Optimal Datacenter Selection for Cloud Services Using Swarm Intelligence

Shusmoy Chowdhury, Ajay Katangur
Department of Computer Science
Missouri State University
Springfield, USA
[sc26s,ajaykatangur]@missouristate.edu

***Abstract*—Cloud computing has become one of the most influential technologies in the computer science field. In the modern-day world, people are dependent on cloud services in every aspect of life. Cloud services process a vast number of user requests through various data centers. So often, data center selection plays a vital role in user satisfaction and the success of cloud services. Researchers are working relentlessly to use the behaviors of nature to solve real-world problems. In this paper, we used swarm intelligence to find optimal data centers for userbases. The swarm intelligence algorithms use their experience and knowledge of their neighbors to direct the algorithm toward an optimal solution. We have designed a particle swarm optimization-based data center selection policy inspired by swarm intelligence. To explore the accuracy and performance of the algorithm, we simulated it with different real-world scenarios using CloudAnalyst. The simulation results of response time and data processing time of the cloud environment exhibit that the proposed data center selection policy outperforms the traditional data center policy, such as optimized response time and closest data center policy.**

***Index Terms*—Cloud Computing, Datacenter Selection, Swarm Intelligence**

I. Introduction

Technology is evolving and changing to make our life easier and more flexible. Nowadays, several aspects of our lives depend on various software and applications. With the introduction of cloud-based software, many aspects of everyday software usage have drastically changed. The impact of cloud computing on the high availability and scalability of resources cannot be overstated. Giant IT companies to small startups have migrated to the cloud to save money and increase company growth [1]. Nowadays, apart from IT companies, the average internet user can easily create, share, and store their digital media in the cloud and access them from anywhere.

Based on the requirement of cloud users, cloud providers offer their services in three forms, and they are IAAS (Infrastructure-as-a-service), PAAS (Platform-as-a-service), and SAAS (Software-as-a-Service) [2].

Having data center (DC) selection policies is crucial because they are responsible for providing end users with different applications and infrastructure services through DCs. With many DCs scattered worldwide, a single data center can be overloaded with requests and unavailable to end users [3]. Thus it will degrade user satisfaction with cloud services. Moreover, to improve the system's efficiency, different studies also emphasize selecting a suitable DC. The criteria for choosing a DC can be minimum response time, processing time, and cost in a way so that performance of the cloud services can be increased. In this paper, we used a swarm intelligence algorithm to find the optimal DC by minimizing response time, and data processing time. In the modern computing world, cloud services are replacing traditional computing technologies and providing clients with services on a pay-per-use basis [4]. Depending on the policy of the DC selection, cloud services can be the most efficient and effective. Choosing the suitable DC for a user base follows the optimized DC selection policy. An optimal DC can process the tasks efficiently and utilize the resources properly to increase the performance of the cloud.

The objective of our model is to minimize response time and processing time for user requests. For this, we will select the optimal DC using the proposed swarm intelligence-based DC selection policy in cloud computing. We will use Particle Swarm Optimization (PSO) as our swarm intelligence-based algorithm. The optimal DC will utilize the storage and resources appropriately to increase the efficiency of the cloud.

Section II summarizes previous work in the area of DC selection policy. Section III provides an overview of applying PSO in the cloud domain. Section IV outlines our proposed PSO-based service broker policy and the motivation behind using PSO. Section 5 explains the cloud workflow and how the service broker policy is integrated into the cloud environment. Section VI describes the experimental setup in CloudAnalyst, which is used to test the performance of our proposed PSO. Section VII analyzes our algorithm's results compared to the Closest Datacenter (CDC) and Optimized Response Time (ORT) policies. Finally, Section VIII concludes this paper with a summary and future extension of this research.

II. Related Work

DC selection policy tries to find the optimal DC that can handle the request of the users effectively and efficiently. The DC selection policy is crucial in achieving user satisfaction and improving cloud performance. Many researchers have done several works on the DC selection policy in cloud computing.

Rakesh Kumar Mishra et al. [5] proposed a priority and extended priority-based Round-Robin DC selection algorithms.

In this algorithm, the requests are distributed based on the rating of DCs. The authors also explain the existing DC selection policy available in the CloudAnalyst. The authors tried to find the optimal value of the rating factor for the task using an extended priority-based Round-Robin algorithm. The rating factor helps to get the desired performance and cost.

Kunal Kishor et al. [6] proposed a broker policy that uses a proportion weight to decide which DC should be selected for servicing a particular user request. Utilizing a proportion weight ensures an equal workload for each DC. The proportion weight is used to assign a comparable proportion of cloudlets to that DC. Using proportion weights is the algorithm's key feature for better performance. However, the paper needs to state how the proportion weights are determined.

Alfonso Quarati et al. [7] proposed a genetic algorithm-based approach for Cloud Brokering, focusing on allocating resources to applications with diverse Quality of Service (QoS) requirements. This approach supports cloud brokering to distribute batches of jobs to hybrid cloud infrastructures. It will address the cloud broker towards finding solutions that best fit the user's needs. The approach is also appropriate for an enterprise with a private cloud that wants to extend its infrastructure by adopting a public cloud solution.

Yacine Kessaci et al. [8] used a pareto-based meta-heuristic approach to design a DC selection policy. The authors proposed a new Multi-objective Genetic Algorithm Cloud Brokering (MOGA-CB) algorithm where the two objectives are response time and cost of the virtual machine. The fitness function represents the clients' satisfaction and maximizes the broker's profit. The proposed algorithm experimented using realistic data of diverse types of Amazon EC2 instances and their pricing history. The results conclude that the algorithm offers efficient Pareto sets of solutions.

Zakaria Benlalia et al. [9] combine cost and efficiency to optimize the cost and response time. In this paper, the proposed approach chooses the best DC with a function of the (efficiency/cost) ratio. The DC with the minimum ratio is selected if available; otherwise, it chooses the closest DC.

Minhaj Ahmad Khan et al. [10] proposed a normalization-based hybrid DC selection approach integrated with throttled round-robin load balancing to improve resource management through cost-and performance-aware provision of cloud services. Their method used hybrid evaluation criteria using normalization for determining the impact of price and performance-oriented parameters in a cloud environment.

Researchers always try to provide an efficient and effective DC selection policy to maintain the quality of services in the cloud. Nature-inspired algorithms exhibit the behavior of nature's different elements and help solve problems using that behavior. In this paper, we used a swarm intelligence algorithm (PSO) to find an optimal DC. Our goal is to provide a DC selection policy using PSO, which will help us ensure maximum cloud usage satisfaction for the end user.

III. Problem Encoding for Swarm Intelligence Algorithm

A. Particle Structure

Particle structure is a vital factor in any swarm intelligence algorithm. Particles represent an individual in a swarm. The best individual in the swarm, as well as the other individuals, will have the same structure. The structure of the particles in our research will be a sequence of DCs allocated to the userbase. Table I represents the structure of a particle in the swarm. The data structure we used to visualize the particle structure is an array. The length of the particle array will be the number of userbases available in the cloud. The array index represents each userbase, and the data at that array index represents the DC allocated to that userbase.

TABLE I
Particle structure of the Swarm Intelligence Algorithm

UB1	UB2	UB3	UB4	UB5	UB6	UB7	UB8	UB9
DC2	DC0	DC1	DC5	DC4	DC3	DC2	DC1	DC3

B. Swarm Size

Swarm size indicates the number of particles that will search for optimal solutions. The more significant number of particles with a uniform initialization procedure ensures the diversity of the swarm. A large swarm covers many parts of the search space in each iteration. It also helps reach a satisfactory solution with fewer iterations than a smaller swarm. Most research in the past has shown that PSO can find optimal solutions using small swarm sizes of 10 to 30 particles [11]–[13]. We have determined the size of swarms with cross-validation using improved performance criteria. We have used 30 as the swarm size for our proposed algorithm.

C. Objective Function

The objective function defines the criteria based on which the best particle in the swarm is determined. The quality of the cloud services relies on the cloud's response time and data processing time. So, the DC selection should be in a way such that the overall response time and data processing time are minimized. The DCs are a collection of physical servers with an abstraction of VMs over each physical server. The cloud's response time and data processing time are impacted by factors such as data transfer cost, region, number of physical machines, peak and off-peak hours in the userbase, and so on. We have considered these factors in our swarm optimization algorithm to find an optimal DC for the user bases. The goal is to optimize the DC's response and data processing time. The optimization or objective function is shown in equation 1

$$F(R_t, D_t) = (\sum R_t + \sum D_t)/N \tag{1}$$

Here,
R_t = Response time of each datacenter
D_t = Data Processing time of each datacenter
N = Number of datacenters

IV. Proposed Particle Swarm Optimization

PSO is one of the most powerful swarm optimization algorithms in the nature-inspired computing algorithm world. PSO is inspired by the foraging and social behavior of the swarm. The swarm searches for food in a cooperative way. Each member in the swarm gains information from their own experience and the experience of the other members and changes the search direction accordingly to locate the food. The PSO algorithm tries to balance the exploration and exploitation of the search space and thus avoid the convergence to a local optimum and ensures an excellent rate of convergence to the optimum. The PSO algorithm will start with initializing particles in the population and then evolves through evaluating the fitness to get the best particle, updating the particle according to its own and neighbor's experience, and terminating when the goal is reached. "Fig. 1" shows the flow chart of the PSO algorithm. The steps performed in the PSO algorithm are further detailed in this section.

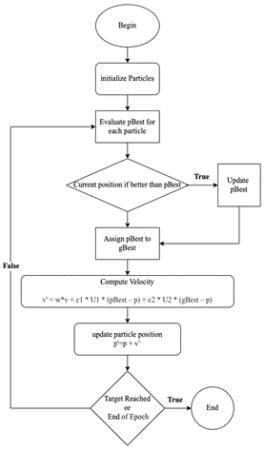

Fig. 1. Flow Chart of Particle Swarm Optimization Algorithm

A. Initializing the Particles

In PSO, the individuals of the populations are called particles. The particles from the initial population are initialized randomly. To find the best solution, PSO uses several particles to constitute a swarm moving around in the search space. Each particle has two principal factors.

1) **Position:** Position will keep the structure of the solution. In our research, it will be an array where the array index represents the userbase number, and the array element will be the DC allocated to the userbase.
2) **Velocity:** Velocity is used to move the particle's position toward the best solution.

B. Evaluating the Particles

There are two criteria that are typically used in PSO for evaluating the fitness of each particle in the population. The first one is the personal best position. Personal best is the best position for an individual in the population. It is updated whenever the fitness value is improved over the previous best. The second criterion is the global best position. The population is sorted based on the fitness value of the individuals. The global best will be the individual who has the best fitness among the individuals in the population.

C. Updating the position of the Particles

The particle's position is regularly updated to move toward the optimal solution. As a result, each particle needs appropriate velocity to move towards the optimal solutions. "Fig. 2" shows the movement of the particles in the swarm. Every particle will move using its personal influence, also known as particle memory influence, and social influence, also known as swarm influence, and try to reach the optimal position in the swarm. The velocity of the particle is calculated using

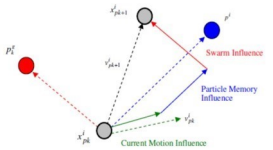

Fig. 2. Movement of the Particle

equation 2 and the new position of the particle is calculated using equation 3.

$$v' = w*v+c_1*U_1*(pBest - p')+c_2*U_2*(gBest - p') \quad (2)$$

$$p' = p + v' \quad (3)$$

Here,
$w * v$= inertia
$c_1 * U_1 * (pBest - p')$= personal influence
$c_2 * U_2 * (gBest - p')$ = social influence
p': particle's new position
p: particle's old position
v': new velocity of the particle
v: previous velocity of the particle
c_1: weight of local information, cognitive constant
c_2: weight of global information, social constant
pBest: best position of the particle
gBest: best position of the swarm
U_1, U_2: random variable

The process of updating the velocity is divided into inertia, personal influence, social influence and acceleration coefficient.

1) Inertia: Inertia helps the particle to move in the same direction and with the same velocity. The weight w is the inertia weight and positive constant, thus used for balancing global search (exploration) and local search (exploitation). It helps search for innovative solutions and finds the regions with the best solutions. We calculate w using equation 4.

$$w = e^{-epoch} \quad (4)$$

Here in equation 4 we have used w as an exponential function of epoch(iteration). As a result with the progress of time, the effect of inertia will become less as the particle will move towards global solution.

2) Personal Influence: It improves individual particles and makes them return to their previous best position.

3) Social Influence: Social influence helps to direct the particles to move towards the best particle of the population.

4) Acceleration Co-efficient: The acceleration co-efficient c_1 and c_2, along with the random variables U_1 and U_2 controls the stochastic influence of the personal and social components in overall velocity. c_1 and c_2 are trusted parameters and represent the level of confidence a particle has in itself and its neighbors, respectively. If c_1 and c_2 are zero, then particles will keep moving at their current speed until they cross a boundary of the search space. In this research, we have made c_1 an exponential function of epoch as shown in equation 5. As a result, with the progress of time, personal experience will have less effect on the movement, and social influence will help the particle move in the direction of the optimal solution.

$$c_1 = e^{-epoch} \tag{5}$$

c_2 will be constant and it's value is 2, U_1 and U_2 are random numbers from 0 to 1.

D. Termination

The PSO algorithm will terminate when the particles in the population reaches the global optimum. The global optimum is defined by comparing the current objective function value of the best individual in the swarm to the current individual running in the DC selection policy. The objective function value is also reviewed for convergence (individuals are consistent with the better fitness value).

E. Motivation to use Particle Swarm Optimization

Cloud services are working continuously every hour with different request sizes and a different number of requests. So, DC selection needs to be efficient and effective with time progress. PSO is inspired by the social behavior of the birds within a flock. In PSO, the particles are moved through a hyperdimensional search space using the social-psychological tendency of individuals. Particles try to replicate the success of other individuals. If one of the particles can find the optimal solution search space, other particles in the swarm will follow the direction and search through the optimal solution search space. The changes to a particle in the swarm are influenced by its experience and knowledge of its neighbors. As a result, particles can stochastically return toward previously successful regions in the search space. So, in a cloud, DC selection will start with swarms of random particles and then move toward the optimal solutions using swarm intelligence. For these reasons, we have considered PSO for the DC selection

V. Cloud Environment

A cloud environment is a complex system with attributes such as userbase, DC, load balancing policy, and service broker policy. "Fig. 3" shows the workflow of the proposed cloud environment. The cloud components have several characteristics: data transfer cost, request size in peak and off-peak hours, latency, bandwidth, request number, etc. The service broker policy is responsible for choosing the DC for the userbase based on these attributes of the cloud. In the beginning, the user will configure the environment according to their requirements, such as:

- the number of DCs needed for their requests
- the request size
- timing of the peak and off-peak hours,
- the number of requests at these hours.
- the DC configuration
- the load-balancing policy

The user request and DC are configured in the service broker policy. The service broker policy will decide the selection of DCs for a particular userbase for a specific time based on these inputs. We have used swarm intelligence-based algorithms, specifically PSO algorithms, to choose the optimal DCs.

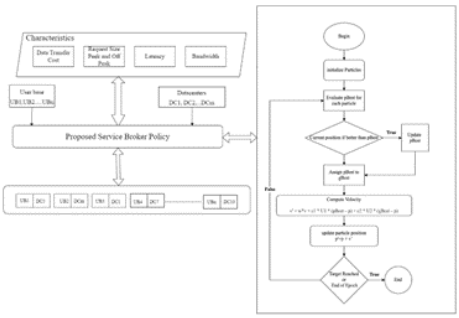

Fig. 3. Workflow of Cloud Environment

VI. Environment Setup

We need an appropriate environment to check the proposed algorithm's accuracy, effectiveness, and efficiency. Many cloud computing providers exist, such as Amazon Web Services (AWS), Microsoft Azure, Google, IBM Cloud, Red Hat, Verizon cloud, and VMware [14]. They provide the user with access to various configurable computing resources (servers, storage, networks, applications). These providers will only let us integrate our proposed solution once we prove the correctness of our algorithms in the cloud environment. Therefore, we need a simulation environment to simulate real-world cloud scenarios and demonstrate the algorithm's efficiency.

Cloudsim [15] is a cloud simulation modeling platform. It allows VMs to be managed by hosts, which DCs manage. Cloudsim does not have any user interface for the users, so it is hard to configure the system according to real-world scenarios. Moreover, the output results are presented in the console, which makes it challenging to generate graphical reports.

Another simulation environment CloudAnalyst [16] platform overcomes these challenges by providing a graphical user interface for the users to configure the cloud environment according to their requirements. "Fig. 4" shows the world view of the CloudAnalyst simulator. The world in the CloudAnalyst platform is divided into six regions defined as R0, R1, R2, R3, R4, and R5. The blue dots in the map represents the userbase, and the red dots represent the DCs of the cloud

services around the globe. The four main vital factors of the CloudAnalyst are userbase, DC, Load Balancing Policy, and Service broker policy. We integrated the proposed swarm intelligence algorithm, PSO, into the DC selection policy.

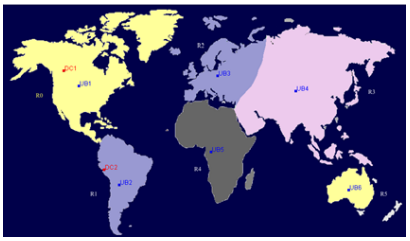

Fig. 4. CloudAnalyst Simulator

A. Userbase Configuration

Userbase defines a group of users from a particular region. The userbase has several properties. The user request per hour can vary for each userbase. Peak hours are a vital factor for the user base. Peak hours define the time when most users are active and when the cloud is processing an enormous number of requests. The average number of users during peak hours is exceptionally high compared to the average number during off-peak hours. Table II shows a sample userbase configuration used in our experiments. We have used similar userbase configurations with varying number of userbases.

TABLE II
USERBASE CONFIGURATION 1 (UBC1)

Name	Region	Request Per User per Hra	Data Size per Request (Bytes)	Peak Hours Start (GMT)	Peak Hours End(GMT)	Avg Peak Users	Avg Off-Peak Users
UB1,UB11,UB14, UB16,UB18,UB40	0	12,60,60, 60,60,60	100	13,3,3, 3,3,3	15,9,9, 9,9,9	400000,10000,10000, 10000,10000,1000	40000,1000,1000, 1000,100,100
UB2,UB6,UB20, UB35,UB39	1	12,12,60, 60,60	100	15,3,3, 3,3	17,9,9, 9,9	100000,80000,1000, 1000,1000	10000,1000,1000, 100,100
UB3,UB7,UB8, UB9,UB10,UB13, UB15,UB17,UB19, UB23,UB24	2	12,60,60, 60,60,60, 60,60,60, 60,60	100	20,3,3, 3,3,3, 3,1,3, 3,3	22,9,9, 9,9,9, 9,2,10, 9,9	300000,10000,10000, 10000,10000,10000, 10000,1000,10000, 1000,10000	30000,1000,1000, 1000,1000,1000, 1000,1000,1000, 1000,1000
UB4,UB12,UB26, UB30,UB32,UB38	3	12,60,60, 60,60,60	100	1,3,3, 3,3,3	3,9,9 9,9,9	150000,10000,10000, 10000,10000,1000	15000,1000,100, 1000,1000,100
UB5,UB21,UB25, UB31,UB33,UB34	4	12,60,60, 60,60,60	100	21,3,1, 15,7,1	23,9,7, 18,8,4	50000,1000,10000, 10000,10000,10000	5000,1000,1000, 1000,1000,1000
UB22,UB27,UB28, UB29,UB36,UB37,	5	60	100	3,15,3, 9,3,3	9,19,9, 12,9,9	10000,10000,10000, 10000,1000,1000	1000,100,100, 100,100,100

B. DC configuration

The DC tries to process the userbase request effectively and efficiently. DC is a collection of physical servers, where each physical server works like a computer's CPU in the cloud. Cloud computing uses an abstraction over the physical servers named VM. The VMs can have the same configuration as physical servers or be different from the physical servers. The combined configuration of the VMs must be equal to that of the available physical servers. Creating more VMs than the available resources or physical servers is impossible. The tasks are divided among the VMs using load-balancing algorithms. The more significant number of VMs enhances the performance of the cloud environment.

The DC is configured as per the parameters defined in Table III. In this research, we have used 20 physical servers in every DC. Each physical server is configured, as shown in Table IV. Each physical server is further abstracted with VMs. We have used 25 VMs for our experiments. They have 1024 MB memory, 10000 MB image size, and 1000 Mbps bandwidth.

TABLE III
DC CONFIGURATION

Region	Architecture	Operating System	Virtual Machine Monitor	Cost per VM Hour	Cost per Mb Memory Hour	Storage cost per Gb	Data Transfer cost per Gb	No of server
0-5	eX86	Linux	XEN	0.1$	0.05$	0.1$	0.1$	20

TABLE IV
PHYSICAL SERVER CONFIGURATION

Memory	Storage	Available Bandwidth	Number of Processor	Processor Speed	Virtual Machine Policy
2048 MB (2GB)	100000 MB	10000	4	100	Time Shared/ Space Shared

C. Load balancing Policy

In the DC, the load balancing policy plays a vital role in managing the user requests among the VMs. DCs are a collection of physical servers with identical or different configurations. Cloud computing uses abstraction over the DCs, which are called VMs. The tasks are processed on these VMs. Load balancing dynamically distributes the workload arriving at the DC among different VMs so that they are not overwhelmed with the tasks or remain idle because of no tasks. A load-balancing policy ensures the proper resource utilization of the DC. Many load balancing policies are available in the cloud services, such as First Come-First Serve (FCFS), Round Robin, Equally Spread Current Execution Load, and Throttled Load Balancing Policy. Round Robin load balancing policy distributes the tasks among the VM in the rational order of the VMs. FCFS chooses the first VM until it is loaded up with tasks and not responding anymore. The throttled load balancing algorithm marks the VM as busy or available to assign tasks. Equally Spread Current Execution load balancing algorithm equally distributes the tasks among different VMs. In this research, we have only used the round-robin load-balancing policy for all the experiments.

D. Experimental Scenarios

We have used five cloud configurations to examine the correctness of our algorithm. Table V shows the cloud configurations of our experiment.

TABLE V
CLOUD CONFIGURATIONS

Configuration Name	User base Configuration	DC Configuration
Cloud Configuration 1 (CC1)	UBC1	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 2 (CC2)	UBC3	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 3 (CC3)	UBC2	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 4 (CC4)	UBC1	4 DCs, 1 in region 0, 1 in region 2, 1 in region 4 and 1 in region 3
Cloud Configuration 5 (CC5)	UBC2	4 DCs, 1 in region 0, 1 in region 2, 1 in region 4 and 1 in region 3

VII. Result Analysis

TABLE VI
AVERAGE RESPONSE TIME IN MILLISECONDS FOR DIFFERENT ALGORITHMS

Configuration Name	Particle Swarm Optimization Algorithm	Optimized Response Time	Closest Datacenter
CC1	937.44	2719.48	2752.09
CC2	1758.34	2500.77	4942.96
CC3	768.55	882.01	2429.67
CC4	904.08	812.23	2419.39
CC5	1407.91	1667.57	3901.81

Table VI and Table VII exhibits the average response time and data processing time between the proposed swarm intelligence (PSO) algorithm and traditional algorithms for different scenarios. The results show that the CDC policy is the worst among all the service broker policies. The CDC policy chooses the nearest DC of the userbase. As a result, the closest DC gets overcrowded with user bases in the same region. ORT policy chooses the DC with the best response time and yields better results than CDC, as evidenced by the results in Table VI and Table VII. But it also overcrowds the DC with the best response time. From Table VI and Table VII, we can see

TABLE VII
AVERAGE DATA PROCESSING TIME IN MILLISECONDS FOR DIFFERENT ALGORITHMS

Configuration Name	Particle Swarm Optimization Algorithm	Optimized Response Time	Closest Datacenter
CC1	568.82	2151.20	2343.70
CC2	1288.32	1977.13	4205.59
CC3	460.72	662.20	2108.89
CC4	575.65	601.34	2100.54
CC5	943.16	1356.07	3514.47

the results of the proposed swarm intelligence algorithm. Our proposed PSO outperforms the traditional algorithms in most scenarios. Table VI shows that the PSO is better than ORT and CDC in all cases except for CC4. In CC4, the PSO lags the ORT by almost 100 milliseconds in response time. The difference is negligible, and the PSO performs better in all other cases, so we can consider that an outlier. But again from Table VII, if we notice the data processing time for CC4, we can see that the data processing time of PSO is better than the other algorithms in CC4 and other cases. Finally, although the PSO sometimes cannot provide a reasonable response time, it will give better data processing time due to optimal DC selection for the userbase.

VIII. Conclusion

In this paper, we have used swarm intelligence to find the optimal DC in cloud services. We have used the PSO algorithm for our DC selection policy. The particles represent the solution structure of the DC selection policy, which is a sequence of DC allocated to different userbases in the cloud services. The particles in the swarm will use their personal experience in the search space as well as the knowledge of their neighbors to move towards optimal solutions. The optimal solution will minimize the response time and data processing time in the cloud environment and thus help increase user satisfaction with the cloud services. The results show that the proposed swarm intelligence DC selection provides better performance (response time and data processing time) than other DC selection policies such as CDC and ORT.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant OAC-1828380.

References

[1] T. Khanom, "Cloud accounting: a theoretical overview," *IOSR Journal of Business and Management*, vol. 19, no. 06, pp. 31–38, 2017.

[2] D. Rani and R. K. Ranjan, "A comparative study of saas, paas and iaas in cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, 2014.

[3] B. Nayak, B. Bisoyi, and P. K. Pattnaik, "Data center selection through service broker policy in cloud computing environment," *Materials Today: Proceedings*, 2021.

[4] R. P. Padhy, M. R. Patra, and S. C. Satapathy, "Cloud computing: security issues and research challenges," *International Journal of Computer Science and Information Technology & Security (IJCSITS)*, vol. 1, no. 2, pp. 136–146, 2011.

[5] R. K. Mishra and S. N. Bhukya, "Service broker algorithm for cloud-analyst," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3957–3962, 2014.

[6] K. Kishor and V. Thapar, "An efficient service broker policy for cloud computing environment," *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 2, no. 4, 2014.

[7] A. Quarati and D. D'Agostino, "Moea-based brokering for hybrid clouds," in *2017 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2017, pp. 611–618.

[8] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 2496–2503.

[9] Z. Benlalia, A. Beni-hssane, K. Abouelmehdi, and A. Ezati, "A new service broker algorithm optimizing the cost and response time for cloud computing," *Procedia Computer Science*, vol. 151, pp. 992–997, 2019.

[10] M. A. Khan, "Optimized hybrid service brokering for multi-cloud architectures," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 666–687, 2020.

[11] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, vol. 2. Singapore: Orchid Country Club, 2002, pp. 692–696.

[12] F. Van den Bergh, "An analysis of particle swarm optimizers [ph. d. dissertation]," *Department of Computer Science, University of Pretoria, Pretoria, South Africa*, 2002.

[13] F. v. d. Bergh and A. P. Engelbrecht, "Effects of swarm size on cooperative particle swarm optimisers," in *Proceedings of the 3rd annual conference on genetic and evolutionary computation*, 2001, pp. 892–899.

[14] "Top 10 cloud computing service providers in 2017," 2017. [Online]. Available: https://www.technavio. com/blog/top-10-cloud-computing-service-providers-2017

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[16] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.