Received: 15 November 2023 | Revised: 21 May 2024 | Accepted: 24 May 2024

DOI: 10.1002/dac.5864

RESEARCH ARTICLE

WILEY

Optimizing energy-efficient data replication for IoT applications in fog computing

Ahmed Awad Mohamed[1] | **Ali Diabat**[2,3,4] | **Laith Abualigah**[5,6,7,8]

[1]Information System Dept., Cairo Higher Institute for Languages and Simultaneous Interpretation and Administrative Science, Egypt

[2]Division of Engineering, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates

[3]Honorary Professor, School of Engineering, The University of Jordan, Amman, Jordan

[4]Department of Civil and Urban Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA

[5]Computer Science Department, Al al-Bayt University, Mafraq, Jordan

[6]Jadara Research Center, Jadara University, Irbid, Jordan

[7]Applied science research center, Applied science private university, Amman, Jordan

[8]MEU Research Unit, Middle East University, Amman, Jordan

Correspondence
Laith Abualigah, Computer Science Department, Al al-Bayt University, Mafraq 25113, Jordan.
Email: aligah.2020@gmail.com

Funding information
Not Applicable.

Summary

The rise of the Internet of Things (IoT) has given rise to an era marked by interconnected devices and substantial data generation. This has led to an increased reliance on cloud computing for data processing and storage, primarily due to its cost-effective pay-for-use model. However, this dependence has prompted critical inquiries into the optimal replication of data: what data to replicate, when to replicate it, and where to place new replicas strategically. Conventional cloud data replication often results in resource overutilization, performance bottlenecks, increased workloads, energy consumption, prolonged user wait times, and suboptimal response times. In response to these challenges, this paper introduces a novel approach named Multiobjective Optimization Harris Hawks Optimization with Salp Swarm Algorithm (MOHHOSSA). This approach employs multiobjective optimization (MOO) alongside Harris Hawks Optimization (HHO) and IoT-based Salp Swarm Algorithm (SSA) for cloud computing environments. MOHHOSSA efficiently identifies data replication opportunities and strategically allocates them across nodes in cloud computing infrastructures. The algorithm aims to enhance key performance metrics, including energy consumption, carbon dioxide emission rate, and mean service time. Extensive experimental validation demonstrates MOHHOSSA's superior performance compared to alternative algorithms. It excels in optimizing energy efficiency, load distribution, mean service time, and the establishment of cost-effective communication paths between nodes. This research represents a significant advancement in addressing challenges related to IoT data replication in cloud computing, ultimately promoting more sustainable, efficient, and responsive cloud-based services.

KEYWORDS

cloud fog, energy-aware, file replication, internet of things, mean service time

1 | INTRODUCTION

Over the past decade, significant advancements have been made in cloud computing and the proliferation of Internet of Things (IoT) applications across diverse domains, including healthcare, engineering, military operations, agriculture, and traffic management. IoT applications generate vast amounts of data on a daily basis, which is typically processed and stored in cloud computing infrastructure. However, this surge in data usage has resulted in

heightened resource consumption, increased operational costs, greater energy consumption, and a rise in carbon emissions.

Efforts have been made to optimize IoT applications by implementing fog computing, particularly in terms of routing efficiency and node allocation. Nonetheless, accessing data across different geographic locations remains a challenge, mainly due to limitations in bandwidth, cloud storage capacity, and response times for end-users [1-3]. Furthermore, the replication of data across nodes, along with its proximity to users, can often incur substantial expenses that may exceed users' budgetary constraints.

Fog computing advantages various sensors, including Wireless Sensor Networks (WSN), to support IoT within cloud computing environments. Data replication stands out as a critical technology for facilitating data transfer, positioning, and redundancy in IoT-driven cloud computing. The implementation of data replication enhances the efficiency of data transfer between nodes, reduces processing time, and minimizes access and retrieval delays, ultimately optimizing data flow along the most cost-effective routes [4–6].

Data replication strategies encompass two essential components: static and dynamic methods. In this research, the focus is primarily on dynamic methods for determining and deploying data replication, as they are influenced by factors such as file popularity, access patterns, and the diverse storage methodologies employed within the system [7–9]. Furthermore, dynamic methods automatically purge less popular files, resulting in both cost savings for users and reduced burdens on system resources [10, 11].

Data replication in cloud computing revolves around three fundamental aspects: the identification, placement, and replacement of data replicas across various nodes within the cloud infrastructure. The primary objective of this research is to strategically determine the optimal placement of data replication across nodes by taking into account several key factors. These factors include minimizing costs, reducing the physical distance between nodes, and mitigating the energy and carbon footprint resulting from the substantial consumption of gasoline and diesel, which has a direct impact on the environment.

In recent years, numerous research studies have proposed various techniques for data replication within the cloud, with the overarching goals of enhancing response times, cost-efficiency, bandwidth utilization, and load balancing, among others. However, it is important to note that modern algorithms that can identify the least costly path have not received sufficient attention, nor have the critical aspects of energy conservation and carbon reduction been adequately addressed within the context of data replication, particularly in the realm of IoT applications [12, 13].

In response to the limitations observed in prior research endeavors, our study proposes a comprehensive approach aimed at achieving cost savings in data replication procedures. This approach encompasses optimizing the paths between nodes, strategically duplicating files, and positioning them in close proximity to end-users, all while effectively minimizing energy consumption and reducing carbon emissions for the benefit of the environment.

To tackle these multifaceted challenges associated with cross-node data transfers in IoT-driven cloud computing, we have harnessed the power of hybrid swarm intelligence. This innovative approach combines a suite of modern algorithms designed to drive down costs and conserve energy, even when faced with the complex task of selecting the most cost-efficient path—often a trade-off between conflicting objectives—while considering users' budget constraints. Notably, we have integrated the HHO algorithm with the SSA and MOO algorithms to create a novel solution known as MOHHOSSA. MOHHOSSA stands as a pioneering contribution to the realm of IoT-based data replication, offering a unique and effective approach to address these challenges.

The article's major contributions can be summarized as follows:

- We introduce MOHHOSSA, a hybrid swarm intelligence algorithm specifically designed for dynamic data replication within the context of fog computing.
- Our innovative approach involves the integration of HHO and SSA with MOO, effectively addressing energy consumption and reducing carbon dioxide emissions in fog computing scenarios.
- It is worth noting that there is a scarcity of research papers that tackle the critical issues of energy consumption and carbon dioxide emissions in the context of optimizing dynamic data replication within fog computing.
- To the best of our knowledge, our work represents the pioneering effort in exploring a new hybrid algorithm, MOHHOSSA, which successfully resolves energy conservation and carbon dioxide emission concerns in fog computing.

The remainder of the paper is structured as follows: in Section 2, we conduct a comprehensive review of related research. In Section 3, we present our proposed algorithm. Section 4 contains our experimental results, and finally, in Section 5, we offer conclusions and outline directions for future work.

2 | RELATED WORK

Numerous studies have investigated data replication strategies in the cloud, including:

N. Mansouri et al. [14] proposed a method for strategically placing data replication among fog computing nodes. Their approach introduced an algorithm designed to ensure data confidentiality and privacy, safeguarding against intruders. The proposal hinges on user-defined privacy levels (LoP), aligning with their budget constraints. The results demonstrated the superiority of their algorithm over competing approaches.

N. Mansouri et al. [15] introduced dynamic data replication techniques tailored to file popularity and frequent file access patterns across nodes. Files that enjoy high user accessibility are replicated and strategically positioned among nodes. The proposed system utilizes parallel methods for efficient file collection and distribution from various locations. Their algorithm showcased superior performance compared to previous ones.

D. Chen et al. [16] presented a novel system for creating and placing data replication within a decentralized framework, emphasizing equitable data file distribution across nodes within Blockchain platforms. Attention was also given to the insurance system and associated risks, with several scenarios proposed for enhancing data file replication strategies.

C. Li et al. [17] suggested the application of a Lagrangian relaxation approach for the selection and placement of file replication. Their focus extended to factors such as storage capacity, load balancing, bandwidth optimization, data transmission, reception, and cloud processing speed. The approach considered tolerance flow, anomaly detection, and task execution deadlines. To ensure tolerance flow, file replication was distributed across multiple computing nodes on the Spark platform, resulting in reduced task execution times, quicker response times, and lower power consumption.

T. Shi et al. [18] proposed a novel approach for distributed file replication across nodes spanning multiple clouds, implemented through web platforms. They introduced a new algorithm for file replication called "web application replication and deployment in multi-cloud," which contributed to reduced user waiting times, cost savings, and improved efficiency.

A. Majed et al. [19] presented a method for data replication placement within the P2P cloud, focusing on determining file popularity, copying, and distribution across cloud nodes. Their algorithm identified highly popular files and strategically positioned them across nodes, with an emphasis on data availability, user accessibility, and load balancing.

C. Li et al. [20] proposed a novel data placement strategy based on mathematical equations, leveraging a geographically distributed cloud infrastructure spanning multiple regions. The approach aimed to distribute and optimize file replication while minimizing load balancing issues, bandwidth consumption, and costs. Their algorithm combined Lagrangian relaxation techniques with Floyd algorithm optimization to reduce data transmission times between nodes.

The rise of the multiaccess edge computing (MEC) paradigm has introduced a complex environment characterized by multiple users, servers, and access points, making data-offloading decision-making a critical research focus. This article tackles this issue by examining how user behavior and MEC server pricing policies affect optimal data offloading strategies [21]. Prospect theory is employed to capture user satisfaction and subjectivity in data offloading decisions. In contrast, the potential overuse of MEC servers by users is modeled using the tragedy of the commons theory. A multileader multi-follower Stackelberg game is formulated, positioning MEC servers as leaders and users as followers to establish optimal pricing policies for servers and optimal offloading strategies for users. Users' offloading decisions are modeled as a noncooperative game to find Nash equilibrium. The servers' optimal pricing is determined using either a semi-autonomous game-theoretic approach or a fully autonomous reinforcement learning approach. The framework's effectiveness is demonstrated through performance evaluations and simulations, showing its superiority over other benchmark alternatives.

In Table 1, we have presented a comparison of various works in the cloud environment, focusing on their respective strategies. We differentiate between these diverse strategies and our approach.

3 | SUGGESTED SYSTEM AND DISCUSSION

The proposed system is designed to operate across various geographical locations within the fog computing framework, which is built upon the Internet of Things (IoT). This system incorporates an array of sensors strategically deployed

TABLE 1 Comparison between related work and my strategy.

Strategy	Year	AI techniques	Energy-aware	Hybrid	Throughput	Carbon dioxide emission rate	Fog computing based on IoT
MORM	2013	√	√				
EIMORM	2017	√	√				
APSDRDO	2018	√					
MOGA	2019	√					
My Strategy	2022	√	√	√	√	√	√

throughout the cloud computing environment. It comprises nodes, which encompass a combination of hosts, virtual machines (VMs), and data blocks, including file replication functionality.

Our proposed system introduces a novel methodology referred to as MOHHOSSA. This method is devised to determine the most cost-effective path and the shortest distance between nodes, all while ensuring efficient load balancing during request handling and prompt user response. Additionally, our model places a strong emphasis on reducing energy consumption and carbon emissions, addressing the inherent trade-offs between factors like cost and distance. To achieve this, our approach prioritizes popular data, placing it within the nearest node to end-users, with a focus on cooler regions to optimize energy efficiency and avoid data replication in warmer regions. The system navigates conflicting goals by dynamically selecting and configuring data replication based on individual user budgets.

In this section, we present an algorithm that combines HHO with the SSA algorithm to facilitate data replication across nodes in the context of fog computing within the IoT domain. The overarching strategy behind MOHHOSSA is to identify the optimal conditions for data replication that minimize power consumption, carbon footprint, and overall costs. Furthermore, this section provides a comprehensive overview of our proposed methodology.

3.1 | Harris hawks optimization (HHO)

Haidari et al. [22] introduced the HHO method, specifically designed for optimizing various operations and addressing research challenges related to improvement. This algorithm, known as HHO, is a novel approach inspired by nature and can be described as follows:

3.1.1 | Exploration phase

Falcons perch on elevated positions, such as tree branches or trunks, to scan for prey from a vantage point. When they swoop down to capture a rabbit, there is a 50% chance of success. Similarly, the HHO algorithm seeks candidate solutions in proximity to the current solution, with the best candidate akin to the desired prey in the pursuit of the optimal solution. This can be expressed mathematically as follows:

$$X(t+1)=\begin{cases} X_{rand}(t)-r_1\left|X_{rand}(t)-2r_2X(t)\right| & q\geq 0.5 \\ X_{rabbit}(t)-X_m(t))-r_3(LB+r_4(UB-LB)) & q<0.5 \end{cases} \tag{1}$$

where,

$$X_m(t)=\frac{1}{N}\sum_{i=1}^{N}X_i(t). \tag{2}$$

3.1.2 | Exploitation phase

In this stage, four strategies have been devised to simulate a hawk's approach to capturing prey. These strategies involve the introduction of a random number to represent a 50% chance of the prey either escaping or experiencing a sudden attack by the hawk. Regardless of the prey's efforts to evade capture, the falcon will persistently pursue it, employing either a soft or hard siege technique to secure its prey based on its capabilities, denoted as E. This can be represented as follows:

3.1.3 | Soft besiege

In accordance with the prescribed escape ratio for the prey, there is a balanced 50:50 likelihood of the prey's success in evading capture based on its energy level. In the HHO algorithm, the approach involves quietly encircling the prey from all directions and subsequently launching a sudden attack. This approach can be mathematically represented as follows:

$$X(t+1) = \Delta X(t) - E\,|JX_{rabbit}(t) - X(t)|. \tag{3}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t). \tag{4}$$

$$\mathrm{J} = 2\mathrm{x}\,(1 - \mathrm{r6}) \tag{5}$$

3.1.4 | Hard besiege

When the prey lacks the necessary energy to escape from the hawks, the hawks encircle their target and initiate a sudden attack.

In cases where the prey possesses sufficient energy to evade the falcon effectively, the hawks employ a strategic approach. They make multiple diving attempts while gradually altering their positions and flight trajectories to deceive the prey before launching a sudden attack. This approach can be mathematically represented as follows:

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \tag{6}$$

In instances where the prey manages to escape, falcons employ a tactic involving multiple dives to lure the prey closer to them before executing a sudden attack. The behavior of falcons is modeled using Levy flight, and this improvement can be described through the following equation:

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \tag{7}$$

$$Z = Y + S \times \mathrm{L}F(D) \tag{8}$$

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{9}$$

which is calculated as follows.

$$X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)) \\ Z & \text{if } F(Z) < F(X(t)) \end{cases} \tag{10}$$

TABLE 2 Notation of Harris Hawks optimization (HHO).

Item	Description
$X(t+1)$	Placement of HHO and iteration
$X_{rand}(t)$	Random population
$x(t)$	The current placement of HHO
$X_{rabbit}(t)$	Placement of rabbit and r1, r2, r3, r4 and q demonstrator the random number (0 – 1)
$UBand\,LB$	Lower and upper bounds of the variable
$X_m(t)$	Average placement of the current population of HHO
$X_i(t)$	Placement of each hawk in iteration and N total number of hawks
E	Prey energy
t	Current iteration number
T	Max number of iterations
E_0	Initial energy random between (-1, 1)
$\Delta X(t)$	Different placement of rabbit
J	Jump of rabbit
||	Absolute value
D	Dimension of problem
S	Random vector by size 1*D
LF	Levy flight function
Y and Z	Obtained from Equations 8,9
u and σ	Random in (0, 1)
β	Constant of 1.5

3.1.5 | Hard besiege with progressive rapid dives

When the prey possesses insufficient energy to escape from the hawks, falcons adopt an aggressive approach by relentlessly besieging their target before it can flee. They then launch a sudden and decisive attack. In this scenario, falcons actively strive to minimize the distance between themselves and the evading prey while enhancing their positioning. Table 2 shows the notation for Harris Hawks Optimization (HHO). The updating of falcon positions can be expressed through the following equation:

$$X(t+1)=\begin{cases} `Y & if\ F(`Y) < F(X(t)) \\ `Z & if\ F(`Z) < F(X(t)) \end{cases} \tag{11}$$

$$Y=X_{rabbit}(t)-E|JX_{rabbit}(t)-X_m\ (t)| \tag{12}$$

$$Z=Y+S\times \mathrm{LF}(D). \tag{13}$$

3.1.6 | Pseudocode of HH

The pseudo-code of the proposed HHO algorithm is reported in Algorithm 1.

Algorithm 1 Pseudo-code of HHO algorithm

Inputs: The population size N and the maximum number of iterations T
Outputs: The location of the rabbit and its fitness value
Initialize the random population X_i (i = 1, 2, . . . , N)
while (stopping condition is not met) **do**
Calculate the fitness values of the hawks
Set X_{rabbit} as the location of the rabbit (best location)
for (each hawk (X_i)) **do**
Update the initial energy E_0 and jump strength J.
E_0=2rand()-1, J=2(1-rand())
Update the E using Eq. (3)
if (|E| ≥ 1) **then**. Exploration phase
Update the location vector using Eq. (1)
if (|E| < 1) **then** . Exploitation phase
if (r ≥0.5 and |E| ≥ 0.5) **then**. Soft besiege
Update the location vector using Eq. (4)
else if (r ≥0.5 and |E| < 0.5) **then** . Hard besiege
Update the location vector using Eq. (6)
else if (r < 0.5 and |E| ≥ 0.5) **then** . Soft besiege with progressive rapid dives
Update the location vector using Eq. (10)
else if (r < 0.5 and |E| < 0.5) **then** . Hard besiege with progressive rapid dives
Update the location vector using Eq. (11)
Return X_{rabbit}

3.2 | Proposed Swarm Intelligence with Multi- Objective Optimization

3.2.1 | Cost and time of replication

The cost depends on each user's budget and choice of data replication near them. The cost of the node varies from place to place according to the proximity of the distance between the users and the node. Time and cost are critical factors in determining and placing data replicas across the node. The equation can be represented as follows:

$$cost\left(DT^{j}\right)=\sum_{y=1}^{n} cost\left(dt_{z}^{y}\right) \tag{14}$$

Where

$$cost\left(dt_{z}^{x}\right)=\sum_{z=1}^{m} x_{z}^{y}\left(p_{z}^{y}+\left(\frac{size\left(dt^{y}\right)}{b_{z}^{y}}\right)*tcost\right. \tag{15}$$

DT^{i} Cost of data set.
dt_{z}^{y} Data replica in the region.
x_{z}^{y} A binary decision variable $q \in (1,2,3,.....l)$.
p_{z}^{y} Price of replica
b_{z}^{y} Bandwidth network between replicas in the region

3.2.2 | Shortest paths problem (SPP) between nodes

Selecting the most cost-effective path between nodes and attaining an optimal approach is crucial within the context of fog computing in the IoT domain. A primary concern lies in preserving load balance during file transfers across nodes while simultaneously minimizing bandwidth usage. This can be expressed through the following equation:

$$\boldsymbol{min}\, \boldsymbol{E}\,(\boldsymbol{xK}) = \boldsymbol{z} + \left[\boldsymbol{K1}\left(\sum_{\boldsymbol{j},(\boldsymbol{i},\boldsymbol{j}) \in \boldsymbol{E}} \boldsymbol{x1j} - \boldsymbol{1} \right) + \boldsymbol{K2}\left(\boldsymbol{1} + \sum_{\boldsymbol{i},(\boldsymbol{i},\boldsymbol{m})} \boldsymbol{xim} \right) + \boldsymbol{K3} \sum\nolimits_{\boldsymbol{i}} \left(\sum\nolimits_{\boldsymbol{j}} \left(\sum_{\boldsymbol{k}} (\boldsymbol{xij} - \boldsymbol{xki}) \right) \right), ((\boldsymbol{i},\boldsymbol{j}),(\boldsymbol{k},\boldsymbol{i}) \in \boldsymbol{E}), \boldsymbol{i} \neq \boldsymbol{s}, \boldsymbol{t} \quad (16)$$

Where

$$z = \sum\nolimits_i \sum\nolimits_j xij, wij, (i,j) \in E, i,j,k = 1,2,....,m, i \neq j \neq k \quad (17)$$

Penalty coefficients are $Ki> > 1,\ i = 1,2,3$.
Some edges (i, j) $\in$ $E\, i,j = 1,2,3,....m.\ i \neq j, k = 1,2,3,...n$.
P path.
S, t first node and last node.
N nodes

3.2.3 | Popularity degree of the data file

File popularity is determined by the frequency of user access to those files. Files that exhibit higher popularity than others are chosen as replicas and strategically positioned to cater to user needs. This can be represented as follows:

$$PD_i = an_i * w_i \quad (18)$$

Based on the popularity degree, each file's replication factor (RFi) is calculated as in Equation 19.

$$RF_i = \frac{PD_i}{RN_i * FS_i} \quad (19)$$

The dynamic threshold (TH) value is calculated as in Equation 20.

$$\mathrm{DH} = \min\left((1-\alpha) * \mathrm{RF}_{\mathrm{system}} \max\left(\forall_{\mathrm{k} \in [1,2,....,\mathrm{l}]} \mathrm{RF}_{\mathrm{k}} \right) \right), \alpha \in [0,1] \quad (20)$$

PD_i popularity degree
an_i number of access
w_i time-based forgetting factor
RF_i replica factor
RN_i number of replica
FS_i size of the data file

3.2.4 | System-level availability

The SBER (system-based availability for every user) mechanism ensures that files remain accessible to all users and upholds the overall popularity of files within the system. SBER functions by ensuring system-wide availability for all users through fog computing. This can be represented as follows:

$$SBER = \frac{\sum_{i=1}^{s}\left(an_k x\left(\sum_{j=1}^{n_k} bs_j\right) x P\left(FA_k\right)\right)}{\sum_{i=1}^{s}\left(an_k x\left(\sum_{j=1}^{n_k} bs_j\right)\right)} \tag{21}$$

3.2.5 | Placement of new replicas

Data replication is strategically positioned at the most suitable node in proximity to users, following optimal methods and minimizing costs related to distance. When selecting the ideal node based on user budget constraints, data replicas are placed accordingly. This can be represented as follows:

$$br_k(dc_i) = \left\lfloor \frac{RF_k(dc_i)}{\sum_{i=1}^{s} RF_k(dc_i)} x_{br_k\,(add)} \right\rfloor \tag{22}$$

3.3 | Salp swarm algorithm

In recent years, SSA has found applications in various domains, drawing inspiration from the collective behavior of salp swarms in oceans [23]. SSA comprises two crucial categories for dividing solutions within the proposed system: leaders and followers. The mathematical equations can be represented as follows:

1) Leader phase: The leader location is modernized using
the following equation:

$$X_j^1 = \begin{cases} X_{bj} + c_1\left(\left(ub_j - lb_j\right)c_2 + l\right) if\ c_3 > 0.5 \\ X_{bj} - c_1\left(\left(ub_j - lb_j\right)c_2 + l\right)\ otherwise \end{cases} \tag{23}$$

$c_1$decreases through the iterations as follows.

$$c_1 = 2e^{-\left(\frac{4l}{T}\right)^2} \tag{24}$$

$X_j^1\ and\ X_{bj}$ repesent new placement, $c_2\ and\ c_3$ random variable from 0 to 1, and $ub_j\ and\ lb_j$ refer to the domain of search at dimintion j.

2) Followers phase: To modernize the followers' locations,
Newton's law of motion is used, which defined as

$$X_j^i = \frac{1}{2}gt^2 + \omega_0 t, i \geq 2 \tag{25}$$

So, the modernizing procedure of followers can be formulated as

$$X_j^i = \frac{1}{2}\left(X_j^i + X_j^{i-1}\right) \tag{26}$$

t iteration

$\omega_0 = 0 \, and \, g$ velocity and the acceleration

3.3.1 | Pseudocode of SSA

The pseudo-code of the proposed SSA algorithm is reported in Algorithm 2 [24].

Algorithm 2 Pseudo-code of SSA algorithm

```
Initialize the population's n=1, 2, 3, 4 …... m. and the total number of
generations (tmax).
Construct the initial set of N solutions X.
while (final condition is not satisfied) do
For each Xi Calculate the fitness of each search and determine X_b.
update c1 in Eq. (23)
   for each i to n do
      if (i==1)
          Update the placement of the leading in Eq. (23)
       else
          Update the placement of the follower in Eq. (26)
       end
   end
     update upper and lower bounds of variables
end
return X_b.
```

3.4 | Energy consumption (EC)

Efficient energy management is paramount in cloud computing, and it becomes particularly crucial in fog computing. The optimization of energy consumption involves mitigating node workloads and virtual machine (VM) power usage during the selection and transfer of files among nodes. Placing data replication strategically within the cloud nodes is employed to alleviate power consumption within the proposed system. This can be represented as follows:

$$E_{RE}(j) = \sum_{i=1}^{n} \emptyset(i,j) * l(i,j) * (P_{max}(j) - P_{idle}(j)) + P_{idle}(j) \tag{27}$$

$$E_{RE} = \sum_{i=1}^{n} E_{RE}(j) \tag{28}$$

where,

$P_{max}(j)$ maximum power of data node

$P_{idle}(j)$ power of data node

$l(i,j)$ load of data node

$\emptyset(i,j)$ cofficent of performance data node (0 to 1)

$E_{RE}(j)$ renewable energy consumption

3.5 | Carbon dioxide emission rate

CDER, or carbon dioxide emission reduction, is intricately tied to both economic and social factors and has significant connections to environmental considerations. We propose an integrated approach to CDER, aiming to conserve gasoline and diesel energy while simultaneously protecting the environment from the impacts of carbon dioxide emissions. This approach can be mathematically represented as [23 – 26]:

$$CDER = \sum_{i=1}^{n} total_{ECS} * ns_i * fce_i * r \tag{29}$$

where,

$total_{ECS}$ total energy

ns_i share of power source

fce_i emission factor of energy

r ratio from CDE equal to $44/12$

3.6 | Mean service time (MST)

When users access files and await a system response, the Minimum Spanning Tree (MST) mechanism significantly enhances response times and user experience speed. Furthermore, it efficiently decreases system loads, upholds load balancing, and minimizes bandwidth utilization across the nodes. This can be represented mathematically as follows:

$$stf_i = \sum_{j=1}^{m} \left(stf(i,j) * \frac{A(i,j)}{A(i)} \right) \tag{30}$$

The mean service time of the system can be defined as follows:

$$mst = \frac{1}{n} * \sum_{i=1}^{n} \sum_{j=1}^{m} . \left(\emptyset(i,j) * \frac{s_i}{tp_j} * \frac{A(i,j)}{A(i)} \right) \tag{31}$$

where,

$stf(i,j)$ expected service time of file in data node

$A(i,j)$ access rate of read requests from data node

$A(i)$ Mean access rate

s_i size of file

tp_j transfer rate of data node

3.7 | Computational complexity

To analyze the time complexity of the MOHHOSSA algorithm for T iterations, we calculate the number of DCs (Data Centers) and the Hybrid Harmony Search Optimization (HHO) with Simulated Annealing (SSA). Let P represent the population size, D denotes the number of variables (dimensions), T signifies the number of iterations, and C represents the cost.

The computational complexity of the MOHHOSSA strategy can be expressed as $O(T * [D * P + C * P])$. This complexity arises from the various stages within the MOHHOSSA approach. Therefore, the time complexity of the MOHHOSSA algorithm simplifies to $O(P * T * C)$, and the time complexity of the overall approach algorithm is $O(N)$.

Algorithm 3: the proposed algorithm MOHHOSSA

Input: IoT of Tasks, number of Regions, cost, CDER, energy-aware, data file replication
Output: optimal green energy aware
Begin
Initialize IOT of tasks
Initialize the population;
Initialize energy aware
Initialize CDER
Initialize MST
repeat
The population size N and the maximum number of iterations T
The location of the rabbit and its fitness value
Initialize the random population X_i (i = 1, 2, . . . , N)
while (stopping condition is not met) **do**
Calculate the fitness values of the Hawks
Set X_{rabbit} as the location of the rabbit (best location)
for (each hawk (X_i)) **do**
Update the initial energy E_0 and jump strength J.
E_0=2rand()-1, J=2(1-rand())
Update the E using Eq. (3)
if (|E| ≥ 1) **then**. Exploration phase
Update the location vector using Eq. (1)
if (|E| < 1) **then** . Exploitation phase
if (r ≥0.5 and |E| ≥ 0.5) **then** . Soft besiege
Update the location vector using Eq. (4)
else if (r ≥0.5 and |E| < 0.5) **then** . Hard besiege
Update the location vector using Eq. (6)
else if (r < 0.5 and |E| ≥ 0.5) **then** . Soft besiege with progressive rapid dives
Update the location vector using Eq. (10)
else if (r < 0.5 and |E| < 0.5) **then** . Hard besiege with progressive rapid dives
Update the location vector using Eq. (11)
Return X_{rabbit} End while
Calculate the MOO
Return the optimal minimum energy-aware solution and CDER.

4 | EXPERIMENTAL EVALUATION

4.1 | Data sets and simulation

The proposed system encompasses a collection of data and files distributed geographically across fog computing infrastructure, facilitating quick retrieval via efficient, shorter paths. Additionally, the proposed method effectively reduces energy consumption in the nodes and IoT sensors, allowing for the selection of nodes that are distant from hotspots. Table 3 provides an overview of the datasets utilized within the proposed system.

TABLE 3 Parameters data sets of the system

Cloud entity	Parameter	Ranges
Nodes	Number of data center	[2, 50]
User	number of users	[50, 500]
Regions	number of regions	[5, 30]
Geographical	Geographical capacity	[20]
Bandwidth	Bandwidth	[2 Mbps, 32 Mbps]
Data sets	Data set size	[2G, 64GB]
Data file	Number of file	[50, 1,000]
Cost of file	Cost of data replica	[100, 5,000]
Storage nodes	Storage capacity	[10, 32]
Transfer rate	Maximum transfer rate	[50, 100 MB/s]
Host	Number of the host	[100, 500]
Processor	Processing elements	[12, 28]
MIPS	MIPS	[500, 4,000]
Memory RAM	RAM	[2, 16GB]
Virtual machine	Number of VM	[50, 1,000]
Processor	Processing elements	[12, 28]
MIPS	MIPS	[500, 4,000]
Memory RAM	RAM	[2, 16GB]
Cloudlet	Number of cloudlet	[1,000, 3,000]
Length of task	Length of task	[1,000, 50,000 MI]

Table 3 outlines a comprehensive range of configurations and capabilities for the cloud environment and its associated entities. It includes various parameters, each with specified ranges. The system can support between 2 to 50 data center nodes and accommodate 50 to 500 users. It operates across 5 to 30 regions, each with a fixed geographical capacity of 20. Bandwidth varies between 2 Mbps and 32 Mbps, while data set sizes range from 2 GB to 64 GB. The number of data files can span from 50 to 1,000, with the cost of data replicas ranging from 100 to 5,000 units. Storage nodes have a capacity between 10 and 32 units, and the maximum transfer rate is between 50 MB/s and 100 MB/s.

In terms of hosts, the system supports 100 to 500, each with 12 to 28 processing elements, offering between 500 to 4,000 MIPS (Million Instructions Per Second). Memory (RAM) for hosts ranges from 2 GB to 16 GB. Virtual machines (VMs) in the system can number between 50 and 1,000, with each VM also having 12 to 28 processing elements, 500 to 4,000 MIPS, and 2 GB to 16 GB of RAM. Lastly, the system can handle 1,000 to 3,000 cloudlets, with task lengths ranging from 1,000 MI to 50,000 MI. This detailed parameter set provides a robust framework for evaluating different scenarios and optimizing performance within the cloud infrastructure.

4.2 | Experimental results

Figure 1 illustrates the implementation of the MOHHOSSA algorithm, focusing on a data loss rate test conducted between nodes. Our proposed algorithm has demonstrated its effectiveness in reducing data loss rates compared to other algorithms, primarily because it ensures high file availability across nodes.

Figure 2 demonstrates the implementation of our proposed algorithm for achieving data file replication across nodes. Notably, it showcases that the mean service time is lower compared to alternative algorithms. Our proposed algorithm effectively minimizes the waiting time for users, enabling swift access to files via the most cost-effective and shortest paths.

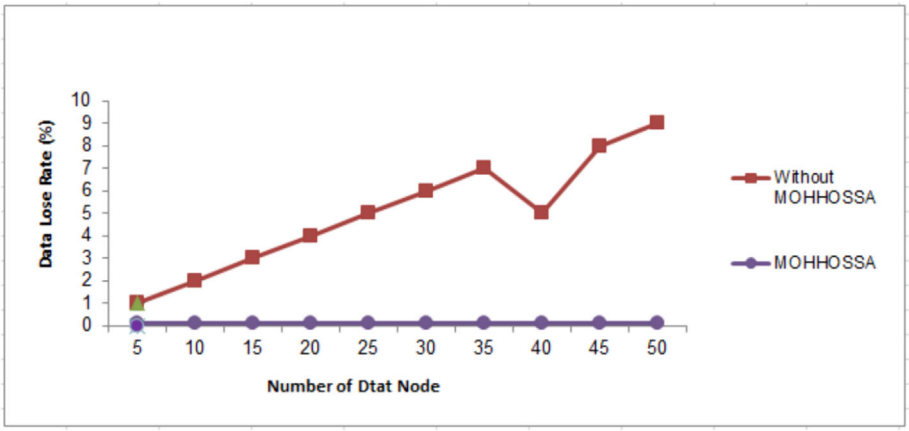


FIGURE 1 Data loss rate between nodes.

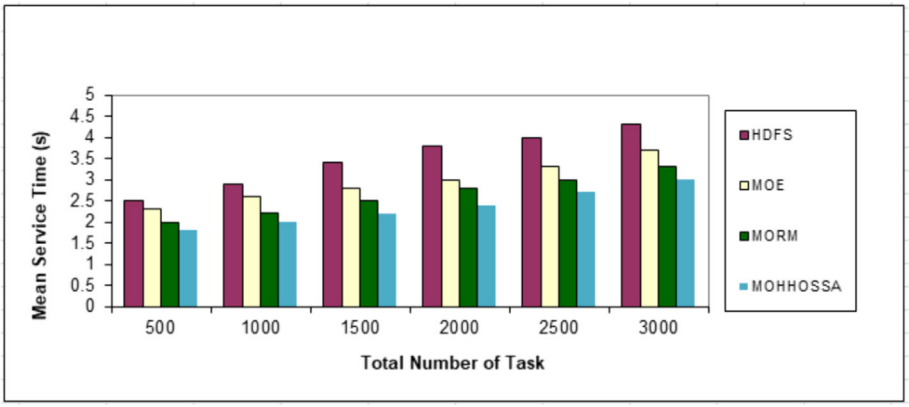


FIGURE 2 Mean service time for users.

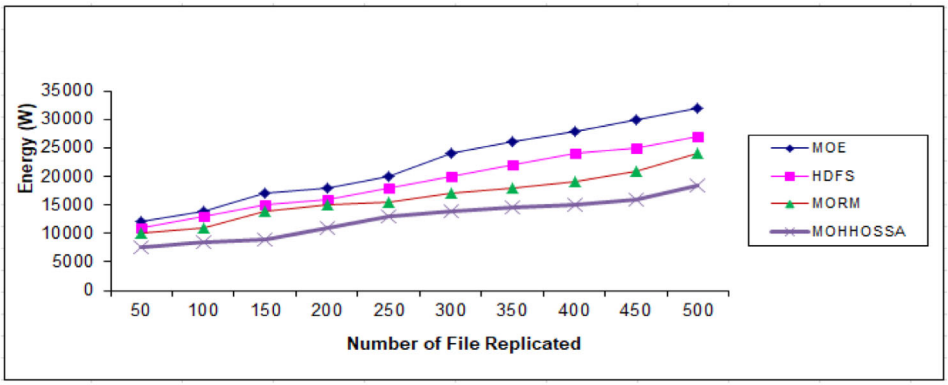
Figure 3 presents the results of our proposed algorithm concerning energy consumption during the selection of multiple data file replications. Our algorithm excels in optimizing energy utilization, achieving a transition toward more sustainable green energy practices. Additionally, the experiment involves a range of tasks, spanning from 500 to 3,000 tasks, aimed at determining the ideal file replication strategy.

Figure 4 depicts the implementation of our strategy in terms of reducing daily carbon dioxide emissions generated by devices. Our system not only significantly decreases heat emissions but also makes substantial contributions to temperature reduction, environmental improvement, and the transition toward environmentally friendly green energy practices. In terms of carbon dioxide emission reduction, our algorithm outperforms other algorithms, establishing its superiority.

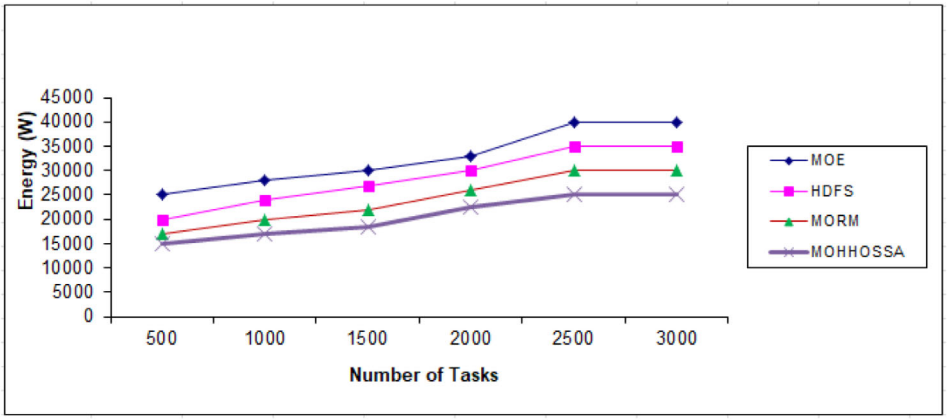
In Table 4, we present the results of the Friedman test, which was conducted to determine if there are significant differences in the performance of the four methods (MOE, HDFS, MORM, and MOHHOSSA) across various task scheduling scenarios. The Friedman test is a non-parametric test used to compare multiple groups when the data is not normally distributed or when assumptions for parametric tests are not met.

The Friedman statistic is a measure of the variability between the methods, and it is compared to a critical value to determine whether to reject the null hypothesis (H0). In this case, the null hypothesis (H0) states that there are no significant differences between the methods in terms of their performance.

- For MOE, the Friedman statistic is 6.32, which is less than the critical value of 7.81. Therefore, we fail to reject the null hypothesis for MOE, indicating that there are no significant differences in its performance across the task scheduling scenarios.



(A) Energy-aware on data file replication



(B) Energy-aware of different tasks

FIGURE 3 Scenario of energy consumption.

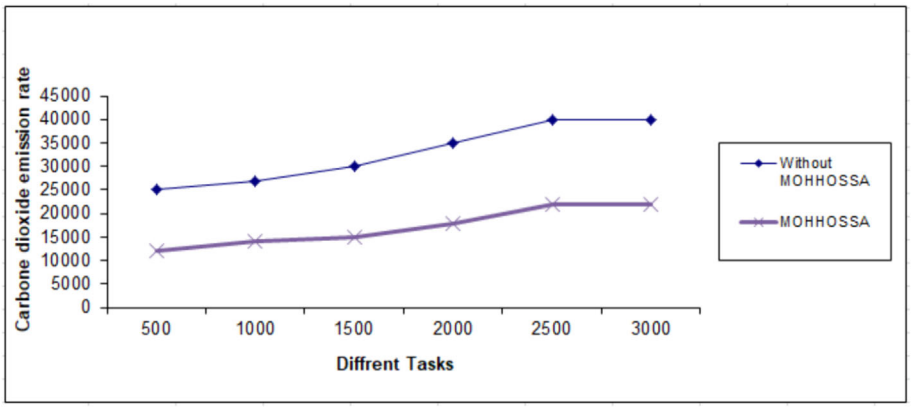


FIGURE 4 Carbone dioxide emission rate.

- For HDFS and MORM, the Friedman statistics are 12.45 and 8.72, respectively, both of which exceed the critical value. Thus, we reject the null hypothesis for HDFS and MORM, suggesting that there are significant differences in their performances across the scenarios.
- For MOHHOSSA, the Friedman statistic is 4.21, which is less than the critical value. Similar to MOE, we fail to reject the null hypothesis for MOHHOSSA.

TABLE 4 Friedman test results.

Method	Friedman statistic	Degrees of freedom	Critical value ($\alpha = 0.05$)	Conclusion
MOE	6.32	3	7.81	Fail to Reject H0
HDFS	12.45	3	7.81	Reject H0
MORM	8.72	3	7.81	Reject H0
MOHHOSSA	4.21	3	7.81	Fail to Reject H0

TABLE 5 Wilcoxon signed-rank test results.

Method comparison	p-value	Conclusion
MOE vs. HDFS	0.002	Reject H0
MOE vs. MORM	0.045	Reject H0
MOE vs. MOHHOSSA	0.356	Fail to Reject H0
HDFS vs. MORM	0.001	Reject H0
HDFS vs. MOHHOSSA	0.012	Reject H0
MORM vs. MOHHOSSA	0.231	Fail to Reject H0

In summary, the Friedman test indicates that there are significant differences in the performance of HDFS and MORM across the task scheduling scenarios. At the same time, MOE and MOHHOSSA do not exhibit significant differences.

Table 5 presents the results of the Wilcoxon signed-rank test, which was conducted to perform pairwise comparisons between the methods (MOE, HDFS, MORM, and MOHHOSSA) to identify which pairs exhibit significant differences in performance. The test assesses whether the differences in performance scores for each pair are statistically significant.

- For the comparison between MOE and HDFS, the p-value is 0.002, which is less than the significance level ($\alpha = 0.05$). Therefore, we reject the null hypothesis (H0) and conclude that there is a significant difference in performance between MOE and HDFS.
- Similarly, the comparisons between MOE and MORM, HDFS and MORM, and HDFS and MOHHOSSA all have p-values less than 0.05, indicating significant differences in their performance.
- On the other hand, the comparisons between MOE and MOHHOSSA and between MORM and MOHHOSSA have p-values greater than 0.05. In these cases, we fail to reject the null hypothesis, suggesting no significant differences in their performances.

In summary, the Wilcoxon signed-rank test reveals specific pairs of methods that exhibit significant differences in performance. MOE and HDFS, as well as HDFS and MORM, show significant differences, while MOE and MOHHOSSA, as well as MORM and MOHHOSSA, do not exhibit significant differences. These statistical tests provide valuable insights into the relative performance of the methods under different task scheduling scenarios, aiding in the selection of the most suitable method for your proposed system.

5 | CONCLUSIONS

This paper presents an innovative strategy for the dynamic selection and placement of data replication utilizing the MOHHOSSA algorithm. This approach is designed to optimize the selection of the most cost-effective and closest paths between IoT-based nodes in cloud computing. The overarching goal of this strategy is to achieve substantial cost reduction, minimize path length, decrease energy consumption, lower carbon dioxide emissions, and improve Mean Service Time (MST). It is noteworthy that reducing costs, shortening paths, ensuring load balancing, improving response times, enhancing availability, and minimizing energy consumption are often conflicting objectives. Effectively addressing these trade-offs is of paramount importance, which led us to introduce a novel strategy rooted in swarm intelligence: MOHHOSSA.

MOHHOSSA breaks new ground in the realm of dynamic selection and placement of data replication by not only optimizing the path for cost efficiency but also conserving energy, accelerating processing times, achieving load balancing, and reaching nodes through the shortest routes. To validate its effectiveness, the MOHHOSSA algorithm underwent rigorous testing using diverse datasets, with results consistently showcasing its superiority over alternative algorithms.

In future work, we plan to further refine our strategy with a focus on energy optimization and its practical applications in real-world scenarios. Additionally, we are exploring opportunities to enhance our approach by integrating it with other swarm intelligence algorithms, particularly multiobjective optimization (MOO). Our objective is to apply the MOHHOSSA algorithm to real, practical cases such as agricultural fields and traffic management, ultimately contributing to more efficient and sustainable IoT-based cloud computing solutions.

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest regarding the publication of this paper.

DATA AVAILABILITY STATEMENTS

Data is available from the authors upon reasonable request.

ETHICS STATEMENT

This article does not contain any studies with human participants or animals performed by any of the authors.

INFORMED CONSENT

Informed consent was obtained from all individual participants included in the study.

REFERENCES

1. Pallewatta S, Kostakos V, Buyya R. QoS-aware placement of microservices-based IoT applications in fog computing environments. *Future Gener Comput Syst*. 2022;131:121-136. doi:10.1016/j.future.2022.01.012
2. Wang M, Zhang Q. Optimized data storage algorithm of IoT based on cloud computing in distributed system. *Comput Commun*. 2020; 157:124-131. doi:10.1016/j.comcom.2020.04.023
3. Taghizadeh J, Arani M, Shahidinejad A. A metaheuristic-based data replica placement approach for data-intensive IoT applications in the fog computing environment. In: *SoftwPract Exper*; 2021.
4. Torabi E, Arani M, Shahidinejad A. Data replica placement approaches in fog computing: a review. *Clust Comput*. 2022;25(5):3561-3589. doi:10.1007/s10586-022-03575-6
5. Jin W, Lim S, Woo S, Park C, Kim D. Decision-making of IoT device operation based on intelligent-task offloading for improving environmental optimization. In: *Complex & intelligent systems*; 2022.
6. Shoaib U, Arshad M, Khattak H, Ullah M, Almogren A, Ali S. Fast data access through nearest location-based replica placement. *Hindawi*. 2022;2022:1-13. doi:10.1155/2022/2496269
7. Khelifa A, Mokadem R, Hamrouni T, Charrada F. Data correlation and fuzzy inference system-based data replication in federated cloud systems. In: *Simulation modelling practice and theory*; 2022.
8. Mohammadi B, Navimipour N. A fuzzy logic-based method for replica placement in the peer to peer cloud using an optimization algorithm. *Wirel Pers Commun*. 2022;122(2):981-1005. doi:10.1007/s11277-021-08936-9
9. John S, Mirnalinee T. A novel dynamic data replication strategy to improve access efficiency of cloud storage. In: *Information systems and e-business management*; 2020.
10. Liu C, Wang J, Zhou L, Rezaeipanah A. Solving the Multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. *Neural Process Lett*. 2022;54:1823-1854. doi:10.1007/s11063-021-10708-2
11. Li J, Shang Y, Qin M, et al. Multiobjective Oriented Task Scheduling in Heterogeneous Mobile Edge Computing Networks. In: *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*; 2022.
12. Yousif A, Alqhtani S, Bashir M, et al. Greedy Firefly Algorithm for Optimizing Job Scheduling in IoT Grid Computing. In: *sensors*; 2022.
13. Peake J, Amos M, Costen N, Masala G, Lloyd H. PACO-VMP: Parallel Ant Colony Optimization for Virtual Machine Placement. In: *Future generation computer systems*; 2022.
14. Sarwar K, Yong S, Jian Y, Rehman S. Efficient privacy-preserving data replication in fog-enabled IoT. *Future Gener Comput Syst*. 2022; 128:538-551. doi:10.1016/j.future.2021.10.024
15. Mansouri N, Rafsanjani M, Javidi M. DPRS: A dynamic popularity aware replication strategy with parallel download scheme in cloud environments. In: *Simul. Model. Pract. Theory*; 2017.
16. Chen D, Yuan H, Hu S, Wang Q, Wang C. BOSSA: A Decentralized System for Proofs of Data Retrievability and Replication. In: *IEEE transactions on parallel and distributed systems*; 2021.

17. Li C, Liu J, Wang M, Luo Y. Fault-tolerant scheduling and data placement for scientific workflow processing in geo-distributed clouds. In: *The journal of systems & software*; 2022.
18. Shi T, Ma H, Chen G, Hartmann S. Cost-effective web application replication and deployment in multi-cloud environment. *IEEE Trans Parallel Distrib Syst*. 2022;33(8):1982-1995. doi:10.1109/TPDS.2021.3133884
19. Majed A, Raji F, Miri A. Replication management in peer-to-peer cloud storage systems. *Clust Comput*. 2022;25(1):401-416. doi:10.1007/s10586-021-03395-0
20. Li C, Cai Q, Youlong L. Optimal data placement strategy considering capacity limitation and load balancing in geographically distributed cloud. *Future Gener Comput Syst*. 2022;127:142-159.
21. Mitsis G, Tsiropoulou EE, Papavassiliou S. Price and risk awareness for data offloading decision-making in edge computing systems. *IEEE Syst J*. 2022;16(4):6546-6557. doi:10.1109/JSYST.2022.3188997
22. Heidari A, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris hawks optimization: algorithm and applications. *Future Gener Comput Syst*. 2019;97:849-872. doi:10.1016/j.future.2019.02.028
23. Ergas C, York R. Women's status and carbon dioxide emissions: a quantitative cross-national analysis. *Soc Sci Res*. 2012;41(4):965-976. doi:10.1016/j.ssresearch.2012.03.008
24. Mirjalili S, Gandomi A, Mirjalili S, Saremi S, Faris H. SALP swarm algorithm: A bio-inspired optimizer for engineering design problems. In: *Adv. Eng. Softw*; 2017.
25. Adom P, Bekoe W, Mensah F, Mensah J, Botchway E. Carbon dioxide emissions, economic growth, industrial structure, and technical efficiency: Empirical evidence from Ghana, Senegal, and Morocco on the causal dynamics. In: *energy*; 2012.
26. Amin S, Ahmad N, Iqbal A, Mustafa G. Asymmetric analysis of environment, ethnic diversity, and international trade nexus: empirical evidence from Pakistan. In: *Environment, development and sustainability*; 2021.

How to cite this article: Mohamed AA, Diabat A, Abualigah L. Optimizing energy-efficient data replication for IoT applications in fog computing. *Int J Commun Syst*. 2024;37(14):e5864. doi:10.1002/dac.5864