Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

ARTICLE

CARE-FSP: A Cost-Aware and Resource-Efficient Hybrid RSO–SFO Framework for Fog-Based IoT Service Placement

Hanene Merouani, Sonia-Sabrina Bendib, Hamouma Moumen* and Lahcene Guezouli

LAMIE Laboratory, Computer Science Department, University of Batna 2, Batna, Algeria
*Corresponding Author: Hamouma Moumen. Email: hamouma.moumen@univ-batna2.dz

Received: 18 February 2026; Accepted: 10 April 2026

ABSTRACT: Efficient service placement is a critical challenge in large-scale Internet of Things (IoT) environments, where fog computing must balance deployment cost and resource utilization under heterogeneous and dynamic conditions. To address this challenge, this paper proposes a hybrid metaheuristic approach that combines Rat Swarm Optimization (RSO) and Sunflower Optimization (SFO), leveraging the strong global exploration capability of RSO and the efficient local exploitation behavior of SFO. The proposed RSO–SFO framework integrates both strategies within a unified fitness function designed to minimize deployment cost while ensuring efficient allocation of fog resources. Extensive simulation results demonstrate that the proposed hybrid algorithm consistently outperforms state-of-the-art optimization techniques, including Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), and the standalone RSO and SFO methods. Specifically, the RSO–SFO approach achieves a fitness improvement of 45.38%, reduces deployment costs by 43.71%, and maintains a high average resource utilization of 78.83%. These results confirm the effectiveness and robustness of the proposed hybrid strategy for optimal service placement in fog-based IoT environments.

KEYWORDS: IoT; Fog Computing; Hybrid RSO–SFO Algorithm; Optimization Techniques; Service Placement.

1 Introduction

The rapid proliferation of Internet of Things (IoT) devices, ranging from low-power environmental sensors and wearable health monitors to industrial actuators and high-definition video cameras, has transformed application domains such as healthcare, smart cities, and autonomous transportation [1]. These devices generate highly heterogeneous data streams, including periodic sensor readings, continuous multimedia feeds, and event-driven alerts. As of 2025, more than 75.44 billion IoT endpoints are expected to be active worldwide, collectively producing approximately 79 zettabytes of data annually [2]. This unprecedented growth in data volume places significant pressure on network infrastructure and centralized processing platforms, motivating a shift toward distributed, edge-centric computing architectures.

Although centralized cloud platforms provide virtually unlimited computation and storage resources, routing all IoT data to remote data centers introduces network congestion and significant communication overhead. Moreover, the continuous transmission of large data volumes leads to increased bandwidth consumption, higher operational costs, and reduced overall system efficiency, limiting the effectiveness of cloud-only solutions for large-scale and resource-constrained IoT applications such as patient monitoring, industrial control, and autonomous driving [1].

Fog computing addresses these limitations by extending cloud capabilities to the network edge, closer to data sources. By enabling local processing, aggregation, and decision-making on fog nodes, this paradigm

improves local resource efficiency and alleviates backbone congestion. Fog-based architectures have therefore become essential for supporting large-scale, resource-constrained, and reliability-critical IoT applications, including smart traffic control, healthcare monitoring, and industrial automation [3].

Despite its advantages, effective service placement in fog computing environments remains a challenging optimization problem. Fog nodes exhibit substantial heterogeneity in computational capacity, memory, storage, and network bandwidth, whereas IoT workloads fluctuate over time. A robust placement strategy must minimize deployment costs, maximize resource utilization, and strictly respect node capacity constraints. Traditional placement techniques often struggle to address this complexity, leading to inefficient resource usage and degraded system performance [4,5]. Several metaheuristic and hybrid optimization approaches have been explored for fog service placement, demonstrating promising results in improving scalability and solution quality. Building on this research direction, the proposed RSO–SFO framework integrates Rat Swarm Optimization and Sunflower Optimization within a coordinated optimization process to leverage their complementary exploration and exploitation capabilities. It is important to note that, although IoT workloads and resource availability evolve over time, service placement decisions are typically made at discrete decision epochs. In this work, a snapshot-based optimization perspective is adopted, in which the placement problem is solved for a given system state that reflects current workloads and fog node capacities. This formulation remains directly applicable to dynamic environments, as the optimization process can be periodically or event-triggered to re-execute and adapt service placement in response to workload variations and changing system conditions.

To address the fog service placement problem, this study proposes a novel hybrid metaheuristic that integrates Rat Swarm Optimization (RSO) and Sunflower Optimization (SFO). The proposed RSO–SFO framework combines the strong global exploration capability of RSO with the focused local exploitation efficiency of SFO to minimize deployment cost and improve resource utilization. The algorithm targets three core objectives: (i) reducing computational, memory, data transfer, and storage costs by assigning services to suitable fog nodes; (ii) maximizing the utilization of available CPU, memory, and storage resources; and (iii) enforcing strict capacity constraints to ensure system stability.

The proposed RSO–SFO framework introduces a coordinated hybrid optimization mechanism in which exploration and exploitation are performed within the same iterative optimization loop. RSO first performs global exploration of the search space, after which SFO refines promising candidate solutions through targeted local improvements under shared constraints. This design establishes an effective exploration–exploitation balance tailored to fog service placement.

The main contributions of this work can be summarized as follows:

- **Structured RSO–SFO Hybrid Framework:** A novel hybrid optimization framework that tightly integrates Rat Swarm Optimization and Sunflower Optimization to balance global exploration and local exploitation for fog service placement.
- **Bi-objective Fitness Formulation:** A unified fitness function that jointly minimizes deployment cost and maximizes resource utilization while respecting node capacity constraints.
- **Comprehensive Performance Evaluation:** Extensive simulation-based experiments demonstrating that RSO–SFO achieves faster convergence, lower deployment cost, and higher resource utilization compared to widely used baseline metaheuristic algorithms.

The remainder of this paper is organized as follows. Section 2 reviews related work on fog service placement and identifies the research gap addressed in this study. Section 3 presents the system model and resource assumptions. The problem formulation and fitness function are described in Section 4, while the proposed RSO–SFO methodology is detailed in Section 5. Experimental setup and results are reported in

Section 6. Section 7 discusses the findings and limitations, and Section 8 concludes the paper and outlines future research directions.

2 Related Work

Existing research on IoT service placement in fog computing can be systematically classified into four main categories: (i) single-objective metaheuristic approaches focusing on a specific optimization metric such as cost, energy consumption, or network-related performance; (ii) multi-objective evolutionary and swarm-based optimization methods that jointly consider multiple conflicting objectives; (iii) hybrid metaheuristic frameworks that combine two or more optimization strategies to balance exploration and exploitation; and (iv) autonomic and QoS-aware frameworks that integrate optimization algorithms with adaptive control or middleware-based decision mechanisms. This categorization provides a structured perspective for analyzing prior work and facilitates a critical comparison of their respective strengths and limitations [6].

Fog computing brings cloud capabilities to the network edge, addressing communication overhead and bandwidth bottlenecks that centralized clouds struggle with in IoT scenarios [7]. Over the past few years, researchers have investigated multiple aspects of this paradigm, including architectural design and resource management, service placement strategies, and security mechanisms. In the following, we review representative works in each category, with particular focus on optimization-based approaches to fog service placement [8].

Apat et al. [9] propose a multi-metaheuristic framework for IoT service placement in fog environments by combining Genetic Algorithms, Simulated Annealing, and Particle Swarm Optimization, including hybrid variants such as GASA and GAPSO. Through extensive simulations, they show that GASA consistently achieves lower makespan, energy consumption, and deployment cost compared to other tested methods. However, their study does not explicitly address resource utilization efficiency, nor does it assess scalability as the number of services and fog nodes increases.

Mehran et al. [10] introduce MAPO, a Pareto-based genetic algorithm for placing IoT applications on fog nodes. Evaluated on both synthetic benchmarks and a real medical monitoring workload, MAPO reduces deployment costs by up to 27%, lowers energy consumption by at least 23%, and accelerates task completion by up to 7.3 compared to earlier methods. Despite these strong multi-objective trade-offs, the approach leaves open questions regarding unused resource capacity and performance robustness in large-scale, geographically distributed fog deployments.

Salimian et al. [11] develop an evolutionary multi-objective PSO framework built on a Fog–Cloud control middleware to deploy IoT services across ten fog nodes handling up to 100 concurrent requests. Their approach jointly optimizes service completion ratio, average waiting time, and per-service cost, achieving notable improvements over round-robin and GA baselines. Nevertheless, the evaluation remains limited to synthetic scenarios, provides limited analysis of objective trade-offs, and lacks validation in real-world or large-scale settings.

Natesha et al. [12] propose a two-tier Docker-based fog provisioning framework that formulates service placement as a multi-objective optimization problem and solves it using an elitist genetic algorithm. Their method reduces deployment cost, execution time, and energy consumption while satisfying quality of service constraints on a real fog testbed. However, the framework does not explore the impact of service interdependencies on placement decisions, nor does it evaluate scalability and resilience under highly dynamic and heterogeneous fog environments.

Niu et al. [13] model the Fog Service Placement Problem as a constrained three-objective optimization problem that maximizes resource utilization while minimizing network latency and placement cost, and

solve it using NSGA-II. Their simulations demonstrate well-distributed Pareto fronts across sparse and dense topologies. However, the approach relies on static network snapshots and lacks mechanisms to adapt to sudden workload variations or to large-scale, geographically distributed deployments.

Liu et al. [14] address IoT service placement using a cloud–fog hybrid model coordinated via Cuckoo Search (CSA). Their method reduces end-to-end delay, response time, and operational cost compared to GWO and PSO, while achieving high resource utilization. Despite these gains, CSA incurs higher runtime overhead, and the authors acknowledge the need for reducing computational complexity to improve scalability.

Gilbert et al. [15] focus on fog node placement in low-voltage distribution networks and introduce FPNSGA, a hybrid of NSGA-II, SMPSO, and the Future Search Algorithm. Their approach improves the HyperVolume metric compared to competing methods. However, scalability to larger networks and validation in diverse real-world scenarios remain unexplored.

Ramzanpoor et al. [16] propose MOCSA, a multi-objective Cuckoo Search algorithm that integrates power conservation, latency minimization, and fault tolerance. Although MOCSA outperforms several evolutionary baselines in simulated environments, its scalability, adaptability to dynamic workloads, and performance on resource-constrained fog hardware are not evaluated.

Ghobaei-Arani and Shahidinejad [17] introduce a three-layer IoT–fog–cloud architecture using the Whale Optimization Algorithm to balance service delay, acceptance rate, resource utilization, and energy consumption. While the method improves energy efficiency and resource usage, it does not investigate scalability, hybridization with complementary heuristics, or robustness under dynamic real-world conditions.

Zhao et al. [18] propose FSP-ODMA, a QoS-driven service placement scheme based on the Open-source Development Model Algorithm. Their solution improves response time, energy efficiency, and acceptance rates in simulation. However, the study does not analyze computational overhead, large-scale scalability, or integration with adaptive optimization techniques.

Wu et al. [19] present an autonomic WOA-based framework that adapts to varying QoS requirements and fog-node capabilities. Their approach achieves higher resource utilization and lower latency than GA, PSO, and SA baselines, demonstrating the benefits of workload-aware placement strategies in practical fog deployments.

Salimian et al. [20] evaluate a GWO-based placement strategy on a ten-node fog testbed handling up to 200 concurrent IoT tasks. By incorporating execution cost and node capacity into the objective function, their scheduler reduces execution time and operational cost. Nevertheless, the scalability and overhead of the approach in large-scale deployments remain open challenges.

Ayoubi et al. [21] introduce MADE, an autonomic loop based on SPEA-II that continuously monitors system state to adapt service placement decisions. Their framework improves resource utilization and reduces latency and cost, highlighting the benefits of autonomic control in dynamic fog environments.

Ben Rjeb et al. [22] develop a hybrid recommendation system combining collaborative and content-based filtering to guide IoT service placement. Their approach improves waiting time and resource utilization without significant overhead, but it does not explicitly address optimization, scalability, or exploration-exploitation balance.

Ali et al. [23] propose a genetic algorithm that jointly optimizes fog colony layout and service placement. By dynamically reshaping node clusters, their method improves response time and resource utilization across diverse network environments.

Toghyani et al. [24] present a SLA-aware framework spanning fog and cloud layers that dynamically reallocates services to meet latency, availability, and cost constraints. While effective in reducing SLA violations, the framework relies on rule-based decisions rather than tightly integrated optimization mechanisms.

Bolettieri et al. [25] introduce a MEC-compliant placement framework that considers service data dependencies and QoS requirements. Their heuristic dynamically allocates compute and cache resources, improving response time and resource utilization, but does not address unified optimization under heterogeneous fog constraints.

Despite the diversity of existing solutions, several common limitations emerge across the literature. Many approaches prioritize either global optimization or local refinement, which may limit the ability to balance both effectively within a unified optimization framework. Hybrid methods often rely on loosely coupled or sequential combinations of algorithms, resulting in limited coordination between the exploration and exploitation phases. Moreover, adaptive and QoS-aware frameworks often introduce additional architectural or computational complexity, which may hinder scalability in large-scale, resource-constrained fog environments.

Table 1 summarizes the key metrics optimized by representative IoT service placement algorithms in fog computing environments.

Table 1: Summary of Optimized Metrics for IoT Service Placement Algorithms in Fog Computing

Ref.	Alg.	Time	Cost	Latency	Energy	Thrp.	Res. Util.	Sec.
[19]	IPGA	Service time	✓	✓	x	x	✓	x
[9]	GA-PSO, GA-SA	Makespan	✓	x	✓	x	x	x
[18]	ODMA	Response time	✓	✓	✓	x	✓	x
[20]	GWO	Response time	✓	x	x	✓	x	x
[17]	WOA	x	x	x	✓	✓	x	x
[10]	NSGA-II	Completion time	✓	x	✓	x	x	x
[11]	PSO	Response time	✓	x	x	✓	✓	x
[21]	SPEA-II	x	✓	✓	x	x	✓	x
[12]	EGA	Service time	✓	x	✓	x	x	x
[13]	NSGA-II	x	✓	✓	x	x	✓	x
[14]	CSA	Delay, response time	✓	x	✓	x	✓	✓
[15]	FPNSGA	Network delay	✓	x	x	x	x	x
[16]	MOCSA	x	x	✓	✓	x	x	x
[22]	HRS	Service time	x	✓	x	x	✓	x
[23]	GA-based Layout	x	x	✓	x	x	✓	x
[24]	QoS-SLA Framework	x	x	✓	x	x	✓	x
[25]	MEC-Aware Placement	Execution period	x	✓	x	x	✓	x
Proposed Work	RSO–SFO	x	✓	x	x	x	✓	x

In summary, existing fog service placement solutions either emphasize broad global optimization or fine-grained local refinement, but rarely achieve a balanced integration of both strategies within a unified optimization framework. To address this gap, the present work proposes a coordinated hybrid optimization approach that combines complementary swarm-based behaviors within the same iterative optimization loop under a shared fitness formulation, tailored to the heterogeneity, scalability, and resource constraints of fog computing environments.

3 System Model

The architecture of the considered IoT–Fog environment is composed of three main components: IoT devices, Fog nodes, and an orchestrator. These elements form a hierarchical structure designed to support efficient data processing and service deployment within distributed IoT environments.

IoT devices represent the lowest layer of the architecture. They consist of interconnected physical objects equipped with sensors and actuators that collect environmental data and transmit it to nearby Fog nodes for processing [2]. These devices generate large volumes of data and typically have limited computational and storage capabilities.

Fog nodes constitute the intermediate layer between IoT devices and cloud infrastructure. They provide computational, storage, and networking resources close to the data sources, enabling local data processing and reducing latency. Fog nodes perform tasks such as data aggregation, temporary storage, and distributed service execution while maintaining communication with both IoT devices and higher-level management components [12].

At the top layer, the orchestrator acts as the central management entity responsible for coordinating system operations. It oversees resource allocation, service placement, load balancing, and system monitoring across the Fog infrastructure. By managing the interaction between Fog nodes and IoT devices, the orchestrator ensures efficient utilization of resources and supports scalable service deployment in the IoT–Fog environment [7].

This hierarchical architecture enables efficient data collection, processing, and service management, allowing IoT applications to meet their latency and resource constraints more effectively than cloud-only solutions.

Figure 1 depicts the IoT–Fog architecture, illustrating how the orchestrator, Fog nodes, and IoT devices work together. This architecture depicts the system's hierarchical structure and data flow, with each component's role in efficiently managing and processing the data.

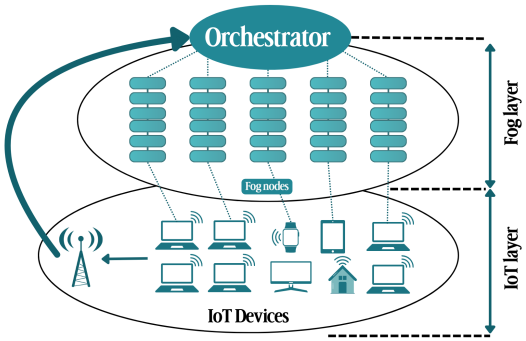

Figure 1: IoT–Fog Architecture with RSO–SFO-Based Orchestration and Closed-Loop Control.

4 Problem Formulation

The service placement problem is formulated considering a static system snapshot corresponding to a specific decision instant. At this instant, the set of active services, their resource requirements, and the available capacities of fog nodes are assumed to be known. While IoT environments are inherently dynamic, this snapshot-based formulation is commonly adopted in the fog and edge computing literature because it enables tractable optimization and can be repeatedly applied whenever significant workload or resource variations occur. Therefore, the proposed formulation provides a fundamental building block for adaptive service placement through iterative re-optimization over time.

The fluctuating usage and availability of computational resources within these nodes influence service deployment on Fog nodes. Each service consumes multiple resources, such as CPU, RAM, and storage, when running on a Fog node. As a result, the resource demands of these services must not surpass the available computational resources of the Fog nodes, which are inherently limited. The following problem formulation delineates the constraints and objectives necessary for optimal service placement within a Fog Computing environment, guaranteeing that each service request adheres to resource limitations.

Let $V_N = \{v_1, v_2, \dots, v_N\}$ is a set of Fog nodes that can be used and $S_M = \{s_1, s_2, \dots, s_M\}$ is a set of services that can be deployed on these Fog nodes. The goal is to determine the optimal placement of these services so that the total cost is minimized and resources are utilized as effectively as possible.

For notational simplicity, the sets V_N and S_M are hereafter denoted by N and M, respectively.

4.1 Objective Functions

It is important to distinguish between the optimization objectives that define the fog service placement problem and the fitness function that guides the metaheuristic algorithm's search process. The objectives represent the problem-level goals to be optimized, while the fitness function serves as an algorithmic evaluation mechanism that combines these objectives into a scalar value for ranking and selecting solutions.

The multi-objective optimization for service placement is defined by Eqs. (1) and (2).

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{M} C_{\text{total}}(v_i, s_j)\, x_{i,j} \tag{1}$$

$$\text{Maximize} \sum_{i=1}^{N} \sum_{j=1}^{M} U_{\text{total}}(v_i, s_j)\, x_{i,j} \tag{2}$$

where

$n_i \in V_N,\ i \in [1, N]$ Fog nodes

$s_j \in S_M,\ j \in [1, M]$ services

$$x_{i,j} = \begin{cases} 1, & \text{if } j \text{ is placed on Fog node } i \\ 0, & \text{otherwise} \end{cases}$$

4.2 Cost

The goal is to reduce the total cost of using services on Fog nodes. This includes costs for CPU and RAM usage, data transfer, and data storage. As shown in Eq. (3), this total cost C_{total} is calculated as the sum of the CPU cost C_{CPU}, the memory cost C_{RAM}, the data transfer cost C_{Data}, and the storage cost C_{Storage}.

$$C_{\text{total}} = C_{\text{CPU}} + C_{\text{RAM}} + C_{\text{Data}} + C_{\text{Storage}} \tag{3}$$

$$C_{\text{CPU}} = \sum_{n \in N} \sum_{s \in M} \text{CPU\_Usage}(n, s) \times \text{Cost\_CPU} \times x_{n,s} \tag{4}$$

$$C_{\text{RAM}} = \sum_{n \in N} \sum_{s \in M} \text{RAM\_Usage}(n, s) \times \text{Cost\_RAM} \times x_{n,s} \tag{5}$$

$$C_{\text{Data}} = \sum_{n \in N} \sum_{s \in M} \big(\text{Data\_Transferred}(n, s)\big) \times \text{Cost\_Data\_Transfer} \times x_{n,s} \tag{6}$$

$$C_{\text{Storage}} = \sum_{n \in N} \sum_{s \in M} \text{Storage\_Usage}(n, s) \times \text{Cost\_Storage} \times x_{n,s} \tag{7}$$

4.3 Resource Utilization

The goal is to make the most of the Fog nodes' resources so that they are used as efficiently as possible. This means maximizing the use of the CPU, RAM, and storage space. The equation (8) gives the objective function for getting the most out of resources.

The total utilization U_{total} is calculated as the sum of the CPU utilization U_{CPU}, the RAM utilization U_{RAM}, and the storage utilization U_{Storage}.

$$U_{\text{total}} = U_{\text{CPU}} + U_{\text{RAM}} + U_{\text{Storage}} \, where : \tag{8}$$

It should be noted that this aggregated formulation is intended to capture the overall level of resource consumption across fog nodes by combining CPU, RAM, and storage utilization. It does not explicitly model imbalance among individual resource dimensions; rather, it serves as a practical global utilization indicator within the optimization framework, while resource feasibility is ensured separately through the corresponding capacity constraints.

$$U_{\text{CPU}} = \frac{\sum_{n\in N}\sum_{s\in M} \text{CPU\_Usage}(n,s) \times x_{n,s}}{\sum_{n\in N} \text{CPU\_Capacity}(n)} \tag{9}$$

$$U_{\text{RAM}} = \frac{\sum_{n\in N}\sum_{s\in M} \text{RAM\_Usage}(n,s) \times x_{n,s}}{\sum_{n\in N} \text{RAM\_Capacity}(n)} \tag{10}$$

$$U_{\text{Storage}} = \frac{\sum_{n\in N}\sum_{s\in M} \text{Storage\_Usage}(n,s) \times x_{n,s}}{\sum_{n\in N} \text{Storage\_Capacity}(n)} \tag{11}$$

4.4 Problem Constraints

1. Each service needs to be deployed on exactly one Fog node.

$$\sum_{n\in N} x_{n,s} = 1 \quad \forall s \in M$$

2. The total resource usage on each Fog node must not exceed its capacity:

$$\sum_{s\in M} \text{CPU\_Usage}(n,s)\, x_{n,s} \leq \text{CPU\_Total\_Capacity}(n) \quad \forall n \in N, \tag{12}$$

$$\sum_{s\in M} \text{RAM\_Usage}(n,s)\, x_{n,s} \leq \text{RAM\_Total\_Capacity}(n) \quad \forall n \in N, \tag{13}$$

$$\sum_{s\in M} \text{Storage\_Usage}(n,s)\, x_{n,s} \leq \text{Storage\_Total\_Capacity}(n) \quad \forall n \in N. \tag{14}$$

4.5 Fitness Function

The fitness function does not introduce new optimization objectives; rather, it aggregates the predefined problem objectives into a scalar value to enable efficient comparison and selection of candidate solutions during the metaheuristic search process. By weighting cost and resource utilization through parameters α and β, the fitness function operationalizes the multi-objective problem in a form suitable for population-based optimization.

$$\text{Fitness} = \alpha \times \text{Cost} + \frac{1}{\beta \times \text{Resource Utilization}} \tag{15}$$

The goal is to minimize the fitness value, which corresponds to achieving an effective balance between low deployment cost and high resource utilization. It should be noted that the fitness function is used as a

relative ranking criterion during the optimization process under fixed experimental settings. Although the cost and utilization terms may operate on different numerical ranges, their contributions are balanced through the weighting coefficients α and β. Consequently, the optimization relies on the relative ordering of candidate solutions rather than the absolute magnitude of the fitness value.

Table 2 presents the mathematical notations and symbols used to formulate the problem described in this paper.

Table 2: Notations and symbols used in the problem formulation.

Symbol	Description
V_N	set of all fog nodes
S_M	set of all services
CPU_Usage(n, s)	CPU usage of service s on fog node n
RAM_Usage(n, s)	RAM usage of service s on fog node n
Data_Transferred(n, s)	Data transferred by service s on fog node n
Storage_Usage(n, s)	Storage usage of service s on fog node n
Cost_CPU	Cost per unit of CPU used
Cost_RAM	Cost per unit of RAM used
Cost_Data_Transfer	Cost per unit of data transferred
Cost_Storage	Cost per unit of storage used
CPU_Capacity(n)	CPU capacity of fog node n
RAM_Capacity(n)	RAM capacity of fog node n
Storage_Capacity(n)	Storage capacity of fog node n
$x_{n,s}$	Binary variable indicating if service s is placed on fog node n
C_{total}	Total service cost
C_{CPU}	CPU cost
C_{RAM}	RAM cost
C_{Data}	Data transfer cost
C_{Storage}	Storage cost
U_{total}	Total resource utilization
U_{CPU}	CPU utilization
U_{RAM}	RAM utilization
U_{Storage}	Storage utilization
α	Weight for cost in fitness function
β	Weight for resource utilization in fitness function

5 The Proposed Approach: Rat Swarm Optimization–Sunflower Optimization

The proposed methodology for service placement in Fog Computing environments combines two complementary metaheuristics, Rat Swarm Optimization (RSO) and Sunflower Optimization (SFO), into a hybrid framework denoted as RSO–SFO. RSO provides broad swarm-based exploration of the placement search space, facilitating the identification of promising regions. The SFO component performs targeted flower-inspired refinements to enhance local exploitation while preserving population diversity. By sequentially combining these two phases within the same iterative process, the proposed RSO–SFO framework aims to improve deployment cost and resource utilization in fog computing environments. The following sections describe the RSO and SFO components in more detail and explain how they are integrated into the proposed RSO–SFO algorithm (Algorithm 1).

5.1 Rat Swarm Optimization (RSO)

Rat Swarm Optimization (RSO) is a population-based metaheuristic proposed in [26], inspired by the cooperative hunting behavior of rat swarms. The algorithm models two main behavioral mechanisms,

namely *chasing* and *fighting*, which guide the search agents toward promising regions of the solution space. During the chasing phase, the intermediate position vector is computed as

$$P = A \cdot P_i(x) + C \cdot (P_r(x) - P_i(x)),$$

where $P_i(x)$ denotes the position of rat i at iteration x and $P_r(x)$ represents the best solution found so far. The parameters A and C control the exploration process; A decreases linearly over iterations to balance exploration and exploitation, while C is a random coefficient used to maintain population diversity. In the proposed hybrid framework, the RSO mechanism provides global exploration before applying the SFO-based local refinement stage.

5.2 Sunflower Optimization (SFO)

Sunflower Optimization (SFO) is a nature-inspired metaheuristic motivated by the heliotropic behavior of sunflowers, which orient their growth toward favorable directions in response to environmental stimuli [27]. SFO can be interpreted as a guided search principle that emphasizes directed local exploitation around promising solutions.

In the proposed hybrid RSO–SFO framework, the SFO phase is implemented as a directed local exploitation operator inspired by the heliotropic principle of sunflower optimization. A selected subset of high-quality candidate solutions, referred to as *sunflowers*, is refined toward the current global-best solution. The position update rule is defined as:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \lambda \left[\mathbf{g}(t) - \mathbf{x}(t)\right] \odot \mathbf{u},$$

where $\mathbf{x}(t) \in \mathbb{R}^D$ denotes the current solution position, $\mathbf{g}(t)$ is the global-best solution at iteration t, $\lambda \in (0, 1]$ controls the exploitation step size, while $\odot$ represents the Hadamard product, and $\mathbf{u} \sim U(0, 1)^D$ introduces stochastic perturbations to mitigate premature convergence. After the update, feasibility is ensured through boundary repair operators that enforce fog-node capacity constraints.

The effectiveness of SFO-inspired mechanisms has been demonstrated in various engineering optimization problems, particularly in structural damage identification [28], and extended to constrained multi-objective optimization frameworks through the development of MOSFO [29]. These studies confirm that heliotropic-inspired local refinement can significantly enhance solution quality when integrated with complementary global exploration strategies. In the proposed hybrid framework, the RSO phase governs the global exploration process, while the parameter λ regulates the SFO-based local refinement intensity.

5.3 Complementarity and Hybrid Framework

The proposed RSO–SFO approach combines two complementary metaheuristics within a coordinated hybrid optimization framework in which exploration and exploitation are performed within the same iterative loop. Within each iteration, the RSO phase first performs global exploration of the search space, after which the SFO phase refines a subset of promising solutions. Each candidate solution is jointly influenced by the swarm-based global search dynamics and heliotropic local refinement, guided by a shared fitness function and common feasibility constraints. This coordinated sequential interaction enables the algorithm to combine broad exploration with focused local refinement without relying on heuristic switching or multi-stage execution, which are commonly employed in conventional hybrid metaheuristics. To the best of our knowledge, the integration of rat swarm dynamics with sunflower-inspired refinement for fog service placement has not been previously reported. The proposed framework integrates these complementary behaviors as follows:

1. **RSO Phase:**
   Each candidate solution evolves according to rat swarm dynamics governed by chasing and fighting behaviors around the best solution found so far. This mechanism enables broad exploration of the search space while preserving population diversity through adaptive search movements, allowing the algorithm to identify promising placement configurations under complex resource constraints.
2. **SFO Phase:**
   A subset corresponding to the top $P_P\%$ of candidate solutions, referred to as *sunflowers*, is refined through a heliotropic movement toward the current global-best solution. These focused updates intensify local exploitation and rapidly improve high-potential solutions identified during the exploration phase.

By embedding these two phases within each iteration, RSO–SFO achieves a principled balance between exploration and exploitation. The wide-ranging dispersal induced by RSO mitigates premature convergence, while the targeted refinement of SFO accelerates local improvements without sacrificing population diversity.

In the proposed hybrid framework, the RSO phase governs the global exploration process, while the parameter λ regulates the intensity of the SFO-based local refinement step. RSO is selected for efficient swarm behavior and its ability to optimize computational cost and resource usage (e.g., CPU cycles and memory footprint), while SFO contributes rapid local improvements through directed search. Together, these mechanisms form a hybrid exploration–refinement framework that balances global search and local improvement during the optimization process. This structured hybridization supports improved optimization performance in Fog service placement scenarios and distinguishes it from standalone and empirically combined hybrid methods. Figure 2 illustrates the step-by-step workflow of the proposed RSO–SFO hybrid algorithm for optimizing service placement in Fog Computing environments.

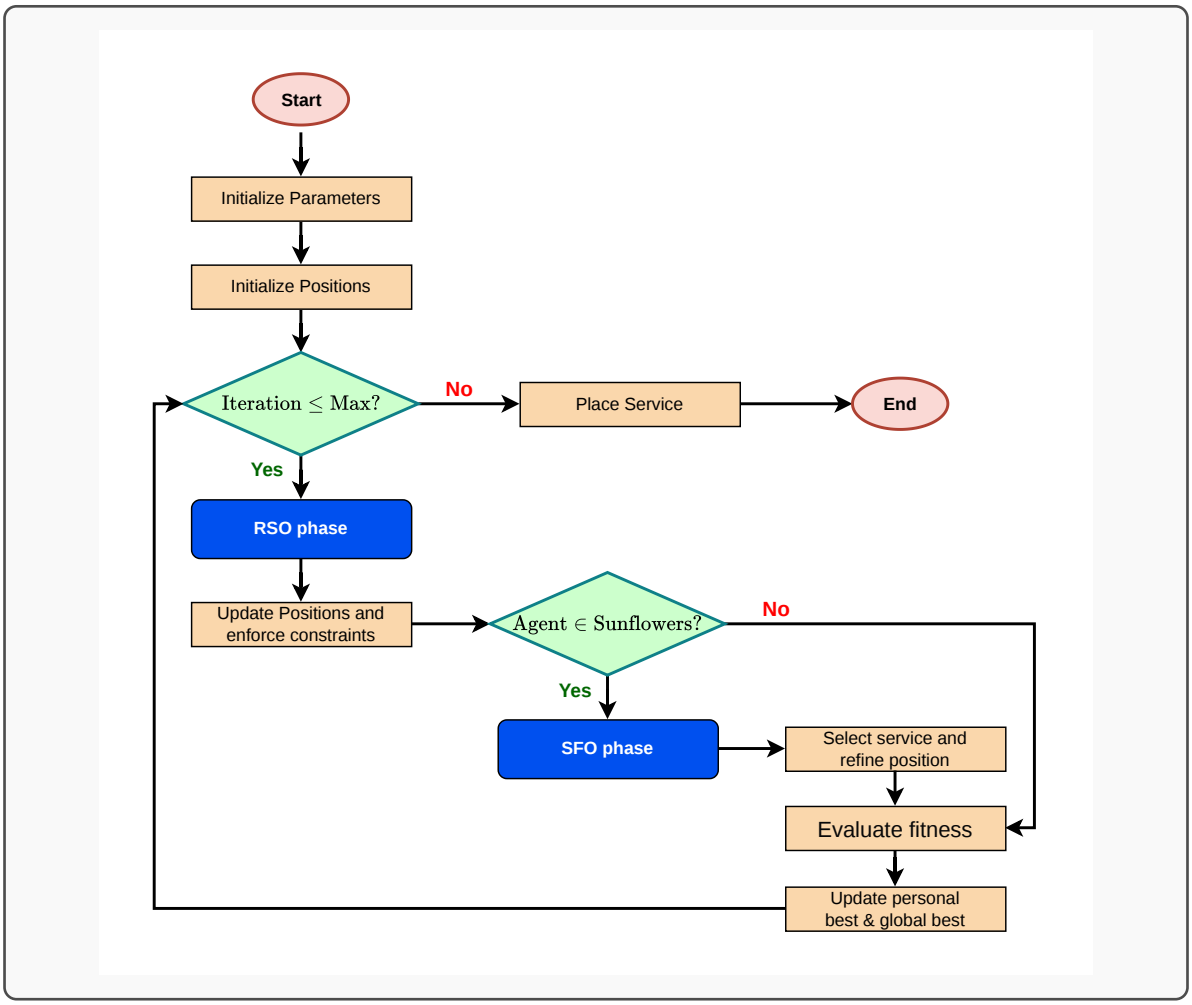

Figure 2: Unified RSO–SFO Hybrid Optimization Framework.

5.4 Proposed Algorithm

We propose a hybrid RSO–SFO algorithm that iteratively alternates between swarm-based exploration and sunflower-inspired exploitation. At each iteration, candidate solutions are first diversified using RSO dynamics and then selectively refined through SFO updates guided by the global best solution. Algorithm 1 presents the RSO–SFO algorithm.

6 Experiments

This section presents the experimental setup and evaluation methodology used to assess the effectiveness of the proposed RSO–SFO approach for fog service placement. The experiments are conducted in a controlled simulation environment designed to analyze convergence behavior, solution quality, and comparative performance against the state-of-the-art metaheuristics under well-defined reproducible conditions.

6.1 Experiment Setup

All experiments were executed on a computing platform whose hardware and software specifications are summarized in Table 3.

It should be noted that the experimental evaluation is intentionally conducted in a small-scale fog environment comprising five fog nodes and 100 services. This setup is designed to provide a

Algorithm 1 Hybrid Service Placement using RSO–SFO

Require: Fog node set $\mathcal{N}$ with capacity constraints, service set $\mathcal{M}$, population size P, maximum iterations T, local refinement coefficient λ, mutation rate μ

Ensure: Best service placement solution $\mathbf{x}^*$, corresponding fitness $f(\mathbf{x}^*)$, cost $C(\mathbf{x}^*)$, resource utilization $U(\mathbf{x}^*)$

Initialize a population $\mathcal{P}_0 = \{\mathbf{x}_i^0\}_{i=1}^P$ with random service allocations
Evaluate $f(\mathbf{x}_i^0)$, $C(\mathbf{x}_i^0)$, and $U(\mathbf{x}_i^0)$ for all $\mathbf{x}_i^0$
Identify the global best solution $\mathbf{x}^* \in \mathcal{P}_0$
$t \leftarrow 0$
while $t < T$ **do**
    for each agent $\mathbf{x}_i^t \in \mathcal{P}_t$ **do**
        Generate candidate solution $\mathbf{x}_i'$ via RSO-based exploration
        Refine $\mathbf{x}_i'$ using SFO-inspired exploitation toward $\mathbf{x}^*$ with coefficient λ
        Discretize $\mathbf{x}_i'$ to obtain binary service placement
        for each service $s \in \mathcal{M}$ **do**
            Select fog node $n = \arg\max_k(x'_{k,s})$
            Set $x_{n,s} = 1$ and $x_{k,s} = 0$ for $k \neq n$
        end for
        Apply mutation operator with probability μ
        Enforce fog node capacity constraints
        Evaluate $f(\mathbf{x}_i')$, $C(\mathbf{x}_i')$, and $U(\mathbf{x}_i')$
        if $f(\mathbf{x}_i') < f(\mathbf{x}_i^t)$ **then**
            $\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i'$
        else
            $\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i^t$
        end if
    end for
    Update global best solution $\mathbf{x}^*$ from $\mathcal{P}_{t+1}$
    $t \leftarrow t + 1$
end while
return $\mathbf{x}^*$, $f(\mathbf{x}^*)$, $C(\mathbf{x}^*)$, $U(\mathbf{x}^*)$

controlled and reproducible testbed for analyzing the convergence behavior, optimization effectiveness, and comparative performance of the proposed RSO–SFO algorithm. Evaluating the metaheuristic behavior on a reduced-scale environment is a common practice in fog computing research, as it allows isolating algorithmic characteristics without introducing confounding factors related to large-scale system heterogeneity.

6.1.1 Fog Nodes Configuration

The simulation configuration consists of five fog nodes, each with varied CPU, RAM, and storage capacity. These nodes are critical for meeting the computational and storage requirements of the deployed services. Table 4 provides a detailed breakdown of each Fog node's resource capacities, including CPU, RAM, and storage.

Table 3: Simulation Environment Specifications

Component	Details
CPU	Intel Core i7-8550U, 1.80 GHz
RAM	16 GB
Storage	1 TB SSD
Operating System	Windows 10 Professional 64-bit
Computational Environment	Jupyter Notebook (Python 3.8)

Table 4: Resource Capacities for Fog Nodes

Resource	Capacity Range	Unit
CPU Capacity	2000–5000	MIPS
RAM Capacity	4096–16,384	MB
Storage Capacity	1000–5000	MB

6.1.2 IoT Devices and Services

A total of 100 services were deployed, representing typical IoT tasks with specific requirements for processing and data transmission. Table 5 summarizes the resource usage values for these services deployed on Fog nodes, detailing the range of CPU usage, RAM usage, data transferred, and storage usage.

Table 5: Resource Usage Values for Services on Fog Nodes

Resource	Description
CPU Usage	100–1000 MIPS (CPU demand per service)
RAM Usage	256–2048 MB (RAM usage in megabytes)
Data Transferred	50–200 MB (Data transferred in megabytes)
Storage Usage	100–1000 MB (Storage usage in megabytes)

6.1.3 Parameter Initialization

The parameters for the hybrid RSO–SFO algorithm were initialized as shown in Table 6.

The same population size and number of iterations were used for all compared algorithms (PSO, GWO, RSO, and SFO) to ensure a fair experimental comparison. The parameter values in Table 6 were selected based on recommendations from the literature and calibrated through preliminary experiments. The population size was set to 30 search agents and the maximum number of iterations to 100 in order to balance exploration capability and computational cost. Additionally, 50% of the population is assigned to the SFO phase ($P_P = 0.5$) to enhance local refinement after the exploration stage. Finally, the fitness function uses weights $\alpha = 0.25$ for deployment cost and $\beta = 0.75$ for resource utilization, reflecting the priority of maximizing Fog node efficiency while controlling operational expenses. At each iteration, the RSO phase is executed first for global exploration, followed by the SFO phase to refine promising solutions locally.

6.2 Experimental Results

In this section, we present a thorough evaluation of the RSO–SFO hybrid algorithm under a range of operating conditions. We simulated a Fog environment with five edge nodes handling up to 100 service requests at varying rates, and measured performance in terms of normalized fitness, resource utilization, and deployment cost. The outcomes from the experiments conducted with the RSO–SFO hybrid algorithm are illustrated within the following figures. Figures 3, 4, 8, and 10 visualize these results, confirming the

Table 6: Simulation Parameters for the RSO–SFO Algorithm

SPV. No.	Simulation Parameters	Value
1	Number of Fog Nodes	5
2	Number of Services to Deploy	100
3	Population Size (Search Agents)	30
4	Number of Iterations	100
5	Proportion of Sunflower Particles (P_P)	0.5
6	Number of Sunflower Particles	15
7	Sunflower Orientation Coefficient (λ)	0.7
8	RSO Control Parameter (R)	5
9	RSO Random Coefficient ($C = 2\,rand()$)	[0,2]
10	Mutation Rate	0.3
11	Weight for Cost (α)	0.25
12	Weight for Resource Utilization (β)	0.75

hybrid's robustness and consistent performance as the number of services increases for efficient service placement in Fog computing scenarios.

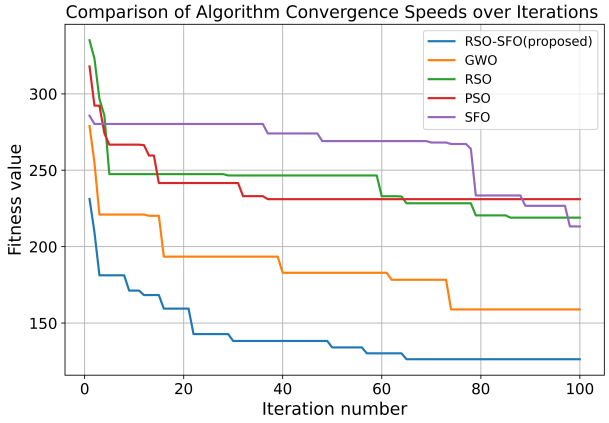

Figure 3: Comparison of Algorithm Convergence Speeds

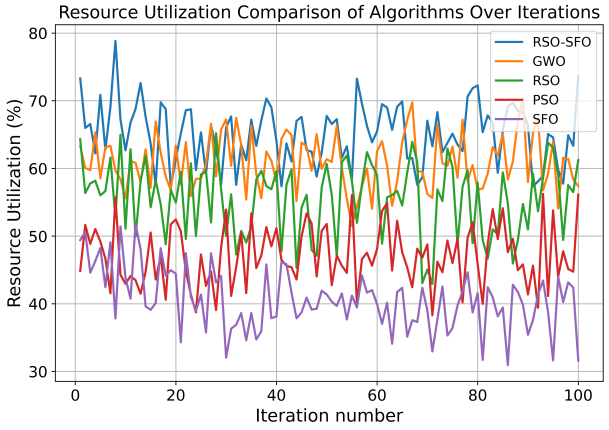

Figure 4: Resource Utilization Comparison of Algorithms Over Iterations

Figure 3 presents the convergence trajectories of five meta-heuristics, RSO–SFO, Gray Wolf Optimization, Rat Swarm Optimization, Particle Swarm Optimization, and Sunflower Optimization, over 100 iterations. The RSO–SFO hybrid clearly outpaces the rest, exhibiting a sharp initial drop followed by a smooth, uninterrupted decline. Gray Wolf Optimization makes rapid initial advances before slowing down in the middle. Rat Swarm and Particle Swarm both move at a consistent pace, with brief plateaus that reflect local search challenges. Sunflower Optimization remains relatively flat until late iterations. These profiles show the value of combining broad exploration with focused local refinement, since the RSO–SFO strategy achieves both immediate and sustained improvements above standalone techniques.

As shown in Figure 4, the RSO–SFO hybrid had the highest resource utilization when compared to GWO, RSO, PSO, and SFO. This suggests that RSO–SFO combines excellent resource allocation with rapid convergence. RSO–SFO exhibits balanced exploration–exploitation dynamics, maintaining utilization between 78.83 % and 57.53 % during 100 iterations. Grey Wolf Optimization has a moderate range (67.34 %–51.48 %), while Rat Swarm Optimization remains consistent at 65.19 %–42.94 %. Sunflower Optimization is the least effective (51.82%–30.92%), while Particle Swarm Optimization spans from 56.22% to 38.29%. These profiles demonstrate that the RSO–SFO framework's employment of complementary heuristics results in significantly higher resource efficiency when compared to solo techniques.

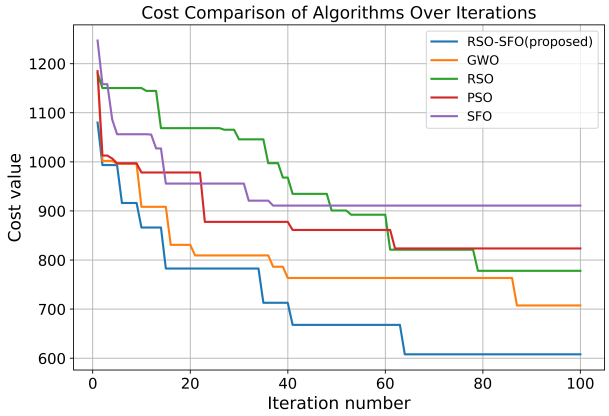

Figure 5: Cost Comparison of Algorithms Over Iterations

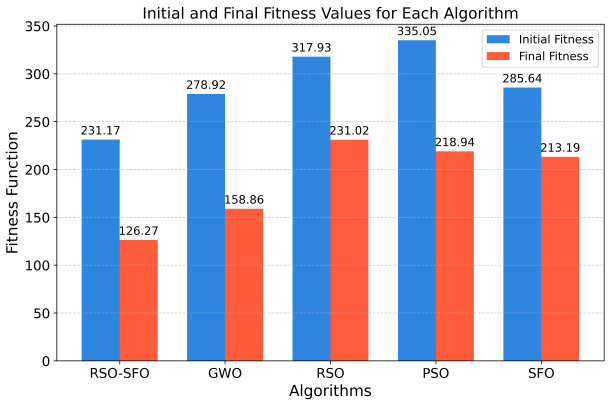

Figure 6: Initial and Final Fitness Values for Algorithms

Figure 5 shows the cost evolution of the five algorithms over 100 iterations. The proposed RSO–SFO hybrid achieves the largest reduction, decreasing the cost by 43.71%. Gray Wolf Optimization (GWO) follows with a reduction of about 40.03%. Particle Swarm Optimization (PSO) reduces the cost by 34.12%, while Rat Swarm Optimization (RSO) and Sunflower Optimization (SFO) achieve reductions of 30.46% and 26.96%, respectively. These results highlight that the hybrid RSO-SFO strategy provides the most effective cost minimization due to the complementary balance between exploration and exploitation introduced by the hybridization.

Figure 6 shows the initial and final fitness values for the five optimizers after 100 iterations. The proposed RSO–SFO hybrid achieves the lowest final fitness value among all compared methods, decreasing from 231.17 to 126.27. Grey Wolf Optimization follows with a final value of 158.86, while Sunflower Optimization, Rat Swarm Optimization, and Particle Swarm Optimization reach final fitness values of 213.19, 218.94, and 231.02, respectively. These results confirm that RSO–SFO produces the best final solution quality among the evaluated algorithms.

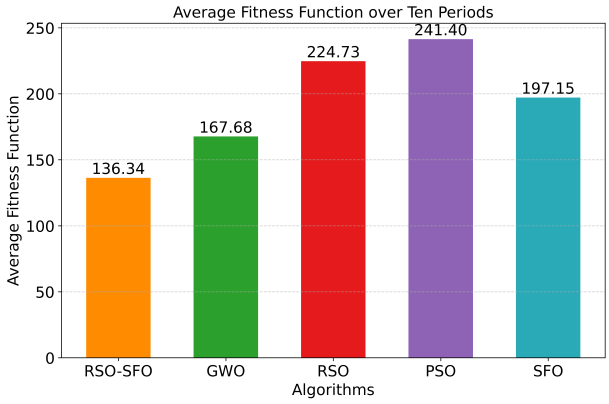

Figure 7: Average Fitness over Ten Periods

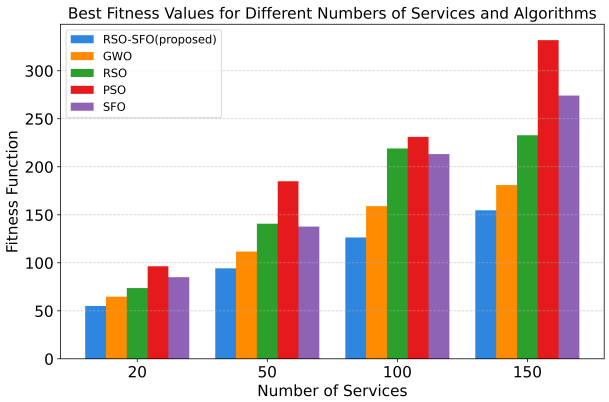

Figure 8: Best Fitness Values for Different Numbers of Services and Algorithms

In Figure 7, you can see the average fitness values for five algorithms: RSO–SFO, GWO, RSO, PSO, and SFO. These were calculated over ten runs. The RSO–SFO hybrid has the lowest average fitness of 136.34, indicating that it is more effective at guiding the search toward high-quality solutions. GWO comes next with an average fitness of 167.68. This indicates that it performs well, but lacks the same level of consistent improvement as RSO–SFO. RSO has an average of 224.73, which is in the middle, while PSO has the highest average fitness of 241.40, which means that convergence is slower and exploration is less effective. SFO is

in the middle of these two extremes, with an average fitness of 197.15. These results show that RSO–SFO works well to maintain the lowest fitness across many trials by combining wide-ranging exploration with focused local search.

The best fitness values for different numbers of services, as illustrated in Figure 8, show the comparative performance of the RSO–SFO hybrid, Gray Wolf Optimization (GWO), Rat Swarm Optimization (RSO), Particle Swarm Optimization (PSO), and Sunflower Optimization (SFO) algorithms. Over all tested configurations (20, 50, 100, and 150 services), RSO–SFO consistently achieves the lowest objective values, highlighting its superior exploration–exploitation balance. GWO and RSO deliver intermediate improvements, whereas PSO and SFO record higher final fitness, indicating slower convergence. These findings confirm that the dual phase rat swarm and sunflower strategy of RSO–SFO provides a clear advantage over standalone heuristics.

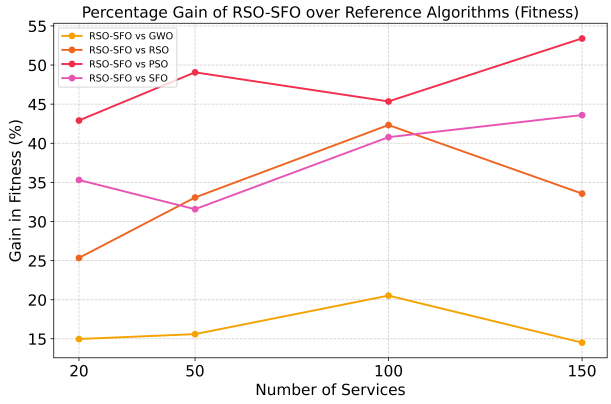

Figure 9: Fitness gain (%) of RSO–SFO over reference algorithms for 20–150 services.

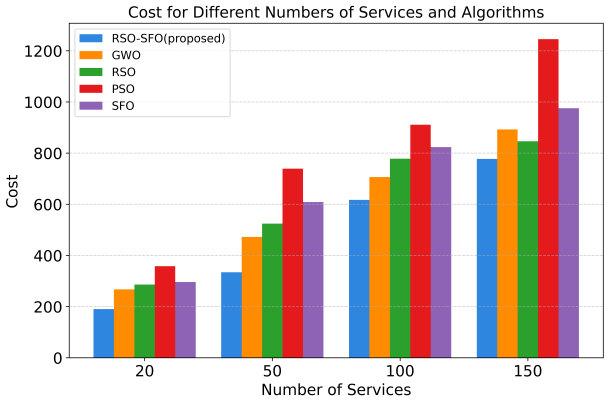

Figure 10: Cost for Different Numbers of Services and Algorithms

For each service count, the percentage reduction in final fitness achieved by RSO–SFO relative to a reference algorithm X is computed as

$$\text{Gain}_{\text{RSO–SFO vs. } X} = \frac{F_X - F_{\text{RSO–SFO}}}{F_X} \times 100,$$

where F_X and $F_{\text{RSO–SFO}}$ denote the final fitness values of algorithm X and the hybrid method, respectively. Figure 9 plots these gains for GWO, RSO, PSO, and SFO at service sizes of 20, 50, 100, and 150. RSO–SFO exhibits a steady advantage over GWO, with gains rising from approximately 15% at 20 services to a peak of 20.6% at 100 services before moderating to 14.5% at 150 services. Against pure RSO, the hybrid's benefit grows from 25.4% to 42.3% on mid-sized instances, reflecting the added value of blending SFO's exploitation into RSO's exploration. The most significant enhancements are observed against PSO (43%–53%), highlighting RSO–SFO's capacity to surpass PSO's premature convergence. Finally, as the problem gets bigger, the gains over SFO go from 35% to 44%, which shows that the hybrid works well in all of the tests.

Figure 10 shows the costs of five meta-heuristic techniques as the number of services goes up from 20 to 150. The RSO–SFO hybrid always has the lowest cost curve, indicating that it can identify the most optimal placement strategies. Particle Swarm and Sunflower Optimizations get a lot more expensive as service loads go up, while Grey Wolf and Rat Swarm Optimization stay in the middle range. This trend indicates that RSO–SFO is more effective at controlling costs while growing. Even though the system is getting more complicated, it keeps its operating costs low by finding a balance between exploration and exploitation.

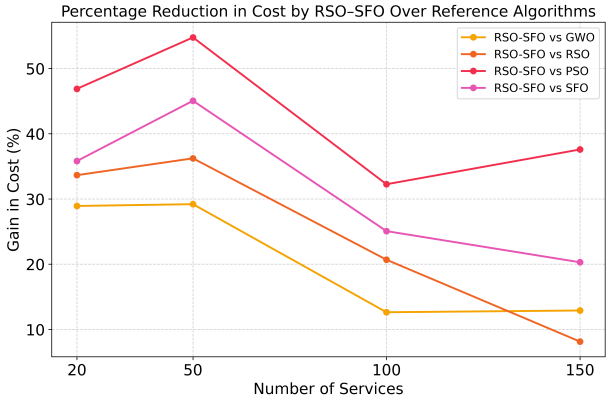

Figure 11: Cost reduction (%) of RSO–SFO over reference algorithms for 20–150 services.

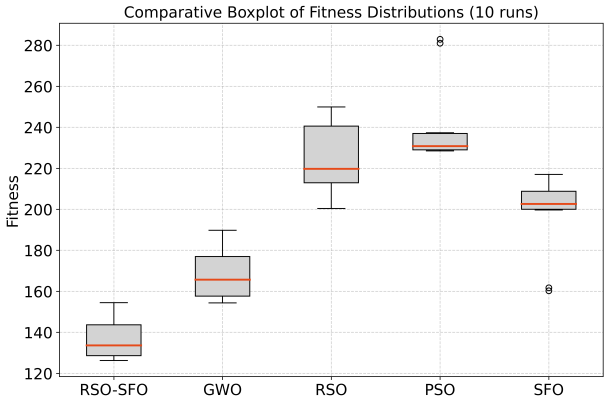

Figure 12: Fitness Distribution Boxplots for All Methods.

For each service count, the percentage reduction in final cost achieved by RSO–SFO relative to each reference algorithm is calculated as

$$\text{Gain}^{\text{Cost}}_{\text{RSO–SFO vs. } X} = \frac{C_X - C_{\text{RSO–SFO}}}{C_X} \times 100,$$

where C_X and $C_{\text{RSO–SFO}}$ denote the final cost values of algorithm X and the hybrid method, respectively. These cost–reduction percentages for service counts of 20, 50, 100, and 150 are shown in Figure 11. As demonstrated, at smaller scales, RSO–SFO saves approximately 29–30% compared to GWO, while the reduction reaches about 34–36% compared to RSO. These gains taper to roughly 13% and 8% at 150 services, respectively. In comparison to PSO and SFO, the hybrid achieves even more substantial improvements, with cost reductions of about 32–55% and 20–45%, respectively, highlighting its strong efficacy and stable performance in reducing operating costs as the number of services increases.

Table 7 shows that the RSO–SFO hybrid achieves the lowest average fitness (136.34) with the smallest dispersion (σ = 9.16). Compared to the Grey Wolf Optimizer (167.68 ± 11.46), RSO–SFO yields an 18.8% reduction in fitness; it outperforms PSO by 43.5% (241.40 ± 20.50). The relative standard deviation (σ/μ) for RSO–SFO is 6.7%, versus 8.6% for GWO and up to 8.5% for PSO, evidencing both superior performance and greater consistency.

Table 7: Comparison of Fitness Results

Method	Average Fitness	Standard Deviation
RSO–SFO	136.34	9.16
GWO	167.68	11.46
RSO	224.73	17.68
PSO	241.40	20.50
SFO	197.15	18.72

Table 8 summarizes two-sided Wilcoxon signed-rank tests (n = 10) comparing RSO–SFO to each benchmark. In every case $W = 0$ and $p \approx 0.00195$ (<0.01), allowing us to reject the null hypothesis of equal median fitness. This confirms that the observed improvements of RSO–SFO are statistically significant and not due to random chance.

Figure 12 visualizes the full distribution of fitness values across the ten runs for each algorithm, highlighting the tighter spread and lower median of RSO–SFO.

Table 8: Wilcoxon Signed-Rank Test for RSO–SFO vs. Other Algorithms (n = 10)

Comparison	W	*p*-value	Significance
RSO–SFO vs. GWO	0	0.00195	$p < 0.01$
RSO–SFO vs. RSO	0	0.00195	$p < 0.01$
RSO–SFO vs. PSO	0	0.00195	$p < 0.01$
RSO–SFO vs. SFO	0	0.00195	$p < 0.01$

In summary, the RSO–SFO hybrid demonstrates superior effectiveness for Fog service placement, achieving the fastest convergence, highest resource utilization, and the lowest operational cost across all tested loads. As the number of services grows, its benefits become more pronounced, demonstrating stable and consistent performance under increasing workload intensity. RSO–SFO is a reliable and efficient approach for large-scale service placement in Fog Computing because it consistently balances global search and local refinement by combining broad rat-swarm exploration with focused sunflower exploitation.

7 Discussion

The following discussion interprets the experimental results, examines their implications, and highlights the limitations of the proposed approach, without reiterating methodological details already presented in earlier sections.

The proposed RSO–SFO hybrid framework combines the exhaustive global exploration capabilities of Rat Swarm Optimization with the focused local exploitation mechanism of Sunflower Optimization. This tightly coupled interaction significantly accelerates convergence and improves the unified fitness function by 45.38% compared to GWO, RSO, PSO and SFO. In addition, the hybrid strategy reduces deployment costs by 43.71% while maintaining an average resource utilization of 78.83%, thereby demonstrating both efficiency and robustness under varying IoT workload conditions. These results confirm that jointly orchestrating complementary search behaviors within a unified optimization loop yields tangible performance gains over standalone and loosely hybridized metaheuristics.

Although latency is not explicitly modeled, improving resource utilization and reducing deployment cost promote balanced service allocation across fog nodes, which may help mitigate resource congestion and processing delays. This provides a foundation for future latency-aware fog service placement models.

From a conceptual standpoint, this work contributes beyond an empirical combination of existing algorithms. The novelty of the RSO–SFO approach lies in its structured hybridization principle, where exploration and exploitation are coordinated at the agent level and governed by a shared fitness function and common feasibility constraints. The proposed framework combines complementary search mechanisms within a unified optimization process. This design aims to maintain an effective balance between exploration and exploitation, which is particularly suitable for constrained fog service placement problems.

The experimental evaluation in this study is conducted under a static optimization setting. The proposed RSO–SFO framework computes service placement decisions based on a snapshot of the system state. Therefore, the experiments assess the convergence behavior and optimization effectiveness of the hybrid metaheuristic under controlled static conditions.

Despite its strong performance, achieving optimal results with RSO-SFO depends on careful parameter tuning, including swarm size, exploration–exploitation balance, and phyllotaxis-related settings. Inappropriate parameter configurations may impair convergence or lead to premature stagnation. Moreover, as with many population-based metaheuristics, the algorithm may still become trapped in local optima when confronted with highly complex or rapidly evolving fog topologies. Addressing these limitations and extending the framework to fully online optimization with explicit workload prediction, real-time

monitoring and continuous reconfiguration constitute promising directions for future research and real-world deployment.

8 Conclusion and Future Work

This paper presents a structured RSO–SFO hybrid approach for Fog service placement, combining complementary swarm-based behaviors within a unified fitness formulation that balances deployment cost and resource utilization. Experimental results demonstrate that the proposed framework outperforms state-of-the-art metaheuristics in terms of convergence speed, cost efficiency and resource optimization. Although latency is a critical concern in fog computing, this study focuses on cost-aware placement and efficient resource utilization, which, in turn, indirectly promote locality-aware and congestion-reducing service assignments. Explicit latency modeling and network-aware optimization will be addressed in future work. Future research will explore online and latency-aware extensions of the proposed framework, including reinforcement learning for dynamic adaptation, self-adaptive parameter control, lightweight variants for resource-constrained IoT gateways, and secure orchestration mechanisms. Quantum-inspired operators also represent a promising avenue for further accelerating convergence in large-scale fog environments.

Acknowledgement: The authors sincerely thanks the supervisors for their academic guidance, support, and valuable insights throughout this work, and acknowledges colleagues for constructive discussions and technical assistance.

Funding Statement: This research received no external funding.

Author Contributions: All authors contributed equally to this work. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: This study does not involve the generation or analysis of datasets. Therefore, data availability is not applicable.

Ethics Approval: Not applicable.

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

References

1. I. Ahammad, "Fog computing complete review: Concepts, trends, architectures, technologies, simulators, security issues, applications, and open research fields," *SN Computer Science*, vol. 4, no. 6, 765, 2023. https://doi.org/10.1007/s42979-023-02235-9.
2. S. P. Malukani and C. K. Bhensdadia, "Fog computing algorithms: A survey and research opportunities," *Applied Computer Systems*, vol. 26, no. 2, pp. 139–149, 2021. https://doi.org/10.2478/acss-2021-0017.
3. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. of the 1st ACM Workshop on Mobile Cloud Computing (MCC)*, New York, NY, USA: ACM; pp. 13–16, 2012. https://doi.org/10.1145/2342509.2342513.
4. R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. Di Martino, K. C. Li, L. T. Yang, and A. Esposito, Eds., Cham, Switzerland: Springer, 2018, pp. 103–130. https://doi.org/10.1007/978-981-10-5861-5_5.
5. M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017. https://doi.org/10.1109/ACCESS.2017.2749422.
6. H. K. Apat, V. Goswami, B. Sahoo, R. K. Barik, and M. J. Saikia, "Fog Service Placement Optimization: A Survey of State-of-the-Art Strategies and Techniques," *Computers*, vol. 14, no. 3, 99, 2025. https://doi.org/10.3390/computers14030099.

7. C. Guerrero, I. Lera, and C. Juiz, "Genetic-based optimization in fog computing: Current trends and research opportunities," *Swarm and Evolutionary Computation*, vol. 72, 101094, 2022. https://doi.org/10.1016/j.swevo.2022.101094.
8. N. Kaliya and D. Pawar, "Unboxing fog security: A review of fog security and authentication mechanisms," *Computing*, vol. 105, no. 12, pp. 2793–2819, 2023. https://doi.org/10.1007/s00607-023-01208-3.
9. H. K. Apat, B. Sahoo, V. Goswami, and R. K. Barik, "A hybrid meta-heuristic algorithm for multi-objective IoT service placement in fog computing environments," *Decision Analytics Journal*, vol. 10, 100379, 2024. https://doi.org/10.1016/j.dajour.2023.100379.
10. N. Mehran, D. Kimovski, and R. Prodan, "MAPO: A multi-objective model for IoT application placement in a fog environment," arXiv:1908.01153. 2019.
11. M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "An evolutionary multi-objective optimization technique to deploy the IoT services in fog-enabled networks: An autonomous approach," *Applied Artificial Intelligence*, vol. 36, no. 1, 2008149, 2022. https://doi.org/10.1080/08839514.2021.2008149.
12. B. V. Natesha and R. M. R. Guddeti, "Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," *Journal of Network and Computer Applications*, vol. 178, 102972, 2021. https://doi.org/10.1016/j.jnca.2020.102972.
13. J. Niu, G. Liu, L. Yu, and J. Wang, "Service placement optimization based on evolutionary algorithm in fog computing," *EAI Endorsed Transactions on Internet of Things*, vol. 7, no. 26, e5, 2022. https://doi.org/10.4108/eai.22-2-2022.173492.
14. C. Liu, J. Wang, L. Zhou, and A. Rezaeipanah, "Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm," *Neural Processing Letters*, vol. 54, no. 3, pp. 1823–1854, 2022. https://doi.org/10.1007/s11063-021-10708-2.
15. G. M. Gilbert, N. Shililiandumi, and H. Kimaro, "Evolutionary approaches to fog node placement in LV distribution networks," *International Journal of Smart Grid (ijSmartGrid)*, vol. 5, no. 1, pp. 23–36, 2021. https://doi.org/10.20508/ijsmartgrid.v5i1.141.g134.
16. Y. Ramzanpoor, M. Hosseini Shirvani, and M. Golsorkhtabaramiri, "Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure," *Complex & Intelligent Systems*, vol. 8, no. 1, pp. 361–392, 2022. https://doi.org/10.1007/s40747-021-00368-z.
17. M. Ghobaei-Arani and A. Shahidinejad, "A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment," *Expert Systems with Applications*, vol. 200, 117012, 2022. https://doi.org/10.1016/j.eswa.2022.117012.
18. D. Zhao, Q. Zou, and M. B. Zadeh, "A QoS-aware IoT service placement mechanism in fog computing based on open-source development model," *Journal of Grid Computing*, vol. 20, no. 2, 12, 2022. https://doi.org/10.1007/s10723-022-09604-3.
19. B. Wu, X. Lv, W. D. Shamsi, and E. G. Dizicheh, "Optimal deploying IoT services on the fog computing: A metaheuristic-based multi-objective approach," *Journal of King Saud University–Computer and Information Sciences*, vol. 34, no. 10, pp. 10010–10027, 2022. https://doi.org/10.1016/j.jksuci.2022.10.002.
20. M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment," *Software: Practice and Experience*, vol. 51, no. 8, pp. 1745–1772, 2021. https://doi.org/10.1002/spe.2986.
21. M. Ayoubi, M. Ramezanpour, and R. Khorsand, "An autonomous IoT service placement methodology in fog computing," *Software: Practice and Experience*, vol. 51, no. 5, pp. 1097–1120, 2021. https://doi.org/10.1002/spe.2939.
22. H. Ben Rjeb, L. Sliman, H. Zorgati, R. Ben Djemaa, and A. Dhraief, "Optimizing Internet of Things Services Placement in Fog Computing Using Hybrid Recommendation System," *Future Internet*, vol. 17, no. 5, 201, 2025. https://doi.org/10.3390/fi17050201.
23. S. Ali and M. Khan, "Optimizing fog colony layout and service placement through genetic algorithms," *Expert Systems with Applications*, 254, 124372, 2024. https://doi.org/10.1016/j.eswa.2024.124372.
24. M. Toghyani, R. Khorsand, and H. Khaksar, "QoS-SLA-Aware Optimization Framework for IoT-Service Placement in Integrated Fog-Cloud Computing," *Journal of Grid Computing*, vol. 23, 1, 2025. https://doi.org/10.1007/s10723-024-09787-x.

25. S. Bolettieri, R. Bruno, and E. Mingozzi, “Application-aware resource allocation and data management for MEC-assisted IoT service providers,” *Journal of Network and Computer Applications*, vol. 181, 103020, 2021. https://doi.org/10.1016/j.jnca.2021.103020.

26. Gaurav Dhiman, Meenakshi Garg, Atulya Nagar, Vijay Kumar, Mohammad Dehghani, “A novel algorithm for global optimization: Rat Swarm Optimizer,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8457–8482, 2021. https://doi.org/10.1007/s12652-020-02580-0.

27. G. F. Gomes, S. S. da Cunha Jr., and A. C. Ancelotti Jr., “A sunflower optimization (SFO) algorithm applied to damage identification on laminated composite plates,” *Engineering with Computers*, vol. 35, no. 2, pp. 619–626, 2019. https://doi.org/10.1007/s00366-018-0620-8.

28. G. F. Gomes and R. S. Giovani, “An efficient two-step damage identification method using sunflower optimization algorithm and mode shape curvature (MSDBI–SFO),” *Engineering with Computers*, vol. 38, pp. 1711–1730, 2022. https://doi.org/10.1007/s00366-020-01128-2.

29. J. L. J. Pereira and G. F. Gomes, “Multi-objective Sunflower Optimization: A hypercubic constrained meta-heuristic,” *Expert Systems*, vol. 40, no. 8, e13331, 2023. https://doi.org/10.1111/exsy.13331.