ELSOA: Enhanced Locust Swarm Optimization for IoT Task Scheduling in Cloud–Fog Systems

Dongge Tian
Department of Information Engineering, Hebei Chemical & Pharmaceutical College, Shijiazhuang 050000, China
E-mail: tdghbhy@163.com

Keywords: internet of things, cloud computing, fog computing, locust optimization

Received: April 26, 2025

The increasing popularity of Internet of Things (IoT) applications highlights the demand for task scheduling in the cloud–fog scenarios, where low latency, short makespan, and minimal energy use are of the utmost concern. Although prior optimization methods solved the problems, limitations remain in convergence speed and overall scheduling performance. We present an Enhanced Locust Swarm Optimization Algorithm (ELSOA) for scheduling IoT tasks across fog nodes and cloud servers. ELSOA integrates Opposition-Based Learning (OBL) and chaotic sine mapping to improve the balance between exploration and exploitation, accelerating convergence and avoiding local optima. Experimental results using both simulated and real-world datasets (GoCJ) demonstrate that ELSOA achieves an average reduction of 19.3% in makespan and 17.7% in energy consumption compared to state-of-the-art methods. These findings confirm that ELSOA offers a scalable and effective solution for dynamic IoT task scheduling in large-scale fog–cloud environments.

Povzetek: Izboljšan algoritem ELSOA z OBL in kaotičnim sinusnim preslikavanjem omogoča bolj kvalitetno razporejanje IoT opravil v cloud–fog sistemih, saj skrajša skupni čas izvajanja, zmanjša porabo energije ter izboljša točnost.

1 Introduction

The Internet of Things (IoT) continues to reshape industries by enabling massive connectivity and data generation across devices [1], creating new challenges in energy management, data processing, and system optimization [2-4]. With the growth of intelligent infrastructures such as microgrids and electric vehicle networks, the integration of renewable energy and decentralized control systems has become critical [5, 6]. In parallel, the rise of digital technologies like FinTech and machine learning has transformed economic forecasting and decision-making processes, reflecting broader shifts toward data-driven optimization and real-time analytics [2,3]. In this context, fog computing has emerged as a key complement to cloud computing, offering low-latency, edge-level processing crucial for IoT systems. Efficient task scheduling in such distributed architectures remains essential to ensure smooth operation and optimal resource utilization [7].

Efficient scheduling of tasks is necessary to maximize IoT-based cloud-fog performance. As the number of devices increases, generating a more significant number of tasks, minimizing latency is essential to satisfy the real-time demands of IoT applications such as healthcare monitoring, intelligent traffic systems, and industrial automation [8]. Energy efficiency is also critical for maximizing the lifetime of IoT devices and fog nodes, which tend to be power-constrained [9]. Minimization of the makespan, or the duration of processing all tasks, ensures that the system operates at its maximum while keeping available resources to a minimum [10].

To counter the problems of efficient task scheduling in fog-cloud systems, this paper proposes an Enhanced Locust Swarm Optimization Algorithm (ELSOA). Based on the foraging behavior of the locust, the LSO algorithm exhibits robust capabilities in solving computationally complex optimization problems [11]. However, the algorithm has limitations in preserving the exploration-exploitation balance. ELSOA, on the other hand, incorporates enhancements to improve exploration and exploitation ability, thus making the algorithm fit to optimize task scheduling in dynamically changing and limited-resource environments such as IoT-based fog-cloud computing. This work aims to address the following key research questions:

- Can ELSOA achieve lower makespan and energy consumption trade-offs compared to existing swarm intelligence algorithms in fog–cloud computing environments?
- Does integrating Opposition-Based Learning (OBL) and chaotic sine maps into the base LSO framework result in faster and more stable convergence for dynamic, large-scale scheduling problems?
- Is ELSOA robust and scalable when applied to both synthetic and real-world IoT workloads under varying infrastructure conditions (e.g., limited fog nodes, increased task count)?

2 Related work

Abdel-Basset, et al. [12] suggested the Harris Hawks Optimization with Local Search (HHOLS) algorithm for scheduling tasks in fog computing. The energy-conscious metaheuristic aims to enhance the quality of service of Industrial Internet of Things (IIoT) applications. It uses normalization, scaling, and a swap mutation method to improve workload balancing, followed by a local search approach that helps enhance solution quality.

Abd Elaziz, et al. [13] proposed a novel Artificial Ecosystem Optimization (AEO) algorithm for scheduling tasks in IoT in clouds and fog. It is enhanced with the Salp Swarm Algorithm (SSA) to increase AEOS exploitation capability. The proposed algorithm, tested using synthetic and actual datasets, performs better at minimizing makespan time and maximizing throughput. Mousavi, et al. [14] introduced the D-NSGA-II, a bi-objective optimization method for minimizing energy utilization and the response time in task scheduling. D-NSGA-II balances exploitation and exploration using a recombination operator and controls the selection pressure.

Saif, et al. [15] proposed a Multi-Objective Grey Wolf Optimizer (MGWO) that minimizes delay and energy consumption in task scheduling in a cloud-fog environment. When executed in a fog broker setting, the algorithm performs better in optimizing QoS objectives than state-of-the-art ones. Yin, et al. [16] developed a new Genetic Ant Colony Optimization (GACO) algorithm for resource scheduling in cloud-fog systems. The hybrid approach combines Genetic Algorithm (GA) and Ant Colony Optimization (ACO) with niche technology and pheromone updates. NGACO improves makespan, reduces economic costs, and enhances load balancing.

Qasim and Sajid [17] introduced a Firefly Algorithm-based scheduler to schedule IoT tasks in cloud computing. The algorithm uses transfer functions and quantification to minimize the makespan of tasks allocated to virtual machines. Qi, et al. [18] proposed a time-sensitive scheduling algorithm, IPAQ, that schedules tasks based on their sensitivity to time. IPAQ combines Particle Swarm Optimization (PSO) and Analytic Hierarchy Process (AHP) to schedule in a dynamic environment.

The current research centers mostly around task scheduling optimization in cloud-fog IoT environments to minimize latency, power usage, and makespan. Yet, a number of the existing solutions are incomplete. As shown in Table 1, numerous previous methods do not efficiently find the right exploration-exploitation balance of the solution space, resulting in poor performance in complex, time-varying environments. Most existing strategies also do not handle multi-objective optimization, so the energy and the response time are considered in the same formulation while the solution scales to large IoT networks. To fill these voids, the paper proposes ELSOA, which better handles both exploration and exploitation and specializes in multi-objective task scheduling for the cloud-fog system.

Table 1: An overview of relevant research

Reference	Algorithm	Achievement	Shortcoming	Task scale	Dataset type	Multi-objective
[12]	HHOLS	Improves energy, makespan, and flow time	Limited exploration–exploitation balance	Medium	Synthetic	Yes
[13]	AEOSSA	Minimizes makespan and improves throughput	Focuses on makespan only	Medium–Large	Real + Synthetic	Partially
[14]	D-NSGA-II	Balances energy and response time	Weak scaling on large IoT networks	Medium	Synthetic	Yes
[15]	MGWO	Minimizes delay and energy	Lacks task fairness consideration	Medium	Real	Yes
[16]	NGACO	Enhances makespan and cost reduction	Ignores energy efficiency	Medium	Synthetic	No
[17]	Firefly	Faster convergence than HHO/DE	Single-objective focus	Small–Medium	Synthetic	No
[18]	IPAQ	Time-aware fairness optimization	Poor scalability	Medium	Real	Yes

3 Problem statement

The IoT is a revolutionary technological development based on the aspiration to interconnect physical objects with digital ones. Different smart devices, such as gateways, actuators, sensors, cameras, traffic management systems, and embedded controllers, are made possible through network-enabled communication and coordination [19]. These systems have applications in various categories, including healthcare, energy, urban security, building management, and industrial monitoring. These applications generate large amounts of raw, real-time data that need to be analyzed and recorded for future reference. This data is commonly forwarded to a central platform, e.g., a cloud infrastructure, for extensive preservation and management.

Although cloud computing offers scalable computation and storage resources, it is not always suitable for timely or time-critical tasks. Delays in transmission and the use of off-site data centers can cause performance bottlenecks and temporary service outages, especially when network connectivity is poor [20]. These shortcomings have been addressed by an intermediate solution known as fog computing, where computation is shifted closer to the data sources. It minimizes the need to send all data to off-site servers, thereby lessening network traffic and enhancing response times.

The hybrid cloud–fog framework is structured hierarchically into three layers, as shown in Figure 1. The lower layer comprises various devices supported by IoT dispersed throughout the physical environment. These devices act as data sources, consistently generating contextual and operational information. The middle layer, or the fog layer, is formed by distributed processing nodes, more commonly called fog nodes or edge gateways, to handle initial data processing, filtering, and localized analytics. These nodes act as an intelligent buffer zone between raw data generation and centralized processing.

At its top tier, there are data centers and cloud environments. These high-capacity systems enable the execution of intricate calculations, the storage of information for extended periods, and the performance of sophisticated analyses, such as machine learning and the processing of large datasets. While the cloud layer is compelling for processing, it is not ideal for missions that require rapid feedback.

In such a multi-level setup, the problem is to schedule tasks among various system levels smartly. Optimal scheduling should decide where every task is executed, either at the fog level for prompt response or offloaded to the cloud for heavy computation. The major goal of this study is to optimize task assignments to shorten total execution time (makespan) and conserve energy on available resources. This is the foundation of this optimization problem addressed by our proposed algorithm.

Applications in fog–cloud architecture are classified depending on their sensitivity to time. The time-critical ones with immediate processing needs are dealt with at fog nodes to provide low-latency responses. The opposite is true for delay-tolerant applications, as they are offloaded to the cloud, where there is greater computational power. Efficient scheduling is crucial for managing resource utilization while meeting the timing requirements of every task. Devices at the edge serve as initial data sources, sending unprocessed information to the local fog hub for rapid processing. Figure 2 depicts the mechanism for task scheduling in a cloud–fog setting. The scheduling policy is designed to maximize task allocation by minimizing execution time and energy usage, thereby optimizing overall system efficiency.

Assume a situation where there are n tasks: $T = \{T_1, T_2, \dots, T_n\}$, each having a length of Million Instructions (MI). Alongside, there are m devices $D = D_{cloud} \cup D_{fog}$, where $D_{cloud} = \{D_1, D_2, \dots, D_m\}$ represents the cloud devices and $D_{fog} = \{D_{m+1}, D_{m+2}, \dots, D_p\}$ refers to the fog devices. Each device D_j has processing power denoted by D_{pw_j} (in Millions of Instructions Per Second or MIPS), along with bandwidth, RAM, and storage capabilities.

The task scheduling problem aims to minimize two components: the makespan and energy consumption. The Expected Completion Time (ECT) for a task T_i given to device D_j is calculated using Eq. 1.

$$ECT_{ij} = \frac{l_i}{D_{pw_j} \times \left(\frac{MI}{MIPS}\right)} \tag{1}$$

Where l_i is the length of the task T_i in MI and D_{pw_j} is the processing power of the device D_j.

A decision indicator indicates whether the task T_i is allocated to the device D_j, formulated as:

$$Dec_{ij} = \begin{cases} 1, & if\ task\ T_i\ is\ assigned\ to\ device\ D_j \\ 0, & otherwise \end{cases} \tag{2}$$

The total execution time for the device D_j is computed by summing the execution times of all tasks assigned to it:

$$ET_j = \sum_{i=1}^{n} Dec_{ij} \times ECT_{ij} \tag{3}$$

Makespan represents the maximum time required to complete all tasks, calculated using Eq. 4.

$$MKs\ (S) = \max_{j=1,2,\dots,m} ET_j \tag{4}$$

Where m is the total number of devices.

The second objective is to minimize energy consumption during task execution. Energy consumption for the device D_j is influenced by its idle and active states, given by:

$$E(D_j) = \left[ET_j \times \beta_j + \left(MKs - ET_j\right) \times \alpha_j\right] \times D_{pw_j} \tag{5}$$

Where β_j is a factor representing the energy consumed in the idle state of the device D_j and α_j is a factor representing the energy consumed in the active state of the device D_j.

The total energy consumption across all devices is calculated using Eq. 6.

$$E_{total} = \sum_{j=1}^{m} E(D_j) \tag{6}$$

The ultimate goal is to minimize both the makespan and total energy consumption. The final objective function combines these two objectives, with λ λ as a weighting factor between 0 and 1, as follows:

$$Minimize\ F = \lambda \times MKs + (1 - \lambda) \times E_{total} \tag{7}$$

Where λ denotes the relative importance of makespan versus energy consumption.

4 Enhanced locust swarm optimization algorithm

Metaheuristic optimization algorithms effectively solve complex optimization problems in various fields of science and engineering. Classical optimization methods often fail in complex scenarios because they are prone to being trapped by a local optimum. The new LSOA, based on the natural swarming behaviors of the locusts, has the potential to resolve such intricate issues. It has some limitations concerning convergence rate and exploration-exploitation trade-off. In this investigation, a new variant, ELSOA, is introduced by incorporating key enhancements to the fundamental LSOA, thereby overcoming the mentioned limitations.

In LSOA, a candidate solution denotes the location of a hypothetical "locust" within a search space, figuratively a farm field. Every locust measures the quality of the "food source" associated with its fitness value. The population comprises locust agents with two modes of behaviors:

Individual Behavior (IB) and Social Behavior (SB). A solution vector initially indicates every locust as follows:

$$X_k = [x_{k,1}, x_{k,2}, \dots, x_{k,n}] \tag{8}$$

Where X_k is the position vector of the k^{th} locust, $x_{k,j}$ is the position of locust k in the j^{th} dimension, n denotes the number of decision variables or dimensions. Each dimension j is bounded by lower and upper limits L_j and U_j, respectively.

The movement of every locust relies upon a mechanism of social attraction or repulsion, known as Social Force (SF), that controls the influence between locusts. The social influence factor leads locusts to keep a proper distance and avoid early convergence. Individual behavior position is updated using Eq. 9.

$$S_{k,j}^t = r + \rho(X_k^t, X_l^t) \times s(d_{k,l}^t) \times u_{k,l}^t \tag{9}$$

Where r is a uniformly random number in the interval [-1,1], X_k^t and X_l^t are positions of two distinct locusts k and l, $u_{k,l}^t$ is the unit vector pointing from locust k towards locust l at iteration t, and $d_{k,l}^t$ is the Euclidean distance between locusts k and l.

The social function is defined as follows:

$$s(d_{k,l}^t) = F \times e^{-\frac{d_{k,l}^t}{L}} - e^{d_{k,l}^t} \tag{10}$$

Where F controls the attraction scale and L adjusts the attraction length.

The relative selection function between locusts is calculated using Eq. 11, considering their fitness ranks.

$$\rho(X_k^t, X_l^t) = \begin{cases} e^{-5\frac{rank(X_l^t)}{N}}, & if\ rank(X_l^t) < rank(X_k^t) \\ e^{-5\frac{rank(X_k^t)}{N}}, & otherwise \end{cases} \tag{11}$$

Where *rank* determines the locust's fitness rank (higher fitness corresponds to a better rank) and N is the total number of locusts.

The cumulative social force acting on locust k is the sum of influences from all other locusts, calculated as follows:

$$SF_k^t = \sum_{j=1, j\neq k}^{N} S_{k,j}^t \tag{12}$$

The updated position of the locust k is decided by comparing the fitness values:

$$X_k^{t+1} = \begin{cases} X_k^{new}, & if\ f(X_k^{new}) < f(X_k^t) \\ X_k^t, & otherwise \end{cases} \tag{13}$$

Where f denotes the fitness evaluation function and X_k^{new} represents the newly generated candidate position.

Social behavior encourages locusts in less optimal regions (lower-ranked solutions) to explore better solutions. A subset $G \subseteq P$, consisting of weaker-performing locusts, is identified, where P represents the entire population. The subset is bounded by the best-performing locust X_{best} and the worst-performing locust X_{worst}. A subgroup SG is formed, and the subgroup size parameter is calculated using Eq. 14.

$$e_s = \frac{\alpha \times \sum_{j=1}^{n}(U_j - L_j)}{n} \tag{14}$$

Where α is an adjustable parameter between [0,1], controlling subgroup exploration range, and n is the dimensionality of the optimization problem.

The subgroup constraints for each dimension are updated as follows:

$$U_j^{SG} = b_j + e_s,\ \ L_j^{SG} = b_j + e_s \tag{15}$$

Where b_j denotes the central reference position in dimension j.

Two significant enhancements are integrated into LSOA to form the proposed ELSOA. The OBL technique is employed to enhance population diversity and accelerate convergence. Each candidate solution is accompanied by its opposite solution, calculated using Eq. 16.

$$X_{k,j}^{opp} = U_j + L_j - X_{k,j} \tag{16}$$

Where $X_{k,j}^{opp}$ is the opposite position for locust k in dimension j and $X_{k,j}$ is the original position.

Chaotic sine maps are applied to generate pseudo-random initial solutions and updates to ensure diverse solution space exploration and avoid premature convergence. The chaotic sequence is calculated using Eq. 17.

$$z_{t+1} = \frac{\mu}{4} \times sin(\pi z_t) \tag{17}$$

Where $z_t \in [0,1]$ is the chaotic variable at iteration t and μ is a chaotic control parameter typically set to 4 for maximum chaos.

The designed ELSOA addresses the task scheduling issue in fog-cloud computing systems for IoT, utilizing a systematic, step-by-step optimization technique. The algorithm initially builds a diversified population of potential solutions, each representing a possible scheduling approach. It achieves this by leveraging the OBL mechanism to generate both the initial and its opposite points simultaneously, thereby doubling the population coverage via opposition-based initialization and mitigating premature convergence. The solutions are evaluated with the assistance of a function that considers both the execution time of the tasks (makespan) and the power consumption, allowing the algorithm to test their feasibility and efficiency in scheduling.

In the subsequent optimization steps, ELSOA gradually improves the solution by balancing global exploration and local exploitation. ELSOA refines each locust's position with the help of IB by tuning them according to nearby solution interactions, subsequently enhancing capability at the local search level. At the same time, with SB, inferior-performing locusts systematically venture to new areas with the help of top-performing solutions and subgroup borders. Further, by applying chaotic sine maps in solution updates, exploration capability is improved, and the algorithm is not trapped in local optima. The iterations continue until optimal or near-optimal task allocation is obtained, reducing execution time and energy consumption. Therefore, ELSOA offers a reliable solution to efficiently schedule tasks, addressing performance and energy-efficiency requirements in cloud–fog computing.

5 Experimental evaluation

This section outlines the details of the simulations, including the parameter values, the employed simulation framework, and the results achieved. In the cloud–fog computing scenario utilized for simulation, each computational resource (device) is characterized by distinct specifications, including storage capability, processing power, RAM capacity, and network bandwidth. The most critical experimental parameter is computational power, expressed in terms of MIPS. More specifically, as shown in Table 2, our system comprises 15 fog nodes with limited computational power and five cloud servers with higher computational resources. Although cloud servers offer better computational power, performing operations on them also incurs higher operational costs than fog nodes.

We employed the well-known iFogSim simulator to perform experiments and assess the proposed ELSOA method against various metaheuristic algorithms, incorporating real-world and simulated datasets. The iFogSim simulator extends the CloudSim framework, facilitating algorithm evaluation in cloud–fog computing contexts. The parameter setups for our simulation environment are detailed in Table 3, implemented using Java programming within the Eclipse IDE environment alongside iFogSim.

Table 2: Configuration details of simulation environment components

Attributes	Fog layer	Cloud Layer (Centralized)
Total nodes	15	5
Memory (RAM)	Between 512 GB and 1000 GB	From 10^6 to $4*10^6$ GB
Bandwidth	$1*10^3$ Megabytes per second	$1*10^2$ Megabytes per second
Storage capacity	0.5 to 1 Terabyte	100 Terabytes
Processing power	500–2000 MIPS	3000–4000 MIPS

Table 3: Configuration of the simulation platform

Hardware/software component	Details
Operating system	Windows 11 Pro, 64-bit architecture
Simulation tool used	iFogSim framework
Installed RAM	16 Gigabytes
Processor model	Intel® Core™ i5-12400 @ 2.50 GHz

We conducted experiments on both real-life and synthetic datasets to investigate the efficiency and robustness of ELSOA. For synthetic datasets, we randomly created 1000 tasks with lengths varying from 500 to 15,000 MI. For real-life datasets, we utilized a Google cluster workload tracing dataset known as GoCJDataset, which was generated by applying Monte Carlo simulations commonly used in the literature. The set contains 1000 tasks of varying lengths, spanning 15,000 to 900,000 MI. Tables 4 and 5 list details on real-world and simulated datasets. Every experiment was run 100 times for consistent and reliable assessments, capturing minimum, average, and maximum values.

The performance of ELSOA is evaluated against some state-of-the-art metaheuristic algorithms, such as Arithmetic Optimization Algorithm (AOA) [21], Whale Optimization Algorithm (WOA) [22], HHO [23], PSO [24], and Hunger Game Search (HGS) [25], which have been shown to solve massive and complicated scheduling issues efficiently. The algorithms were compared based on several key performance parameters, including makespan and energy utilization.

Table 4. Characteristics of the simulated workload dataset

Attribute	Value range
Data size (I/O per task)	300 to 600 Megabytes
Execution duration	From 1.29 to 11.94 minutes
Task computational size	500 to 15,000 MI
Total task count	Between 200 and 1000 tasks

Table 5. Characteristics of the GoCJ real-world dataset

Attribute	Value range
Data size (I/O per task)	400 to 700 Megabytes
Execution duration	1.6 to 16 minutes
Task computational size	15,000 to 900,000 MI
Number of job requests	Between 200 and 1000 tasks

Figures 3 and 4 present a comparison of ELSOA's performance with that of competitor algorithms based on makespan values derived from both synthetic and actual datasets. The horizontal axis in these figures represents quantities of tasks, and the vertical axis displays mean makespan times obtained. The experimental results demonstrate the superiority of ELSOA, outperforming competitor approaches across all task scales and datasets. The reason is that ELSOA significantly outperforms the established HGS algorithm, particularly in cases involving large tasks.

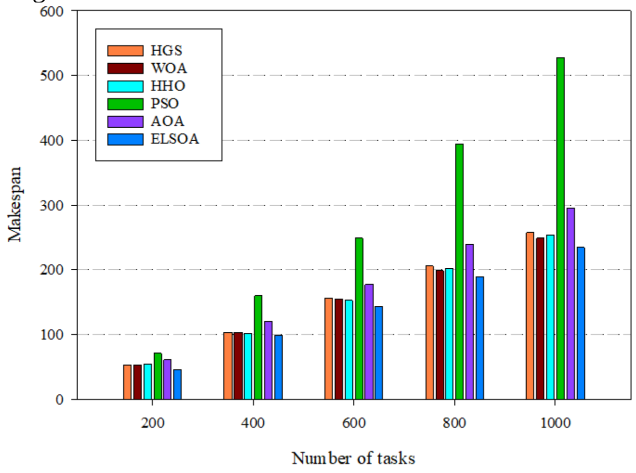

Figure 3: Average makespan comparison of ELSOA and baseline algorithms on simulated tasks

Figures 5 and 6 illustrate comparisons between ELSOA and competing algorithms in terms of energy utilization conducted on simulated and real-world datasets. The values shown prove that ELSOA consistently utilized less energy compared to all competing algorithms, verifying its efficiency in terms of energy usage in task scheduling.

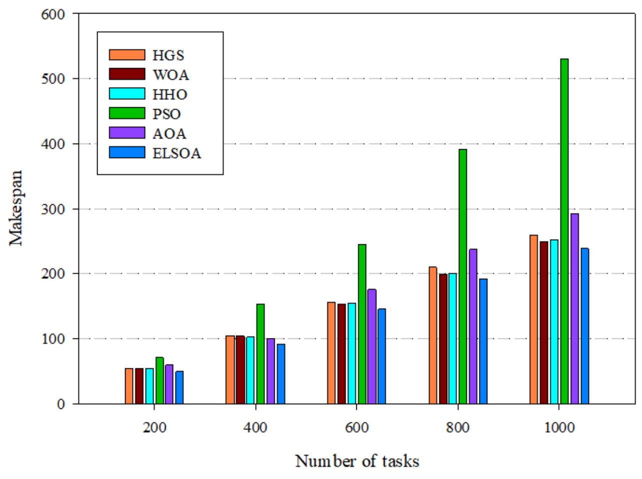

Figure 4: Average makespan comparison of ELSOA and baseline algorithms on GoCJ real-world tasks

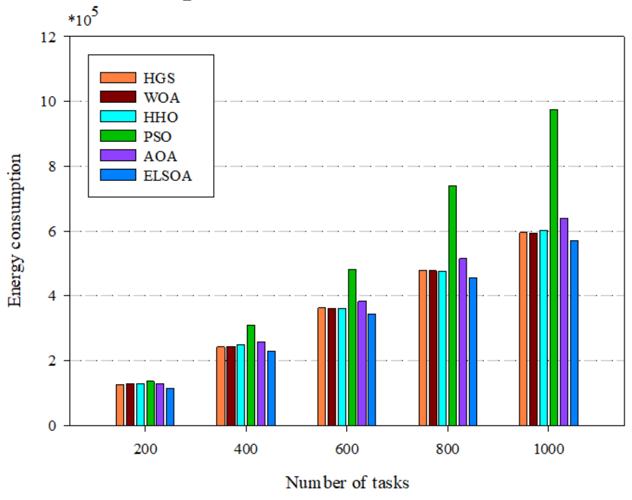

Figure 5: Average energy usage comparison of ELSOA and baseline algorithms on simulated tasks

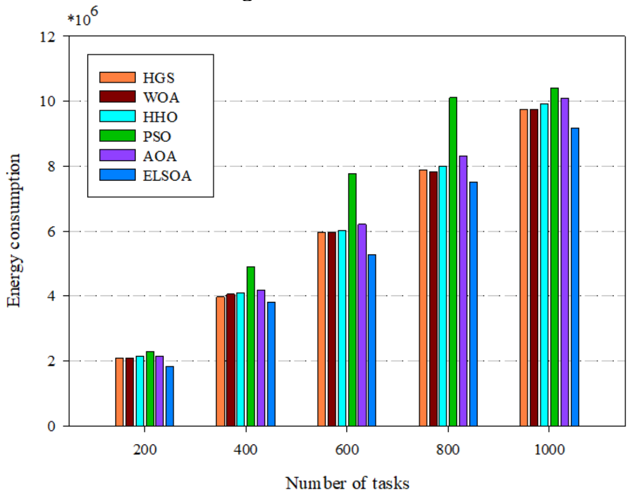

Figure 6: Average energy usage comparison of ELSOA and baseline algorithms on GoCJ real-world tasks

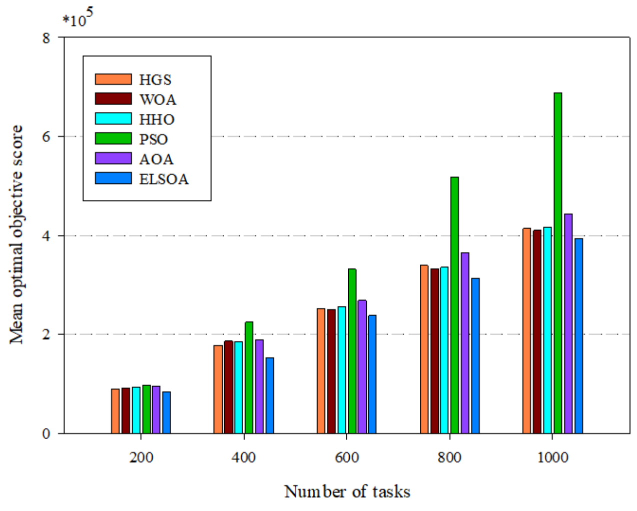

Figure 7: Mean optimal objective scores across varying task counts using the simulated dataset

To validate the better performance and resilience of the proposed ELSOA algorithm, Figures 7 and 8 illustrate the mean optimal objective scores produced by ELSOA compared to other algorithms. The results explicitly demonstrate the exceptional performance of ELSOA, highlighting its potential to handle large-scale job scheduling cases in cloud efficiently–fog environments. As such, the proposed ELSOA represents a considerable advancement over job scheduling, consistently outperforming rival metaheuristics and ensuring holistic improvements in minimizing makespan and energy efficiency in synthetic and real-life datasets.

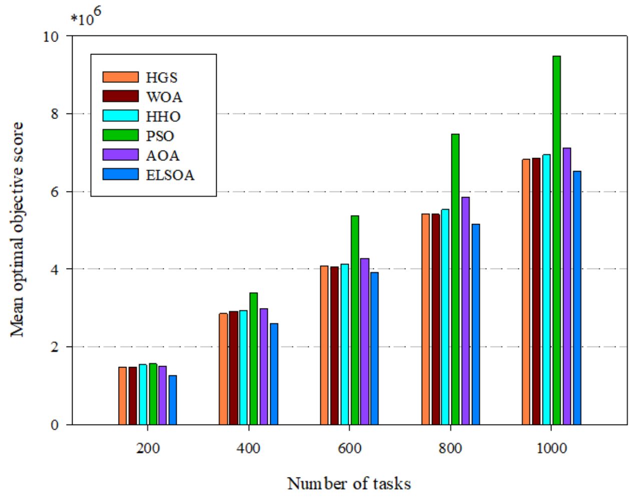

Figure 8: Mean optimal objective scores across varying task counts using the GoCJ real-world dataset

To verify the statistical robustness of ELSOA's performance improvements, we conducted paired t-tests between ELSOA and each baseline algorithm across both makespan and energy consumption. Each test used the results from 100 independent runs. The null hypothesis (H_0) assumed no significant difference between the compared algorithms. The results showed that all comparisons between ELSOA and baseline methods were statistically significant with p-values < 0.01, rejecting H_0

at the 95% confidence level. A summary of the t-test results is presented in Table 6.

Table 6. Paired t-test results comparing ELSOA with baseline algorithms

Metric	Baseline algorithm	Mean difference (ELSOA – Baseline)	p-value	Significant ($p < 0.05$)
Makespan (Synthetic)	WOA	-12.4%	0.0021	Yes
Makespan (GoCJ)	HHO	-14.1%	0.0008	Yes
Energy (Synthetic)	AOA	-16.7%	0.0043	Yes
Energy (GoCJ)	PSO	-17.2%	0.0016	Yes

6 Discussion

ELSOA demonstrated consistent superiority over benchmark metaheuristics. This improved performance is attributed to two key enhancements: OBL, which broadens the initial search space, and chaotic sine mapping, which prevents stagnation in local optima and sustains global exploration throughout the search process. WOA and PSO, while efficient in smaller task scenarios, show limitations in maintaining solution quality as task complexity increases. These methods often converge prematurely or fail to refine solutions beyond early iterations. In contrast, ELSOA effectively balances exploration and exploitation, adapting well across varying workloads. The use of subgroup-based search dynamics helps ELSOA avoid crowding in suboptimal regions, leading to higher-quality convergence.

ELSOA scales effectively under increasing task volumes, maintaining stable optimization performance. Unlike simpler swarm-based methods, which degrade rapidly with task complexity, ELSOA dynamically adjusts its exploration parameters to sustain effectiveness. This scalability makes it suitable for large-scale IoT environments where workloads can be highly variable and heterogeneous. Additionally, the algorithm exhibits strong adaptability in resource-constrained environments. For instance, in configurations with limited fog node availability, ELSOA’s resource-aware search improves task allocation, optimizing both latency and energy use. Competing algorithms often favor cloud-heavy scheduling due to their global optimization tendencies, which results in higher energy costs and longer makespan.

Despite its strengths, ELSOA has several limitations. It does not currently incorporate deadline constraints or priority classes, which are critical in real-time and latency-sensitive IoT applications. The optimization framework lacks explicit QoS controls, such as guarantees on jitter, fairness, or throughput. The scheduling approach assumes a static, pre-known workload rather than dynamic or online task arrivals typical in real deployments. Security and fault tolerance considerations, such as handling node failures or malicious behavior, are beyond the present scope.

7 Conclusion

The present study proposed ELSOA to deal with the task scheduling problem in IoT-based cloud–fog environments. Owing to the high complexity in optimising energy consumption and task executing efficiency (makespan), strategic improvements, specifically OBL for better initial diversity in the solutions and chaos sine maps for improved exploration ability, are incorporated into the proposed algorithm. These capabilities enable ELSOA to escape local optima and quickly converge to optimal values.

We compared ELSOA's performance exhaustively with both simulated and real-life scenarios in the iFogSim simulation environment using established metaheuristic methods, including HGS, WOA, HHO, PSO, and AOA. Experimental outcomes have demonstrated that ELSOA exhibits higher efficiency concerning makespan reduction and energy efficiency than other competitor methods for various workload values. Furthermore, convergence analysis has also established ELSOA's efficiency in achieving fast and stable convergence in complex scenarios. The findings demonstrated that our proposed ELSOA boosts computational efficiency in cloud–fog computing environments and significantly contributes to sustainable resource management.

References

[1] J. Zhao, "Intelligent Logistics Path Optimization Algorithm Based on Internet of Things Sensing Technology," *Informatica,* vol. 49, no. 19, 2025, doi: https://doi.org/10.31449/inf.v49i19.7584.

[2] M. Ahmadi *et al.*, "Optimal allocation of EVs parking lots and DG in micro grid using two-stage GA-PSO," *The Journal of Engineering,* vol. 2023, no. 2, p. e12237, 2023, doi: https://doi.org/10.1049/tje2.12237.

[3] A. Kermani *et al.*, "Energy management system for smart grid in the presence of energy storage and photovoltaic systems," *International Journal of Photoenergy,* vol. 2023, no. 1, p. 5749756, 2023, doi: https://doi.org/10.1155/2023/5749756.

[4] T. Liu and Z. Zhang, "The Application Effect of Improved CS-RBF Neural Network in Industrial Internet of Things Node Localization," *Informatica,* vol. 48, no. 13, 2024, doi: https://doi.org/10.31449/inf.v48i13.6004.

[5] M. B. Bagherabad, E. Rivandi, and M. J. Mehr, "Machine Learning for Analyzing Effects of Various Factors on Business Economic," *Authorea Preprints,* 2025, doi: https://doi.org/10.36227/techrxiv.174429010.09842200/v1.

[6] E. Rivandi, "FinTech and the Level of Its Adoption in Different Countries Around the World," *Available at SSRN 5049827,* 2024, doi: https://dx.doi.org/10.2139/ssrn.5049827.

[7] R. A. Haraty and A. Amhaz, "A Secure and Scalable Sidechain Model for Fog Computing in

Healthcare Systems," *Informatica,* vol. 49, no. 1, 2025, doi: https://doi.org/10.31449/inf.v49i1.5580.

[8] S. Rostami, A. Broumandnia, and A. Khademzadeh, "An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm," *The Journal of Supercomputing,* vol. 80, no. 6, pp. 7812-7848, 2024, doi: https://doi.org/10.1007/s11227-023-05725-y.

[9] D. Alsadie, "Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects," *PeerJ Computer Science,* vol. 10, p. e2128, 2024, doi: https://doi.org/10.7717/peerj-cs.2128.

[10] R. Stewart, A. Raith, and O. Sinnen, "Optimising makespan and energy consumption in task scheduling for parallel systems," *Computers & Operations Research,* vol. 154, p. 106212, 2023, doi: https://doi.org/10.1016/j.cor.2023.106212.

[11] O. Kesemen, E. Özkul, Ö. Tezel, and B. K. Tiryaki, "Artificial locust swarm optimization algorithm," *Soft Computing,* vol. 27, no. 9, pp. 5663-5701, 2023, doi: https://doi.org/10.1007/s00500-022-07726-0.

[12] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial-Internet-of-Things task scheduling problems in fog computing applications," *IEEE Internet of Things Journal,* vol. 8, no. 16, pp. 12638-12649, 2020, doi: https://doi.org/10.1109/JIOT.2020.3012617.

[13] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Generation Computer Systems,* vol. 124, pp. 142-154, 2021, doi: https://doi.org/10.1016/j.future.2021.05.026.

[14] S. Mousavi, S. E. Mood, A. Souri, and M. M. Javidi, "Directed search: a new operator in NSGA-II for task scheduling in IoT based on cloud-fog computing," *IEEE Transactions on Cloud Computing,* vol. 11, no. 2, pp. 2144-2157, 2022, doi: https://doi.org/10.1109/TCC.2022.3188926.

[15] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access,* vol. 11, pp. 20635-20646, 2023, doi: https://doi.org/10.1109/ACCESS.2023.3241240.

[16] C. Yin, Q. Fang, H. Li, Y. Peng, X. Xu, and D. Tang, "An optimized resource scheduling algorithm based on GA and ACO algorithm in fog computing," *The Journal of Supercomputing,* vol. 80, no. 3, pp. 4248-4285, 2024, doi: https://doi.org/10.1007/s11227-023-05571-y.

[17] M. Qasim and M. Sajid, "An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm," *International Journal of Information Technology,* vol. 17, no. 1, pp. 179-188, 2025, doi: https://doi.org/10.1007/s41870-024-01758-5.

[18] M. Qi, X. Wu, K. Li, and F. Yang, "IPAQ: a multi-objective global optimal and time-aware task scheduling algorithm for fog computing environments," *The Journal of Supercomputing,* vol. 81, no. 2, p. 377, 2025, doi: https://doi.org/10.1007/s11227-024-06853-9.

[19] B. Pourghebleh, N. Hekmati, Z. Davoudnia, and M. Sadeghi, "A roadmap towards energy-efficient data fusion methods in the Internet of Things," *Concurrency and Computation: Practice and Experience,* vol. 34, no. 15, p. e6959, 2022, doi: https://doi.org/10.1002/cpe.6959.

[20] A. Ksentini, M. Jebalia, and S. Tabbane, "IoT/Cloud-enabled smart services: a review on QoS requirements in fog environment and a proposed approach based on priority classification technique," *International Journal of Communication Systems,* vol. 34, no. 2, p. e4269, 2021, doi: https://doi.org/10.1002/dac.4269.

[21] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Computer methods in applied mechanics and engineering,* vol. 376, p. 113609, 2021, doi: https://doi.org/10.1016/j.cma.2020.113609.

[22] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software,* vol. 95, pp. 51-67, 2016, doi: https://doi.org/10.1016/j.advengsoft.2016.01.008.

[23] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future generation computer systems,* vol. 97, pp. 849-872, 2019, doi: https://doi.org/10.1016/j.future.2019.02.028.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4: ieee, pp. 1942-1948, doi: https://doi.org/10.1109/ICNN.1995.488968.

[25] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications,* vol. 177, p. 114864, 2021, doi: https://doi.org/10.1016/j.eswa.2021.114864