPU, MM) \qquad (16)$$

$$x_{i,n} \in \{0, 1\}; \quad i \in \mathbf{C_k}; \qquad (17)$$

$$\sum_{i=1}^{k} x_{i,j}; \quad i \in C_k; \quad j \in PM_m \qquad (18)$$

The above-mentioned constraints describe that the cost of application for the microservices cannot be more than the defined budget, and utilization of cloud resource cannot be more than 100% for any resource. Workload over the physical machine cannot be beyond the limit as shown in equation 16. Separate container is allocated for each microservice and several containers can run over a single physical machine based upon the configuration as defined in constraint 17-18.

3.2 Proposed Framework for Microservice in Cloud Computing

Figure 2 shows the primary components of the proposed architecture for the microservices application in the cloud environment. The proposed framework is divided among the following components.

3.2.1 End User Devices: It provides a user interface for cloud users to send the microservices application through various end users' devices. The same user can request the different types of resources to execute microservices or different users can request for the same type of resources. Hence, demand of users is unpredictable.

3.2.2 Workload analyzer: It analyze smooth and Bursty workload demand of end users. Clustering and a ranking system are two modules of the workload analyzer. The clustering component divides end-user jobs into clusters based on their resource requirements. It is used to find tasks with comparable resource requirements. On the other hand, a ranking methodology provides a numerical weight to every task, known as rank. Its QoS requirements determine a task's rank. Tasks are ranked to determine their priority so that resources can be allocated appropriately.

3.2.3 Resource pool: Multiple firms provide cloud computing services, which might be diverse and geographically dispersed. The resource pool connects resource providers who are spread across the globe. It also aids in centralized resource supply and administration by storing resource details such as resource type, number of resources, geographical position, etc. When a user request for the resource, it is allocated from the nearest region based upon the requirement.

3.2.4 Monitoring: It collects consumption statistics for every VM from the appropriate PM's hypervisor. Efficient monitoring aids in making judgments on how to increase the system's behavior, energy efficiency, and other attributes. A monitoring agent operates in a PM's hypervisor and gathers QoS measurements regularly, which is kept in a QoS metric database for later analysis. This article employed low-level measurements like CPU, memory, and network bandwidth utilization to avoid the substantial cost.

3.2.5 Resource scheduling: A resource schedule is a list of activities and resources on a timeline. In other terms, it evaluates when a task should begin or end based on the work's length, QoS requirements, and available resources. At certain times, shared resources are available, and activities are scheduled during those times. There are several scheduling approaches are available in cloud based upon heuristic, meta-heuristic and hybrid approach.

3.2.6 SLA management: The SLA management module keeps track of each user's Service Level Objectives (SLOs) and completion history. A service level agreement (SLA) is a established agreement between the service providers and the users. It includes several service performances measures as well as the SLOs that go with them.

3.2.7 Resource Allocation: The suggested FSWOA, which is a merger of Fine-tuned Sunflower Optimization (FSFO) and Whale Optimization Algorithms (WOA), adopts both algorithms' parametric properties to improve the resources allocation method's efficiency. By evaluating the minimum fitness function, the suggested approach efficiently assigns the work to a Virtual machine. To ensure efficient resource allocation, the suggested FSWOA employs the humpback whales hunting technique and foraging behavior, as well as the unusual behavior of sunflowers.

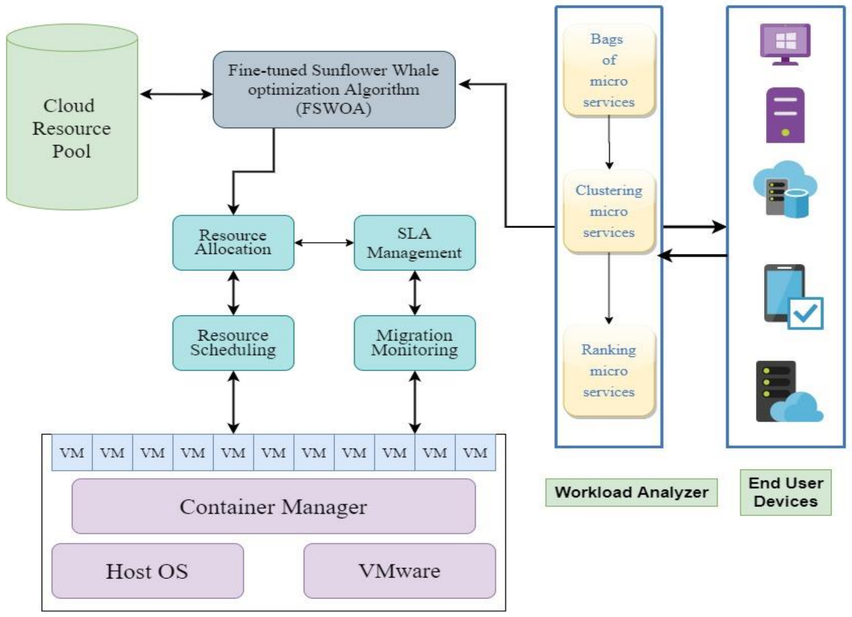

Figure 2 Proposed Framework for microservices in cloud environment

4. Fine-tuned Sunflower Optimization (FSFO):

SFO is an optimization-based approach to enhance the performance of system by improving the QoS parameters or finding the optimal solution of defined fitness/objective functions. It employs three different strategies to regenerate the population [22-23]. The first way, a new plant is developed with the help of two successive plants that are available in the population. The aim of this method's is to explore and utilize the search space in an efficient manner. The aim of next method is to create the new plants that migrate to the best plant. It aids SFO in maximizing the defined search space.

The last approach produces the plants randomly to explore search area and avoid SFO quickly narrowing to a local optimum. Considering the SFO process mentioned above, we have applied a Fine-tuned sunflower whale optimization (FSWOA) technique in this paper. A new strategy for creating a new plant by modifying the finest one is proposed. Defined fitness function is formed for the plants of new population using the above discussed three approaches, and try to fulfil the objective by identifying the finest plant. The aim is to create the new plant by mutation method before the population is regenerated in the next generation. If newly generated plant provides optimal value, then it is substitute with current plant; else, it will die if its quality is lower than the finest one. The update equation of the new plant is expressed in equation 19:

$$S_{new,i} = S_{best,i} + rand\ (0,1).\alpha.\delta(0,1); i = 1 \div p \qquad (19)$$

Here, $S_{new,i}$and $S_{best,i}$represent the regulation variable i of the new and best plants. P denotes problem dimension. α denotes a constant to identify the maximum variation restriction of the variable. $\delta(0,1)$ indicates a function, which have the range of value between 0 to 1. If the $\delta(0,1)$ is equivalent to 0, the $S_{new,i}$is equivalent to $S_{best,i}$ else the $S_{new,i}$will be fixed to new value. The value of $\delta(0,1)$ is identified as shown:

$$\delta(0,1) = \begin{cases} 1; if\ rand(0,1) < R_m \\ \quad 0; otherwise \end{cases} \qquad (20)$$

Wherein R_m is the mutation rate, which has been set to 0.2. It implies that around a quarter of the variables in the s_{best} are updated. The fitness function of the new plant is evaluated, and if it has a higher quality than the most delicate plant, it is employed to substitute the finest plant; else, it will die.

4.1 Proposed Fine-tuned Sunflower Whale Optimization Algorithm (FSWOA):

The suggested FSWOA executes the microservice allocation procedure over cloud resources including time, cost, utilization of resources as significant performance parameters.

Estimation of Fitness function:

The fitness function is calculated to find the perfect solution. As the perfect solution, the fitness value with the lowest number is chosen. Nevertheless, the fitness value is estimated employing the following formula:

$$f = \sum_{l=1}^{p} Q_m + \sum_{m=1}^{q}(C_m + (1 - B_m) + S_m) \qquad (21)$$

Here Q_m indicates the runtime of mth assignment, C_m denotes the cost of mth microservice, B_m represents the resource consumption of mth microservice for an application in cloud platform, $S_m$indicates the skewness. P represents the overall count of assignments. Here, the factors B_m and S_m are expressed as,

$$B_m = \frac{C_m^v \times M_m^v \times U_m^v}{C_m^t \times M_m^t \times U_m^t} \times \frac{X_u}{X_x} \qquad (22)$$

$$S_m = \left(\frac{B_m}{B} - 1\right)^2 \tag{23}$$

Here, C_m^v represents the CPU used by the mth microservice, M_m^v indicates the memory used by mth microservice. U_m^vdenotes the MIPS used by the mth microservice, C_m^t denotes the overall CPU present in the mth microservice. M_m^trepresents memory present in mth microservice, U_m^t indicates the MIPS present in mth microservice, and B denotes the mean resource consumption. X_uIndicates the consumption of time slots, and X_x indicates the highest total slot.

4.2 Working Principle of FSWOA:

The proposed optimization technique FSWOA is used to allocate the cloud resources to microservices as well as deploy the container over the physical machine for the execution of microservices in cloud. The proposed FSWOA is created by combining the characteristics of WOA and ISFO. The equation of WOA is updated with the help of ISFO to implement the allocation approach. The proposed approach is a meta-heuristic enhancement algorithm motivated by the bubble net's hunting technique. Whales are magnificent creatures that are the world's largest animals. Whales are thought to be predators since they never sleep to breathe from the surface of the ocean. Whales contain the same general cells in their brains as humans, which are known as spindle cells. Humans' social actions, feelings, and judgment are all controlled by spindle cells. Whales are the most intelligent creatures because they have twice as many spindle cells as an adult human. The fascinating aspect of whale behavior is their social behavior. Humpback whales are giant baleen whales, and they can exist in clusters or alone. The resource allocation to microservice in cloud model is based on the hunting technique of humpback whales. The humpback whale's hunting habit is known as the bubble net paradigm. The following are the algorithmic steps associated with the proposed FSWOA:

- Initialization of the population: Assume there is an m count of whale W, and the population is initiated as W_τ (τ = 1, 2,…,m). The coefficient vectors $\vec{Y}$ and$\vec{Z}$, present iteration j, and location vector W* are the factors stated in the suggested method.
- Calculate each search agent's fitness function. The fitness function is calculated to achieve the most optimum search agent solution described in equation (21).
- Encircling victim: Humpback whales locate and surround their victim. The present resolution is initially defined as the intended victim that is close to the ideal number. When the best search agent has been identified, the leftover search agents adjust their position based on the best agent's recommendation. As a result, the surrounding behavior of humpback whales is expressed by the equation below:

$$\vec{K} = \left|\vec{Z}.\vec{W}_j^* - \vec{W}_j\right| \tag{24}$$

$$\overline{\vec{W}_{j+1}} = \vec{W}_j^* - \vec{Y}.\vec{K} \tag{25}$$

$$\vec{W}_{j+1} = \vec{W}_j^* - \vec{Y}.\left|\vec{Z}.\vec{W}_j^* - \vec{W}_j\right| \tag{26}$$

$$\vec{W}_{j+1} = \vec{W}_j^* - \vec{Y}.\vec{Z}.\vec{W}_j^* - \vec{Y}.\vec{W}_j \tag{27}$$

Where j represents the present iteration, W^* indicates the location vector of the perfect agent, $\vec{W}$ represents the location vector, ‖ represents the absolute number, and $\vec{Y}$ and$\vec{Z}$ represents the coefficient vectors. The above formula indicates the updated formula of humpback whales depending on the behavior of surrounding victims. The updating formula for fine-tuned sunflower direction towards the sun is expressed as,

$$\vec{M}_{j+1} = \vec{M}_j + d_j^*\vec{S}_j \tag{28}$$

Where M_{j+1} represents the updated location of sunflowers, d_j denotes the step of sunflowers towards the sun's direction. The proposed algorithm (Algorithm 1) for the microservices is given below:

Algorithm 1: Proposed Algorithm for Microservices based Applications

Input: Physical machines PM = $\{PM_1, PM_2, PM_3 \ldots\ldots\ldots PM_m\}$, Containers C = $\{C_1, C_2\ C_3 \ldots\ldots\ldots C_k\}$, and microservices MS= $\{MS_1, MS_2, MS_3 \ldots\ldots\ldots\ldots MS_n\}$
Output: Optimal value of $COST_{A_{app}}$ and Max $ResU_j$

```
1.  Initialize the population $W_\tau$ ( $\tau$ = 1, 2,…,m), iteration t=0 to $t_{max}$
2.  Create the applications as well as microservices
3.  Create the datacenter and physical machine to process the microservices
4.  Defined constraints $R^{BC}$, $Workload_j^{max}(CPU, MM)$, $ResU_j$<=100, and $x_{i,j}$
5.  Identify the best agent
6.  While t< $t_{max}$ do
7.  Compute the optimal value of $COST_{A_{app}}$, $ResU_j$ with $COST_{A_{app}}$<= $R_{A_{app}}^{BC}$
8.  Calculate the $Workload_j$(CPU, MM) with $Workload_j(CPU, MM)$<= $Workload_j^{max}(CPU, MM)$
9.  Map available PM resources with required resources by Containers $C_k$
10. If (j<< $t_{max}$ ) then
11. Update Z and Y
12. Else
13.  Initialization the random location of sunflowers
14. Find the best solution
15. Adjust orientation towards sun
16. Worst m (%) solutions are removed
17. Update the position
18. Check the fitness value
19. If new fitness value is better then
20. Accept the solution
21. t=t+1
22. Otherwise continue up to $t_{max}$
23. End while
24. Return $COST_{A_{app}}$
```

The proposed algorithm, initially defined the control parameters for the input and created the population, cloud datacenter, microservices, and container to execute the services. Our aim is to find the optimal value of $COST_{A_{app}}$ by utilizing the cloud resources maximally. The constraints are defined at the initial stage to get the best solution because utilization of resources and budget should not be beyond the defined limits. After that identify the agent and repeat the loop until find the optimal value of defined objective functions. Start to compute the objective function by deploying the containers over the physical resources in less amount of time and requirement of resources also. Firstly, cloud resources are mapped with requirement of container for the microservices. After mapping the container with cloud resources, microservices are allocated for the execution and compute the workload as well as time and service cost. The proposed framework continuously monitors the microservices assigned to containers and take the necessary step, if there is any possibility of over or underutilization. The proposed algorithm checked the new solution with the current solution, if it is improved, the accept it otherwise process is repeated until the iteration is completed. Finally, algorithm return the optimal value of objective function and improve the significant QoS parameters. The flowchart of the proposed approach is shown in figure 2.

5 Performance Evaluation and Result Analysis

In this section, we check the validation and performance evaluation of the proposed FSWOA techniques for the microservices application in the cloud environment. Furthermore, Concerning the evaluation metrics, we elaborate on the results and discussion obtained on various QoS parameters. Authors have conducted simulations on the

CloudSim, a toolkit among the most widely used simulators. The proposed method is tested in a variety of situations, using Intel Core i7 processors and 16GB of RAM. We consider 30 VM for deploying the container, and the maximum capacity of each VM is in the range of 50 to 100 MIPS. The microservices are arranged in a batch of 50 to 300 in sequential order. We have taken four different datasets for evaluating the performance at the parameters: processing time, memory utilization, CPU utilization, and total cost required to process the microservices. The proposed technique FSWOA performance is against conventional approaches: SFWAO, GA, PSO, and ACO. The comparative result analysis of the proposed algorithm with the existing ones is discussed in the following sub-sections.

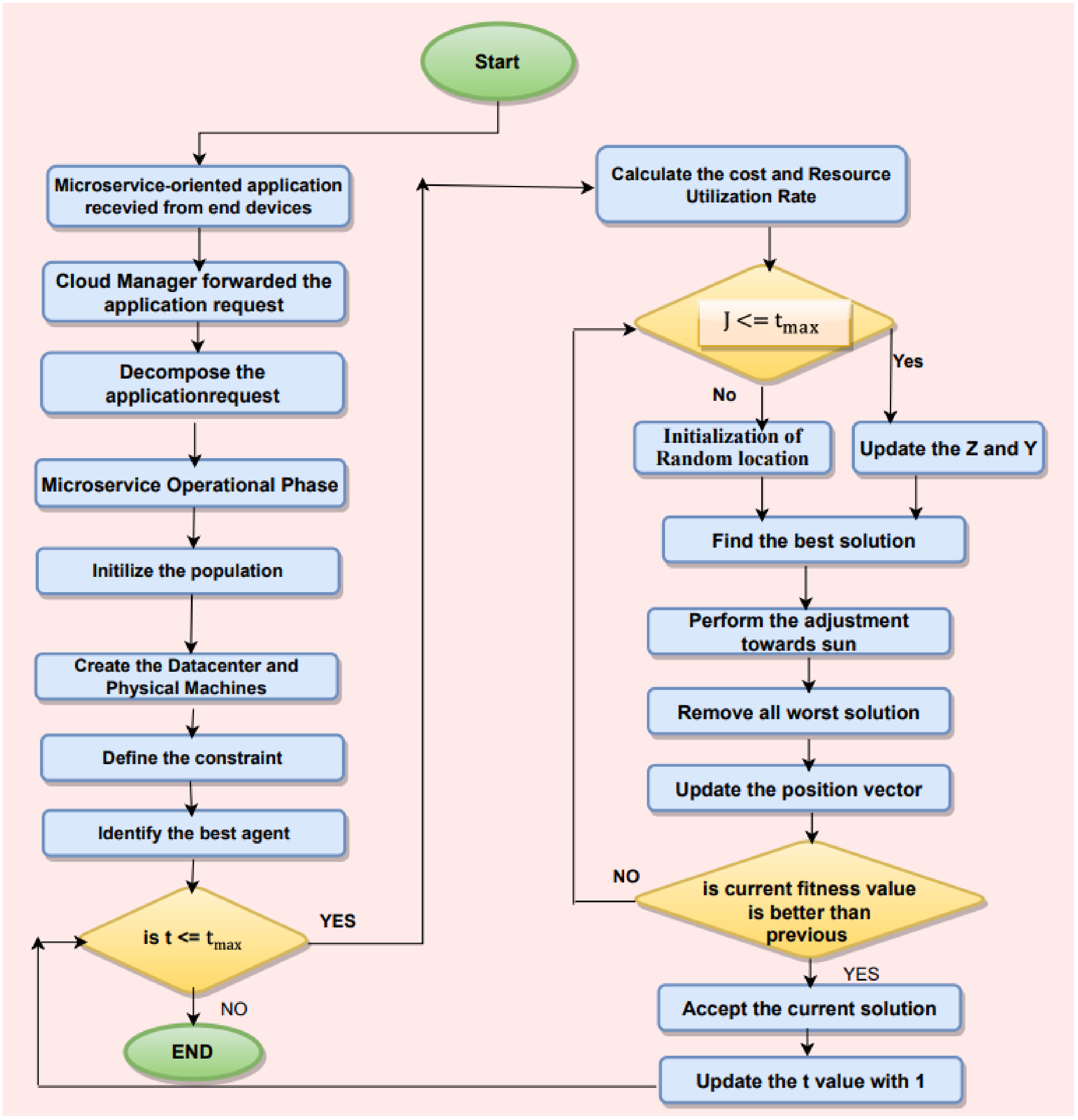
Start
Microservice-oriented application recevied from end devices
Cloud Manager forwarded the application request
Decompose the applicationrequest
Microservice Operational Phase
Initilize the population
Create the Datacenter and Physical Machines
Define the constraint
Identify the best agent
is t <= t_max
YES
NO
END
Calculate the cost and Resource Utilization Rate
J <= t_max
Yes
No
Initialization of Random location
Update the Z and Y
Find the best solution
Perform the adjustment towards sun
Remove all worst solution
Update the position vector
is current fitness value is better than previous
NO
YES
Accept the current solution
Update the t value with 1

Fig. 3 flowchart of proposed methodology

5.1 Processing Time:

The processing time for microservices totally depends on the allocated resources and the deployment time of the container of that microservice. Therefore, the selection of the resource and container affects the processing time parameters. The proposed techniques allocated the resources and container in a manner that minimizes the processing time parameters for microservices. Figure 4 shows the comparative analysis of processing time against the microservices for Datasets 1,2,3, and 4. The result proves that the proposed techniques reduce the processing time up to 4.26%, 11.29%, 17.07% and 24.22 % compared to SFWAO, GA, PSO and ACO to serve the microservice in the range of [50-300]. The number of microservices is represented by the x axis and processing time in seconds is represented by the y axis. More microservices required a greater number of containers, that are deploy over the physical machine, but deployment and execution time is increases in the same order as the microservices is increase. Processing time computational results along with variation and statistics are shown in Table 3.

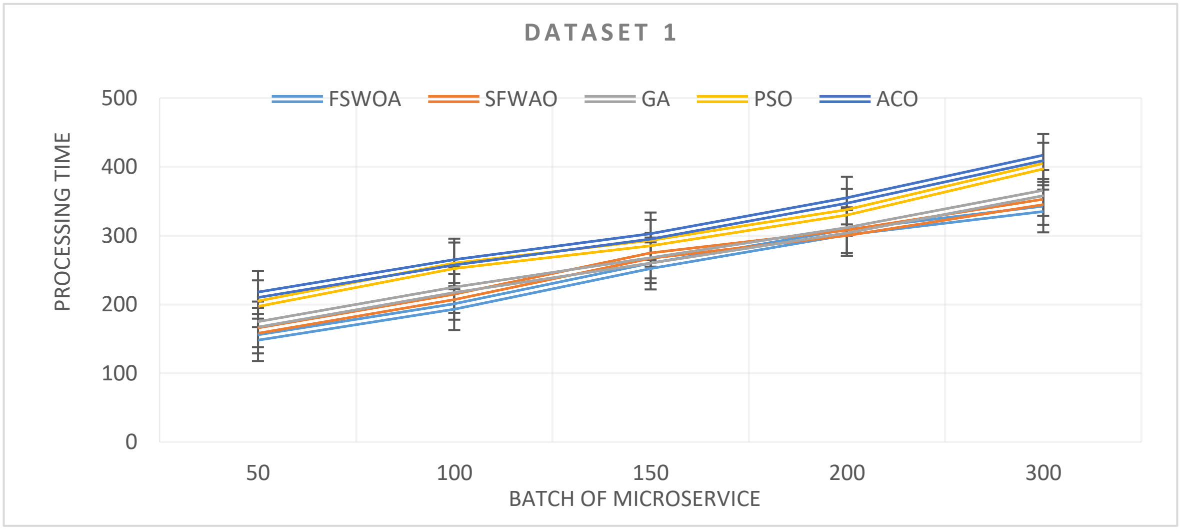

Figure 4(a): Dataset 1

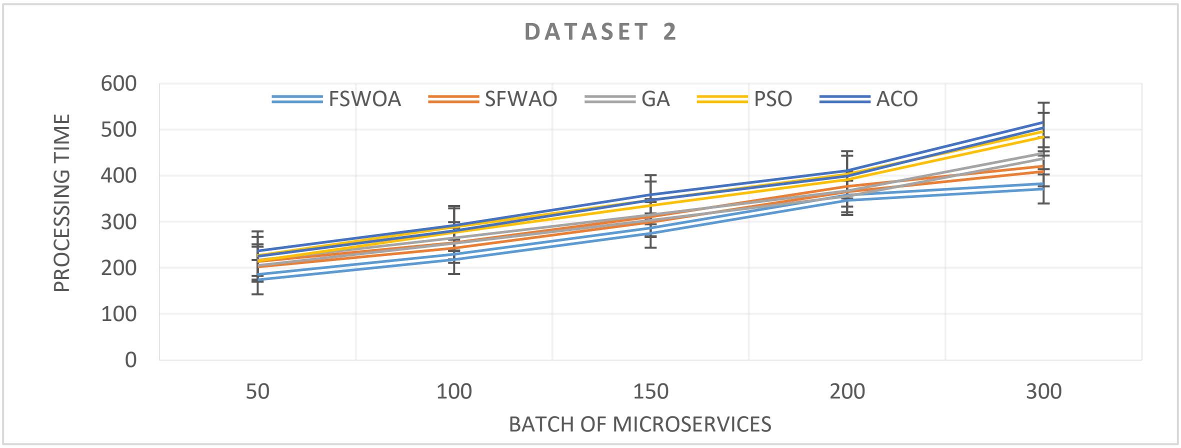

Figure 4(b): Dataset 2

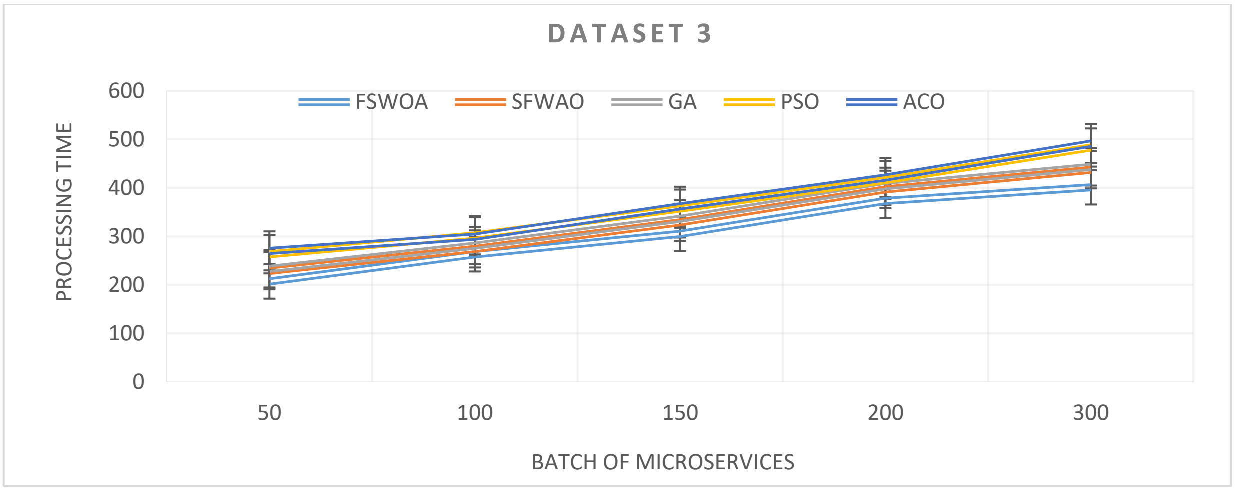

Figure 4(c): Dataset 3

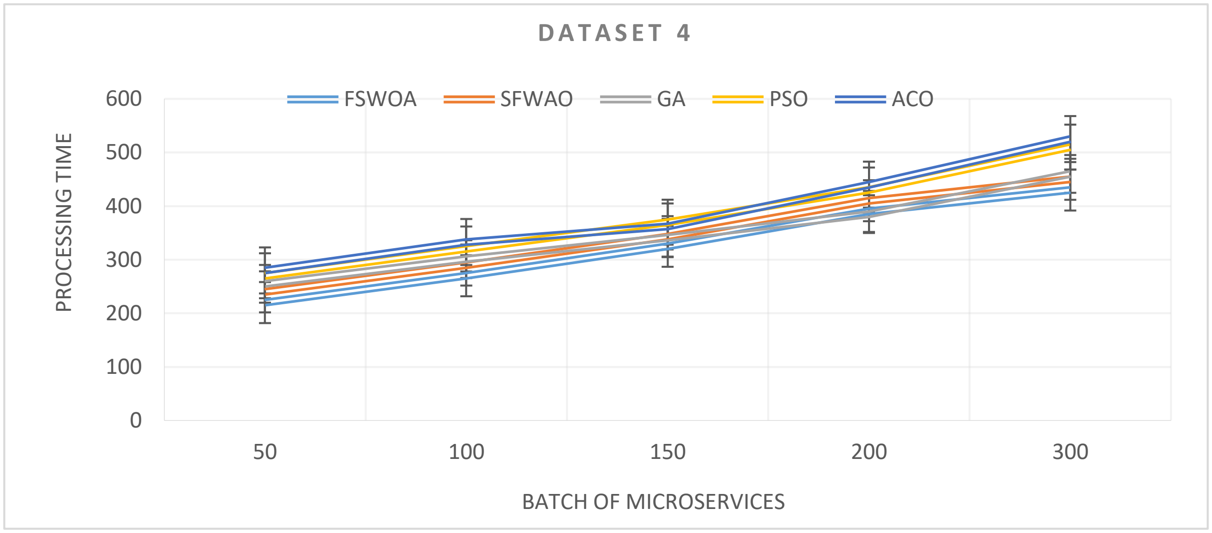

Figure 4(d): Dataset 4

Figure 4: Comparative analysis of processing time against the microservices for the Dataset 1,2,3, and 4

Table 3 Processing Time results comparison for dataset 1,2,3 and 4

Data Set	Statistics	FSWAO	SFWAO	GA	PSO	ACO
Data Set 1	Best	151.96	154.45	167.27	183.25	200.53
	Average	267.67	279.57	301.79	314.95	409.24
	Worst	350.91	358.91	391.45	408.31	410.76
Data Set 2	Best	170.11	176.82	181.67	190.24	191.65
	Average	271.25	273.5	299.12	301.62	300.28
	Worst	320.17	325.54	390.14	405.39	408.97
Data Set 3	Best	201.35	205.3	203.89	217.92	221.21
	Average	272.49	274.83	291.07	300.42	303.53
	Worst	351.27	359.37	394.28	480.79	495.82
	Best	215.67	221.57	235.72	267.58	283.17

Data Set 4	Average	275.61	276.81	293.47	301.7	304.93
	Worst	422.07	436.87	453.21	521.97	532.79

5.2 Memory consumption:

It is defined as amount of memory required to complete the microservice for the application running over the cloud resources. Authors have defined the constraint for the utilization of memory in problem formulation part i.e., memory cannot be used beyond the limit. The proposed framework monitors the percentage of memory utilization continuously, as it goes beyond the limit, it informs to the scheduler for balancing the workload among the resources. Scheduler and resource allocator are responsible for distributed the fair workload and avoid the condition of over and underutilization. Figure 5 portrays the comparative examination of memory utilization concerning the number of iterations. It is evident from the graph that the proposed FSWOA method reduce the memory consumption up to 1.24%, 2.22%, 3.52% and 5.08% compared to SFWAO, GA, PSO and ACO traditional methods. The computational results of memory consumption at diverse synthetic dataset along with variation and statistics are shown in Table 4.

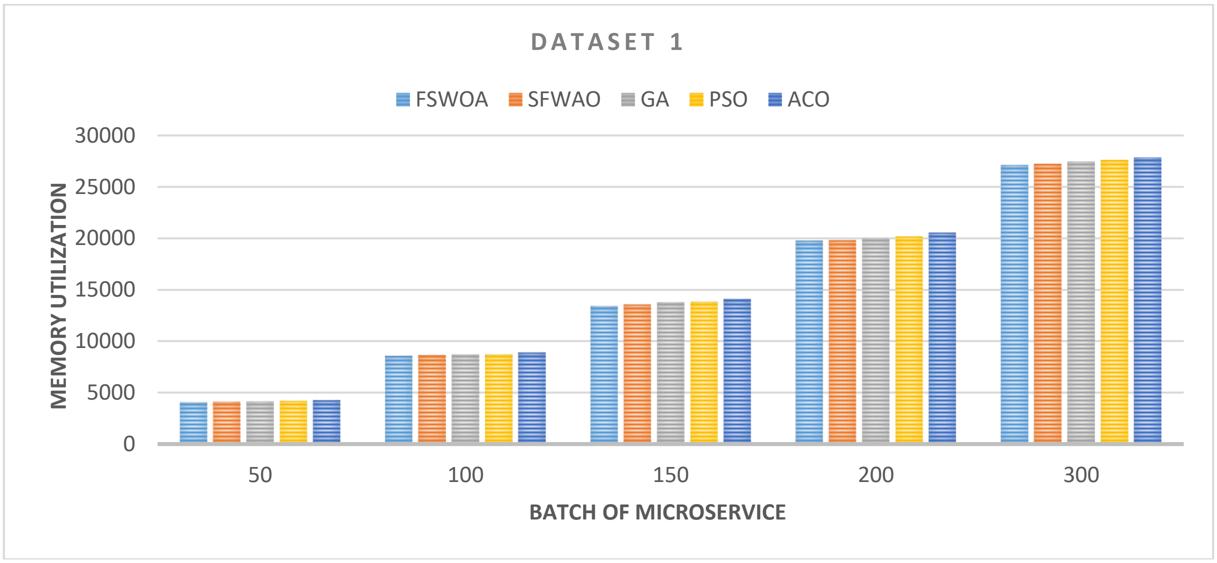

Figure 5(a): Dataset 1

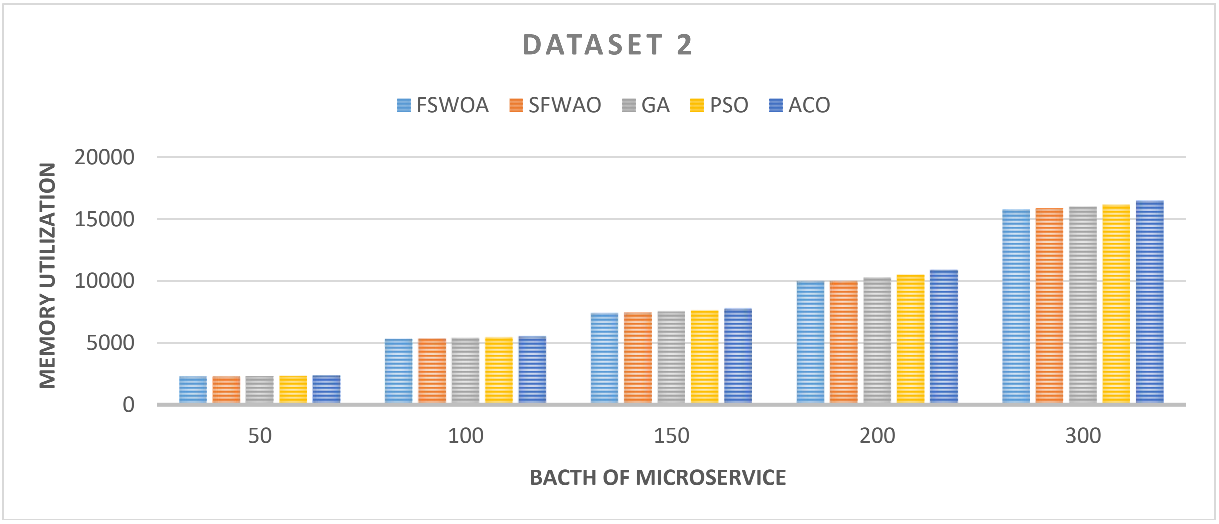

Figure 5(b): Dataset 2

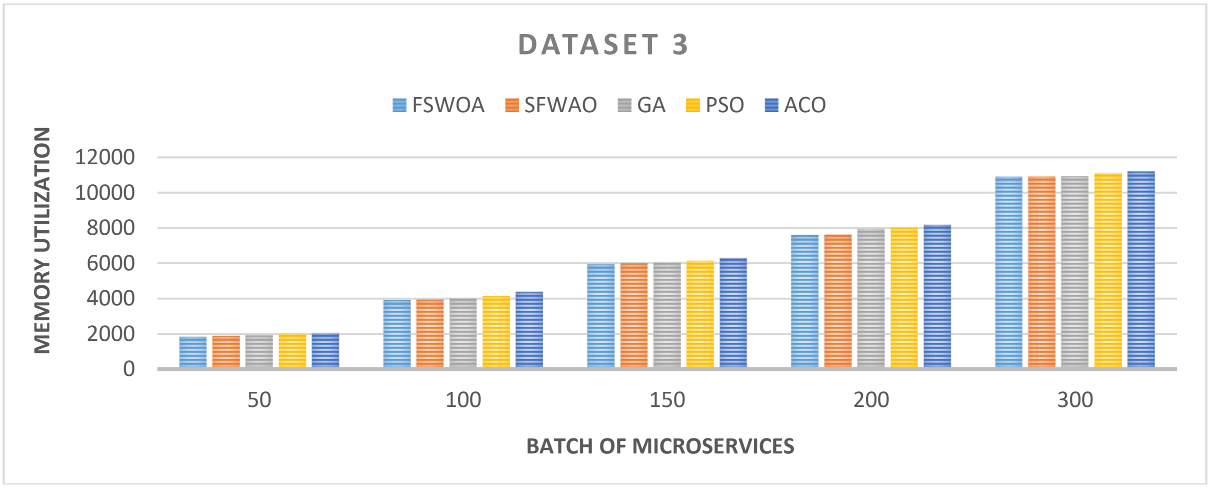

Figure 5(c): Dataset 3

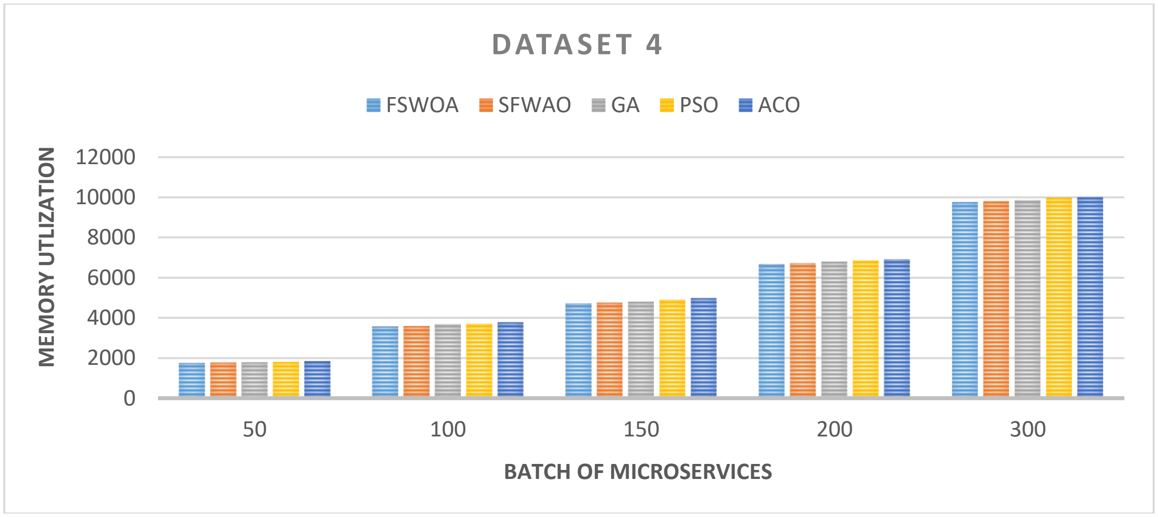

Figure 5(d): Dataset 4

Figure 5: Comparative analysis of memory consumption against the microservice for the Dataset 1,2,3, and 4

Table 4 Memory utilization results comparison for dataset 1,2,3 and 4

Data Set	**Statistics**	**FSWAO**	**SFWAO**	**GA**	**PSO**	**ACO**
Data Set 1	Best	4047.5176	4098.9826	4139.7681	4195.7195	4264.1722
	Average	14587.50508	14670.11106	14810.57192	14911.58438	15143.78296
	Worst	27119.9716	27231.2276	27456.817	27614.4472	27875.3344
	Best	2265.3569	2275.0173	2291.7129	2317.9342	2362.587

Data Set 2	Average	8145.5161	8190.3735	8293.06762	8389.6138	8596.3678
	Worst	15790.2913	15876.2916	15991.7625	16129.3018	16468.633
Data Set 3	Best	1825.9172	1879.4519	1910.3791	1981.6192	2036.134
	Average	6033.234092	6068.02682	6163.9349	6270.3254	6414.265
	Worst	10891.5195	10910.8173	10924.8192	11098.6183	11209.299
Data Set 4	Best	1755.8713	1779.81	1789.5391	1801.5289	1846.885
	Average	5299.53434	5329.88156	5387.50722	5450.00314	5503.2212
	Worst	9755.8821	9800.6371	9863.7347	9981.7259	10001.216

5.3 CPU consumption:

CPU is most significant resource that directly impact at the system performance in the form of energy, cost and time. The utilization of CPU should be maximum to execute the microservices in less amount of time and cost, but not beyond the defined limit. The mathematical model for the CPU utilization is defined in problem formulation section. Authors have taken four different datasets to evaluate the performance of the proposed model, initially 50 microservices are considered to test the performance, then increase linearly 50 to 100, and up to 300. Figure 6 shows the comparative examination of CPU consumption for microservices with different approaches including proposed technique also. The computational results proved that proposed approach utilized the CPU in superior way and improve up to 3.16%, 10.43%, 14.83% and 20.54% compared to SFWAO, GA, PSO and ACO state of art methods. The simulation-based results of cpu consumption at diverse synthetic dataset along with variation and statistics are shown in Table 5.

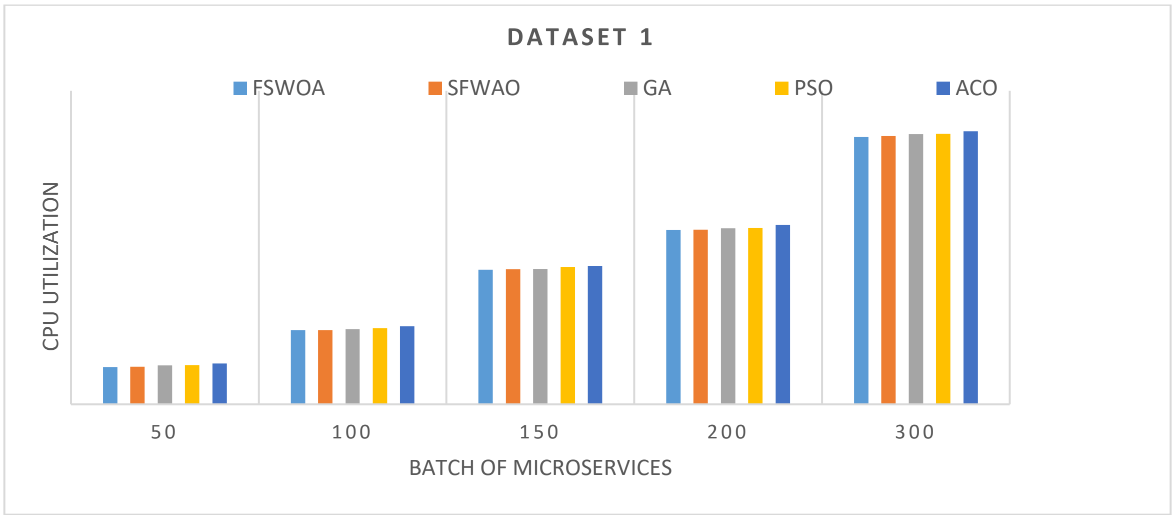

Figure 6(a): Dataset 1

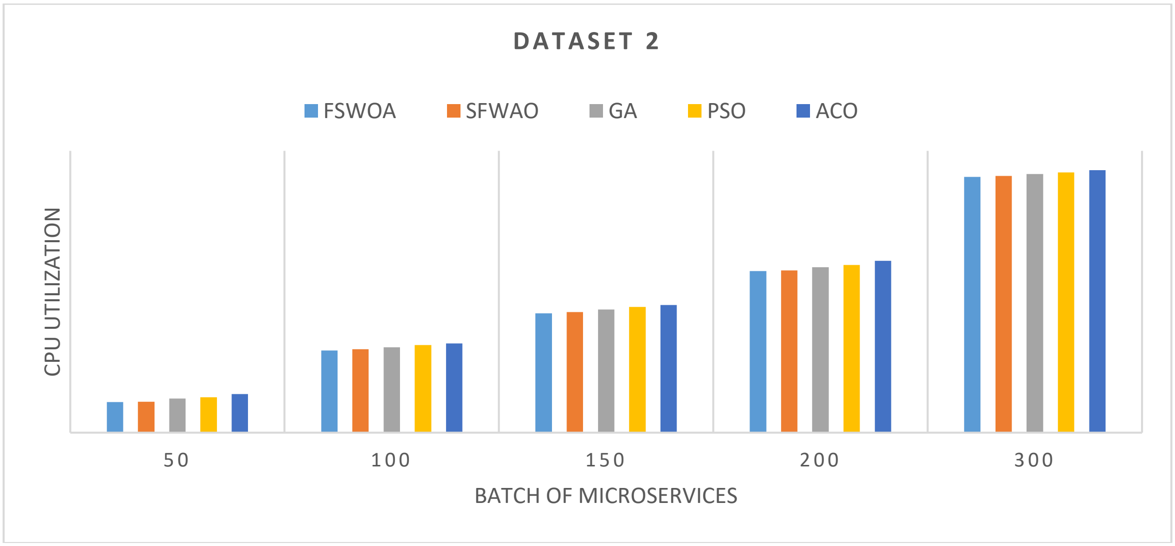

Figure 6(b): Dataset 2

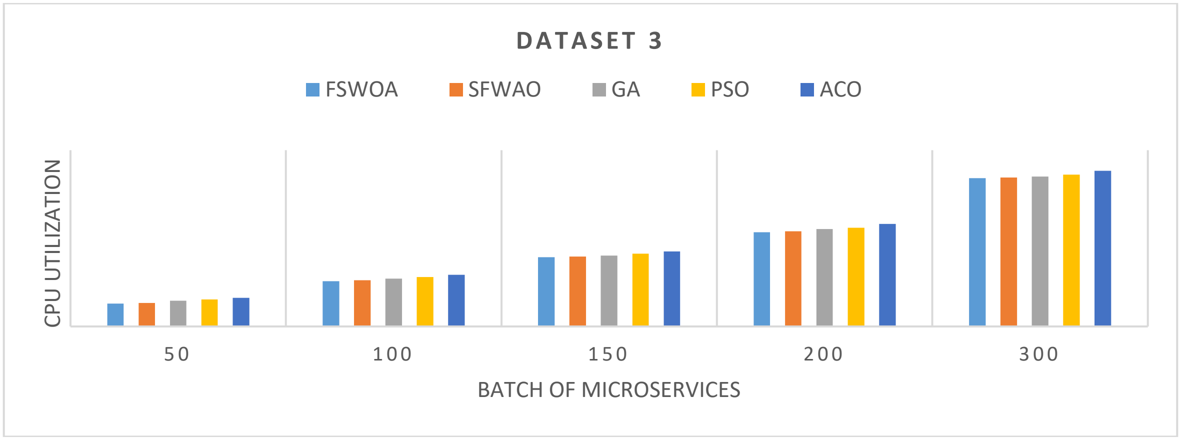

Figure 6(c): Dataset 3

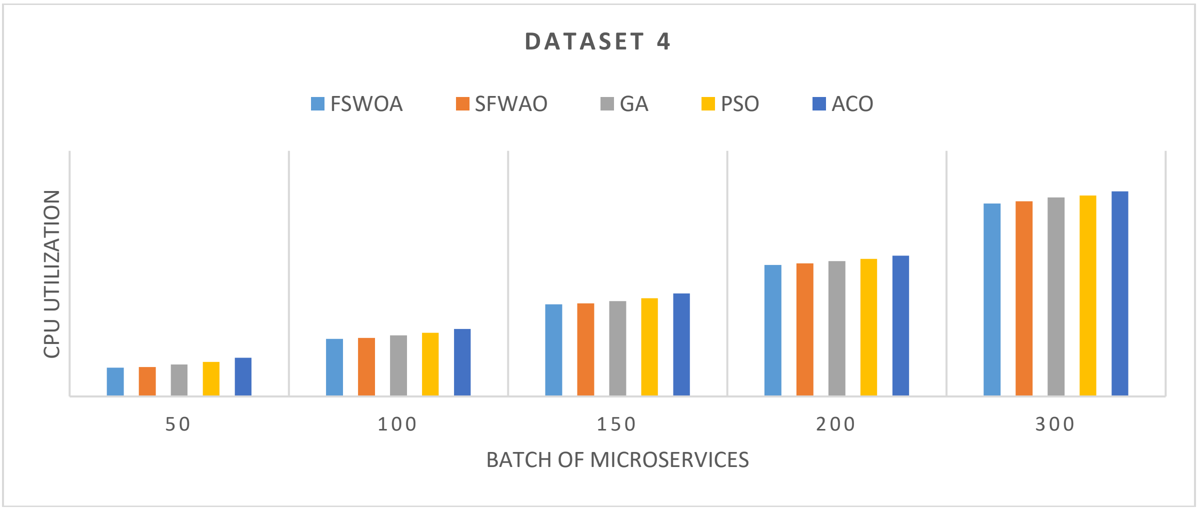

Figure 6(d): Dataset 4

Figure 6:Comparative Analysis of total cost against the microservice for the Datasets 1,2,3, and 4

Table 5 CPU Utilization results comparison for dataset 1,2,3 and 4

Data Set	Statistics	FSWAO	SFWAO	GA	PSO	ACO
Data Set 1	Best	1901.7265	1921.8162	1982.6785	2001.6519	2093.82814
	Average	7019.75716	7041.45078	7102.141144	7142.82448	7249.562972
	Worst	13649.2871	13688.6142	13794.7614	13812.6519	13930.31337
Data Set 2	Best	870.3813	873.8712	965.2396	1006.7611	1095.5856
	Average	3687.75436	3712.85882	3783.43092	3842.94152	3917.294242
	Worst	7260.8341	7285.7125	7341.9123	7391.6193	7455.79995
Data Set 3	Best	781.5194	804.81123	872.9124	917.6223	976.9809
	Average	2586.52204	2614.842106	2667.0511	2721.4452	2814.581592
	Worst	5048.6293	5072.8172	5103.7772	5162.9622	5298.462497
Data Set 4	Best	582.6114	601.6791	653.9223	703.7441	787.62886
	Average	2046.88024	2075.2371	2129.28038	2178.70522	2260.505031
	Worst	3929.0126	3974.8371	4052.9261	4092.0183	4174.10447

Total Service Cost:

The service cost of applications or microservices is depends upon the execution time of microservices and deployment time of containers. To reduce the service cost, authors have proposed a framework that allocate and execute the microservice over the cloud resources in minimum time. In addition, the proposed approach also reduces the deployment time of containers that directly affect the service cost as already discussed in problem formulation part. The cost is a significant parameter that should not be beyond the defined user budget at the time of sla. Authors have considered four diverse datasets to evaluate and test the performance of proposed model. The proposed strategy finds the best resources for the container that will execute the microservices by utilizing the computing resources maximally. The simulation based computational results shown in figure 7 show that proposed strategy achieved the less cost to process the microservices based applications over to other baseline techniques. The computational results of service cost at diverse synthetic dataset along with variation and statistics are shown in Table 6. The figure 7 and table 6 based simulation results proved that proposed framework reduce the cost up to 2.94%, 6.45%, 12.84% and 17.73% compared to SFWAO, GA, PSO and ACO state of art methods.

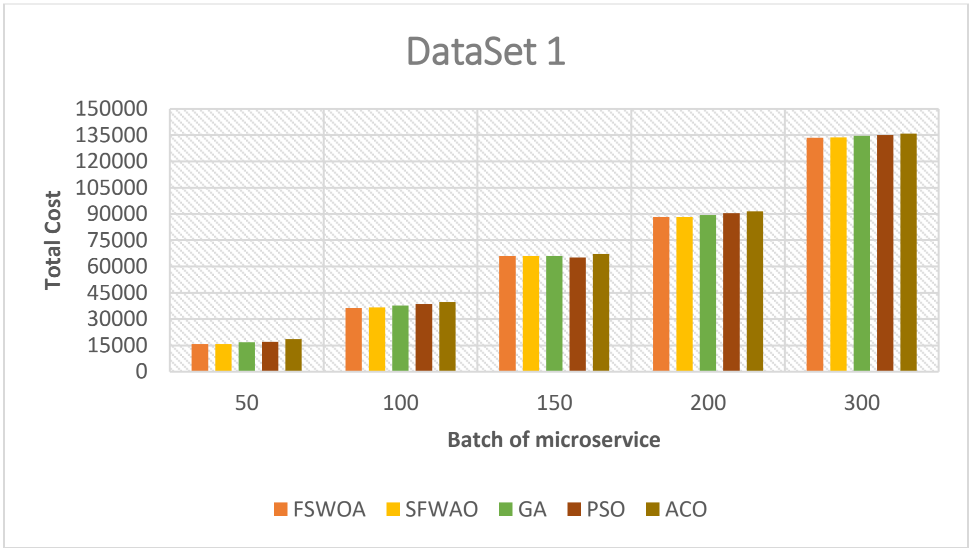

Figure 7(a): Dataset 1

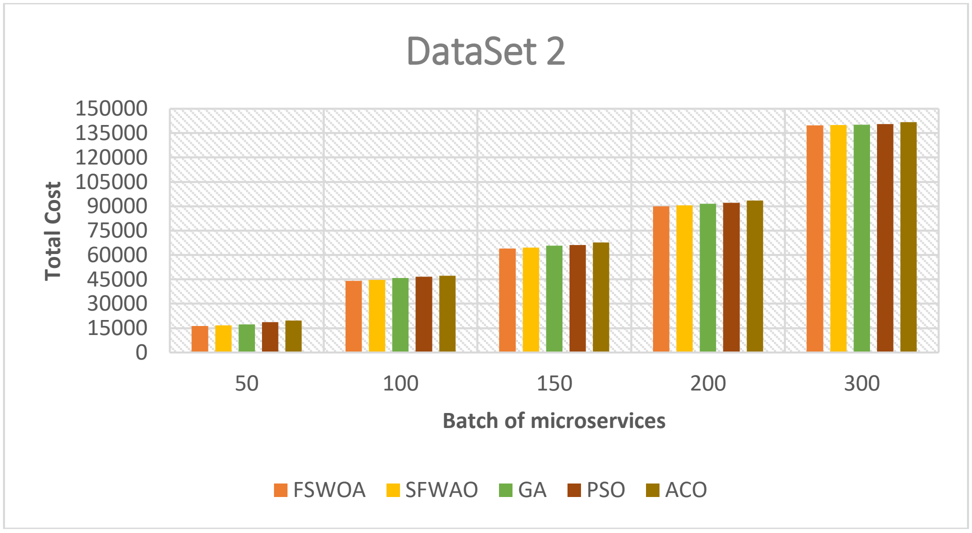

Figure 7(b): Dataset 2

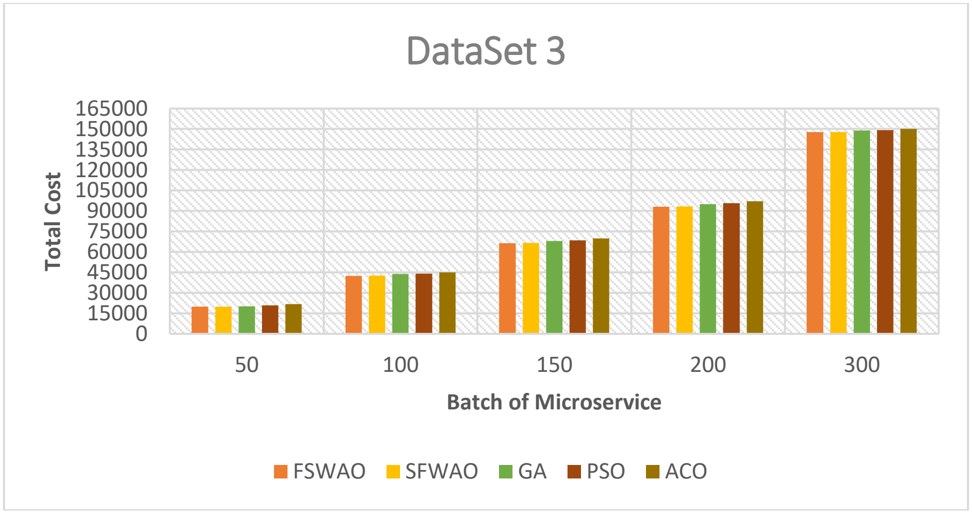

Figure 7(c): Dataset 3

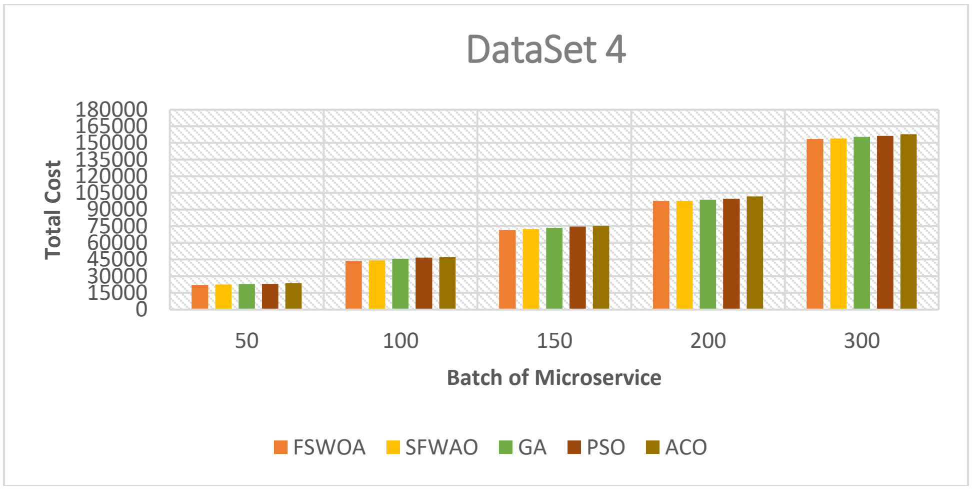

Figure 7(d): Dataset 4

Figure 7: Comparative Analysis of total cost against the microservice for the Dataset 1,2,3, and 4

Table 6 Cost results comparison for dataset 1,2,3 and 4

Data Set	**Statistics**	**FSWAO**	**SFWAO**	**GA**	**PSO**	**ACO**
Data Set 1	Best	15735.9128	15778.0912	16821.091	17027.9816	18495.2906
	Average	68050.09632	68082.75362	68951.03044	69300.13392	70622.17107
	Worst	133698.6518	133734.9637	134782.6693	135164.9263	135994.3507
Data Set 2	Best	16231.8267	16723.971	17348.9162	18623.9846	19731.4627
	Average	70802.25548	71280.30474	72093.73136	72825.35168	74006.51217
	Worst	139836.8724	139932.8712	140117.9262	140628.9367	141804.6228
Data Set 3	Best	19896.7342	19956.8267	20167.9352	20893.8267	21730.7024
	Average	73849.35044	74041.19292	75145.78408	75712.07008	76762.19717
	Worst	147738.9371	147823.8472	148923.8492	149256.9373	150168.4424
Data Set 4	Best	22381.7365	22502.9716	22876.8271	23109.5618	23747.9143
	Average	77967.96014	78273.66588	79355.70914	80235.23648	81239.07068

	Worst	153673.8362	153989.9374	155468.8362	156457.9267	157933.2143

6 Conclusions and Future work

Cloud computing is offering larger number of services through virtualization technology to run multiple applications over single physical machines. An application can be divided in the form of components and each component is called microservices, run independently and required light weighted resources to complete the service, but allocation of efficient resource, utilization of cloud resources, communication overhead between virtual machines, and service cost becomes a challenging issue. To address the mentioned challenges, A QoS-aware resources allocation framework is proposed by the authors for the microservices using Fine-tuned Sunflower Whale Optimization Algorithm (FSWOA). The proposed approach focused at the significant QoS parameters during the allocation of resources for the microservices and deployment of container over the physical resources. The proposed approach reduces the container deployment time and services time, and improve the utilization the CPU and memory maximally that provides the optimal service cost. Further, the proposed approach also monitors the resource and take the necessary action, if resource utilization reaches beyond the defined limits. The proposed approach optimize the QoS parameters based upon the defined function in problem formulation part using FSWOA. The computational results revealed through simulation outcomes that FSWOA outperforms Genetic Algorithm (GA), Particle Swarm Optimization (PSO), ACO and Sunflower Whale Optimization Algorithm (SFWOA) in terms of performance metrics like service cost, memory consumption, CPU consumption, and other parameters. The limitation of the proposed framework is that authors did not test the performance of the proposed approach in real cloud environment and did not consider all the network parameters constraints [28]. Latency is an issue with cloud computing; hence we will develop an intelligent algorithm that will offload the latency sensitive microservices over the fog/edge node without any delay and normal services over the cloud platform in the future [29]. Artificial Intelligence technique will play the vital role to decide the offloading platform at the runtime [30].

References

[1] L. De Lauretis, From monolithic architecture to microservices architecture, in: 2019 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW, IEEE, 2019, pp. 93–96.

[2] Gawali, M.B. and Shinde, S.K., 2018. Task scheduling and resource allocation in cloud computing using a heuristic approach. Journal of Cloud Computing, 7(1), pp.1-16.

[3] Waseem, Muhammad, et al. "Design, monitoring, and testing of microservices systems: The practitioners' perspective." Journal of Systems and Software 182 (2021): 111061.

[4] Ciuffoletti, Augusto. "Automated deployment of a microservice-based monitoring infrastructure." Procedia Computer Science 68 (2015): 163-172.

[5] Linthicum, David S. "Practical use of microservices in moving workloads to the cloud." IEEE Cloud Computing 3.5 (2016): 6-9.

[6] Subhash, L.S. and Udayakumar, R., 2021. Sunflower Whale Optimization Algorithm for Resource Allocation Strategy in Cloud Computing Platform. Wireless Personal Communications, 116(4), pp.3061-3080.

[7] Joseph, Christina Terese, and K. Chandrasekaran. "IntMA: Dynamic Interaction-aware resource allocation for containerized microservices in cloud environments." Journal of Systems Architecture 111 (2020): 101785.

[8] Subalakshmi, N. and Jeyakarthic, M., 2020. Optimal whale optimization algorithm based energy-efficient resource allocation in the cloud computing environment. IIOAB J, 11(2), pp.92-102.

[9] Kumar, A., Sharma, A. and Kumar, R., 2020. A swarm intelligence-based quality of service-aware resource allocation for clouds. International Journal of Ad Hoc and Ubiquitous Computing, 34(3), pp.129-140.

[10] Kumar, Mohit, et al. "ARPS: An autonomic resource provisioning and scheduling framework for cloud platforms." IEEE Transactions on Sustainable Computing 7.2 (2021): 386-399.

[11] Srirama, Satish Narayana, Mainak Adhikari, and Souvik Paul. "Application deployment using containers with auto-scaling for microservices in cloud environment." Journal of Network and Computer Applications 160 (2020): 102629.

[12] Taherizadeh, Salman, and Marko Grobelnik. "Key influencing factors of the Kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications." Advances in Engineering Software 140 (2020): 102734.

[13] He, Xiang, et al. "Programming framework and infrastructure for self-adaptation and optimized evolution method for microservice systems in cloud–edge environments." Future Generation Computer Systems 118 (2021): 263-281.
[14] Shrimali, B. and Patel, H., 2020. Multi-objective optimization-oriented policy for performance and energy-efficient resource allocation in cloud environment. Journal of King Saud University-Computer and Information Sciences, 32(7), pp.860-869.
[15] Kumar, Mohit, et al. "A comprehensive survey for scheduling techniques in cloud computing." Journal of Network and Computer Applications 143 (2019): 1-33.
[16] Chen, X., Li, W., Lu, S., Zhou, Z. and Fu, X., 2018. Efficient resource allocation for on-demand mobile-edge cloud computing. IEEE Transactions on Vehicular Technology, 67(9), pp.8769-8780.
[17] Tseng, F.H., Wang, X., Chou, L.D., Chao, H.C. and Leung, V.C., 2017. Dynamic resource prediction and allocation for cloud data center using the multi-objective genetic algorithm. IEEE Systems Journal, 12(2), pp.1688-1699.
[18] Li-Der, C., Hui-Fan, C. and Fan-Hsun, T., 2018. DPRA: Dynamic Power-Saving Resource Allocation for Cloud Data Center Using Particle Swarm Optimization [J]. IEEE Systems Journal, 12(2), pp.1554-1565.
[19] Kumar, Mohit, and Subhash C. Sharma. "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint." Sustainable Computing: Informatics and Systems 19 (2018): 147-164.
[20] Hasan, R.A., Mohammed, M.N., Ameedeen, M.A.B. and Khalaf, E.T., 2018. Dynamic Load Balancing Model Based on Server Status (DLBS) for Green Computing. Advanced Science Letters, 24(10), pp.7777-7782.
[21] Hammood, O.A., Kahar, M.N.M., Hammood, W.A., Hasan, R.A., Mohammed, M.A., Yoob, A.A. and Sutikno, T., 2020. An effective transmit packet coding with trust-based relay nodes in VANETs. Bulletin of Electrical Engineering and Informatics, 9(2), pp.685-697.
[22] G.F. Gomes, S.S. da Cunha, A.C. Ancelotti "A sunflower optimization (SFO) algorithm applied to damage identification on laminated composite plates" Eng Comput (2018), pp. 1-8
[23] Hussien, A. M., Hany M. Hasanien, and S. F. Mekhamer. "Sunflower optimization algorithm-based optimal PI control for enhancing the performance of an autonomous operation of a microgrid." Ain Shams Engineering Journal 12.2 (2021): 1883-1893.
[24] Vrbaski, Mira, Miodrag Bolic, and Shikharesh Majumdar. "Multi-objective optimization for cloud provisioning: A case study in large-scale microservice notification applications." *2022 9th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2022.
[25] Al Qassem, Lamees M., et al. "Optimal Resource Allocation for Containerized Cloud Microservices." *2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. IEEE, 2022.
[26] Ding, Zhijun, Song Wang, and Changjun Jiang. "Kubernetes-oriented microservice placement with dynamic resource allocation." *IEEE Transactions on Cloud Computing* 01 (2022): 1-1.
[27] He, Xiang, et al. "Online deployment algorithms for microservice systems with complex dependencies." *IEEE Transactions on Cloud Computing* (2022).
[28] Chakraborty, A., Kumar, M., Chaurasia, N. and Gill, S.S., 2023. Journey from cloud of things to fog of things: Survey, new trends, and research directions. Software: Practice and Experience, 53(2), pp.496-551.
[29] Singh, R. and Gill, S.S., 2023. Edge AI: A survey. Internet of Things and Cyber-Physical Systems. Elsevier, 71-92, Vol 3, 2023.
[30] Xu, M., Song, C., Ilager, S., Gill, S.S., Zhao, J., Ye, K. and Xu, C., 2022. "CoScal: Multifaceted Scaling of Microservices With Reinforcement Learning," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 3995-4009, Dec. 2022,