symmetry

MDPI

Article

A Hybrid Heuristic Algorithm Using Artificial Agents for Data Replication Problem in Distributed Systems

Bahman Arasteh [1], Seyed Salar Sefati [2], Simona Halunga [2,*], Octavian Fratu [2] and Tofigh Allahviranloo [1]

[1] Department of Software Engineering, Faculty of Engineering and Natural Science, Istinye University, Istanbul 34460, Turkey
[2] Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest, 060042 București, Romania
* Correspondence: simona.halunga@upb.ro

Abstract: One of the key issues with large distributed systems, such as IoT platforms, is gaining timely access to data objects. As a result, decreasing the operation time of reading and writing data in distributed communication systems become essential demands for asymmetric system. A common method is to replicate the data objects across multiple servers. Replica placement, which can be performed statically or dynamically, is critical to the effectiveness of distributed systems in general. Replication and placing them on the best available data servers in an optimal manner is an NP-complete optimization problem. As a result, several heuristic strategies for replica placement in distributed systems have been presented. The primary goals of this research are to reduce the cost of data access time, reduce the number of replicas, and increase the reliability of the algorithms for placing replicas. In this paper, a discretized heuristic algorithm with artificial individuals and a hybrid imitation method were developed. In the proposed method, particle and gray-wolf-based individuals use a local memory and velocity to search for optimal solutions. The proposed method includes symmetry in both local and global searches. Another contribution of this research is the development of the proposed optimization algorithm for solving the data object replication problem in distributed systems. Regarding the results of simulations on the standard benchmark, the suggested method gives a 35% reduction in data access time with about six replicates. Furthermore, the standard deviation among the results obtained by the proposed method is about 0.015 which is lower than the other methods in the same experiments; hence, the method is more stable than the previous methods during different executions.

Keywords: distributed systems; replica placement; data access time; artificial gray wolf optimization; local best; stability

Citation: Arasteh, B.; Sefati, S.S.; Halunga, S.; Fratu, O.; Allahviranloo, T. A Hybrid Heuristic Algorithm Using Artificial Agents for Data Replication Problem in Distributed Systems. *Symmetry* **2023**, *15*, 487. https://doi.org/10.3390/sym15020487

Academic Editor: Gabriel Ciobanu

Received: 5 January 2023
Revised: 22 January 2023
Accepted: 31 January 2023
Published: 12 February 2023

1. Introduction

Quick access to data objects is a major issue when building large, dispersed systems, such as cloud computing. In a distributed system, retrieving data takes a longer time. Therefore, reducing the operation time of reading and writing data in distributed systems has become a key concern for the designers of these systems. The problem can be solved by using multiple copies of the data on several servers, which will enable quicker access to the data from faraway locations. Replication in this usage refers to the practice of making identical copies of the same data on numerous servers. In general, the placement of replicas, which can be carried out statically or dynamically, is essential to the efficiency of distributed systems.

One of the largest problems with distributed systems is reducing the amount of time needed to access data objects. The primary goals are to reduce costs and speed-up access to data services. The cloud computing system is an example of a distributed system, which consists of several independent computers that users can access as one big system. Data replication is one successful technique to alleviate the data access time in large distributed

systems. The main objectives of data replication in distributed systems are to increase the dependability and efficiency of the system. The quantity of copies (replicas) that are stored on servers has a crucial role in replica placement. Determining the number and location of data replicas in distributed systems is one of the most challenging problems (replica-placement problem). In general, it is best to choose the number of replicas of data objects among many servers in a way that lowers the overall cost of operations. The generation of a minimum number of data replicas, their placement on efficient servers that are easily available, and lowering the system's overall cost of data processing are the study's research objectives. As a result, several heuristic methods for efficiently placing replicas in distributed systems have been presented. Inadequate performance of existing methods in large distributed systems such as cloud and IoT systems, the need for more data servers and, as a result, more memory consumption, inadequate dependability and reliability of the obtained results by the previous methods, and falling in the local optima are the main demerits of the previous methods.

In this study, a discrete and swarm-based technique that combines modified particle swarm optimizer and gray wolf optimizer algorithms (PSGWA) was suggested to handle the problem of data-replica placement and replacement in distributed systems. In the PSGWA, each individual (agent) was implemented as a particle-wolf with the local and global best memory. The local search was implemented by the gray wolf optimization algorithm (GWO or GWA), and the global search was implemented based on the PSO algorithm's imitation. The individuals were divided into a few subgroups of wolves. Each subgroup searches the best solution in the related solution spaces. Each wolf has a local best memory. The best Alpha is selected as the global best of the total population. After combining the subgroups into the main population, each wolf behaves as a particle and imitates the global best individual using its velocity, local best, and position. Each solution explains how to organize data that is replicated over several servers, with the algorithm's main goal being to reduce the cost of data access activities. When compared to other methods now in use, the PSGWA method can reduce the cost of the replica-placement problem due to the algorithm's performance in static scenarios. The principal contributions of this method are as follows:

- In the proposed method, artificial individuals (agents) were developed in such a way that each individual includes local memory to save the visited local best location and velocity; it behaves like a particle (in the PSO) and a gray wolf at the same time. The artificial individuals are divided into subgroups, and each subgroup searches in a separate part of the overall solution space using the GWO imitation strategy. After local searches, the subgroups merge into one population. The best Alpha is selected as the global best of the created main population.
- In the suggested method, the global search is performed using the velocity, local best, and global best of the individuals. The global search was implemented using the PSO imitation strategy. The location and velocity of each individual were imitated using the global best, similar to the particle swarm algorithm. The introduced discrete swarm-based algorithm can be used to solve the discrete optimization problem efficiently.
- Another contribution of this research is the development of the proposed discrete and hybrid algorithm (PSGWA) for solving the data object replication problem in distributed systems. Data replicas should be stored on servers that reduce data access time.

The paper is structured as follows. The second section covers the fundamental definitions, and summaries of earlier studies about the replica-placement problem. The proposed replica-placement method is explained and provided in the third section. In the fourth section, the findings of the suggested method will be assessed with other results in terms of data access time, number of produced replicas, and stability of results.

2. Related Work

Replicas of a data object are extra copies of the data object located on several servers, enabling faster remote access to the data. The technique of making exact duplicates of the same data on other servers is known as replica placement. In general, the placement of replicas, which can be carried out statically or dynamically, is essential to the efficiency of distributed systems. The replica-placement issue has been addressed by several techniques in recent years, each of which has its own specific structure, advantages, and disadvantages. Algorithms such as random, greedy, tree-based, hot spot, and hot region algorithms are the main replica-placement techniques in this field. This kind of algorithm distributes M copies of data items among N servers at random while considering the majority of client workload. This method distributes the data according to the normal distribution. To achieve the desired output, this method should be repeated numerous times, with the best result being selected. The ease with which this method can be put into action is one of its main advantages, but the randomness of the algorithm makes it difficult to achieve the best and most desired results [1].

Greedy Algorithms: The greedy placement algorithm selects a replica version and the optimal server for the replica placement after a comparison of all N servers. After examining the expenses associated with each customer's access to this replica version, the service provider with the lowest cost is selected to supply the replica version. In the subsequent rounds, a similar process is used for the remaining versions. In the case of discussing data access, each client is taken to be using the most recent replica. This method's key benefit is its fast algorithm execution while its main drawback is that it can be challenging to discover the best data distribution strategy [1].

Tree-based Placement Algorithm: Network connectivity is likely to take the form of a tree in distributed systems. Dynamic programming is used to implement this algorithm. This method was initially applied to the placement of the web proxy, but it may also be applied in the future to the placement of the replica in distributed systems. A communication interface between each of the smaller sub-trees that make up this technique's greatest level and the web server is provided for each one. Each client submits a request to the associated interface, which then sends the request to the web server to communicate with it [2].

Hot Spot-based Placement Algorithm: The third strategy seeks to place several data versions close to customers with high workloads. This approach divides data among service providers according to the volume of work that their customers produce; M replicas are supplied to the M service providers who are surrounded by lots of workload. The neighborhood radius around each service provider is employed in this technique to estimate the volume of work traffic [1]. In [3,4], a popular area-based placement technique was published. The major goal of this algorithm is to accelerate placement processes. Using this method, the network is initially split into numerous sections based on node delays; each area is referred to as a cell. Then, each cell is sorted using a radix-type method. It then selects k initial cells to act as placement agent servers. In this algorithm, the network is divided and mapped to Euclidean space using the GNP algorithm. After determining the typical node delay, the GNP algorithm transfers the network space to Euclidean coordinates. The hot area algorithm selects the regions with the largest density of nodes after calculating the node density for each coordinate of each network area.

Scaling and Work-load Aware Replica Management: This approach examined the automatic scaling and dynamic replica-placement mechanisms. The automatic scaling methodology based on service overhead is first recommended to reduce the overall cost of scaling replicas and other resources. A hybrid load forecasting approach is used to predict the workload before allocating resources. The overall cost calculation is generated based on the workload. The optimization problem could be seen as a linear programming problem when the constraints are considered. The best scaling method is then chosen using the Tabu algorithm. The dynamic replica placement's normal processing time is decreased by using the suggested data placement technique. In comparison to the related

strategy, the suggested replica-placement approach provides more equally distributed storage capacity [5].

Flexible Replica-Placement Method: This work presents an adaptive replica-placement technique in an edge computing environment to evaluate the link between user access characteristics, the number of replicas, and the position of the replicas. To determine the ideal position for replicas, this study devised the replica-placement method. This would increase data accessibility and boost cloud storage performance. The technique in this study offers the dynamic replica creation algorithm (DRC-GM), which uses the data block as the unit of data to solve the erratic nature of user access. To meet the criteria for data availability, DRC-GM continuously updates the number of copies to take into account the relationship between the frequency of data access and the number of replicas. The results of the experiments demonstrate that the DRC-GM and RP-FNSG algorithms, when applied in an edge computing environment, can significantly enhance system performance in terms of better prediction accuracy, quicker access response times, higher effective network, and storage space usage, and increased data availability [6].

User-Experience-based Replica-Placement Method: It is recommended to utilize a dynamic replica allocation technique to improve user experience and save storage expenses. The correctness and consistency of the data are further ensured using the replica consistency preservation technique. The suggested resource management method may significantly lower the total cost of the rented nodes and enhance CPU utilization as the length of time increases. For instance, the suggested method's overall cost can be reduced by up to 32.27% and 53.65%, in comparison to the earlier techniques. The suggested replica allocation technique may dramatically reduce both storage overhead and data transmission delay [7].

Replica Placement using Genetic Algorithm: This method uses the genetic algorithm (GA) to make the best placement for the created replicas. The two inputs used in this method are the Euclidean coordinates of the network and the required quantity of placement servers. The first phase computes the Euclidean coordinates of the network, and the second phase computes the density of each area, which includes the number of nodes in the target area. After each region has been aggregated, the center of each region is selected to act as the core of the cluster. The compatibility function for each cluster is then used to compute the density in the subsequent phase. Two clusters are always picked out of the group of good clusters, depending on the magnitude of the density, and the intersection operation is performed. The final stage, which entails a random leap on the new coordinates, then replaces the old coordinates with the new coordinates. This process is continued until there are as many excellent clusters as there are placement servers required [8].

Consistency-based Replica Placement: This method aims to boost the availability by sufficiently increasing the existing replicas. The two goals of reducing the distance between dissimilar exchanges and lengthening the distance between comparable exchanges should typically be achieved. Using Dijkstra's approach, the separation between two sites can be determined by measuring the bandwidth of the shortest link between them. While the plan is put into practice, the network is fixed. The method classifies the websites by color after compiling a list of all the files and mods on the network. The activity results in a group of sites that contain an original file (data object) and zero to many copies (replicas) of the original file. The method is based on selecting the file that is closest to each color, then selecting the file that is farthest from all of the nearby colored files. The most distant file from the closest one is then selected and replaced [9].

Constraint-based Replica Placement: In [10] a hybrid algorithm using the sealion optimization model (SLnO) and grey wolf optimization (GWO) algorithms was proposed for the data replication problem. Apart from this, the mining is carried out under the defining dual constraints of (i) prioritization and (ii) cost. The prioritization falls under two cases, queuing both high- and low-priority data, and the cost relies on the evaluation of storage demand. The high-priority queues are optimized with the GUEES model. Finally, a comparative validation is carried out to validate the efficiency of the adopted model. The network usage is considered as the main performance criteria of this method. The network

usage of the proposed model is 29.23%, 24.57%, 16.8%, and 16.85% higher than the existing methods such as the GWO, SLnO, PSO, and HCS, respectively.

Workflow-based data replication method: In [11] a replication management system which includes dynamic replication creator, a specialized cost-effective scheduler for data placement, a system watcher, and some data security tools for collaborative edge and cloud computing systems were proposed. Considering task dependency, data reliability and sharing, the data scheduling for the workflows is modeled as an integer programming problem. In this study, a faster meta-heuristic algorithm has been proposed to solve it. The experimental results show that the algorithms can achieve much better system performance than comparative traditional methods, and they can create a suitable number of data copies and search for the higher quality replica-placement solution while reducing the total data access costs under the deadline constraint.

Hardware-Efficient Data Replication Method: In [12] a method was proposed to meet the requirements of low hardware overhead and high-performance using hardware-efficient dual-source data replication and local broadcast mechanism (DRLB). DRLB alleviates the inherent limitations of previous data replication mechanisms and reduces the overhead of coherence protocols. The results of experiments on DRLB indicate that it can reduce the execution time by 16%, but only causes 0.82% of extra meta storage overhead, which also outperforms previous state-of-the-art data replication mechanism. The optimized version of DRLB can reduce the execution time by an average of 23%. Table 1 indicates the merits and demerits of the related methods.

Table 1. Overview of the related methods.

Technique	Merits	Demerits
Flexible Replica Placement [6]	Lower response time, higher data availability.	Data-file types and node types have not been considered.
Experience-based Replica Placement [7]	Lower financial cost, lower storage cost and higher performance in large distributed systems.	collaborative resource management has not been considered.
Replica placement using GA [8]	Lower run time and optimal placement.	Lower performance in large distributed systems.
Consistency-based Replica Placement [9]	Higher data availability and lower response time.	Suitable just for the network that is fixed during the execution of the technique.
Constraint-based Replica Placement [10]	Lower response time and memory consumption.	Need more internet bandwidth
Workflow-based data replication method [11]	Lower data access costs under the deadline constraint.	Data and computational scheduler was not considered uniformly.
Priority-based replica management [13]	Lower average response time and fault tolerance capability.	Suitable just for static systems.
PSO and fuzzy-based replica-placement algorithm [14]	Higher data availability and lower response time.	Lower performance for the data writes transaction.
Replica placement with service and content delivery networks [15]	Higher stability.	Local optimum probability.
Correlated data-replicas placement [16]	Lower response time.	Lower performance for the data writes transaction.

3. Proposed Method

In a distributed system, retrieving data may take longer and longer. Because of this, it is now particularly challenging to reduce the amount of time needed for reading and writing data objects in distributed systems. A typical solution to the issue is to use many copies (replicas) of the data on many servers to make it simpler to access from the specific distance. In this work, the PSO and GWO algorithms were combined as a new algorithm (PSGWA) to address the replica replacement issue. The use of the PSGWA as a swarm-based method can reduce data access in static environments as compared to other methods.

3.1. Problem Specification

In a distributed system, data objects need to be replicated on N servers to alleviate the data access time. In this problem, the server number is S(n), and the data object number is Object(k). Additionally, $1 < k \leq K$ and $1 < n \leq N$, where C(n) and V(k) are the corresponding capacities of service provider number n and data object number k. The cost of communication between the service providers S(n) and S(m) will be denoted by the integer l(nm). The cost of data transport between S(n) and S(m) servers is the same as the

cost of communication. It is assumed in this issue that l(nm) = l (mon). Additionally, the number of read and write requests for the Object(k) from S(n) servers is indicated by the variables read(n, k) and write(n, k). Each data object has a primary server as P(k). P(k) is in possession of the Object's initial version (k). It should be noted that Object(k) cannot be transferred to another server in its original form. The servers where Object(k) are replicated are listed for each primary server in a list referred to as RS(k). The nearest server that has the original or replica versions of Object(k) should be chosen for the read operation if server S(n) wants to execute the read command on the data Object(k). To update the Object during the write process, the relevant server sends the main server a data update request (k). In this scenario, P(k) broadcasts a message to all servers hosting Object(k), and each server updates the appropriate data object. The fundamental objective of the placement challenge is to distribute the replicas across the servers to reduce the overall cost of reading and writing operations and increasing the availability of the data objects.

In the first step of the suggested approach, the replica-placement problem can be translated to the common traveling salesperson problem (TSP). For instance, PSGWA was used to evaluate all methods for replicating two data objects over six different servers when there are two data objects and six servers. In this inquiry, each data object has a unique identification number (Kn indicates the unique id of n^{th} data object). There are four separate ways to replicate two data items (k1 and k2) on a server (S). Figure 1a displays configurations for replication of two data objects over six different data servers. The different modes are detailed below.

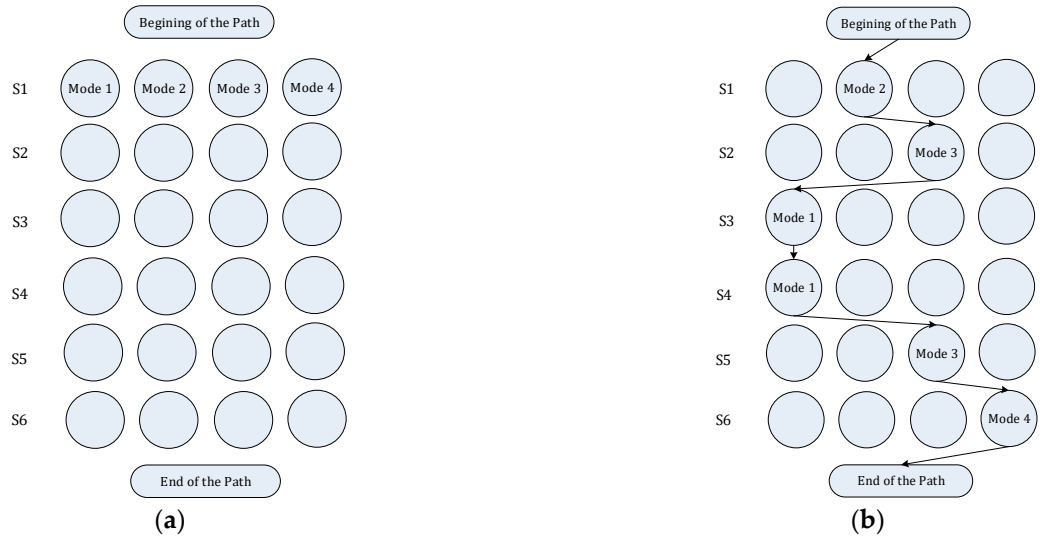


Figure 1. Mapping the search space of the replica-placement problem to the search space of the TSP problem. (**a**) The possible replication models of two data object on six different data servers. (**b**) An example of replica-placement path for replicating two data objects in six servers.

First mode: k1 and k2 are not replicated on S.
Second mode: k1 is replicated on S and k2 is not replicated on S.
Third mode: k1 is not replicated on S and k2 is replicated on S.
Fourth mode: k1 and k2 are replicated on S.

The columns in the resulting graph stand in for the various server modes, while each row in the graph corresponds to a server. Each solution path serves as a model for where the replicas of the data objects should be stored. The fitness value of each path in the graph depends on the objective function. The ideal path is the shortest one (in terms of fitness function), just like the TSP. Figure 1b depicts the replica-placement process for replicating two data items across six servers. Figure 1b shows a placement model of two data object copies on six servers for the specified path ([2,3,1,1,3,4]). The following interpretation can be made of the path shown in Figure 1b:

Server 1 holds just the replica of the k1 data object.
Server 2 holds just the replica of the k2 data object.
Server 3 does not hold any replica.
Server 4 does not hold any replica.
Server 5 holds just the replica of the k2 data object.
Server 6 holds the replicas of the k1 and k2 data objects.

Indeed, the replica-placement problem finally mapped into the TSP problem. Figure 2 shows the final from of the replication graph shown on Figure 1b.

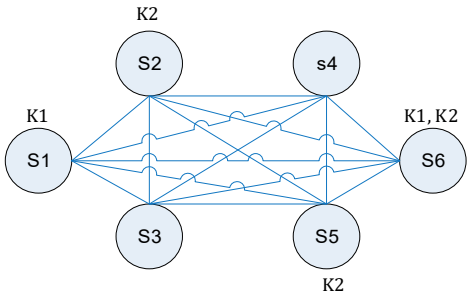


Figure 2. The replica placement graph.

3.2. Replica Placement Using PSGWA

In this study, a hybrid heuristic approach was introduced for solving the replica-placement problem. The PSGWA is implemented as a discrete swarm algorithm to sort out the replica-placement problem. The proposed PSGWA algorithm exploits the classic GWO imitation as its local search and PSO imitation as the global search method. PSGWA is a group-based heuristic algorithm that uses GWO [17] and PSO [18] at the same time. In the PSGWA, each individual has wolf and particle features at the same time. Each wolf has a limited memory for storing the visited local best position. Also, a wolf (individual) has the velocity attribute. In PSGWA, the velocity and local best information are used for global optimization. An intelligent agent (such as a wolf) can use sensors to understand its surroundings and effects to change the environment. The proposed solution to the replica-placement challenge makes up the wolf population. In the replica-placement problem, the structure of a wolf is shown in Figure 3. Each cell's information in a wolf array reveals the server's mode for that instance. The server index can be determined from the wolf array's index. Each wolf in the population does, in fact, display a replication path. The path depicted in picture 1.b is the same path depicted by the wolf array in Figure 3.

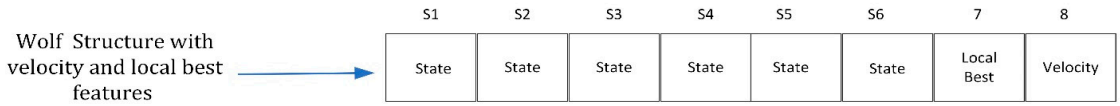


Figure 3. Structure of a wolf in the PSGWA when there are six data servers.

The replication array is equivalent to the wolf array. The word "food" in the approach refers to a wolf's fitness, and the wolf's fitness (replication array) denotes its proximity to the food. The overall cost of the access operation associated with the copies of the data objects stored on several servers shows whether the replication method is appropriate. Any replacement of a replica in the servers will have an impact on the cost and time of data access. A best wolf means that the replica-placement mode has a shorter access time. Both local and global searches are part of the PSGWA. The workflow of the suggested PSGWA algorithm is seen in Figure 4. In the initial population, each wolf array is produced at random. The fitness of each replication array is then assessed using the fitness cost function at the following stage. The population is separated into m subgroups after sorting the replication array according to their fitness.

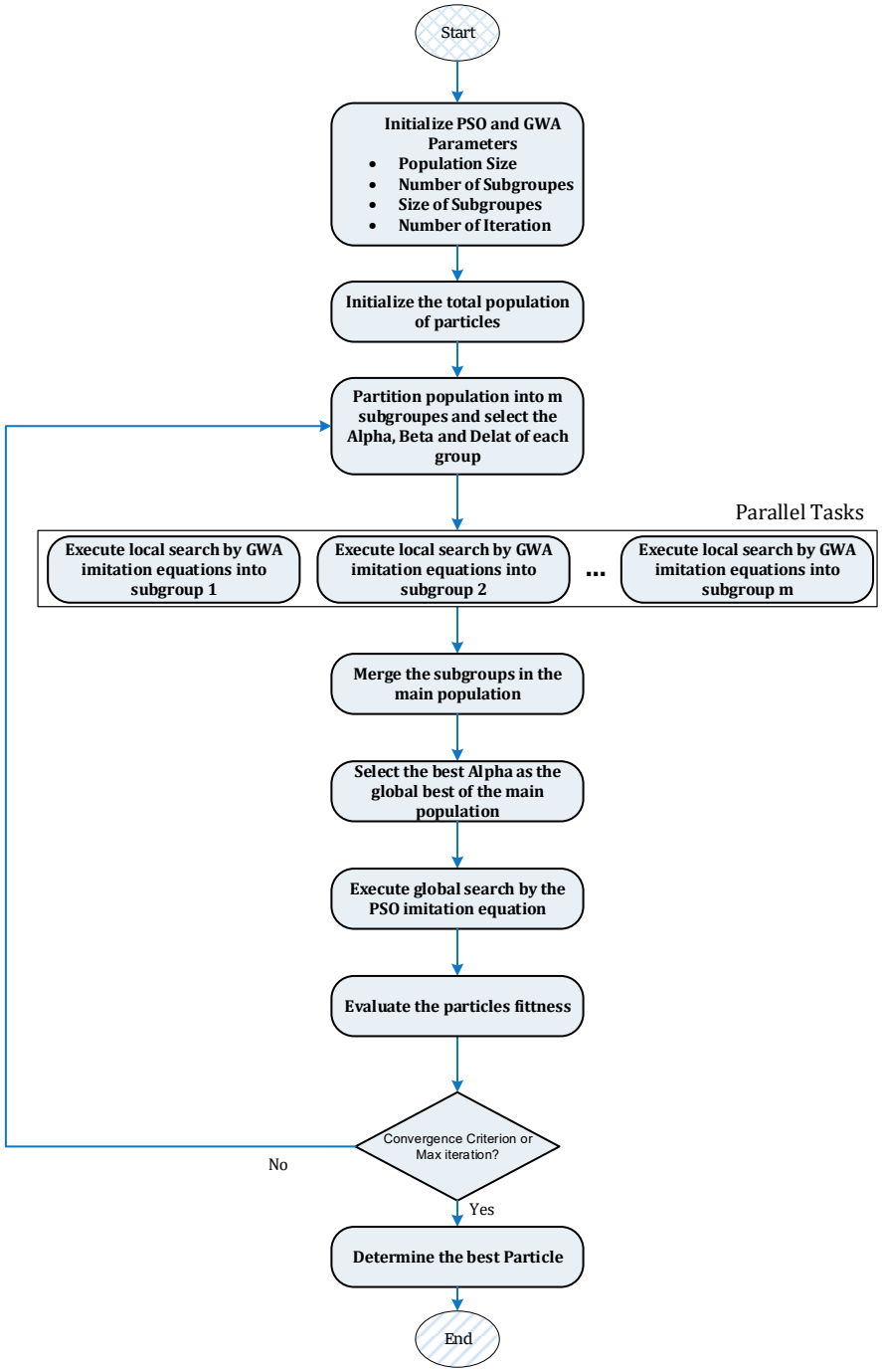


Figure 4. The workflow of the proposed PSGWA replica-placement method.

1. Initialize population size, iteration, number of subgroups, and size of subgroups;
2. Initialize the artificial search-agents with particle and gray wolf features;
3. itr = 0;
4. While (itr < iteration)
5. {
6. Partition the particle population into m subgroups;
7. Select the Alpha, Beta, and Delta of each subgroup;
8. Perform GWO local imitation in each subgroup in parallel using Equations (1)–(10);
9. Merge the subgroups in the main population;
10. Select the best Alpha of the subgroups as the global best;
11. Perform the global search by the PSO algorithm using Equations (11) and (12);

12. Evaluate the fitness of the population using Equations (13)–(16);
13. itr = itr + 1;
14. }
15. Return final population.

Algorithm 1. The pseudocode of the proposed hybrid algorithm with artificial agent. As depicted in algorithm 1, the division of the wolf's population into subgroups is shown in Figure 5. Each subgroup explores its local solution space and consists of the wolves Alpha, Beta, and Delta. The local space has been searched using the suggested imitation technique in the subgroups. The subgroups' Alpha, Beta, and Delta have been used to conduct a local search within each subgroup. Each wolf stores the visited best local position and the velocity. At the end of local search in the subgroups, the least fit subgroups eventually join the best subgroups. The wolves (search agents) of the subgroups are combined into a new population. The best Alpha of the subgroups is chosen as the global best of the total population. On the total population, the PSO traditional imitation algorithm is used for global imitation. In the global imitation, the local best and velocity of each wolf, which were calculated in the subgroups, were used. As shown in Figure 5, the Omega volves mimic the local Alpha, Beta, and Delta (local search). At this point, just the Omega wolves (replication arrays) are modified. Alpha, Beta, and Delta are updated in the subgroups using equations 1–10. According to the new positions of the wolves, Alpha, Beta, and Delta are upgraded. Each subgroup goes through this process a certain number of iterations. If the subgroup members' fitness has not improved, the mutation operator is used as the subgroup's Alpha to create more diversity. Once the local search is over, the wolves in each subgroup are returned to the main population. In the main population the wolves behave as particles and the swarm of particles evolves by the PSO equations (Equations (11) and (12)). Figure 6 shows the reposition of the particles into the main population. Each member of the group is defined by two vectors (speed and position in the search space). The new position of the particles (wolves) is updated by vector of speed and local best. The best position is calculated by the wolf local best ($X^{\text{local best}}$) and the best position of the main population ($X^{\text{global best}}$). Based on the flowchart in Figure 4, these operations are repeated.

$$\vec{a} = 2\left(1 - \frac{iter^2}{maxIter^2}\right) \tag{1}$$

$$\vec{A} = 2 \times \vec{a} \times \vec{r}_1 - \vec{a} \tag{2}$$

$$\vec{C} = 2 \times \vec{r}_2 \tag{3}$$

$$\vec{D}_\alpha = \left|\vec{C}_1 \times \vec{X}_\alpha - \vec{X}\right| \tag{4}$$

$$\vec{D}_\beta = \left|\vec{C}_2 \times \vec{X}_\beta - \vec{X}\right| \tag{5}$$

$$\vec{D}_\delta = \left|\vec{C}_3 \times \vec{X}_\delta - \vec{X}\right| \tag{6}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \times \left(\vec{D}_\alpha\right) \tag{7}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \times \left(\vec{D}_\beta\right) \tag{8}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \times \left(\vec{D}_\delta\right) \tag{9}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{10}$$

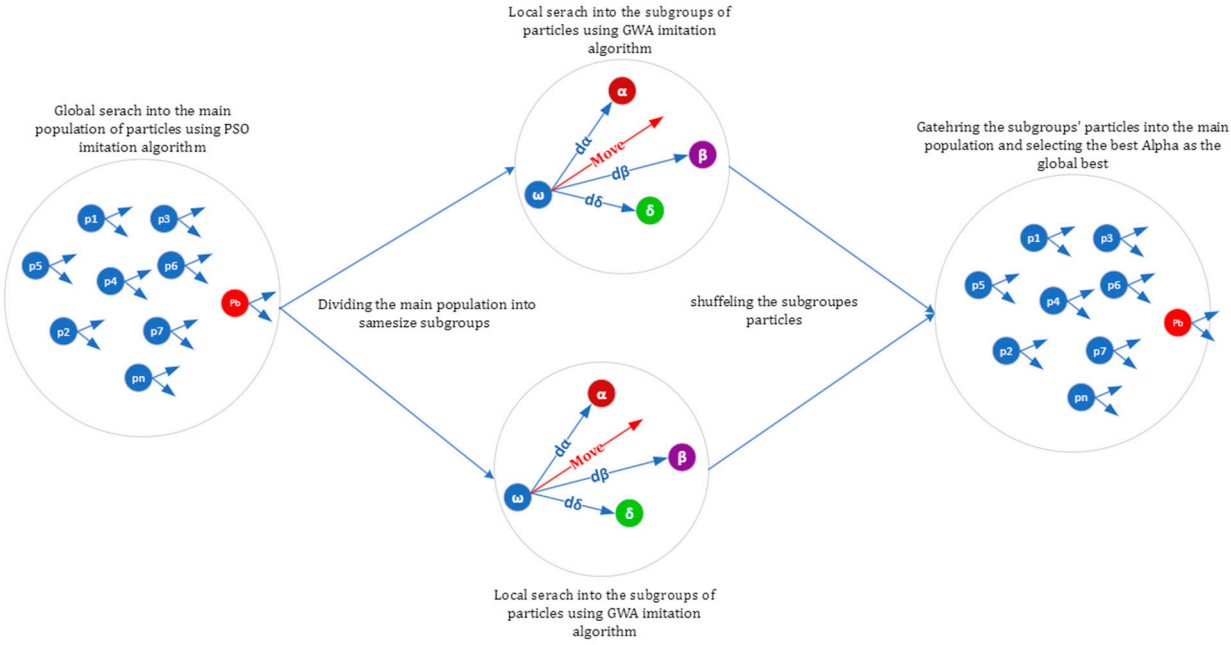


Figure 5. Global search is performed by the PSO imitation and local search is performed in the subgroups by the GWO.

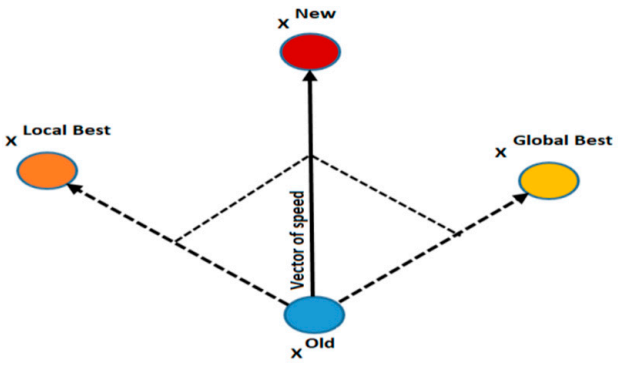


Figure 6. Optimizing the position of each particle in the total population using the best experience of each individual calculated in the local search step.

In the replica-placement problem's graph, one of those edges (or nodes) is removed; the algorithm can still quickly decide the optimum new situation. Therefore, PSGWA can immediately identify the best (shortest) path if a node is eliminated because it is no longer necessary to start at the beginning.

$$Particle\ v_i^{k+1} = wv_i^k + c_1\ r_1 \times \left(pbest_i^k - x_i^k\right) + c_1\ r_1 \times \left(gbest_i^k - x_i^k\right) \quad (11)$$

$$Particle\ x_i^{k+1} = x_i^k + v_i^{k+1} \quad (12)$$

3.3. Objective Function

The total cost of the procedure related to the data object k is specified in Equation (13). In Equation (13), AccessR(k) represents all the read operations for data object k from all of the servers that have received read requests for Object(k). Equation (14) is used to calculate the value of AccessR(k). In Equation (14), NS(nk) stands for the closest or least-priced server to S(n) that has a copy of Object(k). Additionally, AccessW(k) in Equation (13) refers to all writing to or updating operations on data object k from all servers that participated in the update request. Equation (15) is used to calculate the value of Accessw(k). The complete set

of running costs for the entire system for all data objects is determined using Equation (16). The data access-time reduction is the primary task of this study. The PSGWA optimization technique was employed in this study to find a solution to this issue. Equation (13) specifies the fitness function for each of the issue solutions.

$$TOC(k) = Access_R(k) + Access_W(k) \tag{13}$$

$$Access_R = V(k) \times \left(\sum_{n=1}^{N} read(n,k) \times \times l(n.NS(n,k)) \right) \tag{14}$$

$$Access_{Rw} = V(k \times \sum_{n=1}^{N} \left[write(n,k) \times \left[l(n,p(k)) + \sum_{(\forall j \in RS(k), j \neq n)} l(p(k),j) \right] \right] \tag{15}$$

$$TOC = \sum_{k=1}^{K} TOC(k) \tag{16}$$

4. Simulation Platform and Input Data

To evaluate the suggested method's outcomes, they were compared to the results of the GA, ACO, PSO, GWO, and PSGWA with distinct settings; all these algorithms have been implemented along with the proposed method in MATLAB and executed in the same hardware (intel core i7, 16GB RAM) and software (Windows 10) platforms. The developed MATLAB code is freely available. The following are the evaluation criteria used in this study.

- Total data access operations (read and write) cost (TOC).
- The number of replica objects.
- Convergence speed of the algorithms in providing the best solution.
- Reliability of the algorithms in solving the replica-placement algorithms.
- Stability of the algorithm during different executions for a same data object.

Table 2 shows the configuration parameters of the GA, ACO, SFLA, GWO, and PSGWA methods. The initial parameters of the GA are as follows. The initial population is 100 and the chromosome length is a matrix 28 × k, where 28 is the number of available countries (servers) and k is the number of data objects. The crossover probability is 0.8, and the mutation rate is 0.2.

To compare the suggested technique, the European Union standard database (EUData) was utilized, which comprises 28 nodes or servers and the network topology is in the form of a full graph; in this graph (G(V, E)) are the vertices of the graph and show the countries of the European Union (| V = 28 |). In addition, E is equal to the graph's edges and, it indicates the cost of the network communication infrastructure. The simulations were conducted with varying numbers of data objects (the value is between 1 and 5). Table 3 shows the specification of the twenty-eight data-servers used in the simulations. Figure 7 depicts the locations of the twenty-eight data servers throughout various European countries. In the simulation experiments, servers 5, 17, 9, 22, and 11 are considered the major servers for storing original data objects. The proposed method identifies the optimal number of replicas for the original data objects as well as the optimal servers (locations). The volume and location of original data objects used in the simulation are described in Table 4. Table 5 contains a list of data access requests (read and write) used in the simulations. In the simulations, different numbers of data items ($1 \leq k \leq 5$) were used. Each data access request contains five data objects to be accessed, with each number indicating the server.

Table 2. Calibration parameters of different replication algorithms.

Algorithms	Parameters	Value
Genetic Algorithm (GA)	Number of chromosomes	250
	Length of chromosome	28 * k
	Number of iterations	100
	Cross-mutate rate	0.7
	Mutation rate	0.05
Ant Colony Optimization (ACO)	Number of ants	150
	Initial pheromone (τ)	1
	Q parameter	1
	Pheromone power weight (α)	1
	Number of iterations	100
	Evaporation rate (ρ)	0.05
Shuffle Frog Leaping Algorithm (SFLA)	Memeplex Size	12
	Number of Memeplexes	8
	Memplex size	6
	FLA iterations (Beta)	7
	Number of iterations	100
Gray Wolf Optimizer (GWO)	Population size	40
	a	Values between [0, 2] (using Equation (1))
	C	Variable (using Equation (3))
	A	Variable (using Equation (2))
	r1, r2	Random values between [0–1]
Particle-based Gray Wolf Algorithm (PSGWA)	Population size	30
	Particle.W	0.8
	Particle.C1 and Particle.C2	1.8
	a	values between [0, 2] (using Equation (1))
	C	Variable (using Equation (3))
	A	Variable (using Equation (2))
	r1, r2	Random values between [0–1]
	Number of Subgroups	3
	Size of Subgroups	10
	Local Iteration	30

Table 3. Specification of the servers shown in Table 3.

Server Num.	Server Name	Country	Capacity of Server in Gigabytes
1	'FI'	Finland	10,021
2	'SE'	Sweden	21,451
3	'EE'	Estonia	21,110
4	'LV'	Lithonia	30,000
5	'LT'	Lithuania	32,002
6	'DK'	Denmark	34,555
7	'PL'	Poland	46,111
8	'CZ'	Czech	48,121
9	'SK'	Slovakia	42,121
10	'HU'	Hungary	10,001
11	'AT'	Austria	10,021
12	'RO'	Romania	21451
13	'IT'	Italy	21,110
14	'SL'	Slovenia	30,000
15	'BG'	Bulgaria	32,002
16	'GR'	Greece	34,555
17	'CY'	Cyprus	46,111
18	'MT'	Malta	48,121
19	'PT'	Portugal	42,121
20	'ES'	Spain	10,001
21	'FR'	France	34,555
22	'DE'	Germany	46,111
23	'LU'	Luxembourg	48,121
24	'BE'	Belgium	42,121
25	'NL'	Netherlands	10,001
26	'GB'	Great Britain	34,555
27	'IE'	Ireland	46,111
28	'HR'	Croatia	48,121

Table 4. The volume and location of original data objects used in the simulation.

Data Object	K1	K2	K3	K4	K5
Volume in Gigabytes	0.1	0.15	0.16	0.18	0.2
Primary Hosted Server	Server 5 (Lithuania)	Server 17 (Cyprus)	Server 9 (Slovakia)	Server 22 (Germany)	Server 11 (Austria)

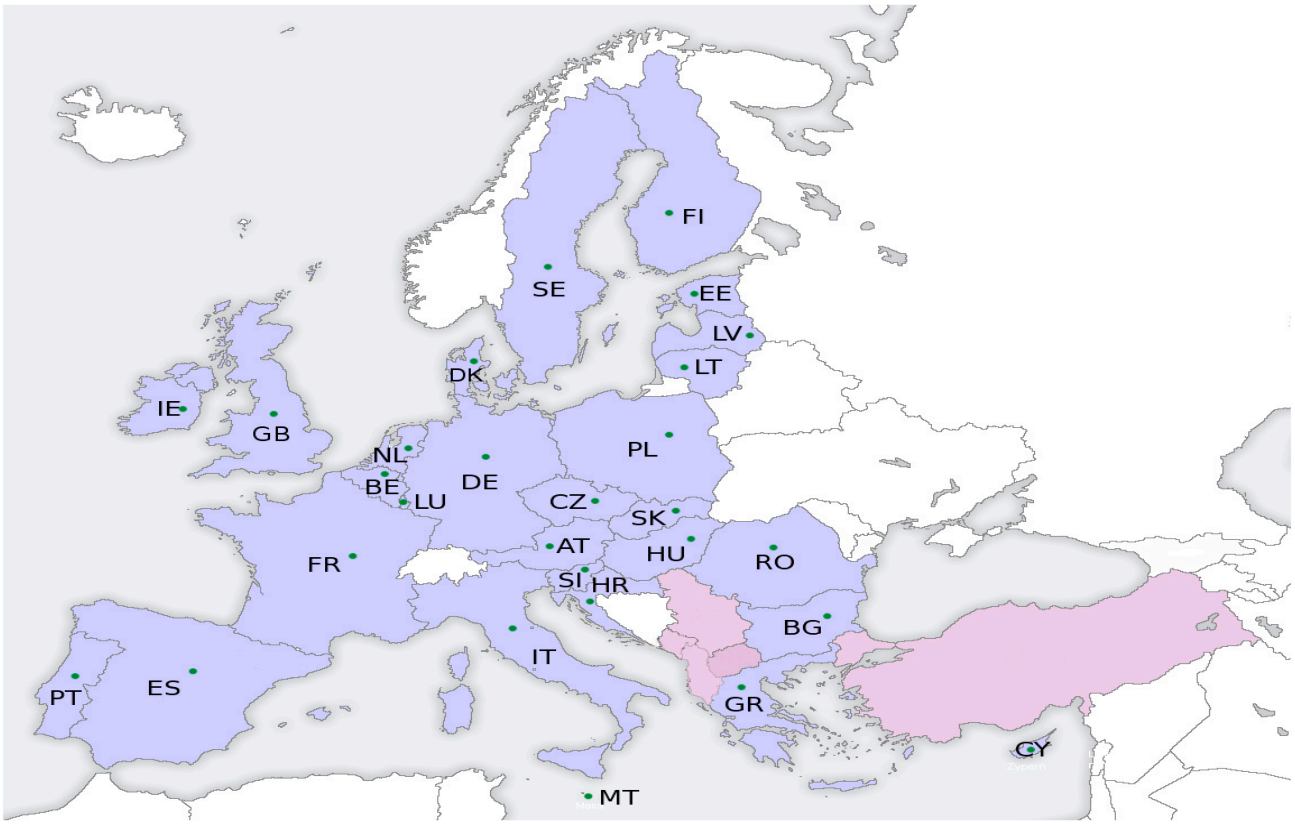


Figure 7. The map of the twenty-eight data servers located in the 28 European countries that was used as dataset in the simulations.

Results

In the replica-placement problem in distributed systems, there are k data objects that must be replicated on N servers. In this case, S(n) represents the server's nth instance, while Object(k) represents data object k. Five series of trials using GA, ACO, SFLA, GWO, PSO, and PSGWA were conducted to establish the ideal position for the replicas. In the initial round of tests, one data object was replicated on twenty-eight servers. To compare the outcomes, data access time (TOC) and the number of replicas produced for the data object were employed. Similar studies have been performed to identify where the best clones of two, three, four, and five data objects may be found. Each method replicates the data objects over many servers. As a result, the data access time and replica count for each algorithm differ. Figure 8 depicts the results of the first experiment. For this experiment, the map of Europe was replicated among twenty-eight servers (Figure 7). During the first stage of this experiment, twenty-eight servers were employed to host copies of data objects generated by various algorithms. Fifty-six read and write operations on the replicas were then performed to calculate the overall data access time (TOC). For the data objects, different algorithms may use a varied number and distribution of servers. As a result, overall access time is determined by the number and location of copies. A similar experiment was conducted with varying quantities of data objects. The suggested strategy aims to discover the best replica location; in this placement, the TOC is at its lowest level. Figure 8 shows the best TOCs obtained by GA, ACO, SFLA, GWO, PSO, and PSGWA. When k = 5, the TOC of the 56 data access operations (given in Table 6) in the replicas created by the GA, ACO, SFLA, GWO, PSO, and PSGWA are respectively 0.6651, 0.8022, 0.8163, 0.8241, 1.0578, and 0.93066 s.

Table 5. The read and write requests used in the simulation.

Request Number	Read Request Array 28 × 5	Write Request Array 28 × 5
1	5, 3, 3, 0, 1	1, 1, 0, 2, 1
2	0, 4, 1, 1, 1	0, 5, 2, 0, 2
3	1, 2, 7, 6, 0	1, 1, 1, 0, 0
4	4, 3, 8, 0, 6	0, 1, 0, 1, 0
5	0, 6, 11, 2, 1	0, 2, 0, 1, 0
6	9, 0, 4, 0, 9;	0, 0, 0, 0, 0
7	3, 5, 2 1, 10	0, 0, 0, 0, 0
8	1, 4, 2, 0, 1	0, 0, 1, 0, 0
9	3, 5, 0, 8, 0	0, 0, 0, 1, 0
10	2, 3, 2, 4, 5	0, 1, 1, 0, 1
11	2, 1, 1, 8, 0	0, 0, 0, 1, 0
12	2, 0, 7, 4, 9	0, 1, 2, 0, 0
13	1, 1, 2, 4, 0	0, 0, 0, 0, 0
14	1, 1, 4, 1, 0	1, 2, 1, 0, 0
15	0, 4, 3, 0, 7	0, 1, 0, 2, 0
16	1, 1, 2, 2, 1	0, 1, 0, 1, 0
17	3, 6, 5, 4, 5	0, 0, 0, 0, 2
18	6, 0, 4, 0, 6	0, 1, 0, 1, 0
19	0, 4, 2, 3, 0	0, 0, 1, 0, 4
20	5, 3, 4, 0, 3	0, 1, 2, 4, 0
21	2, 10, 2, 6, 0	0, 0, 2, 0, 0
22	5, 4, 4, 0, 2	1, 0, 0, 1, 4
23	3, 0, 2, 3, 1	0, 0, 1, 0, 0
24	2, 7, 9, 0, 5	0, 3, 0, 0, 3
25	1, 4, 3, 3, 5	0, 0, 0, 2, 0
26	3, 6, 2, 10, 5	1, 0, 0, 1, 1
27	6, 0, 4, 0, 6	0, 0, 3, 0, 0
28	0, 4, 4, 3, 0	0, 2, 0, 0, 2

Table 6. The slopes of the trendline produced for different algorithms.

Algorithm	Slopes of the Trendline
PSGWA	0.04215
PSO	0.2705
GWO	0.0518
SFLA	0.0518
ACO	0.1132
GA	0.0877

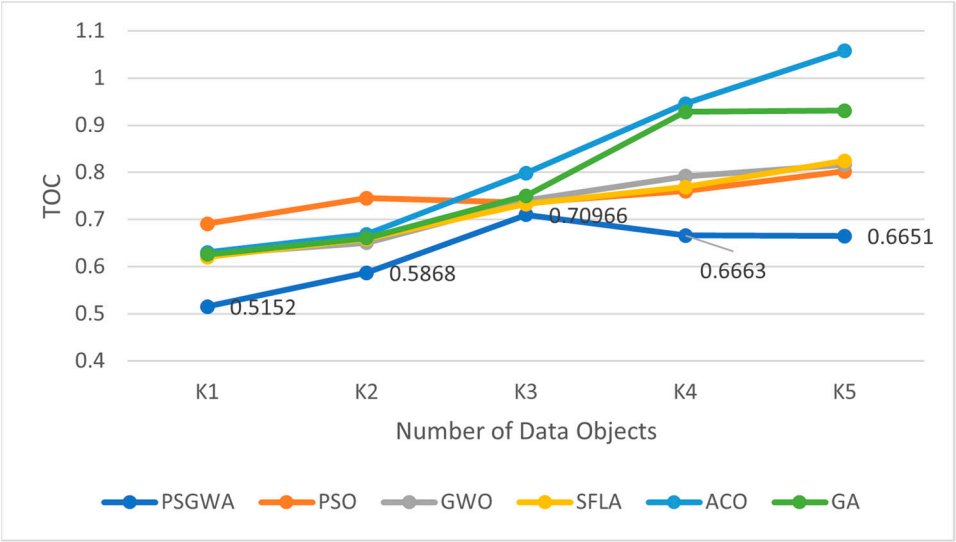


Figure 8. The best value of data access time obtained by different replica-placement algorithms for different number of data objects.

The proposed method provides lower data success time than the other algorithms. The comparable results have been obtained when there are four data objects for replication; the total data access time to the replicas produced for the four data objects by PSGWA, PSO, GWO, SFLA, ACO, and GA are respectively 0.6663, 0.76035,0.79147, 0.76898, 0.94575, and 0.92816 s. Overall, the TOC of the produced replicas by the proposed PSGWA is lower than the TOC of the other algorithms. As shown in Figure 8, PSO and GWO have similar performance in the replica-placement problem in terms of the TOC criterion. The PSGWA exploits the merits of both algorithms and then performs better than the GWO and PSO. The suggested PSGWA has a lower TOC than the other methods in all benchmarks. Figure 9 depicts the reduction in data access time achieved by various algorithms. In all benchmarks, the suggested PSGWA outperforms the previous algorithms in terms of TOC reduction. In the small benchmarks (k = 1), all methods (excluding PSGWA) perform similarly in terms of TOC reduction. In all benchmarks, there are significant disparities in the TOC of various algorithms. The TOC is decreased by 48% when the suggested technique is used for one data object. This figure is 37%, 36%, 37%, 37%, and 31% for the GA, ACO, SFLA, GWO, and PSO algorithms, respectively. In the large benchmarks (k = 5), PSGWA provides 31% TOC reduction that is better than the other algorithms. Regarding the TOC reduction criterion, the proposed PSGWA has a higher performance than the other algorithms. Figures 8 and 9 show how poorly the ACO algorithm worked when confronted with a large number of data objects. The ACO has performance overhead in some benchmarks. Regarding the results of the conducted simulations, PSGWA provides the lowest data access time. The amount of TOC reduction by the PSGWA is higher than the other algorithms.

The suggested method's TOC criterion grows approximately linearly with the number of data objects. A statistical technique known as interpolation is used to approximate an unknown TOC value when the value of K is bigger than five. Interpolation is a technique used to estimate the performance of replica-placement algorithms in huge datasets. Interpolation is possible by using extra known values in the same sequence as the unknown value. If there is a consistent trendline throughout the acquired data points, the performance of the replica-placement algorithms can be predicted. Figure 10 depicts the trendlines derived by several algorithms over the TOC. The trendlines represent the behavior of the algorithms for various data objects. The performance of the proposed strategy is indicated by the slots of the constructed trendlines. The TOC decreases as the slot decreases. The PSGWA's behavior is linear, as illustrated in Figure 10. Table 6 displays the trendlines' slope for the various algorithms. The PSGWA trendline slope is around 0.0421. The PSGWA has the least trendline slope. The highest slope is associated with the PSO. Indeed, the TOC value of

PSO increases more steeply for larger data and it can be used for the considerable number of data objects in distributed systems such as clouds and IoT.

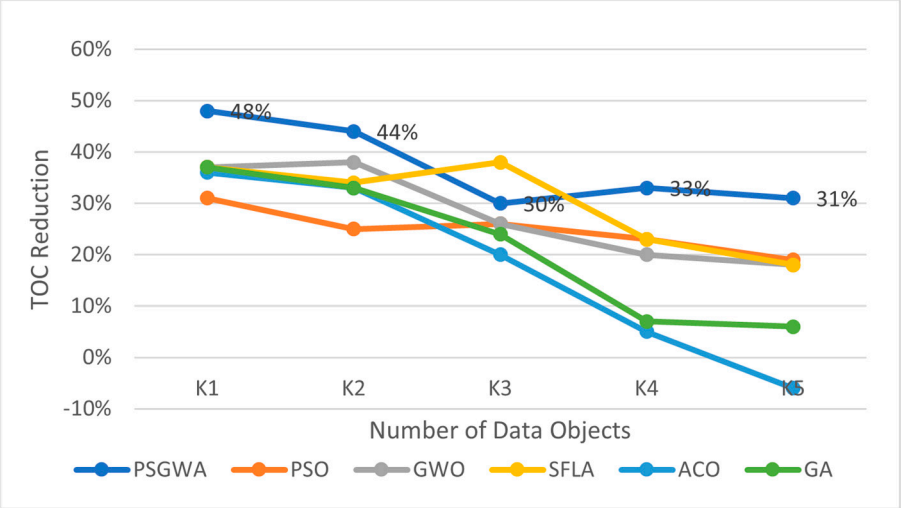


Figure 9. The best percentage of data access time reduction by different replica-placement algorithms for different numbers of data objects.

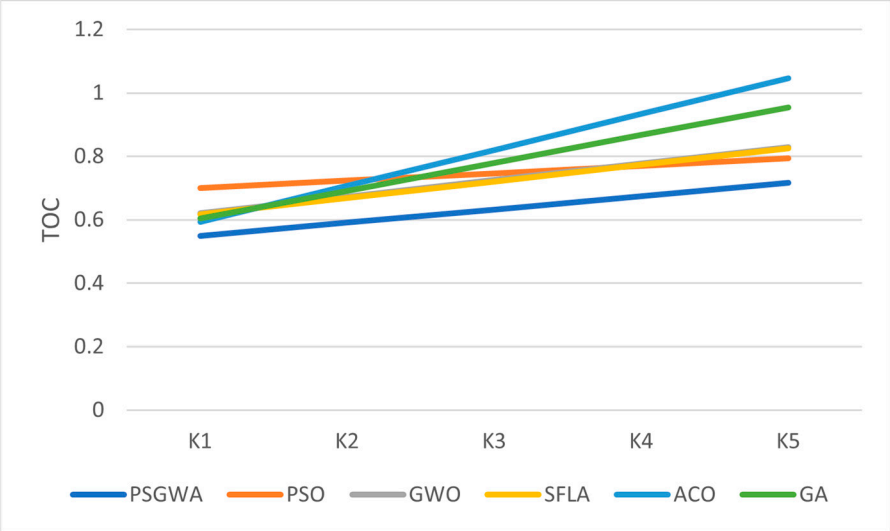


Figure 10. The trendlines of the produced TOC values obtained by different algorithms.

The number of replicas produced by the replica-placement algorithms is another evaluation criterion. The higher the number of produced replicas, the higher storage cost. Figure 11 displays the number of replicas produced by various algorithms for various numbers of data objects. The number of produced replicas by the GA, ACO, SFLA, GWO, PSO, and PSGWA are respectively 5, 6, 5, 5, 5, and 4 when k = 1. In the large benchmark (k = 5), the number of produced replicas is 24, 30, 8, 12, 9, and 7, respectively. In most of benchmarks, the number of replicas produced by the PSGWA is lower than the other algorithms. Overall, the suggested method can achieve larger time savings with fewer replications. The size of required storage by the PSGWA is lower than the other algorithms in the replica-placement problem. The largest time reduction with a limited number of replicas indicates the higher efficiency of the PSGWA over the other algorithms. As shown in Figure 11, in the PSGWA, the rate of increase in the number of replicas increases linearly and with a lower slope than other algorithms; it means that the PSGWA will also have higher performance in the larger number of data objects.

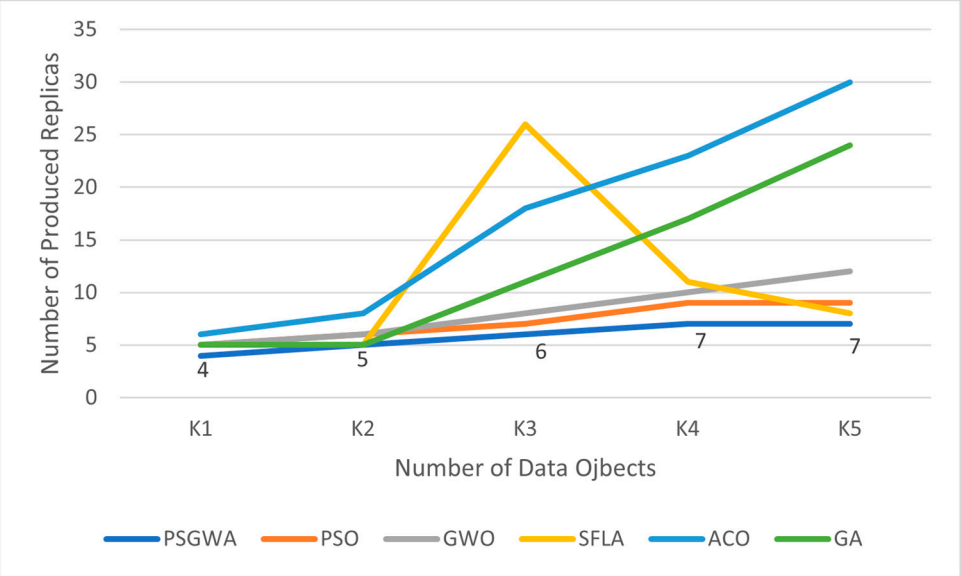


Figure 11. The number of replicas created by different replica-placement algorithms for different numbers of data objects in the best cases.

The other performance criterion of the heuristic algorithms is the convergence criterion. Another set of experiments was run to evaluate the convergence of the replica-placement algorithms. Figure 12 depicts the convergence of various strategies in establishing the appropriate number and location (servers) for a single data object replica. As seen in Figure 12, the proposed strategy has a much faster rate of convergence than the other methods. The PSGWA identifies the optimum servers for the replicas of one data object before ten iterations. In this test (K1), four copies were produced among specified servers, resulting in a 48% TOC reduction. In terms of TOC reduction and convergence speed, Figure 12 shows how the PSGWA outperforms the PSO, GWO, SFLA, ACO, and GA. In the second experiment, the performance of different algorithms was evaluated in replacing two data objects. Figure 13 depicts the convergence of the algorithms when two data objects must be replicated and stored in the proper servers. The results confirm the better performance of the proposed PSGWA in terms of convergence and TOC. The PSGWA algorithm can discover the best replica location in this benchmark experiment before the fifth iteration. The proposed method obtained 44% TOC reduction with only five replicas. Figure 14 shows the convergence speed of different algorithms when there are three data objects for replication over the twenty-eight servers. In this experiment, the value of TOC obtained by the proposed method is lower than the TOC obtained by the other algorithms. As shown in Figure 14, the differences among the TOC of different algorithms are close to each other. The convergence speeds and obtained TOCs by different algorithms are shown in Figure 15. In this benchmark (k = 4), the PSGWA identifies the optimum servers for four data objects' replicas before the 20th iteration. PSGWA results in a 33% TOC reduction with seven replicas when there are four data objects. In this benchmark, PSO and SFLA have similar performance (23% TOC reduction). GWO, as other algorithm, provides a 20% reduction in the data access time. The required number of data servers is lower than the other algorithms. In the other experiments, the performance of the different algorithm has been evaluated with five data objects. Figure 16 shows the performance of the algorithms when there are five data objects for replication. As shown in Figure 16, PSGWA reduces the TOC by 31% with only seven data replicas. In this benchmark, PSO, GWO, and SFLA have similar performance. PSGWA have higher convergence speeds than the other algorithms. GWO contributes 18% TOC reduction by creating twelve replicas. After the proposed method, the PSO requires a lower number of data servers for managing the five data objects than the GWO, ACO, SFLA, and GA.

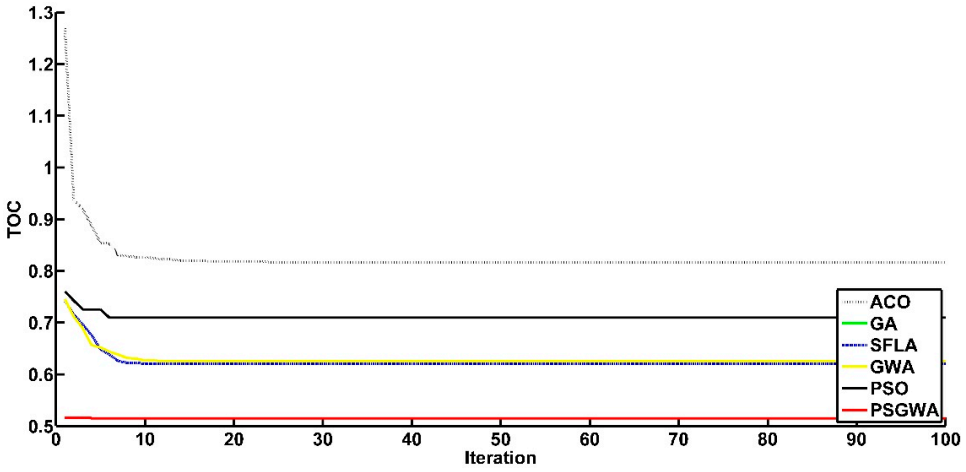


Figure 12. The convergence of different replica-placement algorithms to optimal solutions for one data object (k = 1).

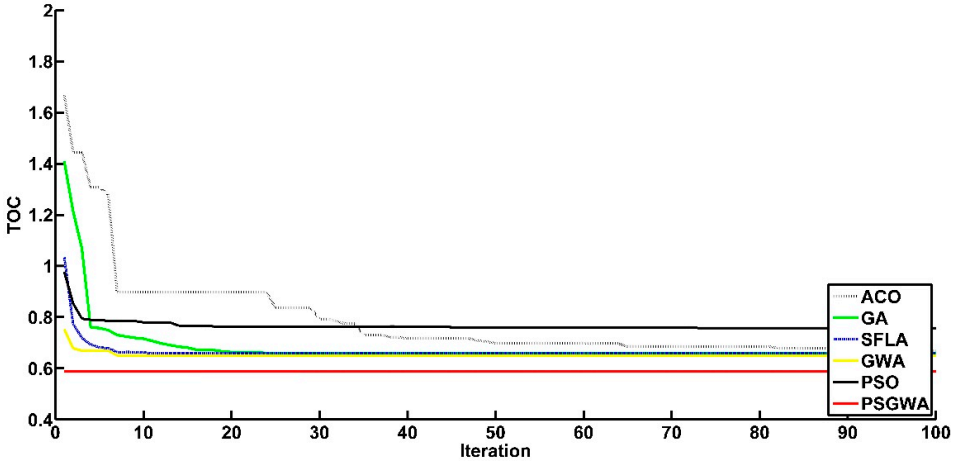


Figure 13. The convergence of different replica-placement algorithms to optimal solutions for one data object (k = 2).

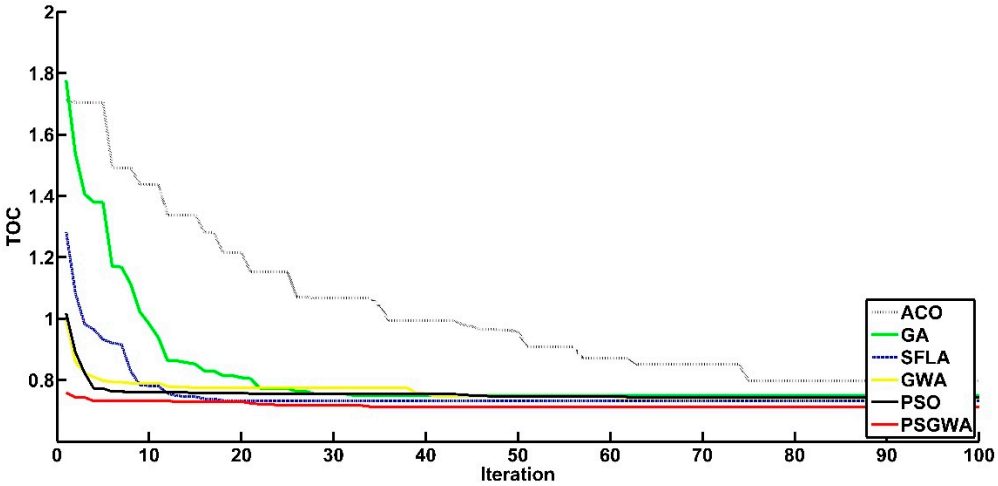


Figure 14. The convergence of different replica-placement algorithms to optimal solutions for one data object (k = 3).

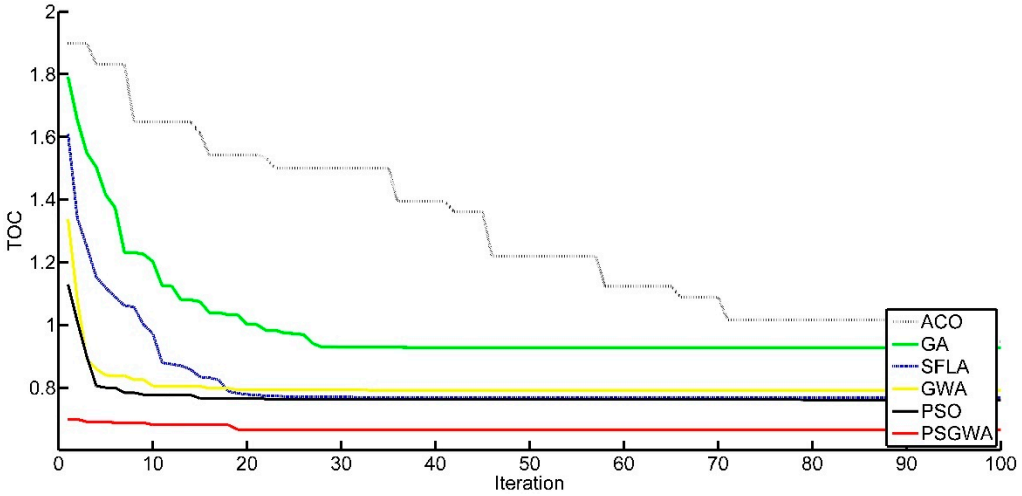


Figure 15. The convergence of different replica-placement algorithms to optimal solutions for one data object (k = 4).

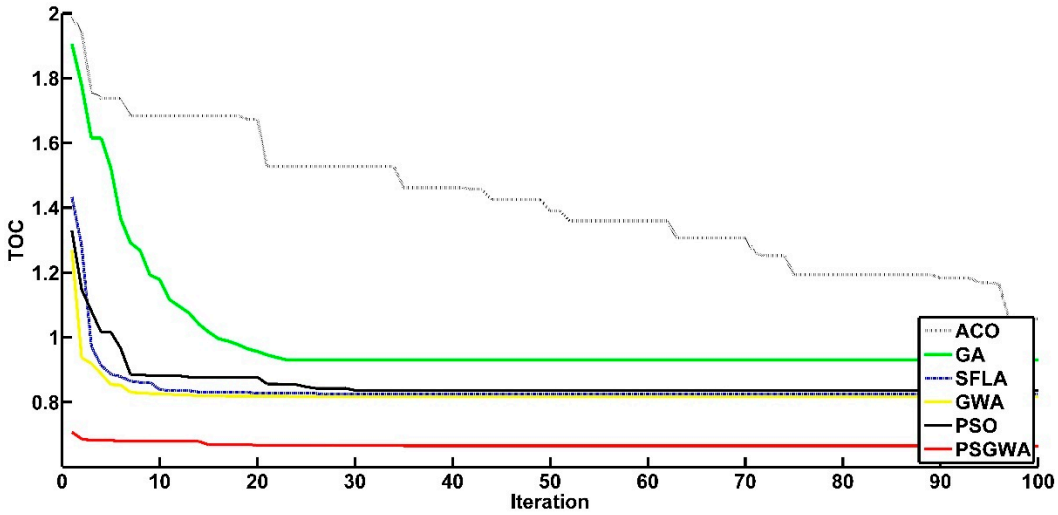


Figure 16. The convergence of different replica-placement algorithms to optimal solutions for one data object (k = 5).

Figure 17 shows the average TOC obtained by different algorithms. The average values of TOC provided by GA, ACO, SFLA, GWO, PSO, and PSGWA are respectively 0.6286, 0.7469, 0.7249, 0.7209, 0.8201, and 0.7790. Indeed, the average value of TOC in PSGWA is lower than the value of TOC in other algorithms. The PSGWA provides lower data access time than the other algorithms. Figure 18 shows the average value of TOC reduction by different algorithms. The average TOC reduction by GA, ACO, SFLA, GWO, PSO, and PSGWA are 35%, 25%, 28%, 30%, 18%, and 21%. Indeed, the proposed PSGWA provides 35% TOC reduction on average. Finally, as shown in Figure 19, the average number of replicas produced by GA, ACO, SFLA, GWO, PSO, and PSGWA are as 6.25, 7.2, 8.2, 12.5, 19.75, and 14.25. Overall, the proposed method manages the data with fewer copies and provides the greatest reduction in data access time. Regarding the results, PSGWA has a significantly higher TOC reduction, faster convergence, and fewer produced replicas (smaller storage spaces) than the other methods.

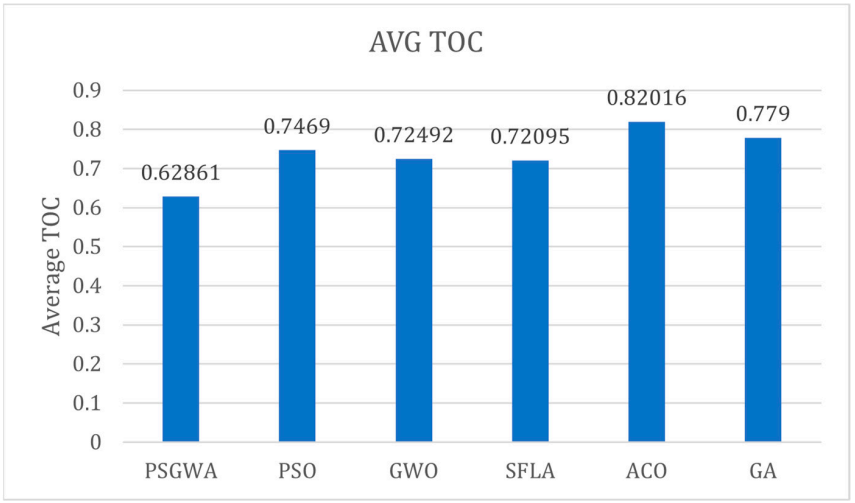


Figure 17. The average TOC of all data objects used in the experiments obtained by different algorithms.

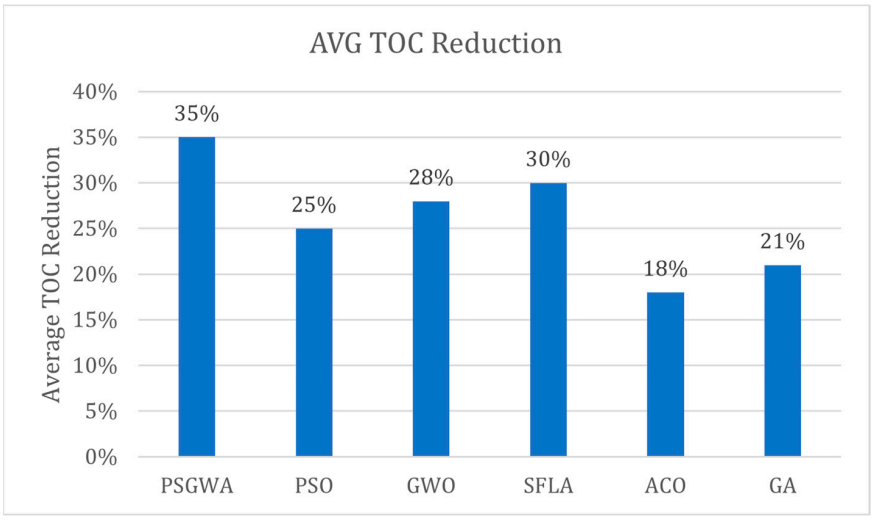


Figure 18. The average TOC Reduction obtained by different algorithms.

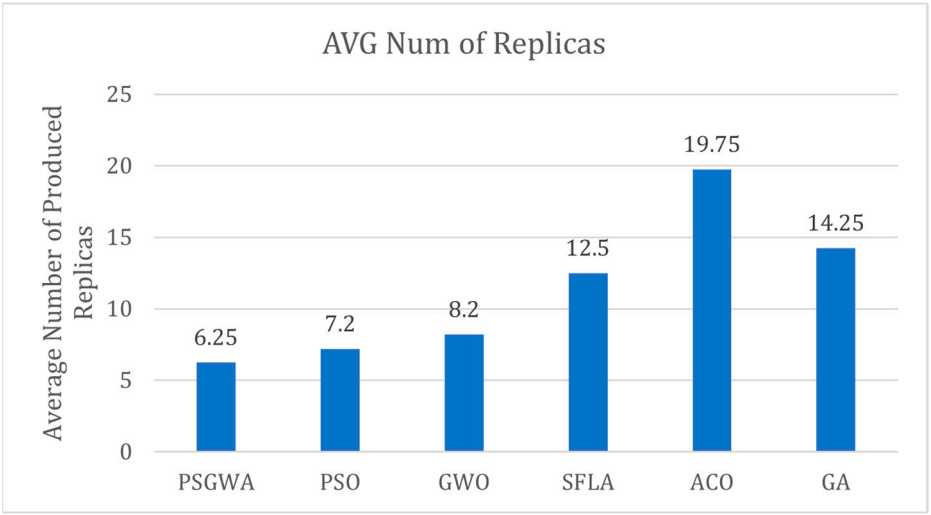


Figure 19. The average number of produced replicas by different algorithms.

The collected results were examined in terms of deviation value to assess the stability of the replica-placement algorithms. Ten runs of each algorithm were examined to evaluate the range of possible outcomes variations. Figures 20 and 21 depict the range of variations between ten results (TOC) from ten executions. PSGWA has the lowest TOC with the lowest variation, according to the results. The lowest variation in the generated outcomes from 10 executions illustrates the PSGWA's stability in the replica-placement problem. Because of the narrow range of change, the proposed solution for the replica-placement problem can produce the same outcomes in different executions. The number of replicas produced by the algorithms for each data object is the other performance criterion. The fewer replicas generated, the fewer servers are required. As illustrated in Figures 22 and 23, PSGWA provides the shortest data access time with the fewest created copies. The key advantage of the recommended method is that it only takes a few replicas to reduce the value of TOC. When K = 5, the PSGWA has the smaller range of changes in the number of replicas than the other algorithms. In fact, PSGWA is more stable than the other algorithms. After PSGWA, the PSO, GWO, and SFLA show similar performance in terms of TOC and stability. The GA and ACO algorithms produce a wider range of outputs than the other algorithms. The ACO does have less stability than the other algorithms, and it may produce different outcomes in different executions.

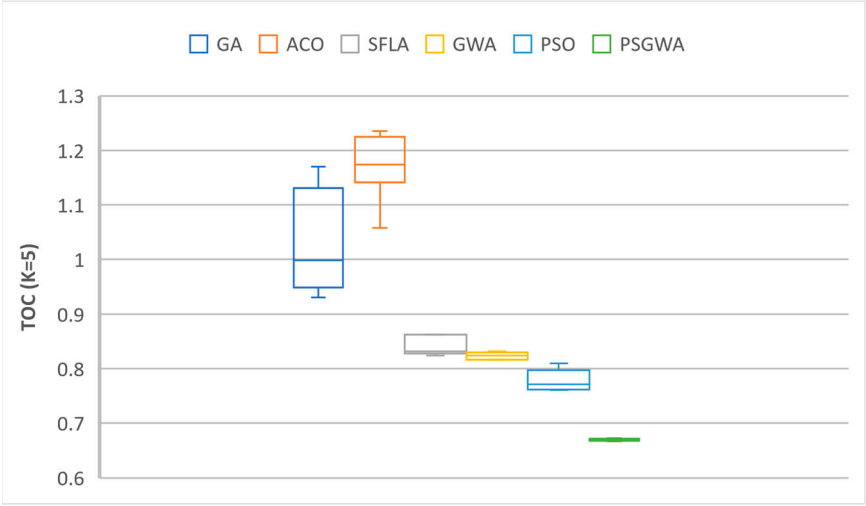


Figure 20. The range of changes in the value of TOC obtained by different algorithms during ten executions when K = 5.

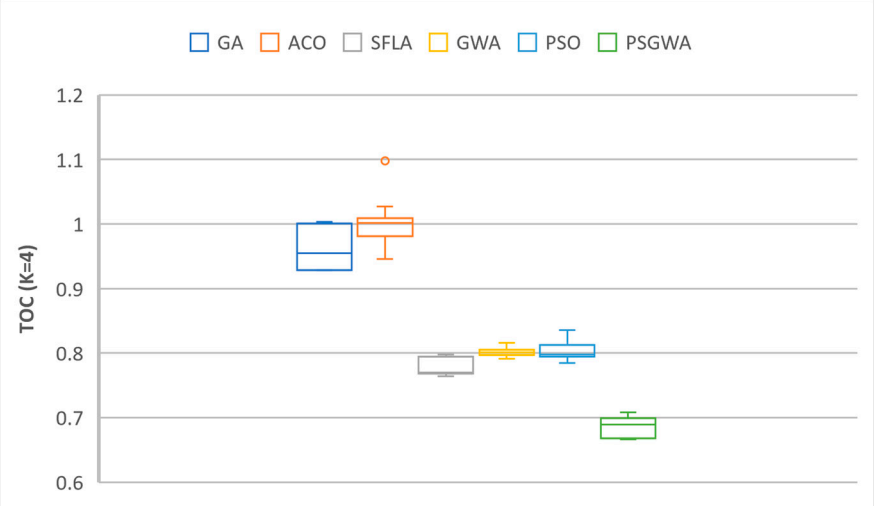


Figure 21. The range of changes in the value of TOC obtained by different algorithms during ten executions when K = 4.

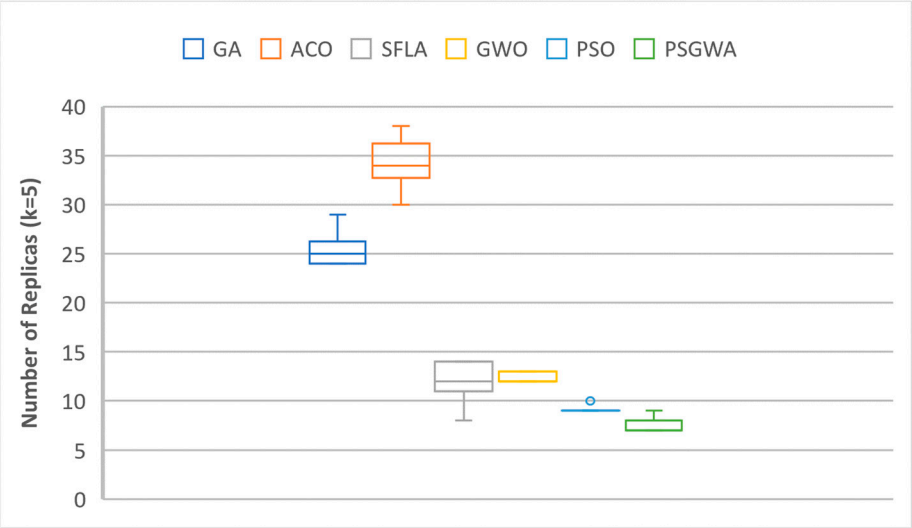


Figure 22. The range of changes in the number of replicas produced by different algorithms during 10 executions when K = 5.

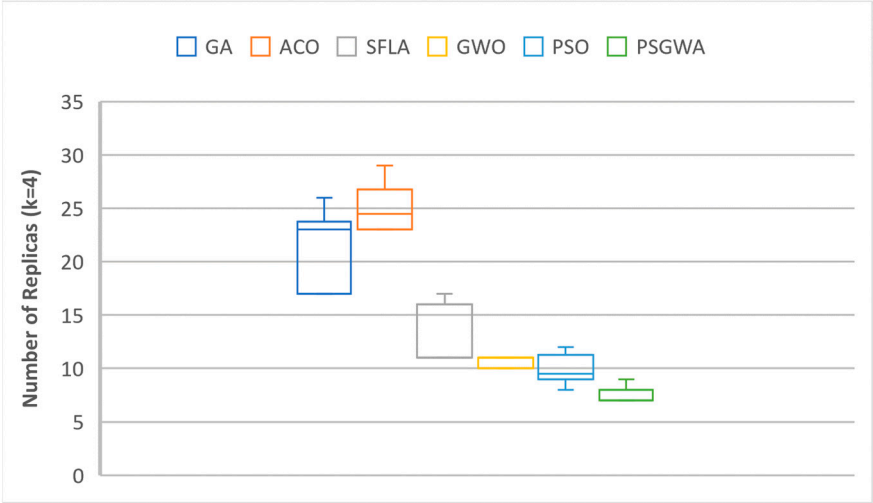


Figure 23. The range of changes in the number of replicas produced by different algorithms during 10 executions when K = 4.

Figures 24 and 25 illustrate the standard deviation (STDV) of the number of replicas and TOC produced by various algorithms. A lower TOC standard deviation indicates that the algorithms are more stable. When k = 4, the standard deviation of the number of produced replicas by the proposed method is 0.707, which is much less than that of the GA, ACO, and SFLA techniques. In this benchmark, the STDV of the PSO and GWO is lower than that of the PSGWA. When k = 5, the standard deviation of the number of replicas produced by the proposed method is 0.710, which is much less than the GA, ACO, SFLA, and PSO techniques. The standard deviation of the TOC provided by proposed method for four data objects is 0.0019, which is considerabbly less than the STDV of other algorithms. İn all benchmarks, the STDV of the GWO is lower than the PSO; indeed, GWO is more stable than the PSO in the replica-placement algorithm. The proposed PSGWA exploits the merits of PSO and GWO at the same time. Overall, the statistics show that the PSGWA is effective as long as GWO is significantly more stable than the other algorithms.

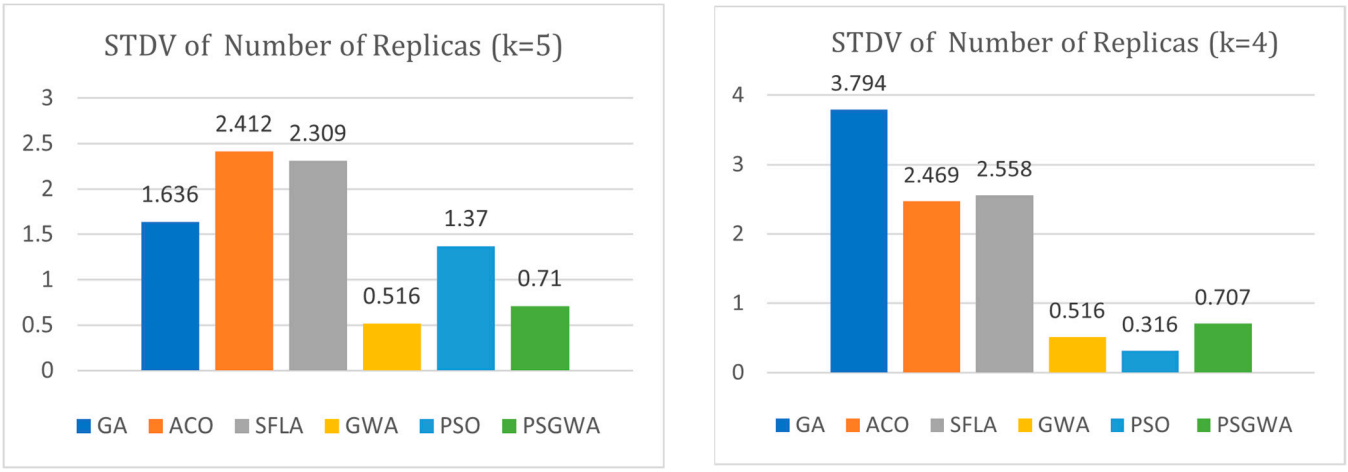


Figure 24. Standard deviation among the number of replicas produced by different algorithms during 10 executions when K = 4 and K = 5.

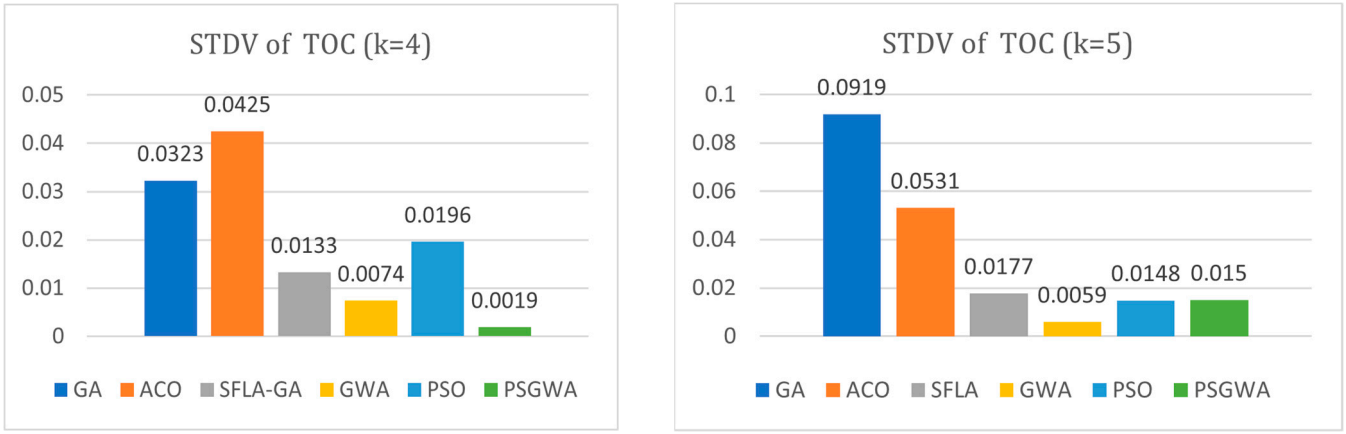


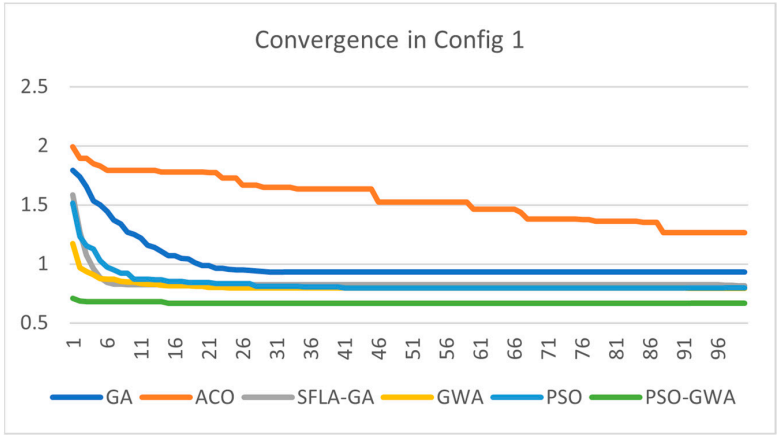
Figure 25. Standard deviation among the TOC obtained by different algorithms during 10 executions when K = 4 and K = 5.

Similar studies were carried out with four different setups in order to validate and trust the results. The principal servers of the original data objects are different in each experiment. Furthermore, the size of data objects varied depending on the arrangement. Each replica-placement algorithm has been tested under various scenarios (configurations). Table 7 displays the final experiment setting parameters. Four separate data objects with varying volumes were stored in selected servers in each configuration. In the configurations, the primary server of each data object is distinct. Figure 26 depicts the performance of the algorithms in the stated configurations. The results show that the PSGWA's performance in finding the best replica-placement is independent of the size and primary location of the data objects. The PSGWA has a higher performance than the other algorithms in all experiments with varying configurations. PSGWA determines the best placement of the data objects before the tenth iteration in first configuration (Figure 26a). In this experiment, approximately, the SFLA, PSO, and GWO have similar performances. The PSGWA yielded a TOC of around 0.665. The second experiment was carried out using a modified arrangement. Data items were saved in various primary servers in the second arrangement. Figure 26b depicts the algorithms' performance in the second configuration; PSGWA converges to the best solution before the tenth iteration. The PSGWA obtained a TOC of around 0.682. In the second experiment, the SFLA and GWO have a similar performance and Pso has a lower performance than the SFLA and GWO. The exploitation

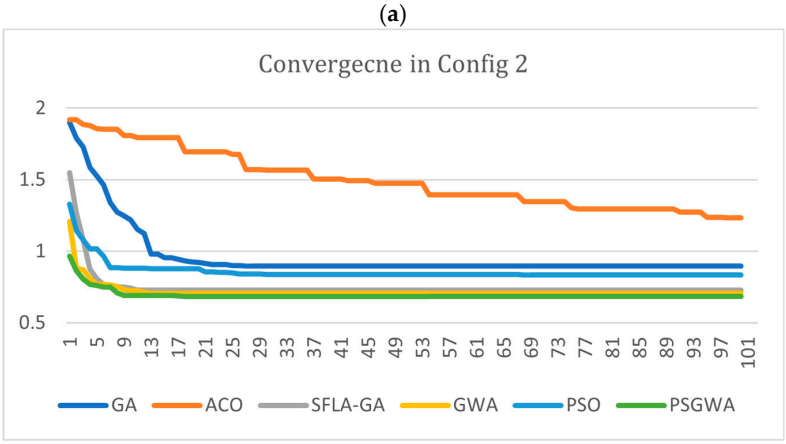
step of the PSGWA was inplemented by GWO; hence, the proposed method benefits from the GWO and PSO. Overall, the PSGWA has a higher performance than the GWO and PSO.

Table 7. Four different workload configurations to validate the performance of different algorithms.

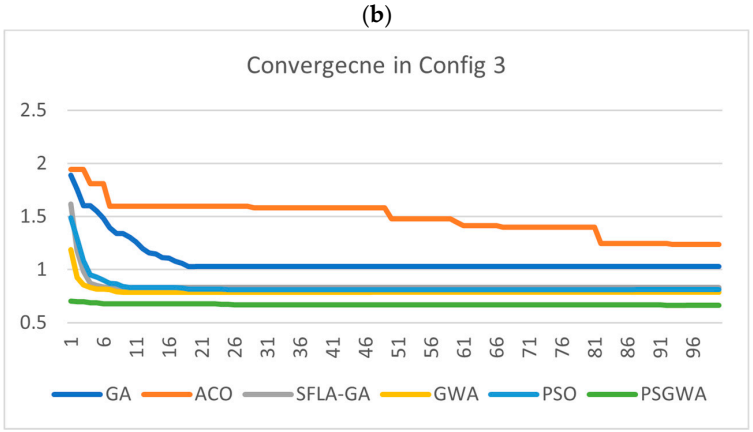
Configuration	Primary Servers	Volume of Data Objects (GB)
Configuration 1	[15, 7, 3, 17, 27]	[0.1, 0.15, 0.16, 0.1, 0.2]
Configuration 2	[1, 28, 11, 5, 3]	[0.7, 0.15, 0.86, 0.9, 0.7]
Configuration 3	[19, 2, 17, 15, 6]	[0.3, 0.25, 0.1, 0.7, 0.25]
Configuration 4	[11, 21, 1, 15, 3]	[0.9, 0.15, 0.3, 0.5, 0.60]



(**a**)

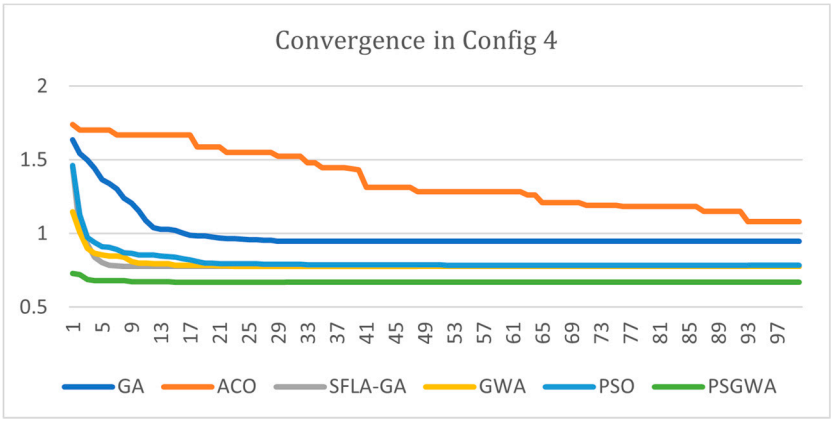


(**b**)



(**c**)

Figure 26. *Cont.*



(**d**)

Figure 26. The behavior of different algorithms during 10 executions when K = 4. (**a**) The performance of the algorithms when the original data objects were stored in [15, 7, 3, 17, 27] servers. (**b**) The performance of the algorithms when the original data objects were stored in [1, 28, 11, 5, 3] servers. (**c**) The performance of the algorithms when the original data objects were stored in [19, 2, 17, 15, 6] servers. (**d**) The performance of the algorithms when the original data objects were stored in [11, 21, 1, 15, 3] servers.

Figure 26c,d show the performance of the replica-placement algorithms in the last configurations 3 and 4. The suggested PSGWA clearly outperforms the previous algorithms in terms of performance and efficiency. In the third and fourth experiments, the performance of the SFLA, GWO, and PSO are like each other. Furthermore, the number of replicas made by the PSGWA is lower than the number of replicas created by the other algorithms. As a result, the suggested solution reduces the overall data access time in distributed systems with fewer data replicas stored on multiple servers. In terms of outcomes, the PSO, GWO, and SFLA outperform the ACO and GA in terms of performance and efficiency. In this study, the PSO, GWO, and SFLA had comparable results. The PSO and GWO produced an average of 7.2 and 8.2 copies, respectively. The PSGWA's average number of created replicas is around six. The average number of generated replicas by the SFLA is about twelve in the conducted experiments. The average TOC decrease by the PSOGWA, PSO, GWO, and SFLA are 35%, 25%, 28%, and 30%, respectively. The SFLA, in fact, has a larger TOC reduction than the PSO and GWO. The performance and efficiency of ACO, the other replica-placement algorithm employed in this investigation, are the lowest.

Figure 27 illustrates the TOC obtained by each replica placement in four different experiments with different configurations. The proposed PSGWA has remarkably comparable results in different experiment configurations. Meanwhile, the other algorithms generate different results (TOC) regarding the initial locations, and the volume of data objects varies. Indeed, the performance of the PSGWA is approximately independent of the data object's specification. Figure 27 indicates that PSGWA has a linear and predictable behavior. Less fluctuation reflects the algorithm's dependability. The linear behavior with a slope of about zero degrees implies that the proposed PSGWA is successful in this problem. The best locations of main data objects and the locations of the related replicas determined by different algorithms is shown in Figures 28–30.

Regarding the features of the problem and the obtained results presented, the metaheuristic method is recommended for this problem because there is not the required prior information of the search space for the gradient-based methods. Also, the gradient-based optimization methods may be very slow and interactable for the large datasets. However, for more confidence, an extensive series of experiments should be developed using the gradient-based methods (as a future study).

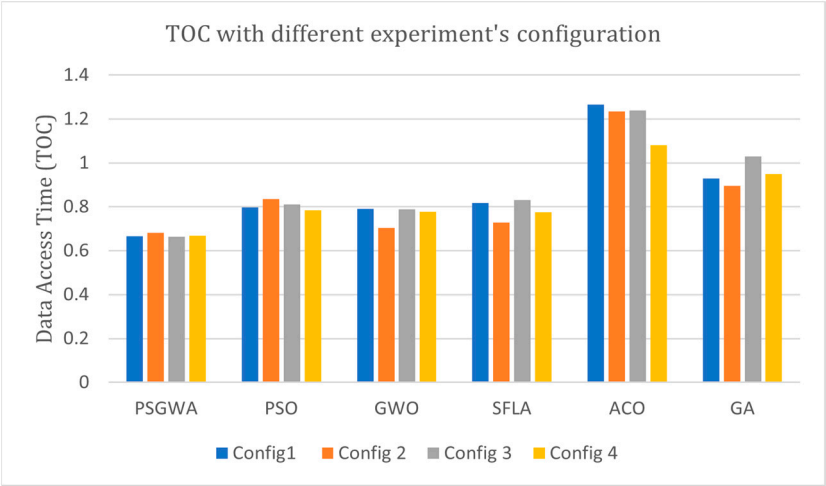


Figure 27. The value of TOC in different algorithms in different configuration.

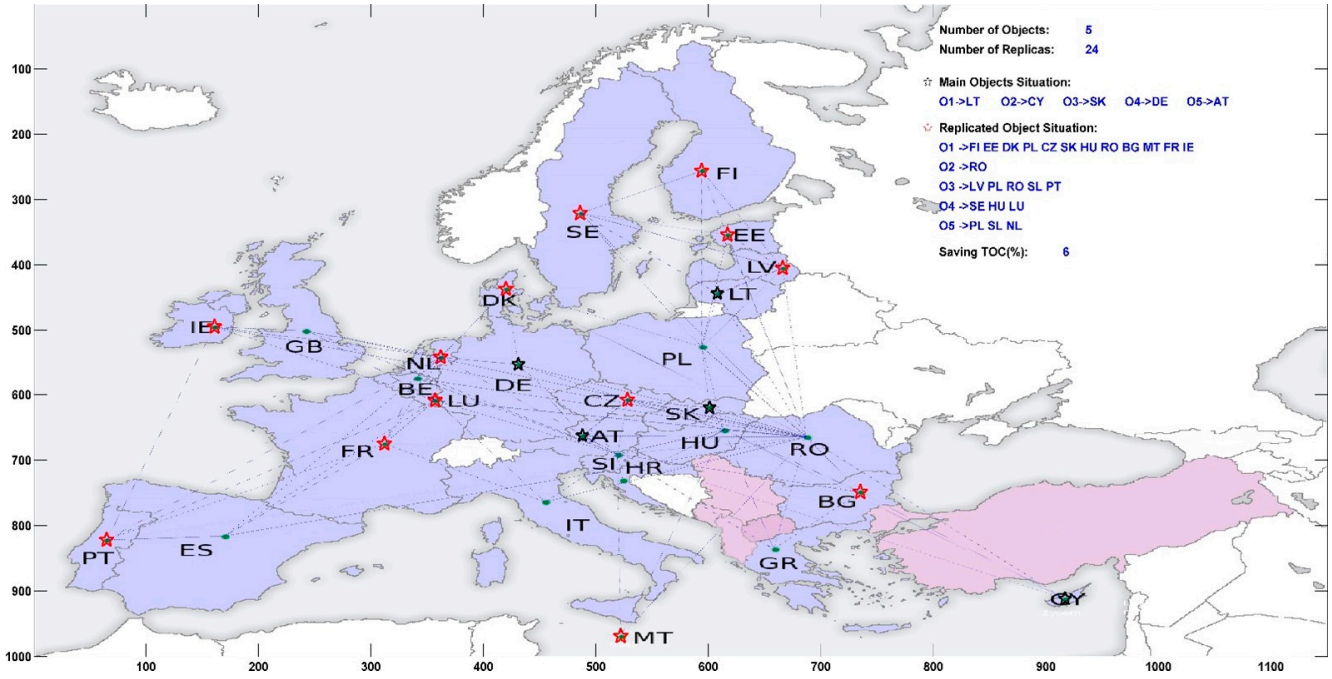


Figure 28. The replication of five data objects by the GA and its location on the Europe map.

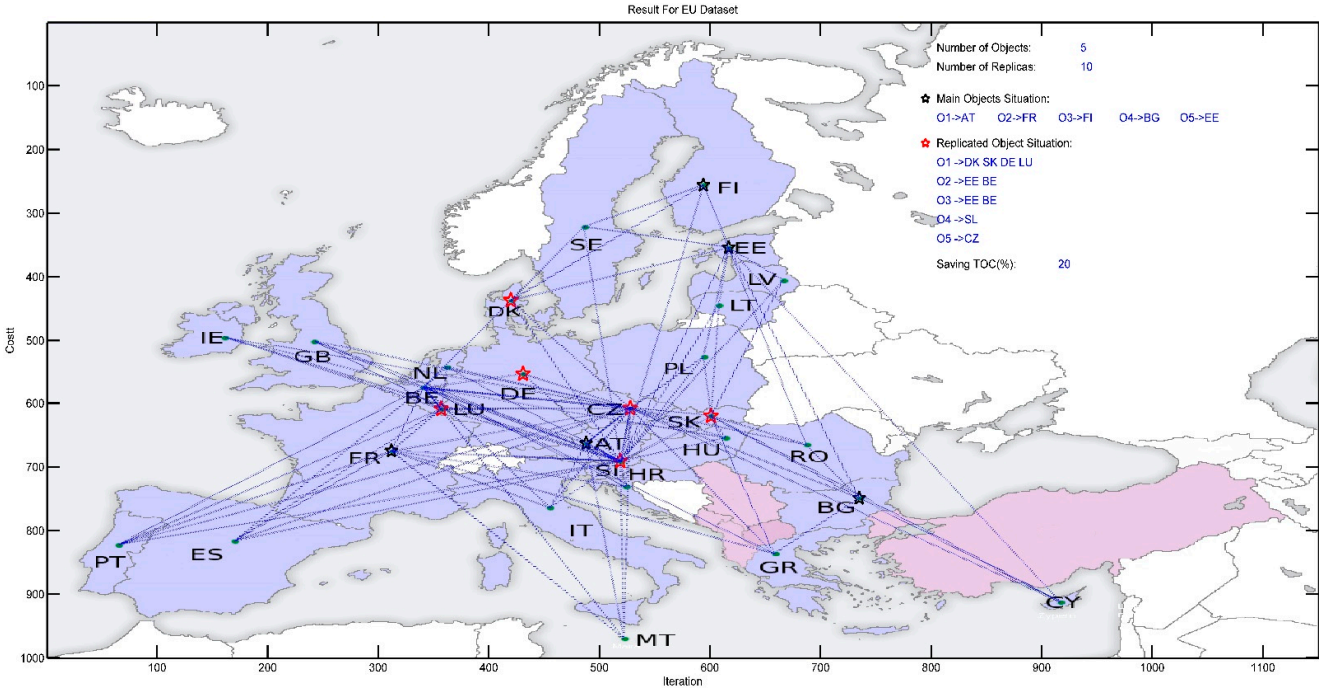


Figure 29. The replication of five data objects by the PSO and its location on the Europe map.

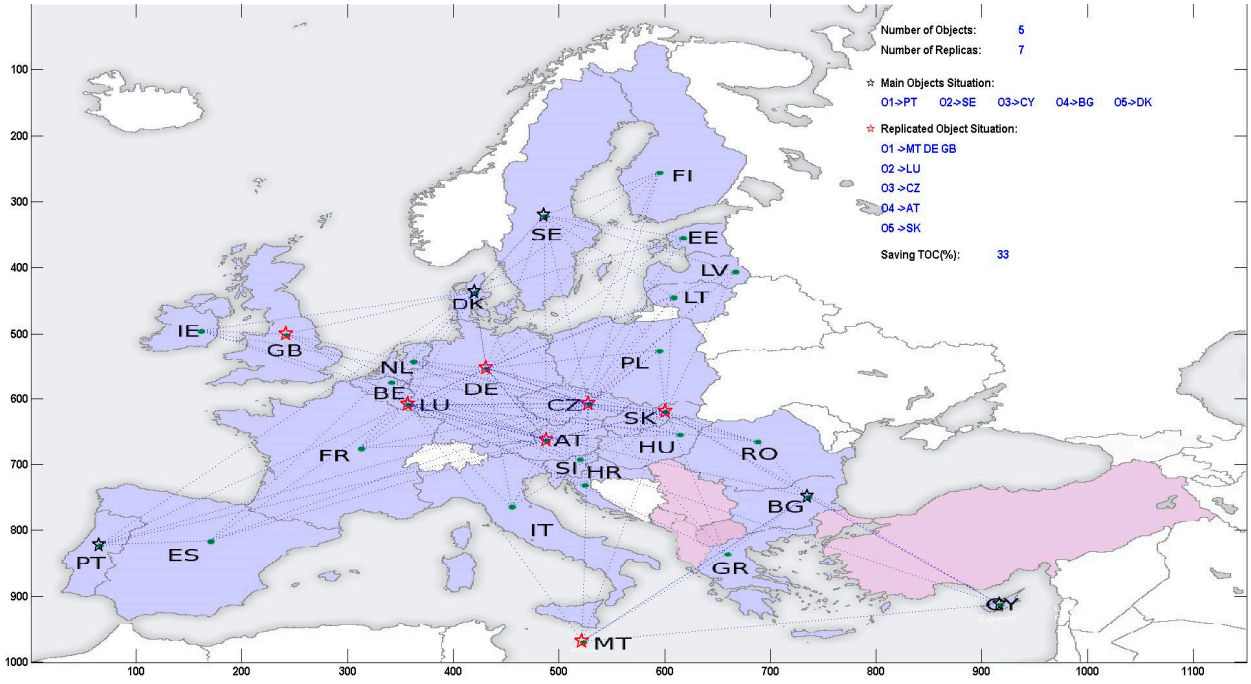


Figure 30. The replication of five data objects by the PSGWA and its location on the Europe map.

5. Conclusions and Future Work

Rapid data access must be addressed in the design of large distributed systems. In distributed systems such as the IoT, replica placement is an NP-complete problem. To reduce the total cost of data operations, the PSGWA technique has been presented as a discrete multi-swarm algorithm for replica management. Using the MATLAB programming language, the proposed approach was applied to a standard dataset. The proposed method is introduced to control and preserve the symmetry of the data access time, in the number of produced replicas, resulting in stability and reliability. On average, PSGWA reduces

data access time by around 35%; the figures for the PSO, GWO, SFLA, ACO, and GA are 25%, 28%, 30%, 18%, and 21%, respectively. The next challenge is to reduce the number of replicas generated for data objects to conserve allocation resources. The average number of data object replicas made by PSGWA, PSO, GWO, SFLA, ACO, and GA is 6, 7.2, 8.2, 12.50, 20, and 14. Indeed, the suggested PSGWA gives the shortest data access time with the fewest storage spaces. Furthermore, the proposed method has a faster convergence rate than PSO, SFLA, ACO, and GA. In this case, the PSGWA technique produced results with less standard deviations than the other algorithm. The suggested technique is more stable than the other algorithms during multiple executions. In fact, the PSGWA can produce the same outcomes in multiple runs. As a result, the PSGWA is more dependable and efficient in the replica-placement problem.

In this study, the system is assumed to be stable overall in this analysis; benchmarking the proposed method in dynamic systems is suggested as one future study. However, the operations (requests) are assumed to be a combination of reading and writing (updating) transactions. The other data transaction was not considered. In the current study's unstable state, the algorithm's execution time to arrive at the best response was not analyzed. The parallelization of the proposed optimization algorithm is the other concern that can be viewed as a future effort. The dynamic form of this technology is anticipated to be another subject of future research because the primary and replica data migrate throughout the operation for unique reasons in a dynamic setting (dynamic system). Reinforcement learning can be used to determine the best potential solution of the replica-placement problem. Reinforcement learning differs from supervised learning in that the training data contains the solution key, while in reinforcement learning, there is no answer, and the reinforcement agent determines what to do to find out the optimal solutions. Hence, development of the reinforcement learning method to sort out the replica-placement problem is one of the interesting future works. A variety of heuristic methods have also been developed and used to manage a wide range of optimization problems in computer engineering [19–28]. Hence. The effectiveness of the other discrete heuristic methods can be evaluated in the replica-placement challenge. Considering the data security in the data replication method is the other future study.

Author Contributions: Conceptualization, B.A. and S.H.; methodology, B.A.; software, B.A.; validation, B.A., S.S.S. and S.H.; formal analysis, B.A. and T.A.; investigation, S.H., O.F. and T.A.; resources, B.A.; data curation, B.A. and S.S.S.; writing—original draft preparation, B.A. and S.S.S.; writing—review and editing, B.A., S.S.S., S.H., O.F. and T.A.; visualization, S.S.S.; supervision, B.A. and S.H.; project administration, B.A., S.H. and O.F.; funding acquisition, S.H. and O.F. All authors have read and agreed to the published version of the manuscript.

Funding: The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The authors have no relevant financial or non-financial conflicts of interest. All authors contributed to this study. The data related to the current study are available on Google Drive and can be freely accessed by the following link: https://drive.google.com/drive/folders/1DpdtwqXiX7_kQSVPvB2QN4AoYxNQKgRu?usp=sharing (accessed on 4 January 2023).

Acknowledgments: This study has been partially conducted under the project 'Mobility and Training for Beyond 5G Ecosystems (MOTOR5G)'. The project has received funding from the European Union's Horizon 2020 program under the Marie Skłodowska Curie Actions (MSCA) Innovative Training Network (ITN), having grant agreement No. 861219.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Genetic Algorithm	GA
Particle Swarm Optimization	PSO
Ant Colony Optimization Algorithm	ACO
Gray Wolf Optimization Algorithm	GWO or GWA
Shuffle Frog Leaping Algorithm	SFLA
Particle Swarm based Gray Wolf Optimization Algorithm	PSGWA
Total data access operations (read and write) cost	TOC
Standard Deviation	STDV
Frog Leaping Algorithm	FLA
Local Best Individual	Pbest
Global Best Individual	Gbest
Velocity of Individual$_i$	Vi
Position of Individual$_i$	Xi

References

1. Qiu, L.; Padmanabhan, V.N.; Voelker, G.M. On the placement of Web server replicas. In Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society, Anchorage, AK, USA, 22–26 April 2001; Volume 3, pp. 1587–1596. [CrossRef]
2. Li, B.; Golin, M.J.; Italiano, G.F.; Deng, X.; Sohraby, K. On the optimal placement of web proxies in the Internet. In Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 21–25 March 1999; Volume 3, pp. 1282–1290. [CrossRef]
3. Szymaniak, M.; Pierre, G.; Van Steen, M. Latency-driven replica placement. In Proceedings of the 2005 Symposium on Applications and the Internet, Trento, Italy, 4 February 2005; pp. 399–405.
4. Ng, T.S.E.; Zhang, H. Predicting Internet network distance with coordinates-based approaches. In Proceedings of the Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 23–27 June 2002; Volume 1, pp. 170–179. [CrossRef]
5. Li, C.; Liu, J.; Lu, B.; Luo, Y. Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment. *J. Netw. Comput. Appl.* **2021**, *180*, 103017. [CrossRef]
6. Li, C.; Wang, Y.; Tang, H.; Zhang, Y.; Xin, Y.; Luo, Y. Flexible replica placement for enhancing the availability in edge computing environment. *Comput. Commun.* **2019**, *146*, 1–14. [CrossRef]
7. Li, C.; Bai, J.; Chen, Y.; Luo, Y. Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system. *Inf. Sci.* **2020**, *516*, 33–55. [CrossRef]
8. Safaee, S.; Haghighat, A.T. Replica placement using genetic algorithm. In Proceedings of the 2012 International Conference on Innovation Management and Technology Research, Malacca, Malaysia, 21–22 May 2012; pp. 507–512. [CrossRef]
9. Abawajy, J.H.; Deris, M.M. Data Replication Approach with Consistency Guarantee for Data Grid. *IEEE Trans. Comput.* **2014**, *63*, 2975–2987. [CrossRef]
10. Rambabu, D.; Govardhan, A. Optimization assisted frequent pattern mining for data replication in cloud: Combining sealion and grey wolf algorithm. *Adv. Eng. Softw.* **2023**, *176*, 103401. [CrossRef]
11. Shao, Y.; Li, C.; Fu, Z.; Jia, L.; Luo, Y. Cost-effective replication management and scheduling in edge computing. *J. Netw. Comput. Appl.* **2019**, *129*, 46–61. [CrossRef]
12. Fu, C.; Zhou, Y.; Han, J. A hardware-efficient dual-source data replication and local broadcast mechanism in distributed shared caches. *Microelectron. J.* **2021**, *118*, 105286. [CrossRef]
13. Subramanyam, G.; Lokesh, G.; Kumari, B.J. A Priori Data Replica Placement Strategy in Grid Computing. *Int. J. Sci. Eng. Res.* **2013**, *4*, 1070–1076.
14. Shamsa, Z.; Dehghan, M. Placement of replicas in distributed systems using particle swarm optimization algorithm and its fuzzy generalization. In Proceedings of the 13th Iranian Conference on Fuzzy Systems (IFSC), Qazvin, Iran, 27–29 August 2013; pp. 1–6. [CrossRef]
15. Kolisch, R.; Dahlmann, A. The dynamic replica placement problem with service levels in content delivery networks: A model and a simulated annealing heuristic. *OR Spectr.* **2015**, *37*, 217–242. [CrossRef]
16. Tu, M.; Yen, I.L. Distributed replica placement algorithms for correlated data. *J. Supercomput.* **2014**, *68*, 245–273. [CrossRef]
17. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
18. Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
19. Jalali, M.; Bouyer, A.; Arasteh, B.; Moloudi, M. The effect of cloud computing technology in personalization and education improvements and its challenges. *Procedia Soc. Behav. Sci.* **2013**, *83*, 655–658. [CrossRef]
20. Keshtgar, A.; Arasteh, B. Enhancing Software Reliability against Soft-Error using Minimum Redundancy on Critical Data. *Int. J. Comput. Netw. Inf. Secur.* **2017**, *5*, 21–30. [CrossRef]

21. Zadahmad, M.; Arasteh, B.; YousefzadehFard, P. A Pattern-Oriented And Web-Based Architecture To Support Mobile Learning Software Development. *Procedia Soc. Behav. Sci.* **2011**, *28*, 194–199. [CrossRef]
22. Bouyer, A.; Arasteh, B.; Movaghar, A. A New Hybrid Model Using Case-Based Reasoning and Decision Tree Methods for Improving Speedup and Accuracy. In Proceedings of the IADIS International Conference of Applied Computing, Salamanca, Spain, 18–20 February 2007.
23. Arasteh, B.; Sadegi, R.; Arasteh, K. Bolen: Software module clustering method using the combination of shuffled frog leaping and genetic algorithm. *Data Technol. Appl.* **2021**, *55*, 251–279. [CrossRef]
24. Arasteh, B.; Razieh, S.; Keyvan, A. ARAZ: A software modules clustering method using the combination of particle swarm optimization and genetic algorithms. *Intell. Decis. Technol.* **2020**, *14*, 449–462. [CrossRef]
25. Arasteh, B.; Miremadi, S.G.; Rahmani, A.M. Developing Inherently Resilient Software Against Soft-Errors Based on Algorithm Level Inherent Features. *J. Electron. Test.* **2014**, *30*, 193–212. [CrossRef]
26. Arasteh, B.; Fatolahzadeh, A.; Kiani, F. Savalan: Multi objective and homogeneous method for software modules clustering. *J. Softw. Evol.* **2022**, *34*, e2408. [CrossRef]
27. Ghaemi, A.; Arasteh, B. SFLA-based heuristic method to generate software structural test data. *J. Softw. Evol. Proc.* **2020**, *32*, e2228. [CrossRef]
28. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]