FACTA UNIVERSITATIS
Series: **Mechanical Engineering**
https://doi.org/10.22190/FUME230707033A

Original scientific paper

DATA REPLICATION IN DISTRIBUTED SYSTEMS USING OLYMPIAD OPTIMIZATION ALGORITHM

Bahman Arasteh[1], Asgarali Bouyer[1,2], Reza Ghanbarzadeh[3], Alireza Rouhi[2], Mahsa Nazeri Mehrabani[4], Erfan Babaee Tirkolaee[5,6,7]

[1]Department of Software Engineering, Istinye University, Istanbul, Türkiye
[2]Department of Software Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
[3]Faculty of Science and Engineering, Southern Cross University, Gold Coast, Australia
[4]Department of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran
[5]Department of Industrial Engineering, Istinye University, Istanbul, Türkiye
[6]Department of Industrial Engineering and Management, Yuan Ze University, Taiwan
[7]Department of Industrial and Mechanical Eng., Lebanese American University, Byblos, Lebanon

Abstract. *Achieving timely access to data objects is a major challenge in big distributed systems like the Internet of Things (IoT) platforms. Therefore, minimizing the data read and write operation time in distributed systems has elevated to a higher priority for system designers and mechanical engineers. Replication and the appropriate placement of the replicas on the most accessible data servers is a problem of NP-complete optimization. The key objectives of the current study are minimizing the data access time, reducing the quantity of replicas, and improving the data availability. The current paper employs the Olympiad Optimization Algorithm (OOA) as a novel population-based and discrete heuristic algorithm to solve the replica placement problem which is also applicable to other fields such as mechanical and computer engineering design problems. This discrete algorithm was inspired by the learning process of student groups who are preparing for the Olympiad exams. The proposed algorithm, which is divide-and-conquer-based with local and global search strategies, was used in solving the replica placement problem in a standard simulated distributed system. The 'European Union Database' (EUData) was employed to evaluate the proposed algorithm, which contains 28 nodes as servers and a network architecture in the format of a complete graph. It was revealed that the proposed technique reduces data access time by 39% with around six replicas, which is vastly superior to the earlier methods. Moreover, the standard deviation of the results of the algorithm's different executions is approximately 0.0062, which is lower than the other techniques' standard deviation within the same experiments.*

Keywords: *Olympiad optimization algorithm, Distributed systems, Replica placement, Data access time, Stability, Availability*

Received: July 07, 2023 / Accepted October 05, 2023
Corresponding author: Erfan Babaee Tirkolaee
Department of Industrial Engineering, Istinye University, 34396 Istanbul, Türkiye
E-mail: erfan.babaee@istinye.edu.tr

1. INTRODUCTION

Quick access to data items would be critical when designing big, distributed systems like cloud computing and the Internet of Things (IoT). Data retrieval takes a longer time in distributed systems. As a result, system designers now place a high priority on lowering the operation time of data read and write in distributed systems. This issue can be addressed by storing several copies of the data across different servers, allowing for faster data access from distant places. Replication in this context refers to generating identical duplicates of the data on other servers. The placement of copies, which could be done either statically or dynamically, is often important to the performance of distributed systems. Decreasing access time to data objects in distributed systems is a challenging issue. Data objects and their replicas are distributed across numerous servers geographically isolated in a distributed system such as IoT. The primary objective of data replication in such systems is to improve the system's dependability and efficiency. The number and location of replicas stored on various servers are critical parameters in the problem of replica placement. The research problem includes determining the optimal quantity of replicas, searching for optimal servers, storing data in the servers, and finally reducing the overall cost of the system for data processing. Determining the data replicas' optimal locations is an NP-complete problem.

To address the problems of placement and replacement of data replicas within distributed systems, the current study proposes a novel discrete optimizer algorithm. The new algorithm is inspired by the process of group teaching and learning of students of a class for an Olympiad exam, which was named the Olympiad Optimization Algorithm (OOA). In OOA, the population of individuals includes and simulates the behavior of a set of students who attempt to get the best learning from their teammates. There is a competition between the individuals (students) to receive the best level of learning and preparation for the Olympiad exam. OOA solves an optimization problem based on the divide and conquer approach, including local and global search strategies. In each iteration, the population is divided into groups of students, and the students are engaged in teaching and learning activities. In OOA, no teacher individuals are involved in the learning process, and the population evolution is performed based on swarm intelligence and group-based learning. The relevant solution spaces for the optimal solution are searched by each subgroup. Solutions describe the quantity of replicas for each data object and their locations on the data servers; the fitness of a solution indicates the data access time provided by the solution. Furthermore, OOA can be utilized in addressing discrete optimization problems. The primary achievements of this study can be outlined as follows:

I. Olympia Optimization Algorithm (OOA) is applied to address the problem as a novel discrete optimization algorithm. This method draws inspiration from the collaborative learning dynamics of students preparing for an Olympiad exam, employing a swarm-based approach. Throughout each iteration, the algorithm simulates the teaching and learning processes among the population members (akin to students), driving the evolution of the population. This proposed algorithm adopts a divide-and-conquer strategy, incorporating both local and global search techniques. The individuals within the population are partitioned into subgroups, each employing a specific imitation mechanism to explore distinct regions within the solution space. During the global search phase, students emulate the behaviors of the best-performing students in the entire population.

II. OOA can be used to address the majority of graph-based optimization problems.
III. OOA is adopted in solving the issue of replica placement in distributed systems as a routing problem in a graph.
IV. A minimum data access time with a lower quantity of produced replicas is provided.
V. The development of a standard case study platform to evaluate the replica placement algorithms is the other contribution of this study. In the case study, the European Union Database (EUData) was employed to evaluate the suggested algorithm; this database contains 28 nodes as servers and a network architecture in the format of a complete graph. The experiments involved running the case study using varying quantities of data items.

The remainder of the current paper is structured as follows. Fundamental definitions associated with the problem of replica placement as well as previous works are reviewed in Section 2. Section 3 states the proposed OOA and its adoption to the replica placement problem. The case study and developed simulation platform were explained in the first part of Section 4. The results attained by the OOA are evaluated and compared with the results of other methods in the second part of Section 4, followed by broad conclusions in Section 5.

2. LITERATURE SURVEY

2.1 Background

'Replicas' are extra copies of data objects stored on various servers to provide quicker remote data access, and 'replica replacement' is the process of generating perfect clones of the same data on various servers. Generally, the performance of distributed systems like IoTs depends on the placement of copies (it can be accomplished either statically or dynamically). In recent years, various strategies have been developed addressing the problem of replica placement, each with its unique structure, advantages, and disadvantages. Algorithms like greedy, tree-based, hot spot, and hot region are this field's primary replica placement techniques. These algorithms consider the majority of client load while randomly distributing M copies of data items across N servers. The data is distributed using the normal distribution approach in this manner. This process must be repeated several times to achieve the desired result, with the best result being chosen. One of the main benefits of this method is its ease of implementation; however, due to the algorithm's randomness, obtaining the most and the best-desired results is difficult [1]. The following is a review of some of the pertinent algorithms applied to the problems of replica placement.

2.2 Greedy and Tree-based Algorithms

After comparing all N servers, the best server, as well as a replica version, is selected by this algorithm. The lowest-cost service provider is chosen for delivering the replica version after evaluating the costs of each customer's access to it. A similar method is applied to the remaining versions in the next cycles. When addressing data access, it is assumed that the most recent replica is used by each client. The main advantage of this algorithm is its quick execution, whereas the main drawback is that determining the

appropriate data distribution strategy could be challenging [1]. This is implemented through dynamic programming. Although this method was initially used in positioning the web proxy, it could also be employed in placing the replicas in distributed systems. Network connectivity in distributed systems will likely take the form of a tree. At the uppermost level of this technique's smaller sub-trees, each one owns a communication interface with the web server. A request is sent by each client to the corresponding interface, which then forwards it to the webserver to connect with that [2].

2.3 Hot Spot-based and Workload-Aware Algorithms

The other technique is storing multiple data replicas close to clients with heavy workloads. Accordingly, data is split among service providers according to the volume of work produced by their clients; to *M* service providers that are surrounded by many workloads, *M* replicas are distributed. This algorithm estimates the volume of work traffic by using the radius of the neighborhood around each service provider [1]. Szymaniak et al. [3] and Ng and Zhang [4] introduced a common area-based placement approach. This algorithm's main objective is to accelerate placement operations. Through the application of this algorithm, the network is divided into multiple regions based on node delays; each region is known as a cell. The cells are then sorted by a radix-type technique, and *k* cells are selected as placement agent servers. The algorithm uses the GNP approach to split the network and map it to Euclidean space. The typical node delay is determined by the GNP approach, before converting the space of the network to Euclidean coordinates. This algorithm computes the density of nodes for each coordinate within every network area and subsequently selects the areas with the node with the highest density.

This method investigates the process of automated scaling as well as dynamic replica placement. To reduce the scaling replicas' and other resources' overall cost, the automatic scaling method based on service overhead is first recommended. Before resource assignment, a hybrid load forecasting technique is utilized for workload prediction. The overall cost is calculated based on the workload. When the constraints are recognized, the optimization problem could be considered a problem of linear programming. Then using the Tabu algorithm, the optimum scaling strategy is chosen. The introduced data placement technique reduces the normal processing time of dynamic replica placement. This replica placement technique offers a more evenly distributed storage capacity than the relevant methods [5].

2.4 User Experience-based Algorithms

This approach offers an adaptable strategy for replica placement within an edge computing environment, with the objective of evaluating the relationships among user access patterns, the quantity of replicas, and their specific placements. In this study, the replica placement approach was created to identify the ideal locations for replicas. This would improve the performance of data access and cloud storage. This method employs the dynamic replica creation algorithm (DRC-GM). DRC-GM addresses the irregular nature of user access by utilizing the data block as the data unit. To meet the data availability requirement, DRC-GM dynamically adjusts the quantity of data copies, taking into account the connection between data access frequency and the number of replicas. The findings indicate that the DRC-GM algorithm can greatly improve system performance in an edge computing environment. This improvement includes enhanced prediction

accuracy, faster access response times, increased utilization of network and storage space, and improved data availability [6]. This method employs a dynamic replica allocation strategy to enhance user experience. The replica consistency preservation method ensures that the data is correct and consistent. The recommended replica management strategy may greatly decrease the rented nodes' total cost (server) as the duration lengthens. In comparison with previous strategies, the suggested method can cut overall costs by up to 32.27% and 53.65%, respectively. The proposed replica allocation technique has the potential to significantly minimize storage overhead as well as the delay of data transmission [7].

2.5 Heuristic-based Replica Placement Algorithms

This approach employs Genetic Algorithm (GA) to identify the optimal locations for the generated replicas. The algorithm takes two key inputs: the Euclidean coordinates representing the network and the required number of servers for placement. The algorithm unfolds in two main steps. First, it calculates the Euclidean coordinates of the network. Second, it assesses the density of each region, including the count of nodes within the target region. Subsequently, it chooses the center for each region to serve as a pivotal point or as the cluster's core after each region is aggregated. In the following phase, the density of each cluster is determined through the compatibility function. Based on the density magnitude, two clusters are consistently selected from the pool of good clusters, and the intersection operation is performed. The prior coordinates are then swapped out for the new coordinates in the third stage, which involves a random leap on the new coordinates. The approach is repeated until many excellent clusters are found as placement servers [8].

This strategy increases availability by enhancing the number of existing clones. Usually, the two objectives of increasing the distance between comparable exchanges while decreasing the distance between distinct exchanges must be accomplished. By measuring the bandwidth of the shortest link between two sites, Dijkstra's approach can be used to determine their distance. While the plan is carried out, the network is fixed. The websites are colorized after the list of every file on the network is collected. A collection of locations is created that consists of an original file (data item) and optionally multiple copies (replicas) of the original file. Subsequently, the file that has the greatest distance from the nearest one is selected and substituted [9]. Table 1 demonstrates a list of relevant methods with their advantages and disadvantages.

The replica placement algorithm can be investigated in static and dynamic models. The dynamic nature of the requests and resources has not been considered in the proposed method. Hierarchical and greedy-based algorithms [14, 15, 16] can be used to replicate the data objects in the dynamic distributed systems. In order to alleviate the replication time, the data servers can be clustered by different clustering and machine learning algorithms [17, 18, 19]. Furthermore, the data servers can be grouped (partitioned) based on their features before being selected by the replication algorithms [20, 21, 22]. Different methods have been proposed to manage the big data objects over the distributed systems' servers [23, 24, 25].

Detecting anomalies in multivariate time series data is of paramount importance for ensuring the overall performance and reliability of cloud-based distributed systems. Given the intricate and rapidly changing nature of these systems, anomaly detection poses a significant challenge. To address these issues, different deep learning-based methods were

proposed [26, 27]. The IoT as a large distributed system with numerous diverse physical devices needs a dynamic services coordination method; the coordinating method can integrate the heterogeneous devices and data storages into the context aware IoT infrastructure. Different studies have proposed the situation aware IoT services coordination approach [28]. Different clustering and classification methods that have been developed in the previous studies [29, 30, 31] can be used to cluster the data objects and data storages in the distributed systems. Furthermore, different metaheuristic algorithms with swarm intelligence features can be used to find out the best location of the data replicas [32, 33].

Table 1 Related methods and their pros and cons

Method	Advantage	Disadvantage
Flexible Replica Placement [6]	Quick response time, high data availability	Data-file types and node types are not considered
Experience-based Replica Placement [7]	High performance in big distributed systems and low financial and storage costs	Collaborative resource management is not considered
Replica placement using GA [8]	Reduced run time and optimum placement	In large distributed systems, it shows a low performance
Consistency-based Replica Placement [9]	Low response time and Higher data availability	It is only suitable for fixed networks during the execution of the method
PSO and fuzzy-based replica placement algorithm [10]	N/A	The data write transaction performance is low
Replica placement with service and content delivery networks [11]	High stability	There is a probability of being trapped in a local optimum
Correlated data-replicas placement [12]	Low response time	The performance for the data writes transaction is low
Priority-based replica management [13]	Low average response time and capability of fault tolerance	It is only suitable for static systems

3. PROPOSED METHOD

Data retrieval could take a very long time in a distributed system. Therefore, decreasing the required time for data objects to read and write in distributed systems is particularly difficult. As discussed earlier, a common approach to address this problem is to store numerous clones of the data on multiple servers (replicas) for easier access from a particular distance. A novel discrete heuristic swarm-based algorithm, OOA, has been proposed in the current study to address the problem of replica placement. Compared to various other methods, the application of OOA can reduce data access time in static contexts.

3.1 Olympiad Algorithm

3.1.1 Algorithm Structure

The proposed OOA employs a swarm-based imitation technique as both its local and global search methods [34]. This heuristic method operates with a population and group-based approach, solving optimization problems through a divide-and-conquer framework. In this approach, each member of the population within OOA mimics the behavior of a student in a classroom setting, particularly those students preparing for an Olympiad exam. The iterative teaching and learning processes among the population members (kind to students) drive the evolution of the population. This novel algorithm follows a divide-and-conquer strategy, integrating both local and global search strategies. Individuals within the population are grouped into subgroups, each employing a specific imitation approach to explore distinct regions within the overall solution space. Fig. 1 provides an overview of OOA's general workflow, illustrating its applicability to solving optimization problems. Competition exists among the individuals (students) as they learn from one another. Each student maintains a memory of their learning rate, representing their position in the learning process. In the OOA, each student is represented as a numeric array, and the student population consists of various solutions.

As depicted in Fig. 1, the initial phase of OOA, following the sorting process, entails dividing the student population into *n* teams, where each team consists of m students. The foremost team is recognized as the global best team, while the final team assumes the role of the global worst team. Within each team, comprised of students, exploration of their respective local solution space takes place. The primary student in the initial team is designated as the global best student, while in each successive team, the first student serves as the local best student.

3.1.2 Learning Procedure

Within OOA, students within teams aim to acquire knowledge from the best student in either the adjacent team (referred to as the local best student) or the top-performing global best team. The learning operator, a fundamental component of OOA, facilitates both local and global search operations. In essence, learning serves as the primary mechanism for OOA to seek out the optimal solution, with the aim of enhancing the population's overall knowledge or fitness. The introduction of the learning operator is intended to bolster the population's knowledge. Individuals (students) are sorted based on their knowledge, as demonstrated in Fig. 2, illustrating OOA's endeavor to enhance the population's knowledge through this operator. The learning operator encompasses four key steps. In the initial step, students within each team seek to learn from the students in their adjacent team. Following the initial phase of the learning operator, knowledge from the first team is propagated to other teams in a manner reminiscent of the bubble sort algorithm. Fig. 3 visually illustrates this first step of the proposed learning operator, wherein a team's knowledge is transferred to the neighboring team. This knowledge transfer process is carried out sequentially, mirroring the operation of the bubble sort algorithm, as knowledge gradually moves from one team to its adjacent counterpart.

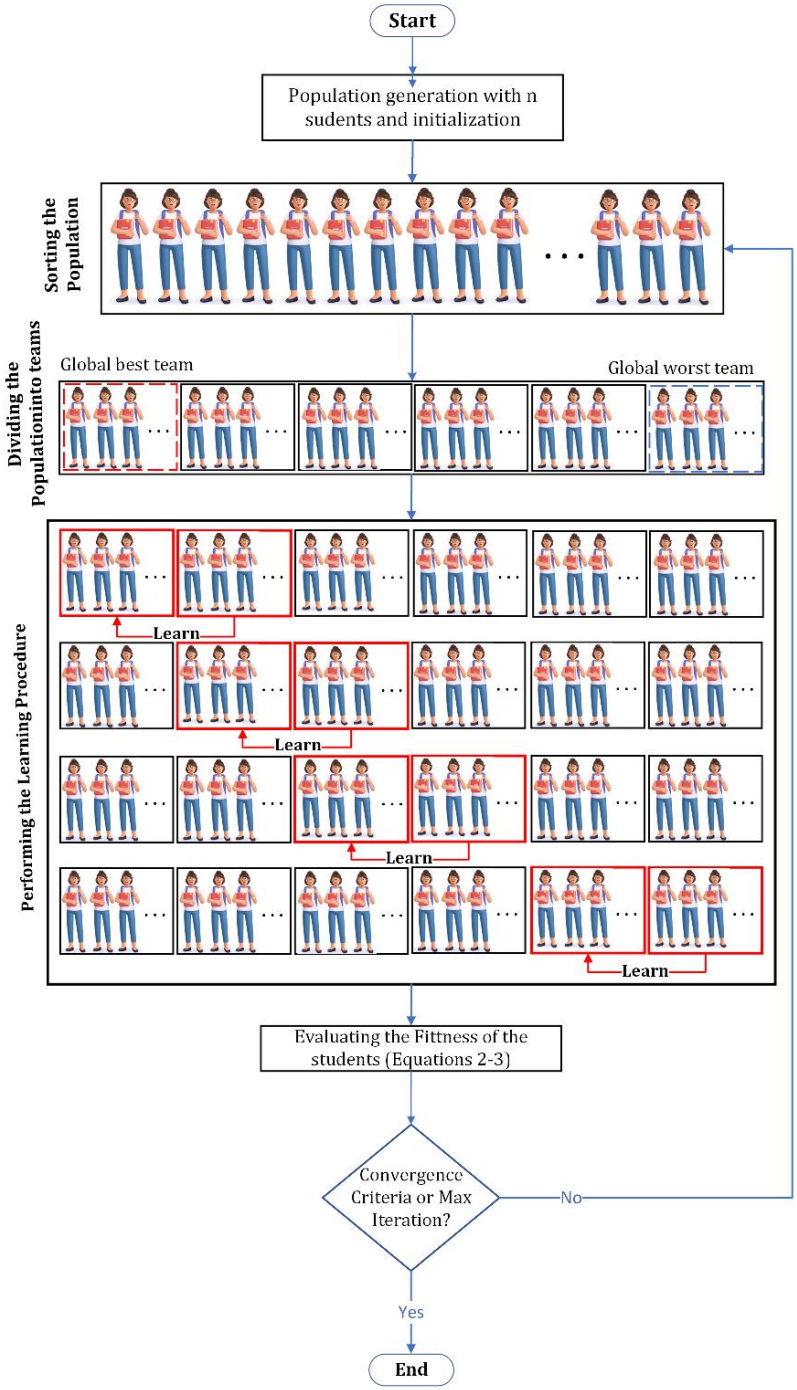

Fig. 1 Workflow of the proposed OOA [34]

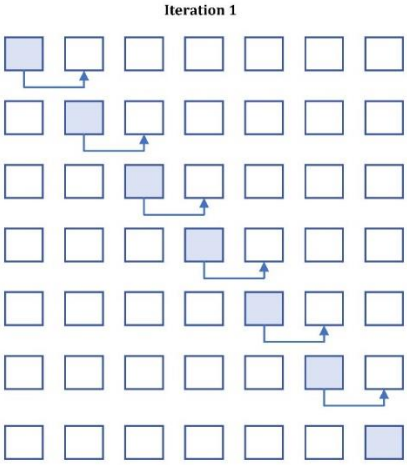

Fig. 2 First step of learning algorithms

As portrayed in Fig. 3, all students within a learning team absorb knowledge from their corresponding peers in the adjacent team located on the left side. If there is no noticeable improvement in the knowledge levels of the less proficient students in the weak team, the second stage of the learning process is initiated. During this second improvement stage, the best student from the learning team (belonging to the right-side team) undergoes a mutation process to introduce diversity. The mutation operator conducts a localized search on the best student from the weaker team. In situations where no substantial improvement is observed in this second step, the third step is put into action. In the third step, all students in the learning team (the weaker team) acquire knowledge from their counterparts in the global best team (the first team).

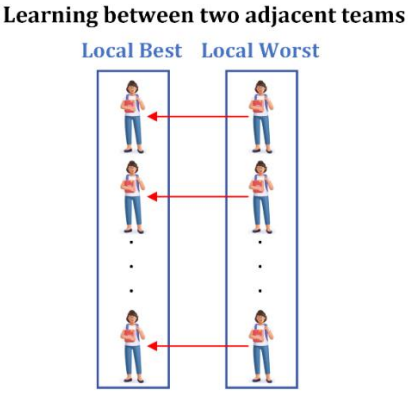

Fig. 3 Learner team's students learn from the adjacent team's students

In situations where the students from the global-best team are unable to effectively instruct the students in the worst team, the mutation operator is invoked on the global best

student. Mutating the global best student introduces a slight degree of diversity, which can help prevent the algorithm from converging to a local optimum. Ultimately, students (referred to as search agents) from various teams are amalgamated to generate a new population. The learning operator is executed iteratively on this student population, as part of the ongoing optimization process. This iterative learning process allows for the continual refinement of the population's knowledge and, consequently, enhances the algorithm's ability to find optimal solutions to the given problem.

3.2 Adapting the Olympiad Algorithm to Replica Placement Problem

3.2.1 Mathematical Specification of The Problem

In distributed systems, replicating data objects on multiple servers is necessary to improve data access time. The problem at hand involves server number *S(n)* as well as data item number (k), where *k* is within the range $1 < k \leq K$ and n is within the range $1 < n \leq N$. The corresponding capacities of service provider *n* and data item *k* are denoted by $C(n)$ and $V(k)$, respectively. The communication cost between service providers $S(n)$ and $S(m)$ is represented by integer $l(nm)$, and the data transport cost between servers $S(n)$ and $S(m)$ is equivalent to the communication cost. It is assumed that $l(nm)$ is equal to $l(mon)$ in this context. Additionally, the variables read(n,k) and write(n,k) indicate the number of read and write requests for the object (k) from servers S(n). Each data object has a primary server, $P(k)$, which possesses the initial version of the object. It is important to mention that object (k) cannot be directly transferred to a different server in its original state. The servers where the object (k) is replicated are recorded in a distinct list referred to as $RS(k)$ for each primary server. When server $S(n)$ intends to perform a read operation on data object *k*, it should select the nearest server that holds either the original or replica versions of the object. During the writing process, the associated server sends a data update request (k) to the main server. In this scenario, the primary server $P(k)$ broadcasts a message to all the servers that host object (k), and every server updates the corresponding data item accordingly. The primary goal of the problem of replica placement is distributing the replicas among servers to minimize the total read and write operations cost and enhance the data availability.

The proposed approach's initial step involves transforming replica placement into the well-known common traveling salesperson problem (TSP). To illustrate this, the Performance-Sensitive Genetic Algorithm (PSGWA) was employed to assess various methods to replicate 2 data items across 6 distinct servers in a scenario where there exist 2 data objects and 6 servers. Each data item is assigned a unique ID number (*Kn* denotes the unique ID of the nth data object). There are four distinct modes of replicating 2 data items (*K*1 and *K*2) on a given server (S). Fig. 4(a) presents the configurations of replicating 2 data items across the 6 data servers. Various modes are as follows:

- 1st mode: Neither *K*1 nor *K*2 is cloned on server *S*.
- 2nd mode: Only *K*1 is cloned on server *S*, but not *K*2.
- 3rd mode: Only *K*2 is cloned on server *S*, but not *K*1.
- 4th mode: Both *K*1 and *K*2 are cloned on server *S*.

The columns of the generated graph represent the different modes of the servers, while each row corresponds to a specific server. Each solution path acts as a representation of the desired locations for the data item's replica. Each path's fitness value in the graph is

determined based on the objective function. Like the Traveling Salesperson Problem, the optimal path is the one that has the shortest distance according to the fitness function. Fig. 4(b) illustrates the process of replica placement to replicate 2 data objects across 6 servers. It presents a placement model that indicates the specified path of the 2 data-object copies on the 6 servers ([2,3,1,1,3,4]). This path can be interpreted as follows:

- Server (1) exclusively stores the *K*1 data item's replica.
- Server (2) exclusively stores the *K*2 data item's replica.
- Server (3) possesses no replicas.
- Server (4) possesses no replicas.
- Server (5) exclusively stores the replica of the *K*2 data item.
- Server (6) stores both *K*1 and *K*2 data items' replicas.

Ultimately, the problem of replica placement was successfully mapped into the Traveling Salesperson Problem. Fig. 5 showcases the ultimate representation of the graph of replication as depicted in Fig. 4(b). The same problem has been addressed using a different technique [26, 27].

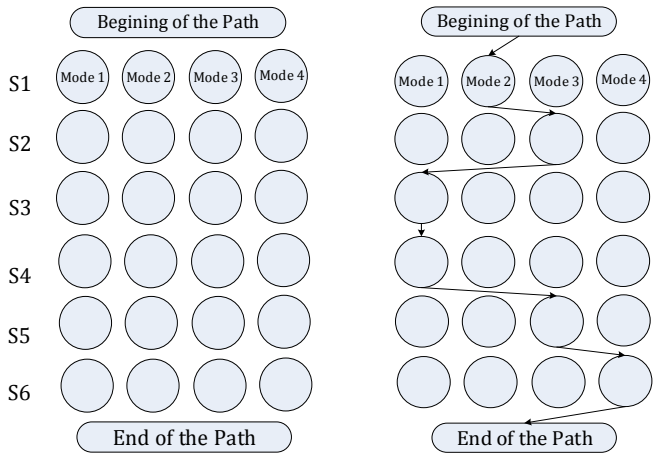

Fig. 4 Mapping the replica placement problem's search space to the TSP problem's search space: (a) Two data object's potential replication model over six 2 data items in 6 distinct servers and (b) A placement path example when replicating distinct data servers.

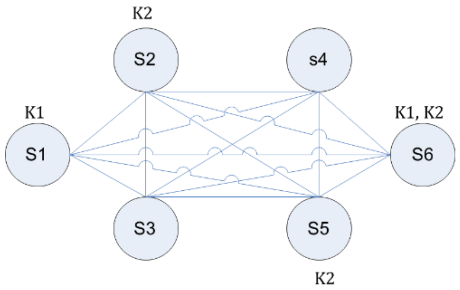

Fig. 5 Replica placement graph

3.2.2 Individual Structure

Each student is represented by a numeric array in the replica placement issue. Fig. 6 displays a student's structure in the replica placement issue. Each cell in a student array contains information about the server mode for that instance. The server index could be calculated using the student array's index. Each student has a replication path within the population. The path in Fig. 4(b) is the same as the one in the student array depicted in Fig. 6.

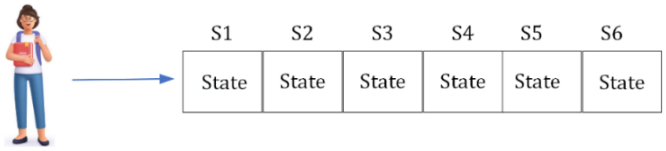

Fig. 6 Structure of a student in OOA is used in solving the problem of replica placement in the case of six data servers

Each replication array's fitness is assessed by applying the fitness cost function. When the replication array is sorted according to fitness, the population is categorized into n teams. The term "student knowledge" in this study refers to a student's fitness; that fitness (replication array) reflects the proximity to the ideal solution. Fitness is proportional to the time it takes to get the data. The required cost and time to access data in the server will be changed by any replica placement. The best student specifies the location of a data replica with the shortest access time. Each student array in the initial population is randomly initialized. If one of the edges (or nodes) in the graph of the problem of replica placement fails, the algorithm could still quickly determine the best new condition. As a result, if a node is removed, OOA can quickly determine the optimal (shortest) path because it is no longer required to start from the beginning.

3.2.3 Fitness Function

The total cost of operations associated with the data item k is calculated using Eq. (1), where $Access_R(k)$ indicates all read operations for data item k from all servers that have received read requests for item k. Eq. (2) calculates the value of $Access_R(k)$. In Eq. (2), $NS(nk)$ denotes the nearest or the least cost server that possesses a clone of object k. Furthermore, in Eq. (3), all write or update operations on data item k from all servers involved in the update request are referred to as $Access_W(k)$. Eq. (3) calculates the value of $Access_W(k)$, and Eq. (4) is used to calculate the total data operation cost (TOC) of the whole system for all data items. The aim is to reduce the TOC parameters, and OOA was used to find the best data replica placement with a minimum amount of TOC.

$$TOC(k) = Access_R(k) + Access_W(k) \tag{1}$$

$$Access_R = V(k) \times \left(\sum_{n=1}^{N} read(n,k) \times l\big(n \times NS(n,k)\big)\right) \tag{2}$$

$$Access_w = V(k) \times \sum_{n=1}^{N} \left[write(n,k) \times \left[l\big(n,p(k)\big) + \sum_{\forall j \in RS(k), j \neq n} l(p(k),j)\right]\right] \tag{3}$$

$$TOC = \sum_{k=1}^{K} TOC(k) \tag{4}$$

4. CASE STUDY AND RESULTS

4.1 Case Study and Simulation Platform

The 'European Union standard Database' (EUData) was employed, as a case study, to examine the suggested algorithm; this case study database contains 28 nodes as servers and a network architecture in the format of a complete graph. The vertices of the graph $G(V, E)$ depict the nations of the 'European Union' ($|V| = 28$). Furthermore, E represents the network communication equipment's cost which is equal to the edges of the graph. To evaluate the introduced method's performance in the EUData case study, a simulation platform was developed in MATLAB. In the developed platform, the other techniques, such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Shuffle Frog Leaping Algorithm (SFLA), and Grey Wolf Optimization (GWO), were implemented along with the OOA. The implemented algorithms were executed with different settings in the developed platform. The simulations were carried out for all algorithms on the same hardware with Microsoft Windows using MATLAB. Table 2 demonstrates the calibration parameters.

Table 2 Calibration parameters of various replication methods adjusted experimentally

Algorithm	Parameters	Value
GA	Quantity of chromosomes	150
	Chromosome length	$28 \times k$
	Crossover rate	0.7
	Mutation rate	0.05
SFLA	Quantity of frogs	96
	Size of Memeplex	12
	Quantity of Memeplexes	8
	FLA iterations (Beta)	7
	Quantity of Iterations	100
GWA	Quantity of wolves	50
	a	[0, 2]
	C	Variable
	A	Variable
	$r1, r2$	Random values in [0, 1]
PSO	Quantity of particles	30
	Inertia Weight	0.8
	Ratio of Inertia Weight Damping	0.99
	Particle.C1 and Particle.C2	1.8
	Quantity of iterations	100
OOA	Quantity of students	40
	Quantity of teams	4
	Size of teams	10
	Learning rate	Random values in [0.2, 0.8]
	Imitation count	1
	Quantity of iterations	100

The following are the examination criteria applied in the current study:

- Total Operation Cost (TOC), which is the data access operations' cost (read and write)
- The quantity of generated replica items by the replica placement algorithm.
- The algorithm's convergence speed.
- Algorithm's reliability in addressing the problem of the replica placement.
- Algorithm's stability during various executions for the same data item.

The experiments involved running simulations using varying quantities of data items, ranging from one to five. The connectivity costs between 28 data servers were uploaded as a matrix and uploaded in a free access repository [58]. The details of 28 data servers employed in the experiment are demonstrated in Table 3 and geographically distributed over the 28 data servers situated all around Europe. In the simulations, servers' number' 5', '17', '9', '22', and '11' hold significant importance as they are considered key servers for storing original data items.

Table 3 Details of data servers

Server number	Server name	Country name	Server capacity in GBs
1	FI	Finland	10021
2	SE	Sweden	21451
3	EE	Estonia	21110
4	LV	Lithonia	30000
5	LT	Lithuania	32002
6	DK	Denmark	34555
7	PL	Poland	46111
8	CZ	Czech	48121
9	SK	Slovakia	42121
10	HU	Hungary	10001
11	AT	Austria	10021
12	RO	Romania	21451
13	IT	Italy	21110
14	SL	Slovenia	30000
15	BG	Bulgaria	32002
16	GR	Greece	34555
17	CY	Cyprus	46111
18	MT	Malta	48121
19	PT	Portugal	42121
20	ES	Spain	10001
21	FR	France	34555
22	DE	Deutschland	46111
23	LU	Luxembourg	48121
24	BE	Belgium	42121
25	NL	Netherlands	10001
26	GB	Great Britain	34555
27	IE	Ireland	46111
28	HR	Croatia	48121

Table 4 lists the five original data items employed in the simulations. All simulations incorporated various quantities of data pieces, ranging from 1 to 5 (where $1 < k < 5$). Each data access request in the simulations involved accessing five data items, with each number representing the corresponding server. Table 5 lists the data-access requests employed in the simulations, encompassing both read and write operations and request settings.

Table 4 Volume/location of original data items employed by simulation

Data Object	***K*1**	***K*2**	***K*3**	***K*4**	***K*5**
Volume (in GBs)	0.1	0.15	0.16	0.18	0.2
Primary Hosted Server	5 (Lithuania)	17 (Cyprus)	9 (Slovakia)	22 (Deutschland)	11 (Austria)

Table 5 Read/write requests employed by simulation (adapted from [28])

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

4.2 Results

When it comes to the challenge of replica placement in distributed systems, k data items are required to be cloned across N servers. In such a situation, S(n) indicates the *nth* instance of the server, whereas Item(k) represents data object k. To identify the most suitable replica locations, a series of experiments were conducted using five different optimization algorithms: GA, ACO, SFLA, GWO, PSO, and OOA. To identify the optimal location for the clones, five sets of experiments were undertaken using GA, ACO, SFLA, GWA, PSO, and OOA algorithms. In the primary set of the test, the data items were replicated on twenty-eight servers. TOC and the quantity of the generated data item clones were utilized for results comparison. Every technique produces a different number of data replicates over different servers. Consequently, there is variation in the time of accessing data and the number of replicas for each algorithm. The first experiment's findings are depicted in Fig. 7. Twenty-eight servers were used in the initial stage of this experiment to hold copies of data items created by each algorithm. The overall data access time was computed by conducting 56 read/write operations on the replicas.

Various algorithms may employ a varying quantity and distribution of servers for storing data items. Therefore, the quantity and placement of copies impact total access time. An experiment with different numbers of data items was conducted similarly. The proposed technique is to identify the optimal replica's locations over the data servers. Fig. 7 displays the optimal TOC obtained from the GA, ACO, SFLA, GWA, PSO, and OOA algorithms. If the value of k is set to 5, the total operation counts for the 56 operations of accessing data are as follows for OOA, PSO, GWA, SFLA, ACO, and GA algorithms: 0.673900 seconds, 0.802200 seconds, 0.816300 seconds, 0.824100 seconds, 1.057800 seconds, and 0.930660 seconds, respectively. According to the results, OOA outperforms the other methods in data success time. When four data objects are replicated, similar results are achieved; the total replica data access time generated by OOA, PSO, GWA, SFLA, ACO, and GA is 0.638000, 0.760350, 0.791470, 0.768980, 0.945750, and 0.928160 seconds, respectively. OOA manages replicas with a lower TOC compared to the previous techniques. Fig. 7 shows that, in terms of the TOC criterion, PSO and GWA perform similarly in the replica placement problem. OOA takes advantage of both algorithms' strengths and outperforms GWA and PSO.

In all benchmarks, OOA has a lower TOC in comparison with the previous techniques. Fig. 8 demonstrates how different methods reduce data-access time. In terms of reduction of TOC, OOA outperforms the prior approaches in all benchmarks, and all algorithms perform equally in the modest benchmarks (k=1). The TOC of various methods varies significantly across all benchmarks. The TOC criteria of the proposed technique demonstrate an almost linear increase as the quantity of data objects grows. When k exceeds five, a statistical interpolation approach is employed to estimate an unknown TOC value. In the context of large datasets, interpolation is an approach employed to estimate the replica-placement algorithms' performance, which involves utilizing additional known values within the same sequence as the unknown value to estimate its value. If a consistent pattern can be observed among the collected data points, it is possible to predict the success of the replica placement techniques.

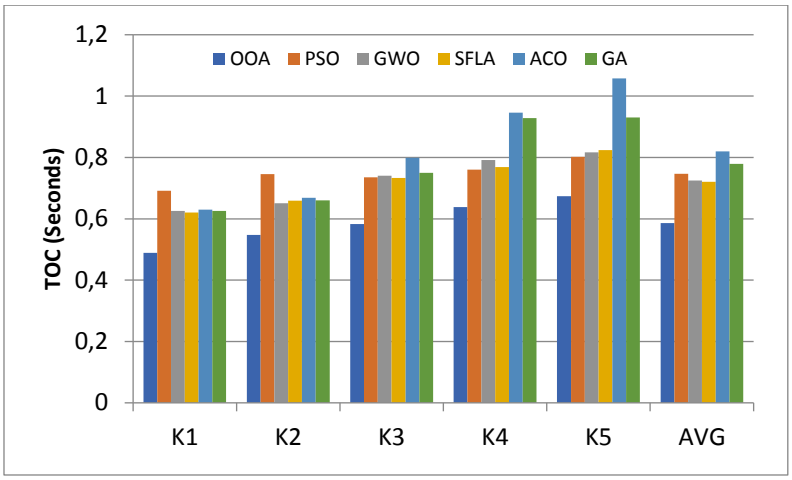

Fig. 7 Greatest time of data access achieved by various methods for various quantities of data items

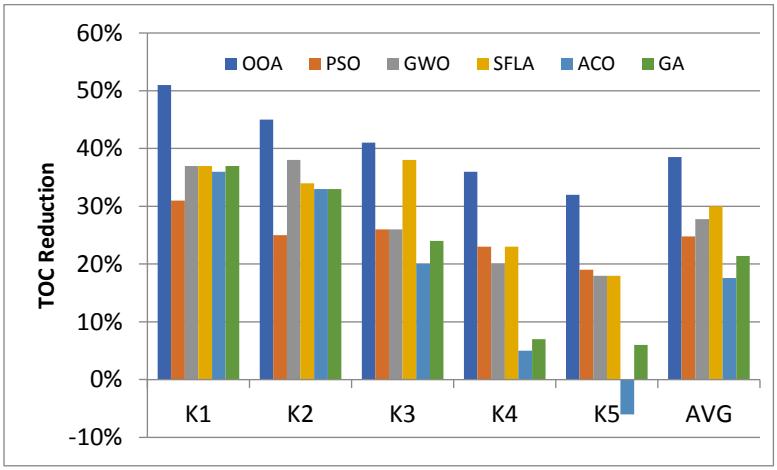

Fig. 8 Replica-placement algorithms achieve varying levels of reduction in data-access time percentages for various quantities of data items

Fig. 9 exhibits the trendlines produced by multiple methods for the TOC, which show the methods' behavior for various data items. The slope of the trendlines represents the new algorithm's performance. Figs. 9 and 10 depict the linear behavior of OOA and the slope of the trendlines for the different methods, respectively. The OOA trendline slope is around 0.045. OOA and PSO show the lowest and highest amount of trendline slope, respectively. PSO's TOC value exhibits a more pronounced increase as the amount of data increases, making it suitable for handling a large quantity of data items in distributed systems like IoT and cloud systems. Another factor for evaluation is the quantity of generated clones

(replicas) by replica placement methods. The higher the storage cost, the greater the number of created clones.

Fig. 9 Trendlines representing the obtained values of TOC through various algorithms

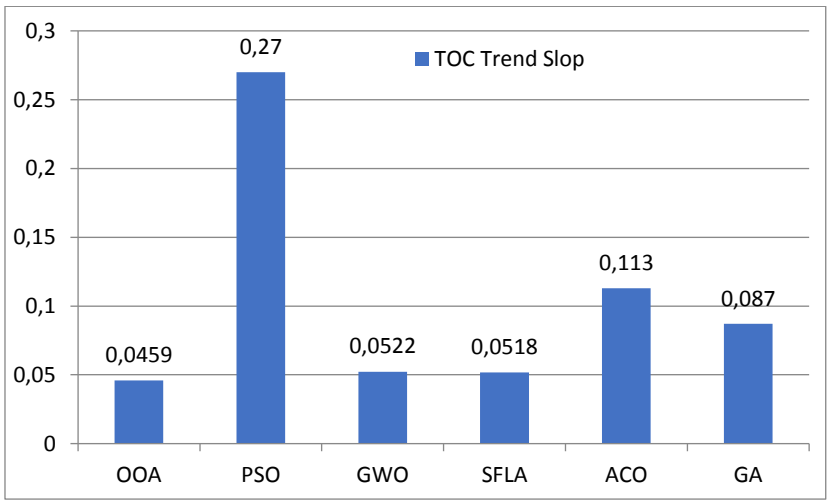

Fig. 10 Trendline slopes are generated for different algorithms

Fig. 11 shows the replica counts generated by various methods for varying quantities of data objects. When k=1, GA, ACO, SFLA, GWA, PSO, and OOA generated five, six, five, five, and four copies, respectively. The number of created copies in the huge benchmark ($k = 5$) is 24, 30, 8, 12, 9, and 7, correspondingly. The OOA produces fewer replicas than the other algorithms in most benchmarks. Overall, the proposed algorithm can save more time with a lower number of replications. OOA requires less storage compared

to other methods in addressing replica placement. The significant reduction in time achieved with a limited number of replications indicates that OOA is more efficient compared to the other methods. As depicted in Figs. 9 and 10, OOA exhibits linear growth in the quantity of replicas but with a smaller slope compared to previous techniques.

The convergence criterion is the heuristic algorithms' other performance requirement. In order to examine the convergence of the algorithms, an additional series of experiments was conducted.

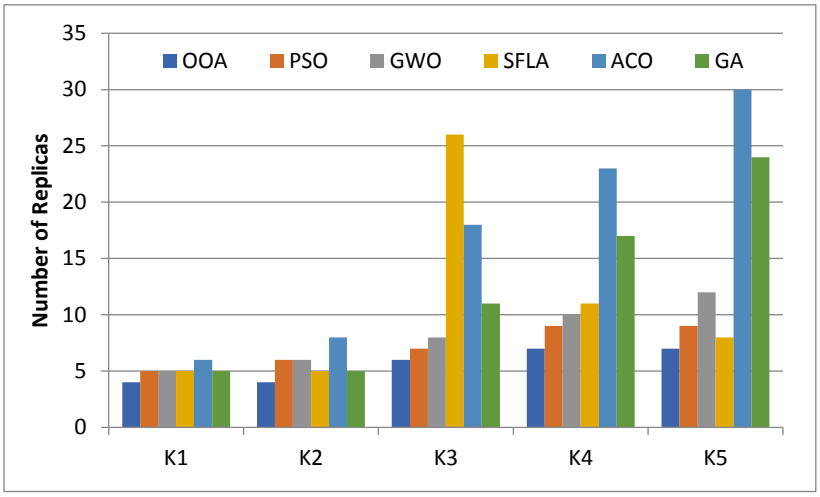

Fig. 11 Replica-placement algorithms generate varying number of replicas for various numbers of data items in optimal scenarios

Notably, the proposed methodology demonstrates significantly faster convergence compared to other methods. The OOA algorithm achieves the optimal server selection for replicas of a single data object within five iterations. In a specific test (*K*1), 4 replicas were distributed across the available servers, which resulted in a TOC reduction of 51%. The results vividly exhibit the superiority of the OOA algorithm over PSO, GWA, SFLA, ACO, and GA concerning the reduction of TOC and speed of convergence. The 2nd experiment evaluated the performance of various methods in replicating and storing 2 data items; the convergence of these methods when confronted with the task of replicating and storing two data items in the appropriate servers. The findings indicate the OOA's superior performance in terms of convergence and TOC. In this benchmark experiment, the OOA algorithm can find the optimal replica site before the 10th iteration. With only four copies, the proposed approach reduced TOC by 45%. The new method's TOC value is lower than the TOC values produced by the other techniques in this experiment. In this benchmark scenario, OOA achieved a 41% reduction in TOC by utilizing six replicas across the 28 data servers. The experimental results demonstrate the superior performance of OOA compared to other algorithms regarding the reduction of TOC and speed of convergence.

The speed of convergence and obtained TOC by different algorithms in the fourth benchmark (k=4) have been shown in Fig. 12. As shown in the results, OOA selects the best servers for replicating 4 data items before the 20th iteration in this benchmark scenario

(*k*=4). When there are four data items, OOA reduces TOC by 36% with seven replicates. PSO and SFLA perform similarly in this benchmark (23% TOC reduction). Similar to other algorithms, GWO achieves a 20% reduction in data access time. Additionally, it requires a lower number of data servers compared to other techniques. In the remaining studies, various algorithms' performance was assessed using five data items. Fig. 13 depicts the methods' performance when there are five data items for replication. OOA decreases TOC by 32% with only seven data replicates, as demonstrated in Fig. 13. PSO, GWA, and SFLA all perform similarly in this benchmark. OOA demonstrates a faster convergence rate in comparison with the other algorithms. GWO reduces TOC by 18% by producing twelve clones. Following the suggested algorithm, PSO requires fewer data servers to manage five data items compared to GA, GWA, ACO, and SFLA.

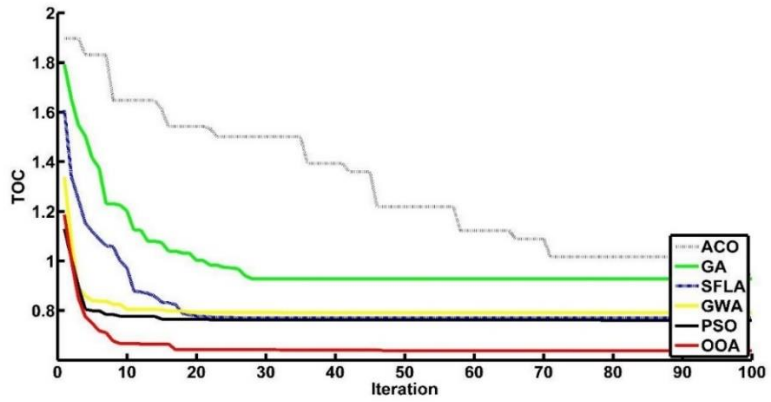

Fig. 12 Convergence of various algorithms towards optimal solutions for *k*=4 (four data objects)

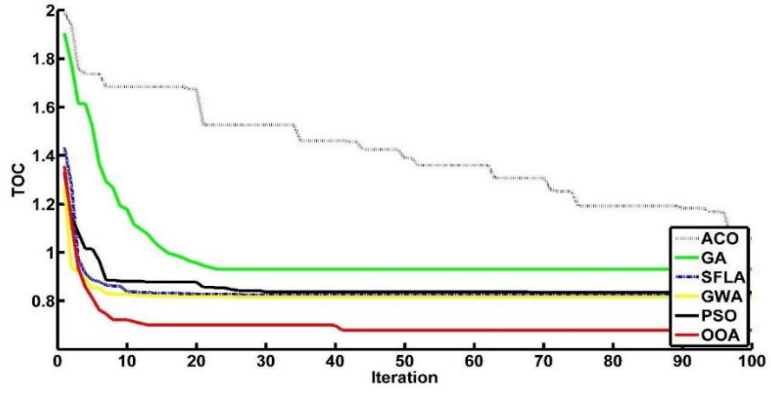

Fig. 13 Convergence of various algorithms towards optimal solutions for *k*=5 (five data items)

The average value of TOC reduction by various approaches for five benchmarks is demonstrated in Fig. 14. On average, the reduction of TOC achieved by OOA, PSO, GWA, SFLA, ACO, and GA algorithms is 39%, 25%, 28%, 30%, 18%, and 21%, respectively. The proposed OOA algorithm achieves an average TOC reduction of 39%. Furthermore, the average quantity of copies generated by OOA, PSO, GWA, SFLA, ACO, and GA algorithms is 6.25, 7.2, 8.2, 12.5, 19.75, and 14.25, respectively. Overall, the suggested strategy maintains data with fewer replicas and reduces data access time the most. In terms of outcomes, OOA greatly surpasses the other algorithms about TOC reduction, quicker convergence, and fewer created clones (lower storage spaces). The findings were analyzed regarding deviation value to determine all mentioned methods' stability. Ten iterations of each method were reviewed to assess the range of probable outcome changes.

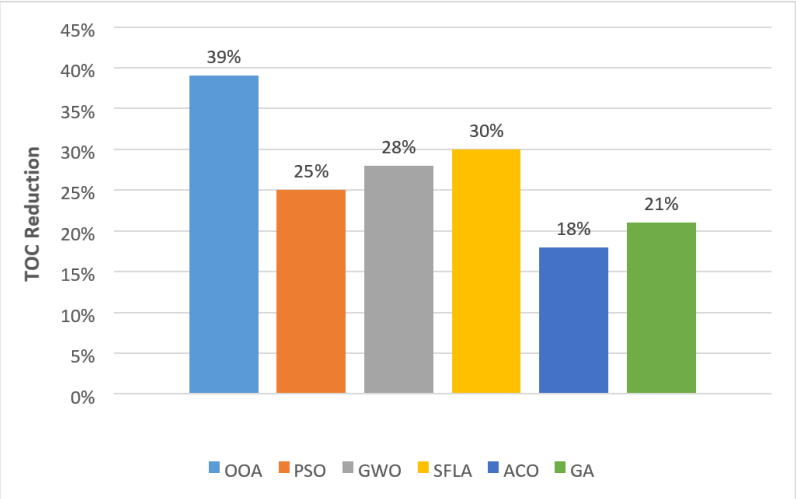

Fig. 14 TOC average reduction

Table 6 indicates the TOC values obtained by six algorithms when k=5 during 10 executions. According to the findings, the average value of TOC obtained from ten executions of OOA is about 0.6917; whereas this figure for GA, ACO, SFLA, GWA, and PSO are 1.0266, 1.1723, 0.8421, 0.8237 and 0.8022 respectively. Indeed, the OOA has more performance compared to the other algorithms in terms of the TOC reduction criteria. Furthermore, the minimum value of TOC provided by OOA is lower than the other algorithms.

The values in the final row of Table 6 illustrate the standard deviation (STDV) among the produced TOC in 10 executions. The least STDV indicates the OOA's higher stability in this benchmark. The increased stability observed in the results obtained by OOA reflects a higher level of reliability. GWO and SFLA have approximately similar performance in this benchmark (k=5).

Table 6 TOC values obtained by various algorithms undergo a range of changes during ten executions when *k*=5

#Runs	GA	ACO	SFLA	GWA	PSO	OOA
1	1.0240	1.0578	0.8625	0.8163	0.7888	0.6981
2	0.9306	1.1785	0.8315	0.8166	0.7977	0.6911
3	1.1277	1.1698	0.8625	0.8315	0.7965	0.6987
4	0.9552	1.2245	0.8241	0.8214	0.8139	0.6787
5	1.1698	1.1554	0.8241	0.8237	0.8116	0.6921
6	1.0578	1.2258	0.8315	0.8312	0.7972	0.6962
7	0.9306	1.1412	0.8287	0.8241	0.7965	0.6917
8	0.9564	1.1941	0.8625	0.8166	0.7843	0.6961
9	1.1412	1.1412	0.8312	0.8287	0.7999	0.6897
10	0.9731	1.2353	0.8625	0.8267	0.835	0.6850
Min	0.9306	1.0578	0.8241	0.8163	0.7843	0.6787
Max	1.1698	1.2353	0.8625	0.8315	0.835	0.6987
AVG	1.0266	1.1723	0.8421	0.8237	0.8024	0.6917
STDV	0.0919	0.0531	0.0177	0.0058	0.0147	0.0062

Regarding the results obtained by six algorithms when *k*=4 during ten executions, the average value of TOC obtained by OOA is about 0.6528; whereas this figure for GA, ACO, SFLA, GWA, and PSO are 0.9581, 1.0019, 0.7770, 0.8018 and 0.7778 respectively. On average, the OOA has a lower TOC than the other algorithm. Like the other benchmarks, OOA provides lower data access time with a limited quantity of produced replicas. The other important criterion is the reliability of the results. In this study, the lower amount of deviation among the obtained results during 10 executions indicates the reliability of the results. As shown in Table 6, GWA has the lowest STDV and highest reliability. After GWA, OOA is more stable (reliable) than the other algorithms. The smaller the STDV, the more stable the OOA in this benchmark. The greater the stability, the greater the dependability of the OOA results. Analyzing results from 10 executions while $k = 5$, the average quantity of replicas produced by OOA is 7, which is smaller compared to the other techniques. The average quantity of replicas generated by GA, ACO, SFLA, GWA, and PSO algorithms is 25.3, 34.4, 12, 12.4, and 9.97, respectively. The average quantity of replicas produced for five data items by GWA and PSO is 12.4 and 9.9 in the simulated distribution systems. Indeed, OOA shows the lowest time of data access with around seven replicas for five data items in the simulated distributed system. OOA is superior to the existing methods when it comes to the quantity of produced replicas. The value of STDV among the results achieved by GA, ACO, SFLA, GWA, PSO, and OOA are 1.6363, 2.4129, 2.3094, 0.5163, 1.3703, and 1.1547, respectively. In this benchmark, GWA has the highest performance from the stability perspective. After GWA, OOA has the lowest STDV and, consequently, the highest stability of the results.

Based on the results of ten executions on the four data object, the average quantity of copies created by OOA is 6.6, which is lower than the other methods. GA, ACO, SFLA, GWA, and PSO generated an average of 21.2, 25.1, 12.9, 10.6, and 9.1 copies, respectively. In the simulated distribution systems, the average quantity of copies generated by GWO and PSO for four data items is 10.6 and 9.1. In the simulated distributed system, OOA

delivers the shortest data access time with around 6.6 replicates for four data items. In terms of the quantity of created clones, the OOA outperforms the other methods. STDV values generated by GA, ACO, SFLA, GWA, PSO, and OOA are 3.7947, 2.4698, 2.5582, 0.5163, 0.316228, and 1.3498, respectively. GWA has the highest performance in this benchmark in terms of stability. Following GWA, OOA has the lowest STDV and hence the best results stability.

To confirm and trust the results, four distinct settings were used in similar investigations. In every experiment, the initial servers of the initial data items are different. Moreover, depending on the layout, data item sizes fluctuate. Each replica-placement method has been thoroughly tested in a variety of settings (configurations). Table 7 shows the final experiment parameters. On each arrangement, 4 various data items with various quantities were stored in chosen servers. Each data object's initial server is specified in the settings. The findings reveal the OOA's performance in determining the optimal replica placement in different configurations. In the conducted testing with varied settings, OOA has better performance with regard to convergence speed and TOC reduction. Regarding the results, SFLA, GWA, PSO, and OOA converged respectively to 0.7377, 0.7069, 0.6987, and 0.7029 after 100 iterations. The results of this experiment indicate that the OOA is superior to the other algorithms. The 2nd experiment was conducted with the 2nd configuration. The provided TOC by SFLA, GWA, PSO, and OOA are 0.8167, 0.7899, 0.7977, and 0.6981, respectively. The OOA is superior to the other algorithms from the TOC and convergence points of view. In this experiment, the SFLA, PSO, and GWA all performed similarly. The third experiment was conducted in different configurations. The obtained TOC by SFLA, GWA, PSO, and OOA are 0.7274, 0.7045, 0.8357, and 0.6787. In this experiment, similar to the previous experiments, the OOA is superior to the other algorithms from the TOC and convergence points of view. Finally, the fourth experiment was conducted in different configurations. The provided TOC by SFLA, GWA, PSO, and OOA are 0.7763, 0.7776, 0.7843, and 0.6850. In most of the benchmarks, OOA finds the optimum solution before the other algorithms. In most trials, the SFLA and GWA perform similarly, whereas GA and ACO perform worse than the other algorithms. Overall, the OOA outperforms both the SFLA, GWA, and PSO.

Table 7 Four workload configs to validate various algorithms' performance

Config. (workload) number	Primary Servers	Data objects' volume in Gigabytes
1	[15, 7, 3, 17, 27]	[0.10, 0.15, 0.16, 0.10, 0.20]
2	[1, 28, 11, 5, 3]	[0.70, 0.15, 0.86, 0.90, 0.70]
3	[19, 2, 17, 15, 6]	[0.30, 0.25, 0.10, 0.70, 0.25]
4	[7, 1, 18, 25, 9]	[0.90, 0.15, 0.30, 0.50, 0.60]

Fig. 15 illustrates the TOC achieved by different replica placement algorithms across four distinct configurations. The proposed OOA algorithm outperforms other methods in performance and efficiency. SFLA, GWA, and PSO algorithms exhibit similar performance in most configurations and benchmarks. Additionally, the quantity of replicas generated by OOA is smaller compared to other methods. This results in a reduced total data access time in distributed systems by minimizing the quantity of data replicas located on multiple servers. Among the algorithms evaluated, PSO, GWA, and SFLA exhibit better performance and efficiency compared to ACO and GA. On average, PSO generates 7.2

replicas, GWA generates 8.2 replicas, and OOA generates approximately six replicas. On the other hand, the SFLA algorithm generates an average of approximately twelve replicas.

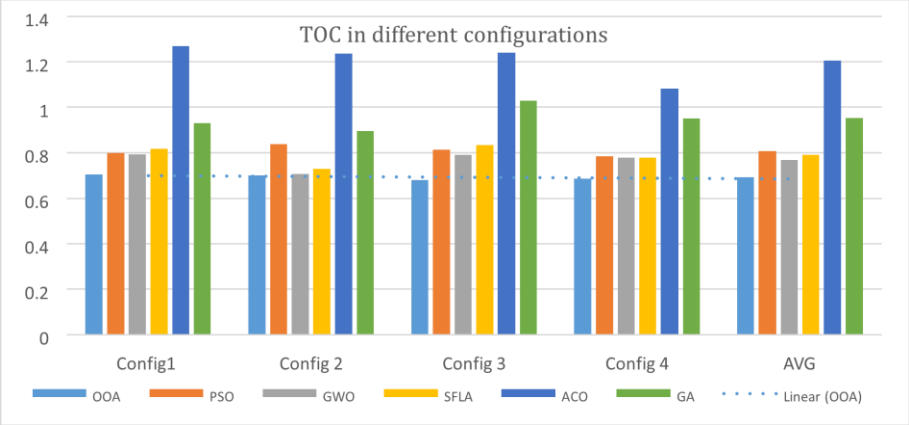

Fig. 15 Replica-placement algorithms' performance within different configurations

The average reduction in total TOC obtained by the OOA, PSO, GWA, and SFLA algorithms is 39%, 25%, 28%, and 30%, respectively, with SFLA exhibiting a higher reduction compared to PSO and GWA. Among the replica placement methods used in the current study, the ACO algorithm demonstrates the least performance and efficiency. OOA, on the other hand, maintains consistent performance regardless of the specific characteristics of the data objects. It exhibits reliable and predictable behavior, as observed in Fig. 15, with reduced fluctuations indicating its reliability. The linear trend with a slope close to 0 degrees further illustrates the effectiveness of the proposed OOA algorithm in the given scenario.

4.3 Discussion

Regarding the TOC reduction criterion, OOA exhibits superior performance compared to the others. Figs. 8 and 9 illustrate the relatively poor performance of ACO when dealing with a larger quantity of data items, as it incurs an overhead of performance in some of the benchmarks. Based on the simulation results, OOA offers the lowest data access time and achieves a higher TOC reduction compared to other algorithms. Figs. 10 and 11 show the linear behavior of OOA and the slopes of the trend lines for different techniques. The slope of the OOA trendline is approximately 0.045. OOA and PSO indicate the lowest and highest trend line slopes, respectively. PSO is suitable for processing large amounts of data in distributed systems such as IoT and cloud systems, as the TOC value increases significantly as the amount of data increases. The OOA's TOC value is lower than the TOC values produced by the other techniques in this experiment. The results demonstrate the superior performance of OOA compared to other algorithms regarding reduction of TOC and speed of convergence. On average, the TOC reductions achieved by OOA, PSO, GWA, SFLA, ACO, and GA algorithms are 39%, 25%, 28%, 30%, 18%, and 21%, respectively. Notably, the proposed OOA algorithm achieves an average TOC reduction of 39%. Furthermore, the results show that the average number of copies generated by the OOA,

PSO, GWA, SFLA, ACO, and GA algorithms are 6.25, 7.2, 8.2, 12.5, 19.75, and 14, respectively. Overall, the proposed strategy manages data with fewer replicas and provides the fastest data access times. OOA far outperforms other algorithms in terms of reduced TOC, faster convergence, and fewer clones created.

The stability of the results is one of the main criteria that should be considered in metaheuristic algorithms. The findings were analyzed regarding deviation value to determine all mentioned methods' stability. Ten iterations of each method were reviewed to assess the range of probable outcome changes. Regarding the findings, the average value of TOC obtained from ten executions of OOA is about 0.6917; whereas this figure for GA, ACO, SFLA, GWA, and PSO are 1.0266, 1.1723, 0.8421, 0.8237 and 0.8022 respectively. Indeed, the OOA has more performance compared to the other algorithms in terms of the TOC reduction criteria. The lowest STDV among the obtained results indicates high OOA stability in this benchmark. On average, OOA has a lower TOC than other algorithms. Like other benchmarks, OOA provides faster data access times while limiting the number of replicas generated. Another important criterion is the reliability (stability) of the results.

In this study, the small variability between the results obtained during the 10 runs indicates the reliability of the results. According to the results, GWA has the lowest STDV and the highest reliability. According to GWA, OOA is more stable (reliable) than other algorithms. The higher the stability, the more reliable the OOA results. The average total TOC reductions achieved by the OOA, PSO, GWA, and SFLA algorithms were 39%, 25%, 28%, and 30%, respectively, with SFLA showing higher reductions compared to PSO and GWA. Among the replica placement methods used in current research, the ACO algorithm has the lowest performance and efficiency. OOA, on the other hand, guarantees consistent performance regardless of the specific properties of the data object. As shown in Fig. 15, it exhibits reliable and predictable behavior, with small variations demonstrating its reliability. A linear trend with a slope close to 0 degrees further clarifies the effectiveness of his proposed OOA algorithm in certain scenarios. To find the optimal values of the OOA parameters, different experiments have been performed with different parameter values. Workload 3 in Table 7 was used in this series of experiments. The experiments have been performed with different values. Table 8 shows the performance of OOA with different values of configuration parameters. The best performance was obtained when the learning rate was about 0.5 or 0.6. The small team size leads to better performance.

Table 8 Calibrating the parameters of the OOA algorithm in workload 3 in Table 7

OOA Calibration Parameters' Values			**Performance**
Num. of teams	Size of teams	Learning rate	TOC Reduction
4	10	0.4	29%
6	7	0.2	30%
8	5	0.6	32%
10	4	0.8	31%

5. CONCLUSION

Rapid data access must be considered while designing big dispersed systems. The task of replicating data in distributed systems like IoT is a problem that falls under the category

of NP-complete problems. The OOA approach, as a novel discrete heuristic algorithm, is introduced as a population-based algorithm to address replica management in order to reduce the cost of data operations. In this study, the EUData was used as a case study. The case study included 28 data servers that were modelled by a complete graph. Regarding the case study, a simulation platform was developed using MATLAB. Different replica placement algorithms were implemented in the simulated platform. This method was evaluated in a simulated distributed system with 28 data servers using the MATLAB platform.

OOA outperforms the previous methods when it comes to data access time, quantity of replicas generated, result stability, and dependability and decreases data-access time by around 39% on average; this value for GA, PSO, GWA, ACO, and SFLA, is 21%, 25%, 28%, 18%, and 30%, respectively. In order to save allocation resources, the next issue is to limit the number of replicas generated for data items. GA, PSO, GWA, SFLA, ACO, and OOA produced an average of 14, 7.2, 8.2, 12.50, 20, and 6 data object replicas, respectively. OOA obtains the least data access time while using the least storage areas. Furthermore, the suggested technique converges quicker than GA, PSO, SFLA, and ACO. OOA yielded lower SDs compared to the other algorithms in this example. In repeated executions, OOA is more stable in comparison with the other algorithms and can provide similar results again and over again. Therefore, OOA shows more reliability and efficiency in addressing the problem of replica placement.

In this analysis, the overall stability of the system is assumed, although only a combination of read and write transactions is considered, while other data transactions are not accounted for. The study does not analyze the execution time of the algorithm to reach the optimal response in an unstable state. Future efforts could focus on parallelizing this method. Additionally, the dynamic nature of the current method is expected to be a topic for future research, as the migration of initial and replica data in a dynamic setting presents unique challenges. Various heuristic/metaheuristics [35-50] and machine learning methods have been developed and utilized in computer engineering to address a variety of optimization problems [51-57]; therefore, assessing these methods' effectiveness in the problem of replica placement would be worthwhile.

REFERENCES

1. Qiu, L., Padmanabhan, V. N., Voelker, G. M., 2001, *On the placement of web server replicas*. Proc. IEEE INFOCOM 2001, Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society, 2001.
2. Li, B., Golin, M. J., Italiano, G. F., Deng, X., Sohraby, K., 1999, *On the optimal placement of web proxies in the internet*, In IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies.
3. Szymaniak, M., Pierre, G., Van Steen, M., 2006, *Latency-driven replica placement*, IPSJ Digital Courier, 2, pp. 561-572.
4. Ng, T. E., Zhang, H., 2002, *Predicting Internet network distance with coordinates-based approaches*, Proc. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 2002.
5. Li, C., Liu, J., Lu, B., Luo, Y., 2021, *Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment*, Journal of Network and Computer Applications, 180(4), 103017.
6. Li, C., Wang, Y., Tang, H., Zhang, Y., Xin, Y., Luo, Y., 2019, *Flexible replica placement for enhancing the availability in edge computing environment*, Computer Communications, 146(10), pp.1-14.
7. Li, C., Bai, J., Chen, Y., Luo, Y., 2020, *Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system*, Information Sciences, 516(4), pp.33-55.

8. Safaee, S., Haghighat, A. T., 2012, *Replica placement using genetic algorithm*, Proc. International Conference on Innovation Management and Technology Research, pp. 507-512, IEEE 2012.
9. Abawajy, J. H., Deris, M. M., 2013, *Data replication approach with consistency guarantee for data grid*. IEEE Transactions on Computers, 63(12), pp. 2975-2987.
10. Shamsa, Z., Dehghan, M., 2013, *Placement of replicas in distributed systems using particle swarm optimization algorithm and its fuzzy generalization*, Proc. 13th Iranian Conference on Fuzzy Systems (IFSC), pp. 1-6, IEEE 2013.
11. Kolisch, R., Dahlmann, A., 2015, *The dynamic replica placement problem with service levels in content delivery networks: a model and a simulated annealing heuristic*, OR spectrum, 37(1), pp. 217-242.
12. Tu, M., Yen, I. L., 2014, *Distributed replica placement algorithms for correlated data*, The Journal of Supercomputing, 68(11), pp. 245-273.
13. Subramanyam, G., Lokesh, G., Kumari, B., 2013, *A priori data replica placement strategy in grid computing*. International Journal of Scientific and Engineering Research, 4(7), pp. 1070-1076.
14. Fan, W., Yang, L., Bouguila, N., 2022, *Unsupervised Grouped Axial Data Modeling via Hierarchical Bayesian Nonparametric Models with Watson Distributions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(12), pp. 9654-9668.
15. Liang, X., Huang, Z., Yang, S., Qiu, L., 2018, *Device-Free Motion Trajectory Detection via RFID*, ACM Transactions on Embedded Computing Systems, 17(4), 78.
16. Lu, C., Zheng, J., Yin, L., Wang, R., 2023, *An improved iterated greedy algorithm for the distributed hybrid flowshop scheduling problem*, Engineering Optimization. doi: 10.1080/0305215X.2023.2198768
17. Wang, Z., Zhao, D., Guan, Y., 2023, *Flexible-constrained time-variant hybrid reliability-based design optimization*, Structural and Multidisciplinary Optimization, 66(4), pp. 89-103.
18. Hu, D., Li, Y., Yang, X., Liang, X., Zhang, K., Liang, X., Taciroglu, E., 2023, *Experiment and Application of NATM Tunnel Deformation Monitoring Based on 3D Laser Scanning*, Structural Control and Health Monitoring, 2023, 3341788.
19. Qu, Z., Zhang, Z., Liu, B., Tiwari, P., Ning, X., Muhammad, K. 2023, *Quantum detectable Byzantine agreement for distributed data trust management in blockchain*, Information Sciences, 637(8), 118909.
20. Li, K., Ji, L., Yang, S., Li, H., Liao, X., 2022, *Couple-Group Consensus of Cooperative–Competitive Heterogeneous Multiagent Systems: A Fully Distributed Event-Triggered and Pinning Control Method*, IEEE Transactions on Cybernetics, 52(6), pp. 4907-4915.
21. Zhou, G., Zhang, R., Huang, S., 2021, *Generalized Buffering Algorithm*. IEEE access, 9, pp. 27140-27157.
22. Ni, Q., Guo, J., Wu, W., Wang, H., 2022, *Influence-Based Community Partition with Sandwich Method for Social Networks*. IEEE Transactions on Computational Social Systems, 10(2), pp. 819-830.
23. Wang, K., Zhang, B., Alenezi, F., Li, S., 2022, *Communication-efficient surrogate quantile regression for non-randomly distributed system*, Information sciences, 588 (4), pp. 425-441.
24. Yuan, H., Yang, B., 2022, *System Dynamics Approach for Evaluating the Interconnection Performance of Cross-Border Transport Infrastructure*, Management in Engineering, 38(3), 04022008.
25. Li, P., Hu, J., Qiu, L., Zhao, Y., Ghosh, B. K, 2022, *A Distributed Economic Dispatch Strategy for Power–Water Networks*, IEEE Transactions on Control of Network Systems, 9(1), pp. 356-366.
26. Song, Y., Xin, R., Chen, P., Zhang, R., Chen, J., Zhao, Z., 2023, *Identifying performance anomalies in fluctuating cloud environments: A robust correlative-GNN-based explainable approach*, Future Generation Computer Systems, 145 (3), pp. 77-86.
27. Deng, Y., Zhang, W., Xu, W., Shen, Y., Lam, W., 2023, *Nonfactoid Question Answering as Query-Focused Summarization with Graph-Enhanced Multihop Inference*, IEEE Transactions on Neural Networks and Learning Systems. doi: 10.1109/TNNLS.2023.3258413.
28. Cheng, B., Wang, M., Zhao, S., Zhai, Z., Zhu, D., Chen, J., 2017, *Situation-Aware Dynamic Service Coordination in an IoT Environment*, IEEE/ACM Transactions on Networking, 25(4), pp. 2082-2095.
29. Lu, S., Liu, M., Yin, L., Yin, Z., Liu, X., Zheng, W., Kong, X, 2023, *The multi-modal fusion in visual question answering: a review of attention mechanisms*, PeerJ Computer Science, 9(5), e1400.
30. Liu, X., Shi, T., Zhou, G., Liu, M., Yin, Z., Yin, L., Zheng, W, 2023, *Emotion classification for short texts: an improved multi-label method*, Humanities and Social Sciences Communications, 10(1), 306.
31. Liu, X., Zhou, G., Kong, M., Yin, Z., Li, X., Yin, L., Zheng, W, 2023, *Developing Multi-Labelled Corpus of Twitter Short Texts: A Semi-Automatic Method*, Systems, 11(8), 390.
32. Lu, S., Ding, Y., Liu, M., Yin, Z., Yin, L., Zheng, W, 2023, *Multiscale Feature Extraction and Fusion of Image and Text in VQA*, International Journal of Computational Intelligence Systems, 16(1), pp. 54-2023.
33. Cao, B., Gu, Y., Lv, Z., Yang, S., Zhao, J., Li, Y, 2021, *RFID Reader Anticollision Based on Distributed Parallel Particle Swarm Optimization. IEEE internet of things journal*, 8(5), pp. 3099-3107.

34. Arasteh, B., Sadegi, R., Arasteh, K., Gunes, P., Kiani, F., Torkamanian-Afshar, M., 2023, *A bioinspired discrete heuristic algorithm to generate the effective structural model of a program source code*, Journal of King Saud University-Computer and Information Sciences, 35(8), 101655.
35. Arasteh, B., Miremadi, S. G., Rahmani, A. M, 2014, *Developing inherently resilient software against soft errors based on algorithm level inherent features*, Journal of Electronic Testing, 30 (2), pp. 193-212.
36. Arasteh, B., Sadegi, R., Arasteh, K, 2021, *Bölen: Software module clustering method using the combination of shuffled frog leaping and genetic algorithm*, Data Technologies and Applications, 55(2), pp. 251-279.
37. ZadahmadJafarlou, M., Arasteh, B., YousefzadehFard, P., 2011, *A pattern-oriented and web-based architecture to support mobile learning software development*, Procedia-Social and Behavioral Sciences, 28, pp. 194-199.
38. Hatami, E., Arasteh, B, 2020, *An efficient and stable method to cluster software modules using ant colony optimization algorithm*, The Journal of Supercomputing, 76(9), pp. 6786-6808.
39. Arasteh, B., Sadegi, R., Arasteh, K, 2020, *ARAZ: A software modules clustering method using the combination of particle swarm optimization and genetic algorithms*, Intelligent Decision Technologies, 14(4), pp. 449-462.
40. Arasteh, B., Najafi, J., 2018, *Programming guidelines for improving software resiliency against soft errors without performance overhead*, Computing, 100(2), pp. 971-1003.
41. Arasteh, B., Fatolahzadeh, A., Kiani, F., 2022, *Savalan: Multi objective and homogeneous method for software modules clustering*, Journal of Software: Evolution and Process, 34(1), e2408.
42. Afshord, S. T., Pottosin, Y., Arasteh, B., 2015, *An input variable partitioning algorithm for functional decomposition of a system of Boolean functions based on the tabular method*, Discrete Applied Mathematics, 185, pp. 208-219.
43. Arasteh, B., 2023, *Clustered design-model generation from a program source code using chaos-based metaheuristic algorithms*, Neural Computing and Applications, 35(4), pp. 3283-3305.
44. Bouyer, A., Beni, H. A., Arasteh, B., Aghaee, Z., Ghanbarzadeh, R., 2023, *FIP: A fast overlapping community-based Influence Maximization Algorithm using probability coefficient of global diffusion in social networks*, Expert systems with applications, 213 (3), 118869.
45. Arasteh, B., Pirahesh, S., Zakeri, A., Arasteh, B., 2014, *Highly available and dependable E-learning services using grid system*, Procedia-Social and Behavioral Sciences, 143, pp. 471-476.
46. Nezhadroshan, A. M., Fathollahi-Fard, A. M., Hajiaghaei-Keshteli, M., 2021, *A scenario-based possibilistic-stochastic programming approach to address resilient humanitarian logistics considering travel time and resilience levels of facilities*. International Journal of Systems Science: Operations Logistics, 8(4), pp. 321-347.
47. Golshahi-Roudbaneh, A., Hajiaghaei-Keshteli, M., Paydar, M. M., 2017, *Developing a lower bound and strong heuristics for a truck scheduling problem in a cross-docking center*. Knowledge-Based Systems, 129, pp. 17-38.
48. Babaeinesami, A., Tohidi, H., Ghasemi, P., Goodarzian, F., Tirkolaee, E. B., 2022, *A closed-loop supply chain configuration considering environmental impacts: a self-adaptive NSGA-II algorithm*, Applied Intelligence, 52(12), pp. 13478-13496.
49. Tirkolaee, E. B., Goli, A., Mardani, A., 2021, *A novel two-echelon hierarchical location-allocation-routing optimization for green energy-efficient logistics systems*, Annals of operations research, 324(11), pp. 795–823.
50. Aghighi, A., Goli, A., Malmir, B., Tirkolaee, E. B. 2021, *The stochastic location-routing-inventory problem of perishable products with reneging and balking*, Journal of Ambient Intelligence and Humanized Computing, 14(10), pp. 6497–6516.
51. Sahebjamnia, N., Goodarzian, F., Hajiaghaei-Keshteli, M., 2020, *Optimization of multi-period three-echelon citrus supply chain problem*, Journal of Optimization in Industrial Engineering, 13(1), pp. 39-53.
52. Mahmood, L., Bahroun, Z., Ghommem, M., Alshraideh, H., 2022, *Assessment and performance analysis of Machine learning techniques for gas sensing E-nose systems*. Facta Universitatis, Series: Mechanical Engineering, 20(3), pp. 479-501.
53. Ewertowski, T., Güldoğuş, B. Ç., Kuter, S., Akyüz, S., Weber, G. W., Sadłowska-Wrzesińska, J., Racek, E., 2023, *The use of machine learning techniques for assessing the potential of organizational resilience*, Central European Journal of Operations Research, 31(1), https://doi.org/10.1007/s10100-023-00875-z
54. Weber, G. W., Arabnia, H., Aydın, N. S., Tirkolaee, E. B, 2023, *Preface: advances of machine learning and optimization in healthcare systems and medicine*. Annals of Operations Research, 328 (1), pp. 1-2.
55. Vasant, P., Zelinka, I., Weber, G. W., 2019, *Intelligent computing optimization*, Berlin: Springer International Publishing.
56. Çevik, A., Weber, G. W., Eyüboğlu, B. M., Oğuz, K. K., 2017, *Voxel-MARS: a method for early detection of Alzheimer's disease by classification of structural brain MRI*, Annals of Operations Research, 258, pp. 31-57.
57. Graczyk-Kucharska, M., Olszewski, R., Golinski, M., Spychala, M., Szafranski, M., Weber, G. W., Miadowicz, M. ,2022, *Human resources optimization with MARS and ANN: innovation geolocation model for generation Z*, Journal of Industrial and Management Optimization, 18(6), pp. 4093-4110.
58. https://drive.google.com/drive/folders/1o50_L9HgmWAa1iMrnZ5O06n46xc2O80d?usp=sharing (last access: 05.07.2023).