Received: September 13, 2020. Revised: December 29, 2020. 271

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Swarm Intelligence-based Approach for Dynamic Data Replication in a Cloud Environment

Ahmed Awad[1] Rashed Salem[2] Hatem Abdelkader[2] Mustafa Abdul Salam[3]*

[1]*Information System Department, Cairo Higher Institute for Languages and Simultaneous Interpretation, Egypt*
[2]*Information System Department, Faculty of Computers and Information, Menoufia University, Egypt*
[3]*Artificial Intelligence Department, Faculty of Computers and Artificial Intelligence, Benha University, Egypt*
* Corresponding author's Email: mustafa.abdo@ymail.com

Abstract: In recent years, there has been increasing interest in cloud computing research, especially replication strategies and their applications. When the number of replicas is increased and placed in different places, maintaining the system's data availability, performance and reliability will increase the cost. In this paper, two multi-objectives swarm intelligence algorithms are used to optimize the data replication selection and placement in a cloud environment. These algorithms are namely, multi-objective particle swarm optimization (MOPSO) and multi-objective ant colony optimization (MOACO). The first algorithm, (MOPSO), is used to find the best selected data replica according to the most popular data replication strategy. The improved time-based decay function (ITBDF), is used to enhance the proposed model. The second algorithm, (MOACO), is used to find the best data replica placement according to the minimum distance, the number of data transmissions and the availability of data replication. A simulation of the suggested strategy has been performed using CloudSim. the Cloud is formed to simulate different kinds of datacenters (DCs) with different structures. Moreover, 21 DCs are used. Each DC consists of a host that contains a set of virtual machines (VMs) that provides blocks of available data replications. Three different data placements for high datacenters were created. A total of one thousand cloudlets are randomly confirmed for the data replication order. All replication files are placed in high datacenters and randomly distributed in the suggested system. The performance of proposed strategy was evaluated relative to many well-known strategies such as, Enhance Fast Spread (EFS), Dynamic Cost-aware Re-replication and Re-balancing Strategy (DCR2S), Genetic Algorithm (GA), Genetic adaptive Selection Algorithm (GASA), Replica Selection and Placement (RSP), Dynamic Replica Selection Ant Colony Optimization (DRSACO), Adaptive Replica Dynamic Strategy (ARDS), Popular File Replication First (PFRF). The experimental results show that MOPSO, achieves better data replication than compared algorithms. Additionally, MOACO, achieves higher data availability, lower cost, and less bandwidth consumption than compared algorithms.

Keywords: Dynamic replication, Cloud computing, Cloudsim, Knapsack problem, Particle swarm optimization, Ant colony optimization and multi-objective optimization.

1. Introduction

Cloud environments provide many services such as pay-per-use virtual computing, network resources and services, which can be developed and have their usage well scaled. The characteristics of the Cloud include the bandwidth, storage and access to important data resources [1, 2]. A Cloud environment consists of software-as-a-service (SaaS), platform-as-a-Service (PaaS) and infrastructure as a service (IaaS) [3]. Moreover, the Cloud is less expensive than other traditional systems and can feature subscription upon request, scalability, elasticity and dynamicity [4]. Load balancing in the Cloud has a great effect on the data transfer rate, performance and low overloads of the Cloud network [5]. It distributes tasks in a virtual machine (VM) to balance the loads in the Cloud to retain low overloads and achieve optimal access to available resources [6].

Replication techniques are generally used in data-intensive applications such as sharing and distribution from distant locations to near locations

through nodes or geographical sites [7]. Replication techniques can be applied as a cluster in a Cloud environment. The cluster allows scalability and data availability to ensure the integrity and consistency between different replications among nodes. This includes reading and writing on data replication through protocols according to the system requirements [8]. In the reviewed literature [9], there are surveys that address static and dynamic replications through the Cloud and in the data grid field. Static and dynamic replications are subject to three important questions that require answers: 1) what data should be replicated? 2) when should the data be replicated? and 3) where should the new replicas be placed? These are three important open questions that need to be addressed for data replication in Cloud environments [10].

In this paper, an efficient dynamic data replication strategy using MOPSO and MOACO are proposed. The proposed strategy is based on the selection and placement of data replicas in different datacenters. It evaluates data replication through seven criteria, such as data replication access, distance, costs, data availability, data replication popularity, and the Zipf and geometric distributions. Therefore, we have access to two optimal selection and placement methods for data replicas in the Cloud.

The proposed model is validated using CloudSim, and the two algorithms MOPSO and MOACO are evaluated. The experimental results show that the proposed strategies outperform the other strategies, saving time, accelerating the access speed, reducing the costs, ensuring high availability and finding the least-cost path among the heterogeneous datacenters in the Cloud.

The remainder of this paper is organized as follows. Section 2 reviews the literature on replication strategies in the Cloud. Section 3 introduces the proposed architecture for replica selection and placement optimization. Section 4 focuses on the suggested MOPSO and MOACO algorithms. Section 5 shows the experimental results and configuration. It also provides the two suggested algorithms with their performance and then compares them to other algorithms. Finally, Section 6 concludes the paper and highlights future research.

2. Related work

Many related studies have researched data replication strategies in the Cloud, such as the following.

N. Mansouri et al. presented a dynamic popularity-aware replication strategy (DPRS) to recover and access data using the 80/20 idea. They employed parallel data downloads from different sites to improve the data file availability and storage space. Network efficiency measures the average service time to execute the required tasks for the set number of files using the Cloud. The mentioned algorithm was compared with five others algorithms (SWORD, Adaptive Data Replication Strategy (ADRS), DRSP, MORM and A2RS) and it was found that DPRS was more efficient than the others [11].

D. Sun et al. designed the modeling a dynamic data replication strategy (MDDRS). The research also conducted an abridged survey of the suitable distribution methodology in the Cloud [12].

N. Kaur and et al. suggested a dynamic, cost-aware, optimized data replication strategy. Their work is an extension of the work in [12] and determined the minimum data replications and file availability near the users in an ordinary tier without losing data. Moreover, it considered a heterogeneous system, which improved the costs of using the knapsack problem [13].

N. Mansouri. showed an ADRS for the placement and replacement of replicas through datacenters. It considers the storage space of the system. When placing new replicas once the space is full, the ADRS will place new replicas instead of old ones. The results confirmed its optimal efficiency over those of the others [14].

K. Kumar et al. suggested a workload-aware data replication strategy called (SWORD) to optimize the resource consumption in Cloud environments. The suggested strategy reduces the average time for either a search request or another process and shares mechanisms in the system environment to analyze work-loads and place data using mathematical models [15].

X. Bai et al. presented a response time-based replica management (RTRM) model that automatically selects replicas and places them in nodes. RTRM depends on diagram theory to solve the problems [16].

D. Yuan and et al. suggested a data placement strategy based on a k-means matrix in cluster form to place data workflows in a Cloud environment containing two algorithms in the building stage and work time. [17].

C. Hamdeni et al. designed a new adaptive technique for calculating the data popularity in database systems through nodes. The experimental results that were achieved using OptorSim showed that the suggested system can adapt the popular file access time and achieve file availability [18].

N. Maheshwari et al. introduced a dynamic energy-efficient data placement and cluster reconfiguration algorithm using GridSim. The

suggested algorithms provide efficient power availability for developing distributed data. [19].

Q. Xu et al. proposed a data-placement strategy based on the genetic algorithm (DPSBGA) to schedule data and optimize the placement of data replications via datacenters. It is clear that using the GA provides a better solution for placing and accessing data replications [20].

Z. Li et al. suggested a different combined replication structure to improve the performance of high-level architecture (HLA). The suggested structure depends on using different synchronization methods to access and select rapid replications for users. Messages are sent and received between users to reduce the time and handle the replications [21].

Z. Tang et al. suggested an intermediate data placement algorithm for load balancing in a Spark environment that attempts to optimize the work among the nodes in order to perform the required processing data tasks through map tasks. The skewed intermediate data blocks (SCID) algorithm attempts to arrange key/value clusters of map tasks according to their size. [22].

T. Yuan et al. suggested causal consistency algorithms for partially and fully replicated systems. The author presents two algorithms to realize the causal consistency of partial and full replications to perform causal consistency tasks in reliable large-scale distributed systems [23].

H. Casanva et al. investigated the impacts of the replication process on an application with large-scale parallel executions and coordinated checkpointing of full replications. They analyze the mean service time to interrupt haphazard distributions and their failures [24].

P. Matri et al. introduced the keeping up with storage: decentralized, write-enabled dynamic geo-replication (DWEDGR) method. DWEDGR allows users to enable access to the nearest replication among nodes without any previous request that supports the changeable data [25].

M. Chang Lee et al. suggested the popular file replicate first (PFRF) strategy, which is based on a star network that accesses and determines the file and popularity of each data replication. Both Zipf and geometric distributions are used to evaluate the suggested algorithm. [26].

3. Methodology and discussionon

3.1 Proposed architecture for replica selection and placement optimization

This section addresses the suggested structural model for sharing data among nodes by selecting and placing replications in Cloud environments, which has been used in many studies [12, 13]. In our case, we used the heterogeneous system with swarm intelligence (SI) techniques to optimize the selection and placement of data replications. The first stage consists of the high datacenters, which are more deeply centralized and possess better data availability, storage space, and performance and greater numbers of hosts and VMs than the other datacenters. The second stage consists of the mid-datacenters, which have fewer whole components than the first stage. The third stage is composed of low datacenters that have fewer whole components than the second stage. Overall, the datacenters are hierarchically related to each other either in the same level or in other levels. The suggested system consists of datacenters, a broker, a replica catalog, a replica management system, a virtual machine, hosts, tasks and hierarchical network users who connect datacenters with each other. To assure that the best SI techniques with replication can be implemented in the optimal suggested system, ACO and PSO can be applied to the selection and placement of replications through nodes. Appling the above SI techniques improves the availability and support, reduces the cost, allows for tasks to be performed, and is quicker than the techniques that are used when selecting and placing replications through CloudSim. Therefore, we can realize the maximum use and optimal replications through datacenters.

The datacenters can be represented in the form of DCs = $\{dc_1, dc_2...dc_n\}$, where n is the number of different DCs in the Cloud environment. The physical machine can be represented as PM = $\{pm_1, pm_2 \ldots pm_x\}$, where x is a group of different pm_s that exist in DCs. Our different system for heterogeneous Cloud datacenters is proposed. The virtual machines can be represented as VM = $\{vm_1, vm_2 \ldots vm_s\}$, where s is the number of VMs that are placed in the PMs. There are different spaces and different operations such as space shards and time shards. The data replication can be represented in the form of F= $\{f_1, f_2 \ldots f_y\}$, where y is the number of different data replications that could be placed inside DCs. The main storage unit is a block and can be represented as B = $\{b_1, b_2 \ldots, b_y\}$, where y is a group of a different data replications stored in DCs. All replication files are placed in high datacenters and randomly distributed in the suggested system. Different values of the probability = pro (bap) can be saved in every DC using the replica catalog that records every replica location in the different DCs.

3.1.1. Data file availability

Generally, availability can be defined as "readiness to offer correct services". The data replications should be made fully available to all users upon request. Therefore, the availability of data is an important issue in the Cloud for users. A system with unavailable data is caused by a data replication or node failure in the Cloud. Data availability is an important issue in the Cloud [23, 24]. The replicas can be located in many DCs, and many replicas can be located in the same DC to ensure the availability of different blocks in the DCs. High DCs have higher costs, better data availability and reliability, and consequently, the blocks that are inside the DCs have high availability. However, low DCs have lower costs, worse availability, and worse reliability, and the blocks that are inside the DCs are likely to have low availability. They can be calculated using Eq. (1) to Eq. (4) as follows:

$$pro(bap_j)_{hig\ DC} > pro(bap_j)_{mid\ DC} > pro(bap_j)_{low\ DC} \quad (1)$$

$$pro(fla_k) = \begin{cases} (1 - \prod_{i=1}^{bnr_k}(1 - pro(bap_j)i))^{nb_k} & for\ case\ 1 \\ \prod_{i=1}^{nb_k}(1 - \prod_{i=1}^{bnr_k}(1 - pro(bap_j)i) & for\ case\ 2 \end{cases} \quad (2)$$

$$\overline{pro(fla_k)} = \begin{cases} 1 - (1 - \prod_{i=1}^{bnr_k}(1 - pro(bap_j)i))^{nb_k} & for\ case\ 1 \\ 1 - \prod_{i=1}^{nb_k}(1 - \prod_{i=1}^{bnr_k}(1 - pro(bap_j)i)) & for\ case\ 2 \end{cases} \quad (3)$$

Let the block available probability Pro (bap_j)

$$high_{dc} = 0.9 > mid_{dc} = 0.6 > low_{dc} = 0.3 \quad (4)$$

Where, B	*blocks*
Pro(fla_k)	*Probability of file availability*
nb_k	*Number of block*
bnr_k	*Number of replica of a data file df_k*
$\overline{Pro(flaJ)}$	*Block unavailability probability ofblock*
na_k	*Number of access task have request*
$\overline{Pro(flak)}$	*Probability of file unavailability*
High DCs	*High Datacenters*
Mid DCs	*Mid Datacenters*
Low DCs	*Low Datacenters*

3.1.2. Replication costs in datacenters

There is a set of DC costs that differentiate low DCs, mid DCs, high DCs, and this set includes the costs, availability probabilities, velocities, reliabilities and performances in the Cloud. The costs of DCs are important factors that include the factors of MOO that select and place replicas through DCs. The total replication costs must be kept as reasonable as possible to save replications through DCs and to ensure that the budget is sufficient for users.

$$cost_k(dc_s) = \sum_{x=1}^{y}\left(cost(dc_y).bnr_k(dc_y)\right) \quad (5)$$

3.1.3. Minimum distance between datacenters

The distance between DCs is calculated according to the access to data replications, which can be calculated using the following Eqs. (6) and (7) [1]. The best solution uses the connection between two DCs (I and j). This equation indicates that dci and dc_j are passed by confirming that this equation assures their nonpassing in an endless ring.

$$Min\ \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \quad (6)$$

St.:

$$\sum_{i=1}^{n} n_i x_{i\geq k,} \qquad x_i\in\{0,1\} \qquad (1\leq i\leq n) \quad (7)$$

3.1.4. knapsack problem

The knapsack problem is NP-hard. Any item has an assigned value and weight. The aim is to minimize the cost in the Cloud by ensuring that the budget is sufficient for users by using the following Eqs. (8) and (9):

$$maximize\ px = \sum_{j=1}^{n} p_j x_j \quad (8)$$

St.:

$$wx = \sum_{j=1}^{n} w_{ij}x_j \leq v_i \quad (9)$$

$x_j \in \{0,1\},\ \ j = 1,2,\ldots\ldots n$
$p = (p_1, p_2, \ldots\ldots p_n)$
$w = w_1, w_2, \ldots\ldots w_n$
$i = 1,2,\ldots\ldots\ldots m$

Each object $j \in J$ has profit p_j and weight w_j in dimension i ($1 \leq i \leq m$)

Binary variable x_j indicates whether object j is included in the knapsack ($x_j = 1$) or not $(x_j = 0)$.

We use the MOO process with AI techniques of PSO and ACO to determine the data availability,

optimal costs and different path distances in DCs. We perform the three processes together based on the AI techniques to optimize the access, selection and placement of replications in the suggested system.

3.2 Proposed PSO and ACO-based algorithm for the cloud environment

The suggested system incorporates two different algorithms: PSO and ACO. PSO selects the data replications and ACO places the data replications. Therefore, the suggested functional system consists of three basic characteristics: file availability, access time and costs. Meanwhile Zipf and geometric distribution are used to distribute the data replications in the Cloud.

3.2.1. The proposed method to determine which file to replicate and when to replicate using PSO

In this subsection, we discuss the PSO algorithm on a wide scale to assess the data replication using CloudSim. To solve the suggested problem, we use MOO along with PSO in the suggested method. The PSO optimizes many problems in the optimization and determination process, and this algorithm has two important aspects: exploration and exploitation. [27-31] In this stage, exploration aims at discovering the optimal search space solutions and local adjacent space. Exploitation searches the recent solutions and selects the best-suggested solutions. A fitness function evaluates the optimally selected data replication for every particle along with the best. Then, the process updates the particle velocity, position and inertia weight using Eqs. (10), (11), and (12) as follows. [32]. We update the velocities for every dimension as follows:

$$V_{i,j}^{k+1} = W.V_{i,j}^{k} + C1R1\left(pbest_{i,j}^{k} - X_{i,j}^{k}\right) + C2R2\left(gbest_{i,j}^{k} - X_{i.j}^{k}\right) \quad (10)$$

Where

$V_{i,j}^{k+1}$ *Represents the new velocity of a particle*

$V_{i,j}^{k}$ *represents its current velocity.*

$C1, C2$ *positive constants acceleration parameters.*

$pbest_{i,j}^{k}$ *personal best position partical.*

$X_{i,j}^{k}$ *position of ith partical in jth swarm.*

$R1, R2$ *two random variables in the range [0,1].*

$gbest_{i,j}^{k}$ *gloabel best position partical.*

$$X_{i,j}^{k+1} = X_{i,j}^{k} + V_{i,j}^{k+1} \quad (11)$$

Where

$X_{i,j}^{k+1}$ *new position of partical.*

k *iteration population.*

i 1,2,3, …. m m *is the number of members in a iterations.*

j 1,2,3, …. d d *is the size of the swarm.*

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}}.iter \quad (12)$$

Where

w *inertia weight*

w_{max} *initial value of inertia weight.*

w_{min} *final value of inertia weight.*

$iter_{max}$ *maximum number of iterations*

$iter$ *current iteration number.*

When the replica access task is performed by users, we determine that the algorithm accesses and selects the optimal replica. It calculates the fitness function of the optimally selected replica for every particle and uses MOO to determine the shortest paths, lowest costs and lowest access times for data replication in the Cloud.

3.2.2. Analyze the type of fragment's and access time to replication

The improved time-based decay function (ITBDF) is used to determine the priorities, data replication weights or importance's in DCs along with different accesses and data interval times. For this concept, the ITBDF weights the access to recent data replications and distinguishes previously accessed data. The ITBDF gives priority and weights to data replication access based on the PSO that is implemented in our system. The ITBDF has a concept called data replication that continuously analyzes DCs functions over time. It can be calculated through the following Eqs. (13) and (14):

$$ITBDF(t_a, t_b) = \mathrm{e}^{-(\mathrm{ta}-\mathrm{tb})\mathrm{k}} \quad (13)$$

$k \in \{1,2,3, \ldots \ldots\}$

$$ITBDF(t_a, t_b) = e^{-(\Delta t)k} \quad (14)$$

$where\ \Delta t = (t_a - t_b)$

Where:
t_a is current time.
t_b starts time.
k value is increase.
e exponential function decay.

$$RF_k = \frac{\sum_{t_i=t_b}^{t_a}(na_k(t_i,t_i+1).ITBDF(t_i,t_a))}{bnr_k . \sum_{i=1}^{nb_k} sb_i} \quad (15)$$

where:
t_a *is the current time*
t_b *is the start time*
na_k *is thenumber of accesses*
bnr_k *is the number of replicas*
nb_k *is the number of blocks*
sb_i *is the size of a block*

Algorithm 1. The Proposed MOPSO for Selecting Data Replicas in a Cloud Environment
Input: Size of Population
Number of Particles
Number of Iterations
Datacenters
Data Availability
Improved Time-Based Decay Function
Output: Selected Optimally Best Replica
Total Execution Time and Costs
Initialization:
Define Values of parameters, Size of Pop, Num of Iterations and Num of Particles;
Initialize set values of particle swarm (Num of Iterations and Num of Particles);
Initialize availability and unavailability probabilities;
Initialize best replica according to costs and time;
Repeat
For j=1 to number of particles;
$P_{velocity}$ ←Random velocity();
$P_{position}$ ←Random position();
$P_{best} \leftarrow P_{position}$
End for
For each data replication in DC do;
Calculate the ITBDF of the data replication;
Calculate the replica factor of the data replication;
Calculate the costs of the data replication;
End for
If $\alpha \leq 0$ then
Exploitation
Else
Exploration
Select best data replication;
End if
Until maximum number of iterations is reached or access solution found;
Return the optimal best replica solution;

3.2.3 The proposed placement of new replications using ACO

In this section, we discuss applying the suggested ACO algorithm on a wide scale to perform the placement in CloudSim. In this stage, replica management decides the new replica placements based on the replication costs and space of DCs. Thereafter, data replications are made available to users and data access occurs according to the suggested system. The ACO algorithm is used to optimally place replicas in DCs to meet user requests, which are created according to the suggested system.

ACO is the most important algorithm to find the best shortest and least expensive way. The total pheromones at positions on the different routes are measured using the objective function, and we also calculate the transition from DC_i to DC_j according to the following equations from Eqs. (16) to (21): [33].

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{s\in k}[\tau_{is}]^{\alpha}[\eta_{ij}]^{\beta}} \\ 0 \quad otherwise \quad if\ j \in allowed\ k \end{cases} \quad (16)$$

The calculation of the next DC that is selected by ant k is as follows:

$$j = \begin{cases} \arg\max_{s\in allowed\ k}\{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}\}, & \text{if } q \leq q_0, \text{if } q > q_0 \\ j \quad \text{otherwise} \end{cases} \quad (17)$$

The calculation of the detection array of the ant proceeds according to the following Eq. (18):

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (18)$$

The pheromone values on routes are updated after every repetition. When ants reach the end of their travel path, the pheromone value is a positive constant. The updated local pheromone value can be calculated as follows Eq. (19):

$$\tau_{ij} = (1-p)\tau_{ij} + p_{\tau 0}, \forall (i,j) \in t_k, where\ (0 < p \leq 1) \quad (19)$$

After evaporation, every ant adds pheromones to the routes according to the set method, and the updated global pheromone value is calculated Eq. (20) as follows:

$$\tau_{ij} = (1-p) + p.\sum_{k=1}^{m} \Delta_{\tau ij} \quad (20)$$

$\Delta_{\tau ij}$: is the amount of pheromone that is added by ant k on their route. It can be represented Eq. (21) as follows:

$$\Delta_{\tau ij} = \begin{cases} \frac{1}{c^k} & if\ \forall(i,j) \in t^k \\ 0 & otherwise \end{cases} \quad (21)$$

We find that the algorithm places and accesses optimal replicas based on geometric and Zipf distributions. The calculations of the fitness function for every MOACO of the replica positions through DCs in the Cloud occur as follows.

Algorithm 2. The Proposed MOACO for the Placement of Data Replicas in a Cloud Environment
Input: Number of Ants
Number of Iterations
Datacenters
Zipf, Geometric Distribution
Min Distance between Datacenters
Output: Selected Optimally Best Data Replica Placement
Total Execution Time and Costs

Initialization:
Define Values of parameters, Num of Iterations and Num of Ants;
Initialize availability and unavailability probabilities;
Initialize distance between DCs;
Initialize costs of data replication and size;
Initialize optimal best data replication placement in DC solution;
Repeat
For I=1 to (Num of ants);
Step = step + 1;
Set all ant distribution in DC;
End for
Repeat
For each DC in current system;
Calculate desirability of the movement;
Calculate probability of the movement;
End for
If $q \leq q_0$ then
Exploitation
Else
Exploration
End if
Set local pheromone update;
Set global pheromone update;
Set determine replica placement in DC;
Until all replicas are selected;
Until all replicas are placed;
End for
End for
If the storage space of the DC is insufficient;
Then
Apply the global update rule;
Else if
Delete small replica popularities;
End if
Until maximum number of iterations is reached or access solution is found;
Return the optimally best data replication placement in the DC;

3.2.4 Zipf and geometric distributions

Zipf and geometric distributions are used to select and place replicas through DCs in the Cloud. A distribution follows users' behaviors to determine the more popular data replications and places those data replications in DCs that are nearer to users. A Zipf distribution is used to randomly model the size and selects the replica locations according to their popularity and rapid user access:

$$p(f_i) = \frac{1}{i^a} \quad (22)$$

where i = 1, 2. . . N; and α is a factor determining the data access distribution, where $0 \leq \alpha < 1$.

A geometric distribution provides the most popular data that are more distinct with more access and higher weights. The data can be distributed through DCs and more random solutions can be discovered through the search space. Therefore, the most dispersed distribution is called the geometric distribution and it has a different formula:

$$p(i) = (1-p)^{i-1}.p \quad (23)$$

where i = 1, 2. . . n and $0 < p < 1$. A larger p represents that a smaller portion of the data are repeatedly accessed.

4. Experimental evaluation

4.1 Experiments of optimization

This section discusses the experimental results and configures the replication model in the suggested Cloud. Further, it places the replications by using the suggested PSO and ACO algorithms. These algorithms are performed on CloudSim. The execution time, costs, access speed, high replication availability and placement efficiency of this method are compared with those of other algorithms.

4.2 Configuration details

The Cloud is formed to simulate different kinds of DCs with different structures. Moreover, 21 DCs are used. The system configurations are shown in Table 1. Each DC consists of a host that contains a set of VMs that provides blocks of available data replications. We created three different data placements for high datacenters. A total of one thousand cloudlets are randomly confirmed for the data replication order. PSO and ACO algorithms parameters are shown in Tables 2 and 3. A comparison between related work and proposed model is shown in Table 4.

Table 1. Simulation parameters of the configuration system

NO	Proposed System	High DCs	Mid DCs	Low DCs
Datacenters (DCs)				
1	No. of DCs (21)	1	5	15
2	Cost of DCs	500	300	100
3	No. of hosts per DC (110)	20	30	60
Host				
4	How many processing elements per host?	12 - 16	4 - 8	1 - 4
5	How many MIPS per Processing Element?	1000 - 2000	500 - 1000	100 - 500
6	How much Bandwidth per Processing Element	5 – 10 GB	2 – 4 GB	1 – 2 GB
Virtual machines (VMs)				
7	Number of VMs (470)	200	150	120
8	MIPS	800	400	200
9	Memory ram	2 GB	1 GB	512 MB
10	Bandwidth	10 GB	2 GB	1 GB
11	No of processing elements	8 - 16	4 - 8	1 - 4
12	Cloudlet scheduler	time &space shard		
13	VM scheduler			
Cloudlet				
14	Cloudlet task	1000 task		
15	Length of task	1000 - 20000		
File				
Using zipf distribution and geometric distribution				
A three different data files (3 files) are placed in the Cloud storage environment, with each size in the range of [0.1, 10] GB.				
16	No of file (3)	A	B	C
17	Cost of replication	500	300	100
Users				
18	No. of users	10 – 50		

Table 2. PSO parameters

No.	parameters	values
1	*No. of particles*	50
2	C_1	2
3	C_2	2
4	R_1	[0 –1]
5	R_2	[0 –1]
6	w_{max}	0.9
7	w_{min}	0.4
8	*No. of iteration*	1000
9	W	1

Table 3. ACO parameters

No.	parameters	values
1	α	1
2	β	2
3	p	0.3
4	q	1
5	m	50
6	$p_{\tau 0}$	0.9

Table 4. Comparison between related work and my proposed

Strategy	Year	Availability	Load Balancing	Heterogeneity	Knapsack problem	Distance	Least Cost Path	Optimal Location	Path Length Tasks	Replica decision
Fuzzy-FP [34]	2016	√	√							
EFS [35]	2011	√	√							
DCR2S [13]	2016	√	√	√	√					
ACO [36]	2016	√	√							
GASA [37]	2016	√	√							
DRSACO [38]	2013	√	√							
Genetic [39]	2015	√	√							
My Strategy	2019	√	√	√	√	√	√	√	√	√

5. Results and discussion

This paper uses CloudSim to conduct the experiments on the dynamic selection and placement of replicas in the Cloud. It uses the MOPSO and MOACO algorithms and verifies the experimental results by comparing them with other algorithms.

5.1 The selecting the optimally best replica

Fig. 1 shows the influence of using the MOPSO, DCR2S and EFS on the replication costs as the users' number of tasks increases. It is obvious that our MOPSO method decreases the replication costs compared with the DCR2S and EFS. The MOPSO algorithm has been shown to select replicas by

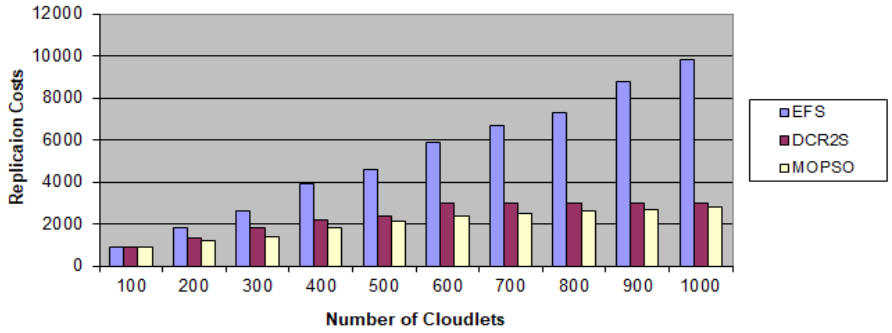


Figure. 1 Replication costs and number of cloudlets

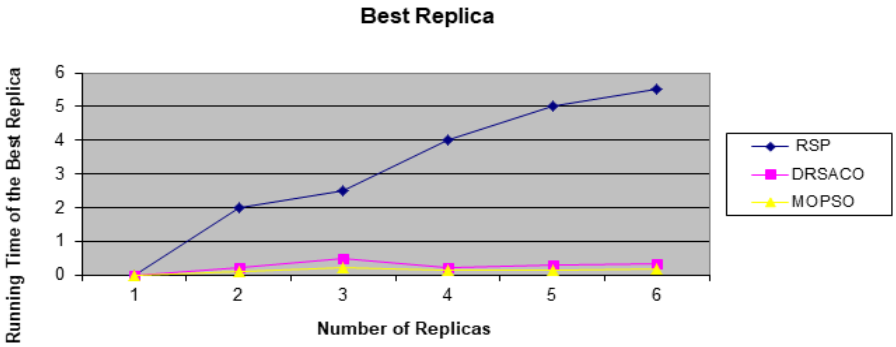


Figure. 2 Probability of data availability and no of replicas

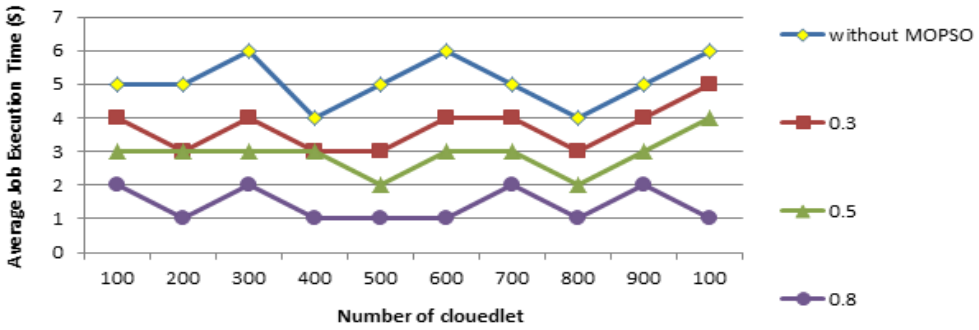


Figure. 3 Different number about Zipf and geo in cloudsim

changing the number of particles and the number of iterations, as shown in Table 3. The replication costs are very expensive for the EFS compared to the DCR2S, where the replication costs equal the budget. There are three gaps: "constant cost, optimize replica and waiting time for users". Hence, MOPSO is highly beneficial in optimizing the cost of replication as well as high availability and ITBDF. Finally, optimization of selecting replicas is performed through cloudlets.

In Fig. 2, the experimental results show the access ability with the most popular replica through the MOPSO algorithm to select optimal replicas that are based on ITBDF. These findings have therefore clarified that our algorithm is more effective in the execution time necessary to access optimal replicas than that of other algorithms.

5.2 Different number of replicas using the zipf and geo distributions:

In Fig. 3 shows the requesting the number of data replications, the average response time of the selection of data replication greatly increases Our MOPSO strategy shows a reduced average response time within the requested number of data replications by users.

5.3 The optimally best placement of replicas:

This experiment assesses the use of the MOACO algorithm to evaluate the placement of replicas through the ant's behavior and the number of cycles through DCs. To assess the optimal placements in

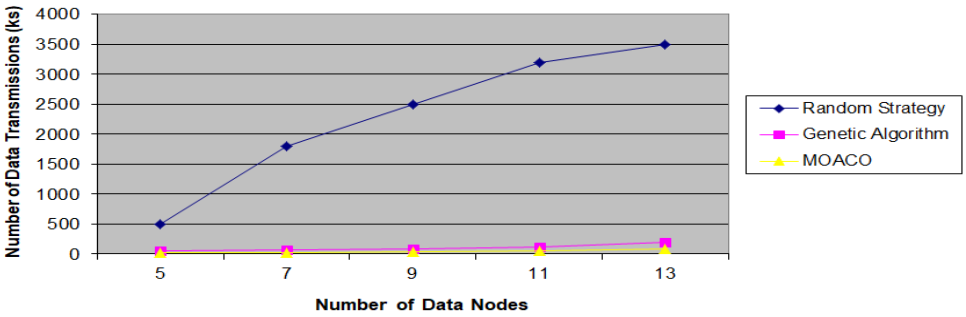


Figure. 4 The data transmissions with different data nodes

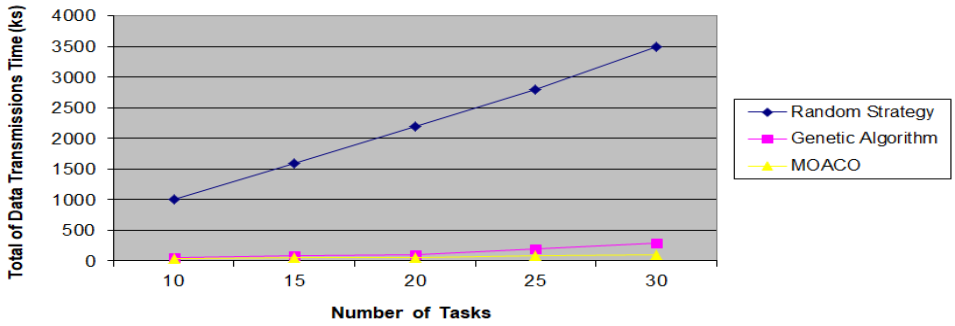


Figure. 5 Total data transmission and number of tasks

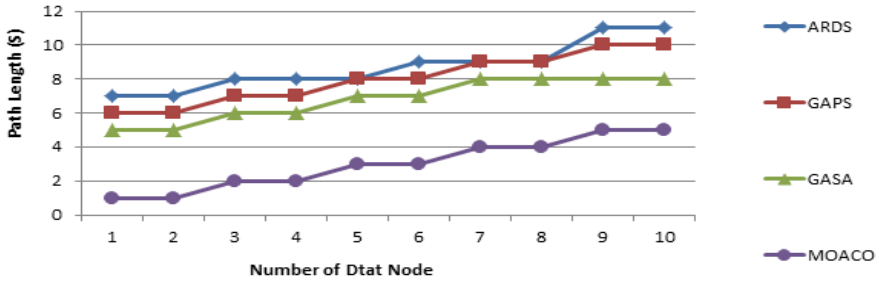
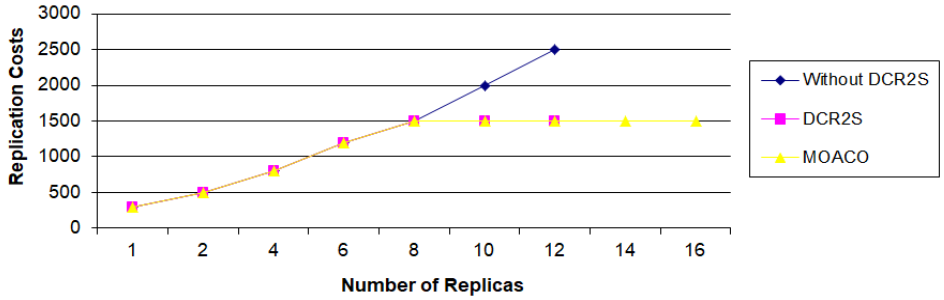


Figure. 6 Path length of data transmission between other strategies in DCs

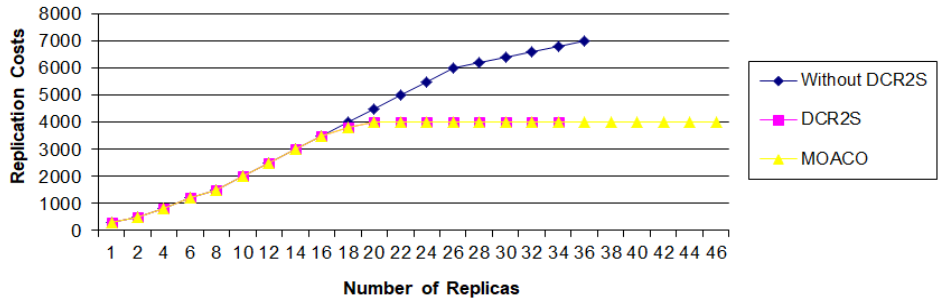
DCs, we use different distributions such as Zipf and geometric distributions, in CloudSim. The experiments show that our MOACO strategy provides the best results, and its results are compared with those of other algorithms. The suggested strategy is evaluated for many aspects, including the least-cost path and the distance to reach the optimal replica placement locations in DCs according to Zipf and geometric distributions. Several approaches are implemented to assess the transmission process in DCs such as the number of tasks and number nodes. In Figs. 4 and 5 MOACO is used to determine the number of replications through DCs and the data transmission rate that reduces time and costs. The results show that our strategy is superior to other algorithms. The comparison of the results of our strategy with the results of other algorithms shows that our strategy is superior to other algorithms with respect to the time and costs of optimizing the placement of replicas.

5.4 Different number of replicas using the Path length based on MOACO:

In this section, the calculation of the shortest path based on ACO with MOO is shown, and the new strategy approach to solve optimal placement is provided. Fig. 6 shows the comparison between the MOACO algorithm and GASA algorithm based on the shortest path problem, which affects the decreased shortest path of optimal data replication in nodes. Therefore, when requesting the number of data replications, the shortest path of data replications greatly increases. Our MOACO strategy has been shown to reduce the shortest path within the requested number of data replications by users. Regarding the comparison of the GASA algorithm in reducing the shortest path, we note that the MOACO algorithm is superior to the GASA algorithm.



(a)



(b)

Figure. 7 Different scenarios about cost of data replication with the number of replicas: (a) the first type of scenario cost replication and (b) the second type of scenario cost replication

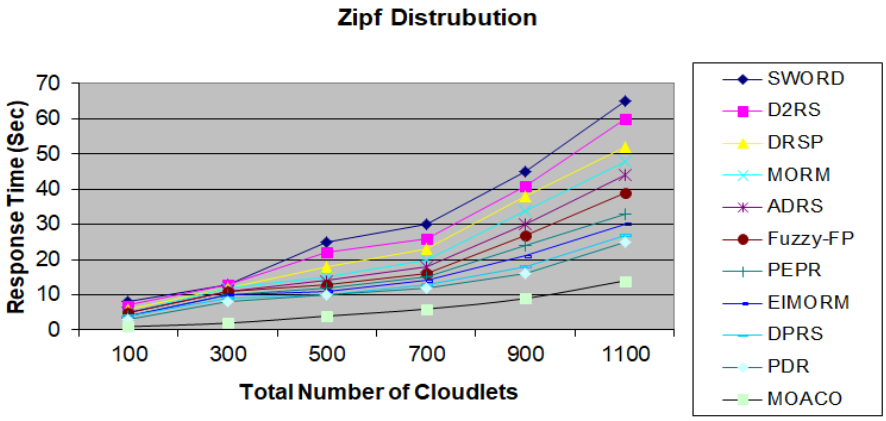


Figure. 8 Average response time for different data replication strategies in the cloud

5.5 Cost of replication and number of replicas using the knapsack problem

Fig. 7 shows the comparison between the MOACO algorithm and DCR2S algorithm based on the budget of replication with the knapsack problem, which affects the decreased budget of data replication. Therefore, when requesting the number of data replications, the costs of data replication greatly increase. There are many different scenarios from (a) and (b), and we have made different budget scenarios from 500 to 5000. Thus, the MOACO algorithm can enhance the budget of data file replication with the knapsack problem when users require the number of data replications.

5.6 Performance evaluation

5.6.1. Average response time

In Fig. 8 the average response times of the replication strategy using Zipf and geometric distributions are given. Through the experiments, we see that our suggested strategy reduces the average response time by 12% more than the familiar PDR algorithm. As the number of user tasks to determine and placement replications increases, the average response time rapidly increases but our strategy efficaciously reduces the time.

5.6.2. Effective network usage (ENU)

In Fig. 9 the effective network usage measures the data percentage that passes through a network in the domain on a scale from 0 to 1. It includes the number of suitable accesses, the response time and the replications to other places that are closer and have lower costs among DCs. It shows that the bandwidth use was more efficient and effective with our strategy. Further, the most familiar algorithm, the PDR, has an ENU that improves by approximately 25% through the Cloud.

5.6.3 Storage usage

In Fig. 10, the different storage use rates in our heterogeneous system assess the relative size of the data when transferring and storing data replications with respect to the DCs' size. Nevertheless, it is another measure of the costs and transfer time among DCs. By comparing our strategy with the most familiar algorithm, the PDR, the results show that our strategy is superior and that it provides the best percentage at approximately 20%. Moreover, it maintains the most popular data replication in near places.

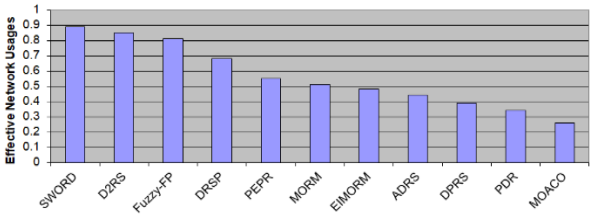


Figure. 9 Effective network usage for different data replication strategies in the cloud

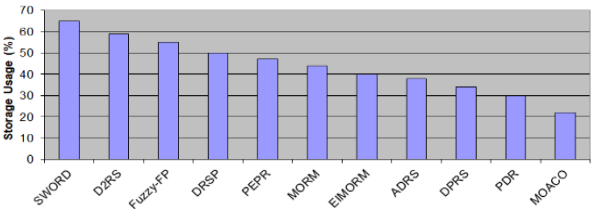


Figure. 10 Storage use rates for different data replication strategies in the cloud

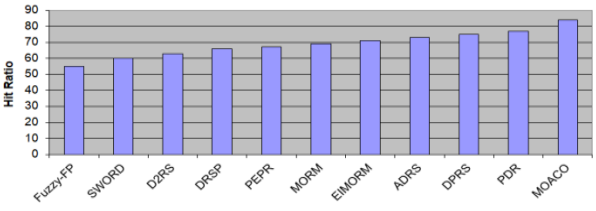


Figure. 11 Hit ratios for different data replication strategies in the cloud

5.6.4. Hit Ratio

Fig. 11 the hit ratio is represented as the percentage of accessible data that are either closer to or farther from tasks as distances between 1000 to 3000. Our strategy achieved a higher hit ratio compared with the most familiar PDR algorithm, which shows that our method can be accurately applied to model users' behavior when accessing closer and farther replications.

6. Conclusion

The Cloud environment is one of the most important scenarios that can be leveraged to achieve high availability and optimize the performance of replicas. This paper proposed two swarm intelligence algorithms (MOPSO and MOACO) for dynamic data replication and placement. MOPSO determines the optimal access to the most popular data replications and selects the optimal replication using ITBDF. MOACO is used to optimize the placement of data replicas, which was previously determined using MOPSO in suitable sites near users. The suggested system architecture was constructed and implemented using CloudSim. The performance of proposed model was compared with different replication algorithms such as, EFS, D2RS, ADRS, DRACO, and GA. The simulation results showed that the proposed algorithms were more efficient and better than the compared algorithms.

In future work, the suggested system architecture will be assessed using a real computing environment. Also, the knapsack problem will be improved to optimize costs, storage space, waiting times, data availability, performance and access speed for data replications through the cloud.

Conflicts of Interest

The authors declare no conflict of interest about this research.

Author Contributions

Conceptualization: Ahmed Awad, Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam. Methodology: Ahmed Awad, Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam. Software: Ahmed Awad, Mustafa Abdul Salam. Validation: Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam. Formal Analysis: Ahmed Awad. Investigation: Ahmed Awad. Resources: Ahmed Awad, Mustafa Abdul Salam. Data Curation: Ahmed Awad, Mustafa Abdul Salam. Writing - Original Draft: Ahmed Awad. Writing - Review & Editing:

Ahmed awad, Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam. Visualization: Ahmed Awad, Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam. Supervision: Rashed Salem, Hatem Abdelkader, Mustafa Abdul Salam.

References

[1] R. Salem, M. Abdul-Salam, H. Abdel-kader, and A. Awad, "Anartificial bee colony algorithm for data replication optimization in cloud environments", *IEEE Access*, Vol. 8, pp. 51841–51852, 2020.

[2] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", *Future Generation Computer Systems*, Vol. 25, No. 6, pp. 599-616, 2009.

[3] R. Calheiros, R. Ranjan, A. Beloglazov, C. 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.

[4] N. Rajeshirke, R. Sawant, S. Sawant, and H. Shaikh, "Load Balancing in Cloud Computing", *International Journal of Recent Trends in Engineering & Research*, Vol. 3, No. 3, pp. 260–267, 2017.

[5] E. Ghomi, A. Rahmani, and N. Qader, "Load-balancing Algorithms in Cloud Computing: A Survey", *Journal of Network and Computer Applications*, Vol. 17, pp. 1–62, 2017.

[6] H. Ahn, K. Lee, and Y. Lee, "Dynamic erasure coding decision for modern block-oriented distributed storage systems", *J Supercomput*, Vol. 72, No. 7, pp. 1312–1341, 2016.

[7] N. Maheshwari, R. Nanduri, and V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework", *Future Generation Computer Systems*, Vol. 28, pp. 119–127,2014.

[8] B. Milani and N. Navimipour, "A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions", *Journal of Network and Computer Applications*, Vol. 64, pp. 229–238, 2016.

[9] Z. Fadaie and A. Rahmani, "A new Replica Placement Algorithm in Data Grid", *International Journal of Computer Science Issues*, Vol. 9, No. 3, pp. 491-507, 2012.

[10] A. Rajalakshmi, D. Vijayakumar, and K. Srinivasagan, "An Improved Dynamic Data Replica Selection and Placement in Hybrid Cloud", *International Journal of Innovative Research in Science, Engineering and Technology*, Vol. 3, No. 3, pp. 2187-2192, 2014.

[11] N. Mansouri, M. Rafsanjani, and M. Javidi, "DPRS: A dynamic popularity aware replication strategy with parallel download scheme in cloud environments", *Simulation Modelling Practice and Theory*, Vol. 77, pp. 177–196, 2017.

[12] D. Sun, G. Chang, S. Gao, L. Jin, and X. Wang, "Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments", *Journal of Computer Science and Technology*, Vol. 27, No. 2, pp. 256 – 272, 2012.

[13] N. Gill and S. Singh, "A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers", *Future Generation Computer Systems*, Vol. 65, pp. 10–32, 2016.

[14] N. MANSOURI, "Adaptive data replication strategy in cloud computing for performance improvement", *Front. Comput. Sci.*, Vol. 6, pp. 1-11, 2015.

[15] K. Kumar, A. Quamar, A. Deshpande, and S. Khuller, "SWORD: workload-aware data placement and replica selection for cloud data management systems", *The VLDB Journal*, Vol. 1, pp. 1-26, 2014.

[16] X. Bai, H. Jin, X. Liao, X. Shi, and Z. Shao, "RTRM: A Response Time-Based Replica Managementm Strategy for Cloud Storage System", In: *Proc. of International Conf. on Grid and Pervasive Computing*, Wuhan, China, pp.124-133, 2013.

[17] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows", *Future Generation Computer Systems*, Vol. 26, pp. 1200-1214, 2010.

[18] C. Hamdeni, T. Hamrouni, and F. Charrada, "Adaptive measurement method for data popularity in distributed systems", *Cluster Comput*, Vol. 19, pp. 1801-1818, 2016.

[19] N. Maheshwari, R. Nanduri, and V. Varma, "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework", *Future Generation Computer Systems*, Vol. 28, pp. 119– 127, 2012.

[20] Q. Xu, Z. Xu, and T. Wang, "A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing", *International Journal of Intelligence Science*, Vol. 5, pp. 145-157, 2015.

[21] Z. Li, W. Cai, and S. Turner, "Un-identical federate replication structure for improving performance of HLA-based simulations",

Simulation Modelling Practice and Theory, Vol. 48, pp. 112–128, 2014.

[22] Z. Tang, X. Zhang, and K. Li, "An intermediate data placement algorithm for load balancing in Spark computing environment", *Future Generation Computer Systems*, Vol. 78, No. 1, pp. 287-301, 2018.

[23] T. Hsu, A. Kshemkalyani, and M. Shen, "Causal consistency algorithms for partially replicated and fully replicated Systems", *Future Generation Computer Systems*, Vol. 86, pp. 1118-1133, 2018.

[24] H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni, "On the impact of process replication on executions of large-scale parallel applications with coordinated checkpointing", *Future Generation Computer Systems*, Vol. 51, pp. 7–19, 2015.

[25] P. Matri, M. S. Perez, A. Costan, L. Bouge, and G. Antoniu, "Keeping up with Storage: Decentralized, write-enabled dynamic geo-replication", *Future Generation Computer Systems*, Vol. 86, pp. 1093-1105, 2017.

[26] M. Lee, F. Leu, and Y. Chen, "PFRF: An adaptive data replication algorithm based on star topology data grids", *Future Generation Computer Systems*, Vol. 28, pp. 1045–1057, 2012.

[27] O. Hegazy, O. Soliman, and M. Abdul-Salam, "A Machine Learning Model for Stock Market Prediction", *International Journal of Computer Science and Telecommunications*, Vol. 4, No. 12, pp. 17-23, 2013.

[28] O. Hegazy, O. Soliman, and M. Abdul-Salam, "Comparative Study between FPA, BA, MCS, ABC, and PSO Algorithms in Training and Optimizing of LS-SVM for Stock Market Prediction", *International Journal of Advanced Computer Research*, Vol. 5, No. 18, pp. 35-45, 2015.

[29] O. Hegazy, O. Soliman, and M. Abdul-Salam, "LSSVM-ABC Algorithm for Stock Price prediction", *International Journal of Computer Trends and Technology*, Vol.7, No. 2, pp. 81-92, 2014.

[30] O. Hegazy, O. Soliman, and M. Abdul-Salam, "Optimizing LS-SVM using Modified Cuckoo Search Algorithm (MCS) for Stock Price Prediction", *International Journal of Advanced Computer Research in Computer Science and Management Studies*, Vol. 3, No. 2, pp. 204-224, 2015.

[31] O. Hegazy, O. Soliman, and M. Abdul-Salam, "FPA-ELM Model for Stock Market Prediction", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 5, No. 2, pp. 1050-1063, 2014.

[32] K. Kavitha and R. Neela, "Optimal allocation of multi-type FACTS devices and its effect in enhancing system security using BBO, WIPSO & PSO", *Journal of Electrical Systems and Information Technology*, Vol. 5, No. 3, pp. 777–793, 2018.

[33] Y. Moon, H. Yu, J. Gil, and J. Lim, "A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments", Human-centric Computing and Information Sciences, Vol. 7, No. 28, pp. 1-10, 2017.

[34] P. Elango and D. Kuppusamy, "Fuzzy FP-Tree based Data Replication Management System in Cloud", *International Journal of Engineering Trends and Technology*, Vol. 36, No. 9, pp. 481-489, 2016.

[35] M. Bsoul, A. Al-Khasawneh, E. Abdallah, and Y. fKilani, "Enhanced Fast Spread Replication strategy for Data Grid", *Journal of Network and Computer Applications*, Vol. 34, pp. 575–580, 2011.

[36] N. Navimipour and B. Milani, "Replica selection in the cloud environments using an ant colony algorithm", In: *Proc. of International Conf. on Control Engineering and Communication Technology*, Moscow, Russia, pp. 1-9, 2016.

[37] T. Junfeng and L. Weiping, "Pheromone-Based Genetic Algorithm Adaptive Selection Algorithm in Cloud Storage", *International Journal of Grid and Distributed Computing*, Vol. 9, No. 6, pp. 269-278, 2016.

[38] L. Wang, J. Luo, J. Shen, and F. Dong, "Cost and time aware ant colony algorithm for data replica in alpha magnetic spectrometer experiment", In: *Proc. of International Conf. on Big Data*, Santa Clara, USA, Vol. 1, pp. 247-254,2013.

[39] L. Cui, J. Zhang, L. Yue, Y. Shi, H. Li, and D. Yuan, "A Genetic Algorithm Based Data Replica Placement Strategy for Scientific Applications in Clouds", *IEEE Transaction on Services Computing*, Vol. 11, pp. 727-739, 2018.