Journal of Engineering Management and Soft Computing (JEMSO)

Journal of Data Analytics and Intelligent Decision-making

Journal homepage: jdaid.qom.ac.ir

University of Qom

Load-Tolerant Multi-Objective Optimization for Resource Allocation in SDN-Based IoT

Mohammad Rostami*[a], Afsaneh Fatemi[b], Salman Goli-Bidgoli[C]

a. Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.
b. Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.
c. Faculty of Electrical and Computer Engineering, University of Kashan, Kashan, Iran.

ARTICLEINFO

Keywords:
Quality of service
Load balancing
Load tolerance
Internet of Things
Software-defined networking
Grey Wolf Optimizer

ABSTRACT

The increasing workload in Internet of Things (IoT) environments leads to network congestion and resource imbalance, significantly degrading the Quality of Service (QoS). Software-Defined Networking (SDN) provides a flexible control paradigm for improving QoS and load balancing. However, most existing approaches are limited to multi-objective formulations and do not explicitly address overload conditions. Therefore, there is a need for many-objective QoS optimization frameworks that can jointly consider multiple conflicting objectives while maintaining load tolerance in SDN-based IoT systems. In this paper, a load-tolerant many-objective resource allocation framework is proposed to simultaneously optimize user and service provider QoS requirements. Specifically, cost and response time are minimized for users, while energy consumption is minimized and resource utilization is maximized for service providers. A Pareto-based many-objective evolutionary optimization process is employed to generate diverse non-dominated solutions, which are evaluated under overload conditions. To address infeasible allocations under overload, a load-tolerance mechanism is introduced to identify admissible solutions and improve system robustness. This mechanism enables sustained task allocation even when Pareto-optimal solutions violate load threshold, thereby increasing the task acceptance rate under overload conditions. Simulation results demonstrate that the proposed Many-objective Grey Wolf Optimization (MaGWO)-based framework improves resource allocation cost by 13.85%, response time by 17.2%, energy consumption by 15.8%, and resource utilization by 10.25%. Furthermore, the proposed load-tolerant strategy increases the task acceptance rate by 8.4% compared to NSGA-III and Many-objective Particle Swarm Optimization (MaPSO).

* Corresponding author.
E-mail addresses: m.rostami@comp.ui.ac.ir (M. Rostami), a_fatemi@eng.ui.ac.ir (A. Fatemi), salmangoli@kashanu.ac.ir (S. Goli-Bidgoli)

Received 26 Nov 2025; Received in revised form 27 Jan 2026; Accepted 31 Jan 2026
Available online 30 Mar 2026
3115-8161

Cite this article: Rostami, M., Fatemi, A., Goli-Bidgoli, S. (2026). Load-Tolerant Multi-Objective Optimization for Resource Allocation in SDN-Based IoT. *Journal of Data Analytics and Intelligent Decision-making*, 2(1), 31-57.
https://doi.org/10.22091/jdaid.2026.14627.1022

1. Introduction

With the continuous growth of networking infrastructures and the increasing number of IoT devices, the demand for various services has surged substantially. This rise in service requests leads to increased traffic and load imbalance across resources, ultimately degrading the QoS (Keshari et al., 2021; Sandanasamy & Charles, 2025). In this context, SDN can effectively meet diverse network requirements. By providing a global view of the network and awareness of its operational state, SDN facilitates programmability, traffic management, and efficiency improvement for IoT services. Consequently, integrating SDN into IoT enables intelligent real-time traffic management and effective load control, playing a crucial role in enhancing QoS. In fact, optimal resource allocation helps prevent congestion and significantly improves network performance.

In SDN-based IoT environments, employing an efficient resource allocation algorithm that considers task characteristics is essential for managing computational resources and delivering high-quality services (Cao et al., 2021). Resource allocation, as a critical QoS optimization process, can enhance network performance, accommodate the increasing number of user tasks, and prevent violations of service-level agreements (SLAs) between users and service providers (Belgacem, 2022). The SLA defines the expected service level and performance requirements negotiated between users and service providers (Alsadie, 2021). Therefore, resource allocation plays a vital role in improving QoS for both end-user tasks and service providers.

Optimization problems, particularly many-objective ones, often rely on metaheuristic algorithms to obtain optimal or near-optimal solutions. Due to inherent conflicts among different objectives, no single solution can simultaneously optimize all objectives in many-objective problems. Instead, optimization yields a diverse set of candidate solutions from which decision-makers can evaluate and select. Metaheuristic algorithms have been widely applied in recent years, efficiently exploring the search space and converging to near-optimal solutions in reasonable time. These algorithms generate sets of non-dominated Pareto-optimal solutions, allowing users to choose the most suitable solution based on environmental conditions, requirements, and priorities, as no single optimal solution universally dominates others.

Several well-known metaheuristic algorithms have been utilized to obtain optimal solutions, including the Non-dominated Sorting Genetic Algorithm III (NSGA-III) (Imene et al., 2022), Multi-objective Grey Wolf Optimizer (MoGWO) (Mirjalili et al., 2016), and Many-objective Particle Swarm Optimization (MaPSO) (Figueiredo et al., 2016). These algorithms leverage the concept of Pareto dominance to judge solution quality and aim to identify solutions close to the global optimum.

Despite extensive research on QoS-aware resource allocation in SDN-based IoT environments, several critical limitations remain insufficiently addressed. Existing QoS improvement techniques focus on optimizing performance metrics under normal conditions and implicitly assume balanced workloads. As a result, their effectiveness degrades significantly when the system experiences overload or traffic surges.

Another overlooked aspect is the explicit consideration of load imbalance as an optimization factor. While metrics such as delay, energy consumption, and cost are commonly optimized, the relationship between load imbalance and QoS degradation, particularly its impact on response time violations, and reduced task acceptance is rarely quantified or systematically incorporated into the optimization process. Consequently, there is a clear research gap in developing a unified framework that simultaneously addresses many-objective QoS optimization and load imbalance under overload conditions.

This study proposes a GWO–based approach to address the resource allocation problem for improving many-objective QoS and maintaining load balancing in SDN-based IoT environments. The proposed algorithm addresses QoS provisioning by simultaneously

optimizing four qualitatively distinct and conflicting objectives from both user and service provider directions. Specifically, it aims to minimize cost and response time for end users, while concurrently minimizing energy consumption and maximizing resource utilization for service providers. The concurrent optimization of these heterogeneous objectives introduces complex trade-offs that cannot be adequately captured using low-dimensional or aggregated formulations, thereby necessitating a many-objective optimization framework. In essence, the proposed approach aims to dynamically align infrastructure resource allocation with network traffic demands by assigning computational tasks to resources according to many-objective QoS requirements.

Existing MaGWO-based approaches for QoS optimization generally assume the existence of solutions under fixed resource and workload constraints. However, under overload conditions which are common in SDN-based IoT environments, these assumptions often break down, resulting in task rejections or degraded QoS. To address this critical limitation and enhance the effectiveness of MaGWO in improving QoS while maintaining load balance under overload scenarios, the following key contributions are made:

- Development of GWO for constrained many-objective QoS optimization, in which Pareto dominates the leadership hierarchy.
- Ensuring QoS across four objectives of cost, response time, energy consumption, and resource utilization while complying with constraints such as throughput, reliability, and availability.
- Introducing a resource allocation approach that identifies balanced solutions among conflicting objectives, evaluates overload conditions, and ultimately selects a single implementable solution from the Pareto-optimal set using a preference-based decision-making mechanism.
- Designing a load-tolerance mechanism to handle scenarios in which no solution satisfies both Pareto-optimality and load-balancing requirements, thereby increasing task acceptance rates.
- Demonstrating superior performance of the proposed method, achieving better convergence and more uniformly distributed Pareto solutions compared to other algorithms.
- Differentiation existing GWO by introducing a load-tolerant many-objective approach, ensuring deployable solutions even when QoS constraints cannot be simultaneously satisfied.

The rest of the paper is organized as follows. Section 2 reviews related research on QoS improvement. Section 3 explains the MaGWO algorithm. Section 4 presents the proposed approach. Section 5 provides simulation results and performance analysis. Finally, Section 6 concludes the paper.

2. Literature Review

Improving QoS through optimal task placement across resources plays a significant role in enhancing performance and maintaining the stability of network infrastructures. Efficient load management supports scalable network expansion and ensures smooth operational capability.

According to previous studies in IoT and SDN, QoS improvement and load balancing are two critical challenges within these environments. Through its ability to share resources efficiently and manage the network intelligently, SDN acts as a key enabler for enhancing the performance of IoT applications and meeting QoS requirements. Numerous techniques with varying QoS parameters have been proposed to efficiently allocate tasks to resources in SDN-based IoT environments. These methods span a wide range of mechanisms, such as scheduling, migration, routing, clustering, offloading, prediction, allocation, aggregation, and prioritization.

This section reviews the most relevant studies on QoS improvement using metaheuristic algorithms. Various metaheuristic algorithms, such as genetic algorithms, particle swarm

optimization, ant colony optimization, and GWO, have been widely employed to solve QoS optimization problems. Table 1 presents a summary of the most notable QoS and load-balancing approaches.

Many recent studies have explored many-objective optimization in IoT and SDN environments by enhancing or hybridizing population-based swarm intelligence algorithms. These approaches typically generate a diverse set of optimal solutions. Metaheuristic techniques are particularly effective in resource allocation problems under QoS constraints, as they help identify high-quality approximate solutions in NP-hard scenarios. Since no single optimization method can guarantee optimal performance across all problem types (Rostami & Goli-Bidgoli, 2024), existing algorithms must be adapted and extended to meet the needs of newly emerging optimization challenges.

While previous studies have proposed various methods for facilitating network management, only a limited number have explored approaches specifically tailored to SDN-based IoT environments. Moreover, the concept of IoT integrated with SDN is relatively new, and innovative strategies for managing IoT traffic and resources are still evolving. Most existing studies focus on cloud environments and emphasize objectives such as cost, response time, energy consumption, and resource utilization.

In this study, particular attention is paid to resource management and QoS. Therefore, investigating such a combined approach is essential, especially given the rapid development of IoT applications and the increasing adoption of SDN.

This paper proposes a MaGWO-based framework for addressing QoS and load balancing in SDN-based IoT environments, with particular emphasis on overload conditions that are often overlooked in existing studies. Unlike prior approaches that focus on performance optimization under balanced workloads, the objective here is to determine task allocation strategies that enhance QoS for both users and service providers while preserving load-balancing through load-tolerant mechanisms when system resources become overloaded.

Table 1
Approaches for Improving QoS and Load Balancing

Ref.	Year	Network Type	Technique	Objectives	directions	Algorithm	Objective Type	Overload handling	Acceptance rate
Hashemi et al.	2021	Cloud	VM allocation	Energy consumption, VM allocation time	Service Provider	Improved GWO	Multi-objective	No	No
Cao et al.	2021	Fog-Cloud/SDN	Allocation	Delay, cost, load balancing, stability	User, Service Provider	Two_Arch2	Many-objective	No	No
Alsadie	2021	Cloud	Scheduling	Makespan, resource utilization, imbalance degree, throughput	Service Provider	Multi-objective GWO	Multi-objective	No	No
Keshari et al.	2021	SDN-IoT	Clustering	Communication cost	Service Provider	Cluster-based GWO	Single-objective	Partial	No
Cao et al.	2021	5G IoV-SDN	Hierarchical clustering	Stability, load balancing,	Service Provider	Two_Arch2	Many-objective	No	No

				delay, energy consumption					
Balicki	2022	Cloud	VM migration	Energy consumption, reliability, CPU load, cost	Service Provider	Quantum-inspired many-objective PSO	Many-objective	No	No
Imene et al.	2022	Cloud	Scheduling	Execution time, energy consumption, cost	User, Service Provider	Many-objective Genetic Algorithm	Multi-objective	No	No
Sefati et al.	2022	Cloud	Allocation	Makespan, response time, resource utilization	User, Service Provider	Multi-objective GWO	Multi-objective	No	No
Saif et al.	2023	Cloud-Fog	Scheduling	Delay, energy consumption	User, Service Provider	Multi-objective GWO	Multi-objective	No	No
Mohammadi et al.	2023	SDN-IoT	Clustering	Energy consumption, network lifetime, throughput	Service Provider	Improved Sailfish Optimization	Multi-objective	No	No
Singh & Chaturvedi	2024	Cloud-Fog	Scheduling	Makespan, Cost, Energy Consumption	User, Service Provider	Hybrid GA-modified PSO	Multi-objective	No	No
Badr et al.	2024	IoT-Cloud-Fog	Scheduling	Makespan, Energy Consumption, Cost	User, Service Provider	Multi-objective PSO	Multi-objective	No	No
Tyagi et al.	2024	SDN-IoE	Migration	Delay, RTT	User	Cheetah Optimization	Multi-objective	Partial	No
Hussaini et al.	2025	IoT	Allocation	Cost, delay, energy consumption	User, Service Provider	Multi-objective Grey Wolf and Whale	Multi-objective	No	No
Sandanasamy & Charles	2025	SDN-IoT	Allocation	Response time, throughput, load balancing, resource utilization	Service Provider	AHP-TOPSIS based server selection	Multi-objective	Partial	No
Nain et al.	2025	SDN- Edge	Allocation	Makespan, resource utilization, load balancing	User, Service Provider	Average-Based load balancing algorithm	Multi-objective	Partial	No
Hosseinzadeh et al.	2025	SDN-IoT	Scheduling	CPU, memory, storage utilization, latency, energy consumption	User, Service Provider	Deep Q-Network	Many-objective	No	No
Singh et al.	2025	IoT	Architecture	Energy, latency,	User, Service Provider	Hybrid quantum-	Many-objective	No	Indirect

				coverage rate, service cost		based GWO and Whale			
Rameshet al.	2025	Cloud	Scheduling	Cost, execution time, load balancing	User, Service Provider	Hybrid GWO	Multi-objective	No	Indirect
proposed	2025	IoT/SDN	Allocation	Cost, response time, energy consumption, resource utilization	User, Service Provider	Improved GWO	Many-objective	Yes	Yes

As summarized in Table 1, recent GWO-based and hybrid metaheuristic approaches predominantly focus on improving convergence behavior or solution diversity under QoS constraints. A clear pattern emerging from the comparison is that most existing studies assume balanced workloads and do not explicitly address scenarios in which Pareto-optimal solutions violate load-balancing requirements.

Moreover, Table 1 reveals that the majority of existing approaches are formulated as multi-objective optimization problems, load imbalance is often treated as a secondary outcome rather than a first-class optimization concern, and explicit load-tolerance or overload-handling mechanisms are rarely incorporated. Consequently, their performance degrades significantly when the system operates under overload scenarios. Although several studies jointly address QoS optimization and load balancing, their ability to maintain QoS under overload conditions remains limited.

These observed gaps highlight the need for a unified framework that jointly addresses many-objective QoS optimization and load tolerance. Motivated by these limitations, the proposed approach integrates Pareto-based many-objective optimization with a load-tolerant decision-making mechanism to ensure QoS solutions under overloaded conditions.

It is worth noting that some existing studies partially mitigate overload through clustering, migration, or load balancing strategies. However, as presented in Table 1, explicit load-tolerance mechanisms and direct evaluation of task acceptance rate are largely absent, highlighting the distinct contribution of the proposed approach.

3. Many-Objective Grey Wolf Optimization (MaGWO)

The MaGWO algorithm is designed based on the basic grey wolf optimizer and can effectively approximate Pareto-optimal solutions for problems with multiple objectives. The archive component is a fixed-size memory responsible for storing and retrieving the approximate Pareto-front solutions of the non-dominated wolves during the optimization process as the global best solutions. The archive capacity is determined based on the size of the initial population. When the archive reaches full capacity, the crowding-distance mechanism is used to remove some solutions (with equal rank) from the most crowded region (archive pruning). Then, the new solutions must enter the less crowded region, which affects the performance of the algorithm. According to the search process, the algorithm is allowed to quickly converge to the non-dominated solutions and also preserve the diversity of the non-dominated solutions (improving the quality and uniformity of the distribution of the solution set). The diversity operator prevents the solutions from falling into local minima.

The leader selection strategy of the wolves as a many-objective optimization component is based on the alpha, beta, and delta wolves for updating and replacing the solutions stored in the archive. The non-dominated wolves are better in terms of balance between the QoS objective values and are placed in different fronts or ranks. The wolves in each front must be updated and sorted so that a good spread of solutions is achieved. The omega wolves follow these three

wolves in search for the global optimum. This technique generally processes and refines the population previously generated in the later stages.

The grey wolves' strategy during the hunting process involves repeating three main steps: search, encircling, and attacking (Makhadmeh et al., 2023; Saif et al., 2023). Equation (1) is for the search phase, while Equation (2) is for the encircling phase.

$$D = \left|C.X_p(t) - X_{GW}(t)\right| \quad (1)$$

$$X_{GW}(t+1) = X_p(t) - A.D \quad (2)$$

Parameter D is the distance between the prey and the wolf; X_p is the prey (target) position; X_{GW} is the current wolf position; A and C are coefficient vectors; and t indicates the current iteration count. The coefficient vector C is for the global search of the prey (exploration), and the coefficient vector A is used for random behavior during optimization, exploration, and avoidance of local optimum (Gohil & Patel, 2022). The value of parameter A decreases linearly depending on 'a' during iterations (Hashemi et al., 2021). Equations (3) and (4) are used to compute vectors A and C.

$$A = 2a.r_1 - a \quad (3)$$

$$C = 2.r_2 \quad (4)$$

r_1 and r_2 are random vectors in the interval [0,1], allowing grey wolves to reach any random position around the prey using Equations (1) and (2) (Gohil & Patel, 2022). Vector 'a' varies linearly in the interval [2, 0] during the iteration period of the algorithm and is computed according to Equation (5).

$$a(t) = 2 - t\left(\frac{2}{\text{Maxiter}}\right) \quad (5)$$

Where 't' is the current iteration and Maxiter is the maximum number of algorithm iterations. After calculating the fitness of the search agents, the three wolves closest to the answer are selected, and the remaining wolves update their position relative to the three superior wolves using Equations (6) and (7) to find the optimal prey/solution. The update of the omega wolf position X(t+1), with respect to the optimal position of the three superior wolves to find the optimal solution (position) in the search space, is computed according to Equation (8). Equations (6), (7), and (8) are used to implement the attacking mechanism.

$$D_a = |C_1 \times X_a - X|,\ D_\beta = \left|C_2 \times X_\beta - X\right|,\ D_\delta = |C_3 \times X_\delta - X| \quad (6)$$

$$X_1 = X_a - A1 \times (D_a),\ X_2 = X_\beta - A2 \times \left(D_\beta\right), X_3 = X_\delta - A3 \times (D_\delta) \quad (7)$$

$$X(t+1) = \frac{\text{X1+X2+X3}}{3} \quad (8)$$

Non-Dominated Sorting

Solutions are sorted based on the concept of non-dominance. In the non-dominated sorting method, solutions that are not dominated by other solutions are assigned Rank 1, placed in the first front, and removed from the population. This process continues until all solutions are placed into different fronts.

Pareto Dominance

If the goal of optimization is minimization, we say that vector $x \epsilon R^n$ dominates vector $y \in R^n$, meaning (x < y):

$$\forall i \in \{1,2,\dots,m\}: f_i(x) \leq f_i(y) \text{ and, } \exists j \in \{1,2,\dots,m\}: f_j(x) < f_j(y). \quad (9)$$

Solution $x \epsilon R^n$ is Pareto optimal if and only if:

$$\nexists y \in \mathbb{R}^n,\ x < y \quad (10)$$

Crowding Distance

After front classification, another criterion which is used to evaluate the solutions within the same front is the crowding distance. The crowding distance of each solution in the Pareto front equals the distance of that point from its neighboring points. First, for each objective function k, infinite crowding distance is assigned to the points that have maximum and

minimum values of that objective. For other points, Equation (11) is used, and the crowding distance of solution x is computed according to Equation (12).

$$cd_f(x_{[i,f]}) = \frac{f_k(x_{[i+1,k]}) - f_k(x_{[i-1,k]})}{f_k^{max} - f_k^{min}} \quad (11)$$

$$cd(x) = \sum cd_f(x) \quad (12)$$

In the above equation, f_k is the k-th objective function, and f_k^{max} and f_k^{min} are the maximum and minimum values of that function. Additionally, $x_{[i+1,k]}$ is the next neighboring point, and $x_{[i-1,k]}$ is the previous neighboring point of $x_{[i,k]}$. The crowding distance for the solutions of each front is calculated separately. Comparisons between distances are also performed only among the solutions within the same front. The pseudocode of the MaGWO is presented in Algorithm 1.

Algorithm 1: Pseudocode of the MaGWO

1: Initialize population of GWs
2: Initialize a, A, and C
3: Initial archive
4: Calculate the objectives values for each grey wolf
5: Calculate the fitness value for each grey wolf
6: Add non-dominated solutions to the archive
7: Select the best leaders from an archive (Xα, Xβ, Xδ)
8: t = 1
9: while (t < MaxIt) do
10: for each grey wolf do
11: Update the position of grey wolf through Equations (6)–(8)
12: end for
13: Update a, A, and C through Equations (3)–(5)
14: Calculate the objective values for all GWs
15: Update the archive according to the obtained non-dominated values
16: if the archive is full then
17: Calculate crowding distance (cd) through Equation (11)
18: archive sort in descending according to the values of cd
19: Delete the Min. value of cd in the archive
20: end if
21: t = t + 1
22: end while
23: return archive
24: return the Xα as the best solution in the search space

The non-dominated sorting technique is adopted to define the social hierarchy, along with the crowding distance for selecting the next generation. Each solution in the archive is assigned a rank which represents its non-dominated level (rank one indicates the first-best level, rank two indicates the second-best level, and so on). As a result, the archive is divided into different fronts based on rank. Generally, throughout the iterative stages of the algorithm, the non-dominated solutions obtained in each iteration are compared with the existing solutions in the archive. The crowding distance mechanism is considered for improving the non-dominated solutions in the archive to achieve the optimal Pareto front. Solutions are updated in the archive according to the crowding distance concept, enabling diversity and exploration in new spaces and prevents premature convergence. This leads to faster convergence, enhanced global search capability, and thus, avoids getting trapped in local optima.

It should be noted that the proposed MaGWO does not merely append Pareto-based operators to the classical GWO. Instead, the hunting mechanism and leadership structure are

reinterpreted in a many-objective context, where alpha, beta, and delta wolves are selected from different regions of the Pareto front to balance convergence and diversity. The proposed adaptations and novel mechanisms are detailed in Section 4.

3-1. Representation of Solutions in Discrete Optimization Problems

One of the fundamental aspects of solving optimization problems using metaheuristics is the representation of solutions. In the allocation problem, each solution is a mapping of resources to tasks. Considering that the GWO was developed for continuous problems, and resource-to-task allocation is a discrete problem, the minimum position rule is proposed to determine the position of the solution.

The position of the i-th solution is represented as a vector W_i:<w_{i1}, w_{i2},..,w_{in}> containing continuous values. Using the minimum position rule, the continuous values of the position vector are mapped to discrete values S_i:< s_{i1}, s_{i2},.., s_{in}>, and the allocation vector P_i: <p_{i1}, p_{i2},..,p_{in}> for the i-th solution is obtained using the function $p_{i,k} = \mathrm{mod}(s_{i,k}, m+1)$ (the remainder of dividing $s_{i,k}$ by $m+1$). In other words, each element of the vector P_i (e.g., $P_{i,j}$) indicates the resource on which task j should be executed. It is worth noting that the fitness function values (cost, response time, energy consumption, and resource utilization) are calculated based on the allocation vector. For example, the calculation of P_i for allocating resources to 10 tasks with 3 resources is presented in Table 2. (Here, n and m represent the number of tasks and resources, respectively).

Table 2
An Example of Discrete Solution Representation

Tasks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\mathbf{W_i}$	4.76	7.94	3.27	3.58	3.99	9.45	5.09	8.85	2.93	5.68
$\mathbf{S_i}$	9	3	4	5	1	7	10	2	8	6
$\mathbf{P_i}$	1	3	0	1	1	3	2	2	0	2

4. Proposed Approach

The novelty of the proposed framework lies in the coupling of many-objective optimization with load-tolerance–aware feasibility analysis, enabling practical deployment in SDN controllers where a single robust decision must be executed under dynamic workload conditions. In this section, the assumptions and the description of the research problem are first presented, followed by the analysis, formulation, and time complexity of the problem.

4-1. Fundamental Assumptions

To continue the research, the assumptions related to the problem must be stated. Controller reliability and network stability assumptions are consistent with scenarios where redundancy and failover mechanisms are deployed at lower layers of the SDN architecture.

- Each user task is assigned only to a resource that meets its computational requirements.
- The resource demand information, provided by the user, is accurate and reliable.
- Each task submitted by the user must be executed on the assigned resource until its completion.
- The tasks submitted by the user are independent of one another.
- During the allocation process, multiple tasks can be assigned to each resource, resulting in varying workloads across resources.
- Based on system reliability, the probability of controller failure and network node malfunction is assumed to be zero.
- The network is reliable and fault-free, such that no links fail.

- The capacities (hardware) of the resources are considered identical.
- The minimum and maximum overload thresholds for resources are considered similar.
- Tasks are aggregated discretely within specific time intervals and enter the system for resource allocation (task start delays are not considered).

The above assumptions are adopted to simplify the system model and to focus on the effectiveness of the proposed many-objective optimization and load-tolerance mechanisms. While real SDN-based IoT environments may involve controller failures, network disruptions, and heterogeneous resource capacities, incorporating all such factors simultaneously would obscure the impact of the optimization strategy itself. Therefore, these assumptions serve as a baseline modeling choice rather than strict operational constraints.

4-2. Problem Description

Section 3 presented the general principles and standard mechanisms of the Many-Objective Grey Wolf Optimizer (MaGWO). To avoid redundancy, this section does not repeat the core algorithmic foundations. Instead, it focuses on the problem-specific adaptations and extensions introduced in this work, including the integration of load-tolerance mechanisms, SDN-aware constraints, and acceptance-based decision-making under overload conditions.

In IoT applications based on software-defined networking, one of the main concerns of service providers is delivering high-quality services to users, which requires optimizing resource utilization according to the current network conditions. To achieve user and provider satisfaction, two key aspects must be considered: improving the utilization of available resources and maximizing request acceptance, both aimed at optimizing QoS and maintaining load balancing.

Optimal resource allocation to tasks, in a way that enhances QoS and load balancing, is essential. Given the presence of a software-defined network controller with global visibility and programmability, it becomes possible to effectively make decisions regarding QoS and network load balancing. Essentially, the purpose of resource allocation is to assign tasks to resources while ensuring QoS and achieving load balancing.

The allocation of computational tasks to a set of computing resources is a many-objective optimization problem, where minimizing response time, resource leasing cost, and energy consumption, along with maximizing resource utilization, is necessary. However, the resource allocation problem is nondeterministic polynomial-time and, due to the number of QoS parameters involved, it is considered NP-hard (Cao et al., 2021). Therefore, heuristic (approximate) and metaheuristic approaches are often proposed as efficient methods for handling task execution. The proposed resource allocation framework is shown in Figure 1.

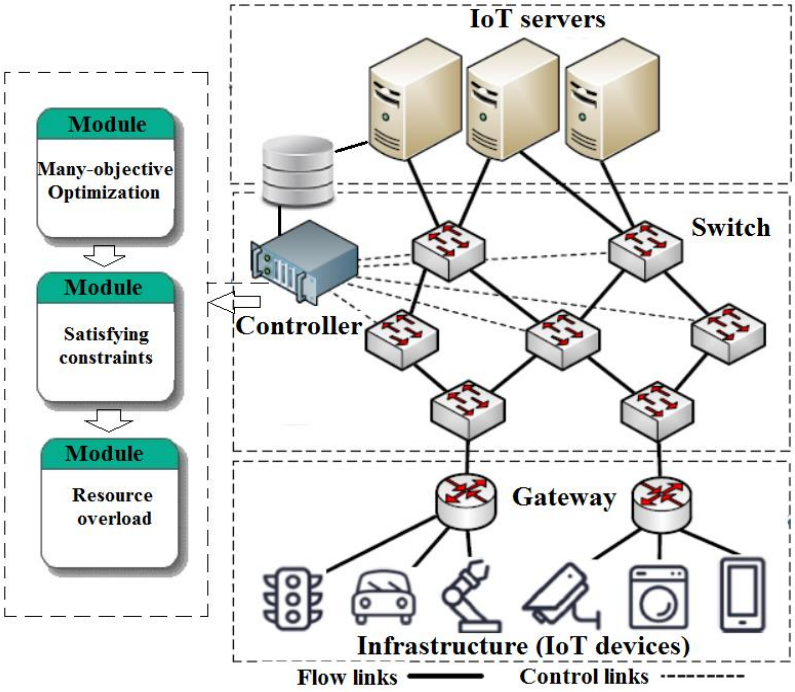

Figure 1
The Proposed Framework Considering QoS Improvement and Load Balancing

User tasks are sent to the controller for processing. The controller, as the main component, is responsible for assigning tasks to resources with the objective of QoS and load balancing. Using resource workload information and user task characteristics, the controller calls a model to map tasks onto resources. The model employs a MaGWO to obtain a set of solutions representing trade-offs among QoS objectives while satisfying constraints, based on the current resource consumption levels. Accordingly, tasks are allocated to computing resources on different servers through virtual machines.

In this study, workload refers to the number of tasks per unit time on the processor resource. The heuristic load balancing model defines the workload within a certain range (from minimum overload to maximum overload). When a task arrives, if the minimum overload threshold is satisfied, the task is accepted. If not, the overload tolerance is increased up to the maximum overload threshold; therefore, the task can be accepted if possible. Naturally, if the maximum overload threshold is violated, the task will not be accepted. Workload tolerance refers to the ability of a resource to process tasks under overload conditions, enabling the network to maintain flexibility and achieve proper load balancing even when resources face higher loads. The load-tolerance mechanism is evaluated under controlled conditions to assess its fundamental behavior.

In the proposed method, based on prior research, the most effective parameters are selected: cost and response time from the user perspective; energy consumption and resource utilization from the service provider perspective as objectives; and throughput, availability, and reliability as constraints. Since the problem is many-objective and some objectives conflict, the algorithm maintains trade-offs among them and generates feasible solutions, while constraints determine whether solutions are acceptable. Ultimately, the non-dominated solutions are presented as the optimal solution set in the form of a Pareto front. This means that the Pareto set contains optimal or near-optimal solutions that balance the objectives and satisfy the constraints. From the Pareto front, solutions that meet the minimum overload threshold, representing a range of load-balanced states, are selected as those with desirable load balancing.

If no overlap exists between Pareto-front solutions and load-balanced solutions, the resource workload is increased iteratively up to the maximum overload threshold (overload tolerance) until solutions are reached that both lie on the Pareto front (balancing objectives and constraints) and satisfy load balancing emerge. If no such solutions appear even after increasing the workload to the maximum overload threshold, then those tasks will be rejected.

The pseudocode of the proposed approach, presented in Algorithm 2, outlines the main framework for resource allocation. The optimization algorithm repeatedly executes in several iterations to improve QoS objectives. Then, the optimal allocation solutions are evaluated using availability, reliability, and throughput constraints to assess resource overload. If all solutions result in overload, resource overload tolerance is applied up to the maximum threshold to reveal possible feasible solutions. If no solution appears, the tasks are rejected.

As shown in line 1, the algorithm generates the initial set of wolves. Lines 2 to 5 evaluate the fitness function, assess the non-dominated solutions generated by the initial population, and store them in the archive. Then, as shown in lines 6 to 24, the positions of the wolves are updated until the termination condition is met, and the non-dominated solutions are stored in the archive. Line 25 computes the constraints of the obtained solutions. In lines 26 to 30, if no solution (i.e., a constraint-satisfying solution) exists and throughput exceeds the workload threshold, workload tolerance is applied up to the maximum overload threshold to reveal feasible solutions.

Lines 31 and 32 from the obtained Pareto-optimal and constraint-satisfying solutions, a preference-based ranking is applied using the weighted fitness function to select a single implementable solution. In line 34, tasks are assigned to resources based on the optimal solution. The algorithm aims to design a decision-making system that maintains QoS parameters and load balancing when allocating resources.

Algorithm 2: Pseudo-code of the MaGWO algorithm.

Input:

Parameters of GWO

Parameters of load balancing

Output:

Optimal task allocation solution

Assumption:

Input solution Si, Fitness Function, Cost, MIPS, MI, and Energy consumption.

Initialization:

the number of tasks, the number of CPU, Max_{iter} , C, A, a, and population size. n Archive

Start:

1: Generate the initial population Si, i = 1, 2,…,n randomly, between lower bound and upper bound

2: *for* all Si, do

3: Evaluate the fitness function of the population using Equation (21)

4: Find the non-dominated solutions and add to the archive

5: *end for*

6: *Repeat*

7: Set t = 0

8: *while* ($t < Max_{iter}$)

9: *for* each search agent

10: Update the position of the current search agent using Equation (8)

11: Convert continues to integer coding

12: *end for*

13: *Apply the boundary limitation*

14: Update a, C, A
15: Calculate fitness of all search agents
16: Find the non-dominated solutions
17: Update the archive according to the obtained non-dominated values
18: If archive size>nArchive
19: Select the nArchive solutions based on Crowding Distance
20: ***end***
21: Update Xα, Xβ, Xδ based on Equation (8)
22: t = t + 1
23: ***end while***
24: return archive
25: Calculate the restrictions values for solutions by equations (18), (19), and (20)
26: If not, improve solution and throughput > load-limit
27: ***while*** (load-limit<=Max-overload || archive== ∅)
28: load-limit +=α
29: Go to 19
30: ***end while***
31: Sort solutions in the order of fitness
32: return best solution Xα
33: ***end***
34: Allocate tasks to resources
35: stop

The diagram of the proposed design is illustrated in Figure 2. The GWO solves the problem through successive iterations. Using the fitness function, it can obtain better solutions at each iteration step, and the next population is selected based on improvements over the previous stage. A higher number of iterations increases the algorithm's execution time for searching and therefore increases the likelihood of finding more optimal solutions. The proposed method preserves a set of solutions across different iterations and strives to identify the best possible solutions based on QoS criteria.

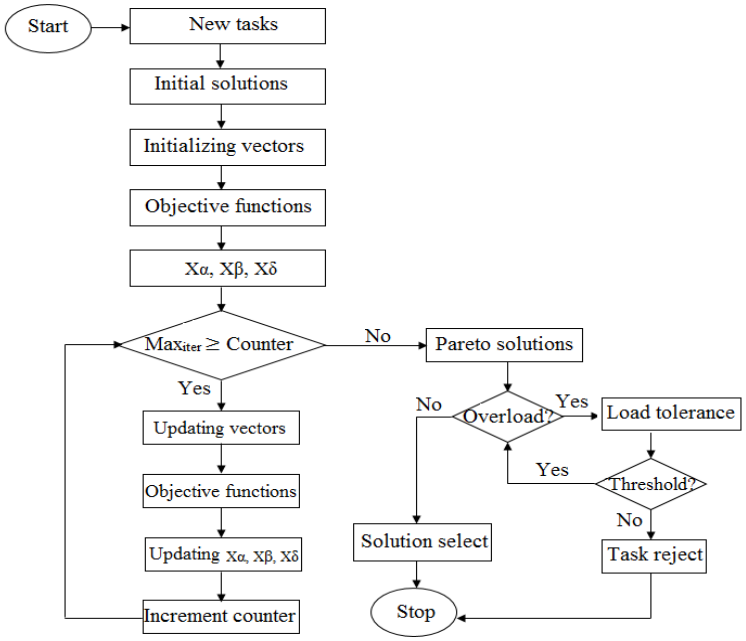

Figure 2
Diagram of the Proposed Design

The proposed framework is built upon the standard Many-Objective Grey Wolf Optimizer (MaGWO) whose core mechanisms, namely population initialization, leadership hierarchy (α,

β, δ), position updating equations, and Pareto-based dominance evaluation, are adopted from established studies in the literature. These components provide the baseline many-objective search capability.

In contrast, several key components are newly introduced in this work to address limitations observed in existing MaGWO-based QoS optimization approaches. Specifically, the proposed framework incorporates: (i) an load-tolerance mechanism that enables the system to operate under overload conditions by prioritizing feasible and admissible task allocations, (ii) a decision-level integration between many-objective optimization outputs and SDN controller constraints. These extensions collectively differentiate the proposed method from conventional MaGWO implementations and enable robust QoS provisioning under dynamic and overloaded SDN-based IoT environments.

4-3. Analysis and Problem Formulation

The many-objective optimization model includes a vector of four objective functions/parameters for resource allocation. Cost, response time, energy consumption (power usage), and resource utilization are considered the main variables and evaluation metrics.

Although the number of QoS objectives considered in this work may seem limited, their concurrent optimization fundamentally alters the problem structure. Cost, response time, energy consumption, and resource utilization capture qualitatively different and often conflicting aspects of system performance, spanning both user and provider directions. Reducing these dimensions through aggregation-based formulations requires predefined weights, which introduce subjective bias and remain highly sensitive to traffic dynamics and workload variations.

Furthermore, as the number of objectives exceeds three, Pareto dominance relationships become increasingly weak, causing a large proportion of candidate solutions to be mutually non-dominated. Under such conditions, conventional multi-objective optimization algorithms experience reduced selection pressure and impaired diversity maintenance, which in turn degrades convergence behavior and limits the effective exploration of trade-offs among QoS dimensions.

To address these challenges, the proposed framework adopts a many-objective optimization paradigm that preserves diverse Pareto-optimal solutions without prioritizing any single QoS dimension. This formulation is particularly critical under overload conditions, where conflicts between user-oriented objectives and infrastructure-level constraints intensify and cannot be adequately captured by simpler multi-objective or weighted approaches.

4-3-1. QoS Objectives

Cost: Cost represents the total amount the user pays to the service provider for using computing resources, calculated based on task execution time in dollars. Since resources have different processing speeds and task sizes vary, the cost of using different resources for processing tasks is not identical. The cost of tasks submitted by users equals the total cost users incur for task acceptance. Cost is calculated using equation (13):

$$C = \sum_{j=1}^{M} \sum_{i=1}^{Nj} \frac{MI_{i,j}}{MIPS_j} \times Pcostj \qquad (13)$$

Here, $MI_{i,j}$ is the number of million instructions for task i on resource j, and $MIPS_j$ is the number of million instructions per second for resource j. *P*costj represents the cost per second of using processing resource j, and $\frac{MI_{i,j}}{MIPS_j}$ represents the execution time of task i on resource j. N_j denotes the number of tasks on resource j, and M denotes the total number of resources.

Response Time: Response time is the duration from the start of executing the first task to the completion of the last task equivalently, from assigning the first resource to the release of the last one measured in seconds. Response time is evaluated using equation (14):

$$\text{RT}= Max_{j=1}^{M}[\frac{\sum_{i=1}^{Nj} MI_{i,j}}{MIPSj}] \quad (14)$$

The maximum total processing time of tasks on resource j is considered the response time. Here, N_j denotes the number of tasks assigned to resource j.

Energy Consumption: Efficient resource allocation considering power consumption is a key factor for service providers. Energy consumption relates to the processing capacity of resources and the size of processed tasks. Higher processing capacity and a larger number of processed tasks lead to more energy usage. To minimize energy consumption, resource management techniques must maintain a balance between energy and resource utilization. Energy consumption is considered as electrical power used, measured in watts. The energy consumed by resource j is calculated using equation (15):

$$\text{E} = \sum_{j=1}^{M} \sum_{i=1}^{Nj} \frac{MI_{i,j}}{MIPS_j} \times Wj \quad (15)$$

This is the product of the total processing time of tasks on resource j and its power consumption W_j. The total of these values represents the total energy consumption of all resources.

Resource Utilization: Resource utilization is the ratio of the busy time of a resource to the total observation time. It must be optimized to achieve effective load balancing. Resource utilization is computed using equation (16):

$$\text{CU} = \frac{\sum_{j=1}^{M} \frac{\sum_{i=1}^{Nj} MI_{i,j}}{MIPS_j} / M}{Max_{j=1}^{M}[\frac{\sum_{i=1}^{Nj} MI_{i,j}}{MIPSj}]} \quad (16)$$

This formula calculates the average execution time of tasks across resources divided by the maximum execution time, resulting in the overall resource utilization.

Normalization of QoS Parameters

Given that QoS parameters use different measurement units, the values of each must be normalized within the range 0–1. Normalization ensures that all parameters become homogeneous and comparable. Moreover, all parameters must be measured on a single unified scale so the fitness function can be evaluated. Normalization is performed using equation (17):

$$\text{Norm(Obj)} = \frac{\text{Obj}-\text{Min (Obj)}}{\text{Max(Obj)}-\text{Min(Obj)}} \quad (17)$$

4-3-2. Constraints

Considering the impact of the problem space, decision variables, and QoS objectives on optimization, constraints play a crucial role in many optimization models. These constraints directly influence task execution and, consequently, the QoS objectives. The constraints in the optimization problem are presented as follows.

Throughput: It is defined as an appropriate performance metric representing the number of tasks completed by a resource per unit of time. This metric is an important indicator for evaluating the resource overload problem and is denoted by T. A higher number of executed tasks per unit time indicates increased throughput, and a higher throughput value reflects a better performance of the resource allocation method. As throughput increases, due to improved tolerance to computational load, the task acceptance rate also increases. The calculation of throughput is expressed in Equation (18):

$$T = \frac{\sum_{j=1}^{M} MIPS_j}{\sum_{j=1}^{M}\sum_{i=1}^{Nj} MI_{i,j}} \quad (18)$$

Throughput is obtained by dividing the total MIPS of processing resources by the total MI of tasks being processed on the resources.

Accessibility: Accessibility is defined based on the successful access of tasks to resources, where resources are evaluated to accommodate task demands. This metric is denoted by A. The accessibility of each solution can be calculated by the ratio of accepted tasks to the total number of submitted tasks. The user specifies the access requirement for task execution and expects the task to be completed within that access level. Therefore, the access threshold is received as an input from the user. Accessibility for each solution is defined using Equation (19):

$$A = \frac{\sum_{j=1}^{M} Nj}{T} \times 100 \quad (19)$$

Where T is the total number of tasks submitted by the user, and N_j is the number of tasks completed on resource j.

Reliability: Reliability represents the stable conditions required for executing tasks over a given time period. It reflects the success of task execution on resources. Based on the number of tasks running on resources and the task rejection rate, the reliability of each solution is calculated. Reliability can also be used to evaluate resource accessibility and is denoted by R, expressed as a percentage. If the number of rejected tasks is represented by Υ_j, and the number of accepted tasks on resources is N_j, then reliability can be calculated using Equation (20). As N_j increases, the reliability value also increases.

$$R = 1 - \frac{\sum_{j=1}^{M} Y_j}{\sum_{j=1}^{M} Nj} \times 100 \quad (20)$$

4-3-3. Fitness Function

It is important to note that the weighted fitness function is not used during the many-objective optimization process. The MaGWO algorithm relies solely on Pareto dominance to explore and construct the Pareto front. The fitness function is applied only after obtaining the Pareto-optimal solution set, as a decision-making criterion to select a single solution for implementation.

The many-objective optimization fitness function is formulated by applying the objectives of minimizing cost, response time, and energy consumption, and maximizing resource utilization. The fitness function, by balancing the desired objectives, provides a criterion for comparing task allocation solutions to dedicated and shared resources, ultimately enabling the selection of the best solutions. In this regard, a value is assigned to each solution such that the closer this value is to zero, the better the solution. In general, the fitness function value is used for ranking solutions to support decision-making. The value of the fitness function is calculated using Equation (21).

Minimize: $F(s) = \{a \times \mathrm{Norm}(C(s)) + b \times \mathrm{Norm}(RT(s)) + c \times \mathrm{Norm}(E(s)) + d \times (\frac{1}{Norm(CU)}(s))\ for\ all\ s \in S\}$ (21)

$$s \in S\ ,\ 0 \preceq a,b,c,d \preceq 1,\ a+b+c+d=1$$

Subject to: $T_{t+1} \preceq C_T$, $A_{t+1} \succeq C_A$, $R_{t+1} \succeq C_R$

Where T, R, and A represent the throughput, reliability, and accessibility constraints of the problem, respectively, and C_T, C_A, and C_R are their corresponding threshold limits. In the fitness function equation, the four parameters, response time, energy, cost, and resource utilization, are first computed and normalized using Equations (13) to (16), as previously described. Then, by applying the priority weights (a, b, c, d), the final fitness value is obtained. The objectives are combined in the fitness function through the weights that define the priority of each

objective in the overall quality of the solution. The weights must be defined such that their sum equals 1.

4-4. Time Complexity Analysis

The time complexity analysis of the proposed approach consists of the following stages:

First stage: The time complexity of obtaining the Pareto-front solutions using the metaheuristic algorithm depends on the population size, the number of iterations, and the evaluation cost of each solution. The size of the initial random population of solutions is denoted by N, and the number of algorithm iterations by T. For each iteration of the algorithm, the following steps are examined:

- In the first iteration, evaluating the solutions takes O(N) time, during which the non-dominated solutions are stored in the archive.
- Updating the positions of the solutions in a D-dimensional space (the number of problem variables) takes O(N) time.
- In each iteration, the non-dominated solutions stored in the archive are updated. Each new solution in the next iteration is compared with the solutions already in the archive. Since the archive size is proportional to N, comparing each of the maximum N new solutions with an archive of size N requires $O(N^2)$ time per iteration. Therefore, in each iteration, the time complexity is $O(N^2 + N + N) = O(N^2)$, and over T iterations, the total time complexity becomes $O(T \times N^2)$.

Second stage: This stage evaluates the constraints on the Pareto-front solutions and selects the solution based on the fitness function. The size of the Pareto front can be as large as the population size N. Evaluating the constraints for N Pareto solutions requires O(N) time. Then, the fitness function is evaluated for the N solutions to select the optimal solution, which also takes O(N) time. Therefore, the total time complexity is O(N) + O(N) = O(N).

Third stage: If no feasible solution appears, the flexibility of the throughput constraint is applied, requiring the re-evaluation of the constraints and the solution selection process. Let K denote the possible incremental adjustments of the constraint up to its maximum threshold. For each increment in K, up to N solutions must be examined. Since K is assumed to be constant, evaluating the accessibility constraint takes O(N), the reliability constraint takes O(N), and evaluating the throughput constraint at K levels takes $O(K \times N)$. Therefore, the total time complexity becomes $O(K \times N + N + N) = O(K \times N)$. Finally, the total time complexity of the proposed approach is:

$$O(T \times N^2) + O(N) + O(K \times N) = O(T \times N^2)$$

5. Simulation and Evaluation of the Proposed Approach

This section describes the experimental setup, performance metrics, and experimental results. The discussion of results includes the evaluation and comparison of the proposed approach based on the MaGWO with the identified many-objective metaheuristic algorithms.

5-1. Experimental Configuration

The proposed approach was simulated and evaluated in the MATLAB R2022b environment. This environment supports the implementation of most multi-objective and many-objective algorithms. The experiments were simulated on a machine equipped with an Intel (R) Core (TM) i7 CPU, 2.8 GHz, 16 GB RAM, and a 1 TB SSD running a 64-bit Windows 11 operating system.

MATLAB was selected as the simulation environment due to its flexibility in implementing many-objective evolutionary algorithms and its extensive support for numerical analysis. The

use of synthetic workloads allows precise control over task arrival rates, sizes, and resource characteristics, ensuring fair and reproducible comparisons across all evaluated algorithms.

5-2. Simulation Parameters

Given the use of MATLAB and the GWO, this section presents the parameters used in each of these components. In the experiments, a data center consisting of multiple computing resources was defined. Each of these resources is capable of being shared among multiple tasks. The more powerful the hardware (MIPS) of the resources, the higher the access cost and energy consumption assigned by the service provider.

Computing Resource Parameters: The software specifications of the resources based on Google servers are provided in Table 3 (Amazon, 2024). During the task allocation process, the number of tasks assigned to each resource varies, resulting in different resource loads.

Table 3
Technical Specifications of Resources

Value	Details	Task Type
8	Number of resources	Small
1000-5000	Processing power	
5-10	Cost (USD)	
5.5-25	Energy (Watt)	
10-80	Input task rate	
100-4000	Task size (MI)	
1	Number of controllers	
19	Number of resources	Medium
1500-12000	Processing power	
8.5-11	Cost (USD)	
6-27	Energy (Watt)	
100-700	Input task rate	
300-5500	Task size (MI)	
1	Number of controllers	
25	Number of resources	Large
2000-17000	Processing power	
6.5-15	Cost (USD)	
6.2-29	Energy (Watt)	
1000-1600	Input task rate	
350-6200	Task size (MI)	
1	Number of controllers	

The ranges of task sizes, arrival rates, processing capacities, and energy consumption parameters are derived from commonly adopted cloud and IoT configurations reported in prior studies, ensuring realistic workload modeling.

GWO Parameters: Table 4 presents the simulation parameters of the many-objective GWO algorithm (Khan & ur Rasool, 2024).

Table 4
GWO Parameters

Parameters	Value
Population size (number of solutions)	100
Maximum number of iterations	200
Archive size	100
C_1, C_2	1.49445
r_1, r_2	Random numbers between 0 and 1
Number of objectives	4

Number of constraints	3
Number of resources (processors)	8-25
Number of controllers	1
Types of workloads	Small, Medium, Large
Resource overload threshold	0.90

To ensure reproducibility, all experiments were conducted using fixed and explicitly defined parameter settings. The population size and maximum number of iterations were selected based on preliminary experiments, and were kept constant across all comparative methods. The control parameters C_1 and C_2 were set according to commonly adopted values in the GWO literature.

Each experiment was independently executed 200 times to mitigate the effect of stochastic variations. A fixed random seed was used for each run to ensure fair comparison across algorithms, while different seeds were applied across runs to capture performance variability. All reported results correspond to the average values obtained over these independent executions.

5-3. Simulation Results and Analysis

The results obtained from the Many-Objective Gray Wolf Optimization (MaGWO) algorithm were evaluated and compared with the many-objective Genetic Algorithm (NSGA-III) and Particle Swarm Optimization (MaPSO) algorithms, considering their prevalence in research and their Pareto-front-based approach. Subsequently, to measure the algorithm's performance, the scenario in Table 5 was implemented in the Genetic Algorithm, Particle Swarm Optimization, and the proposed GWO algorithm.

Table 5
Proposed Scenario for Algorithm Evaluation

Data Type	Number of Infrastructure Providers	Number of Service Providers	Number of Tasks
10-1600	4-9	8-25	Synthetic

Table 6 lists the number of accepted and rejected tasks in two cases with and without load tolerance.

Table 6
The Number of Accepted and Rejected Tasks

Number of tasks input	Normal		Load tolerance	
	Allocated	Rejected	Allocated	Rejected
20	18	2	18	2
30	25	5	26	4
40	33	7	34	6
50	37	13	38	12
60	41	19	44	16
70	52	18	55	15
80	58	22	63	17

Figure 3 exhibits the experimental results of the comparison of the acceptance rate of tasks according to Table 6. The load-balancing strategy based on load tolerance causes more tasks to be accepted than load-balancing without load tolerance.

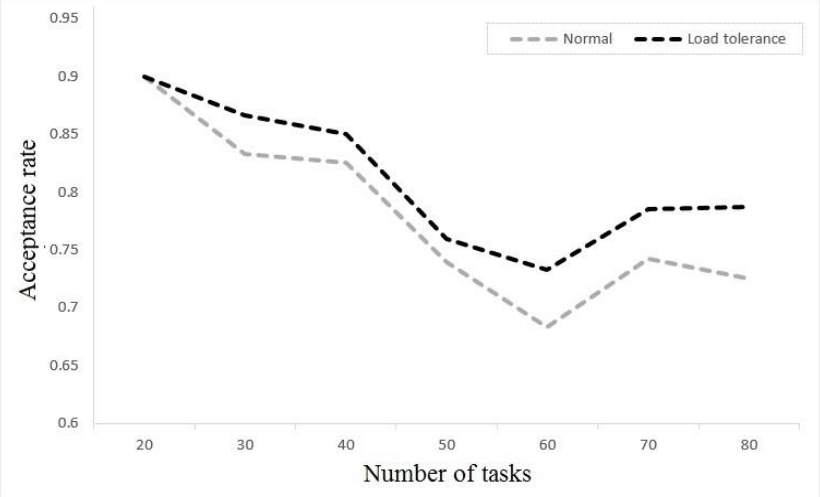

Figure 3
Tasks Acceptance Rate

Although several compared methods optimize QoS metrics, most of them rely on either objective functions or limited multi-objective formulations. Such approaches assume trade-offs among objectives, which become ineffective under overloaded or highly dynamic workload conditions. In contrast, the proposed many-objective formulation maintains a diverse set of non-dominated solutions, enabling the SDN controller to select feasible task allocations even when certain QoS objectives cannot be simultaneously satisfied. This characteristic is particularly evident under overload scenarios, where aggregation-based approaches tend to converge to solutions that violate load constraints, resulting in lower task acceptance rates.

The experimental results demonstrate that preserving Pareto diversity across multiple QoS dimensions allows the proposed MaGWO-based framework to achieve higher robustness compared to simpler formulations. This is reflected in the consistently higher task acceptance rate and balanced QoS performance observed in overloaded conditions.

Figure 4 examines the energy consumption for each algorithm. The number of tasks is categorized into small, medium, and large types. The results indicate that the proposed method consumes less energy compared to the Genetic Algorithm and Particle Swarm Optimization. The reduction in energy consumption in the proposed approach is due to optimal resource selection and better decision-making by the MaGWO algorithm. (100× Energy)

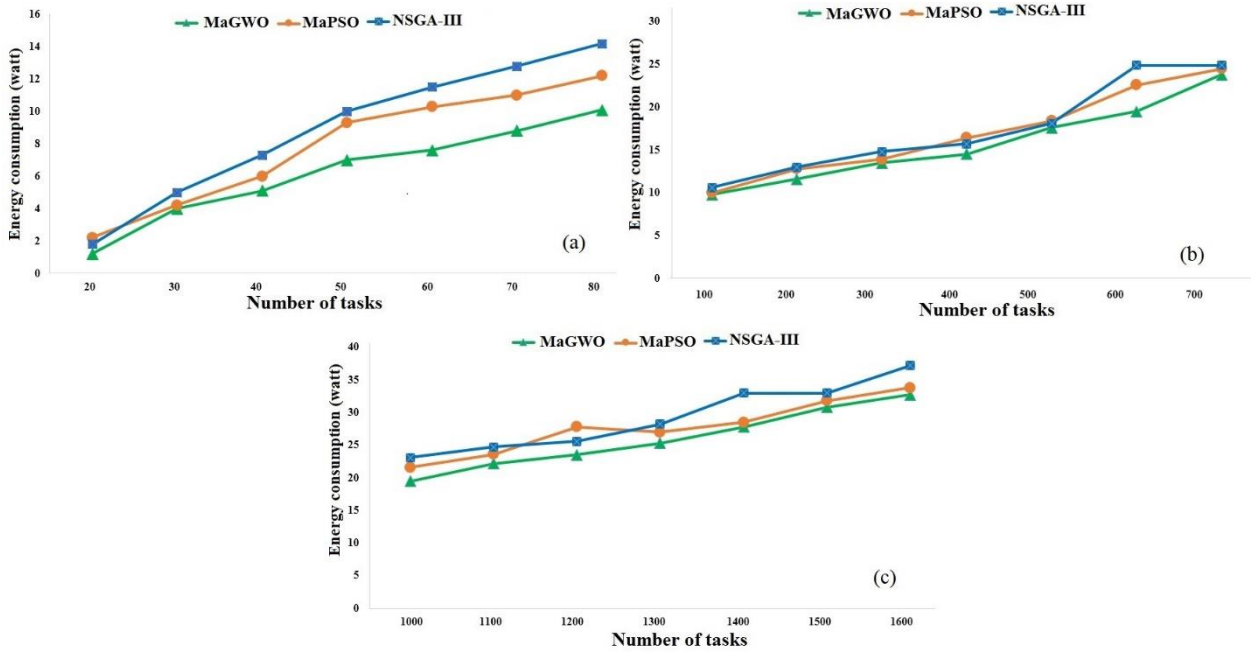

Figure 4
Evaluation of Proposed Algorithm Based on Energy Consumption for Different Task Types:
a- Number of Small-Sized Tasks; b- Number of Medium-Sized Tasks; c- Number of Large-Sized Tasks

Figure 5 illustrates the response time across the three algorithms under study. For this purpose, the response time of each task is calculated to obtain the total time spent processing the tasks. It can be observed that the GWO algorithm achieves better results compared to the other algorithms, allowing tasks to receive processing service in a shorter response time. The above chart demonstrates the superior performance of the proposed algorithm with respect to the response time parameter.

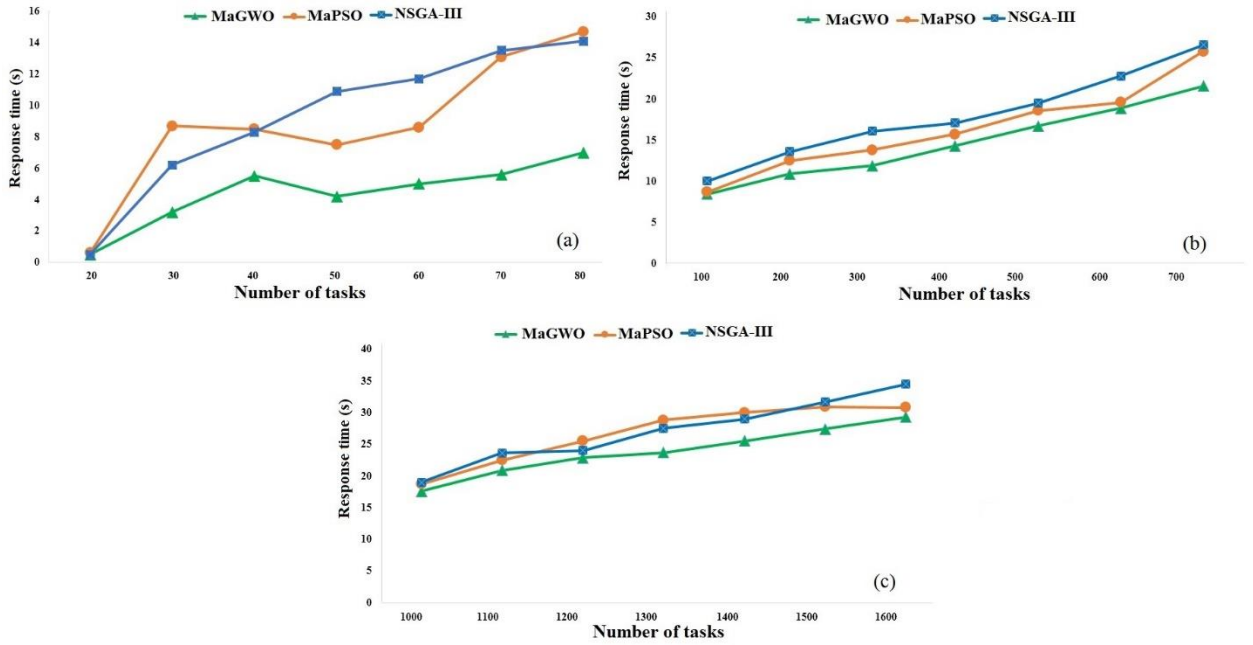

Figure 5
Evaluation of Proposed Approach Based on Response Time for Different Task Types:
a- Number of Small-Sized Tasks; b- Number of Medium-Sized Tasks; c- Number of Large-Sized Tasks

In the proposed method, by distributing tasks among resources, it can be observed that increasing resource utilization leads to shorter response times and greater user satisfaction. The results can be clearly observed in Figure 6.

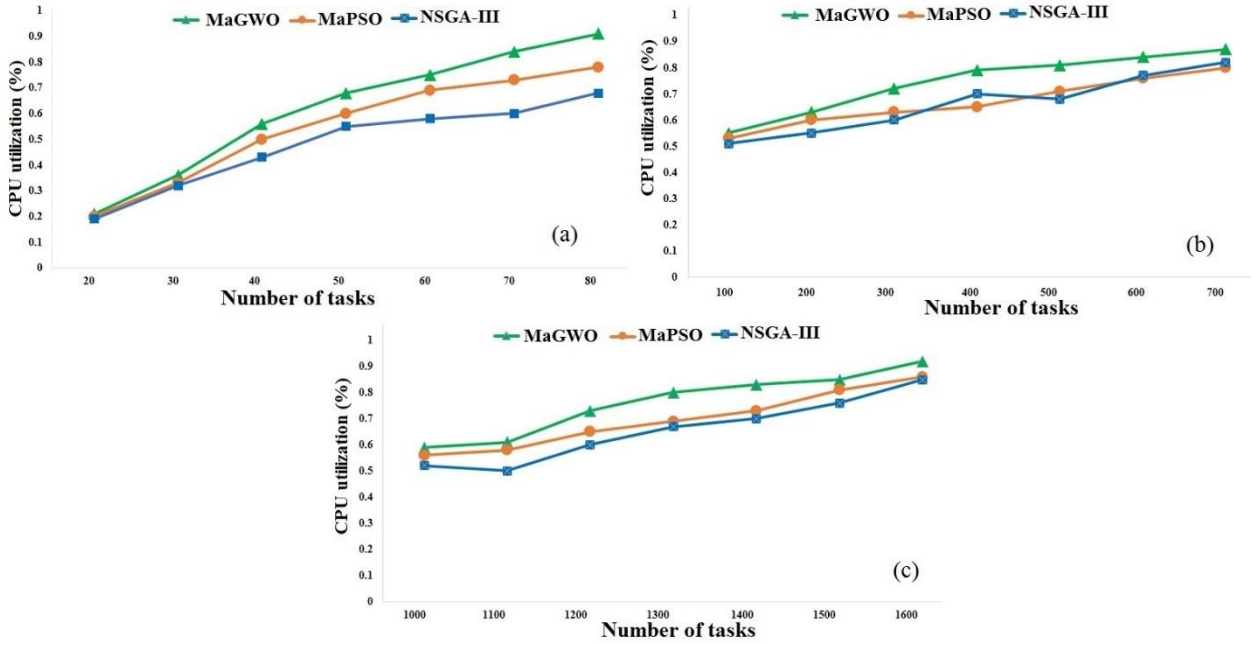

Figure 6
Comparative Results Based on Resource Utilization for Different Task Types:
a- Number of Small-Sized Tasks; b- Number of Medium-Sized Tasks; c- Number of Large-Sized Tasks

Figure 7 shows that the simulation results of the GWO algorithm achieve optimal performance in cost reduction compared to other algorithms, successfully reducing costs more than the other methods. The cost evaluation metric refers to the amount paid by the user for each resource, calculated based on the amount of resource utilized. (10 × Cost)

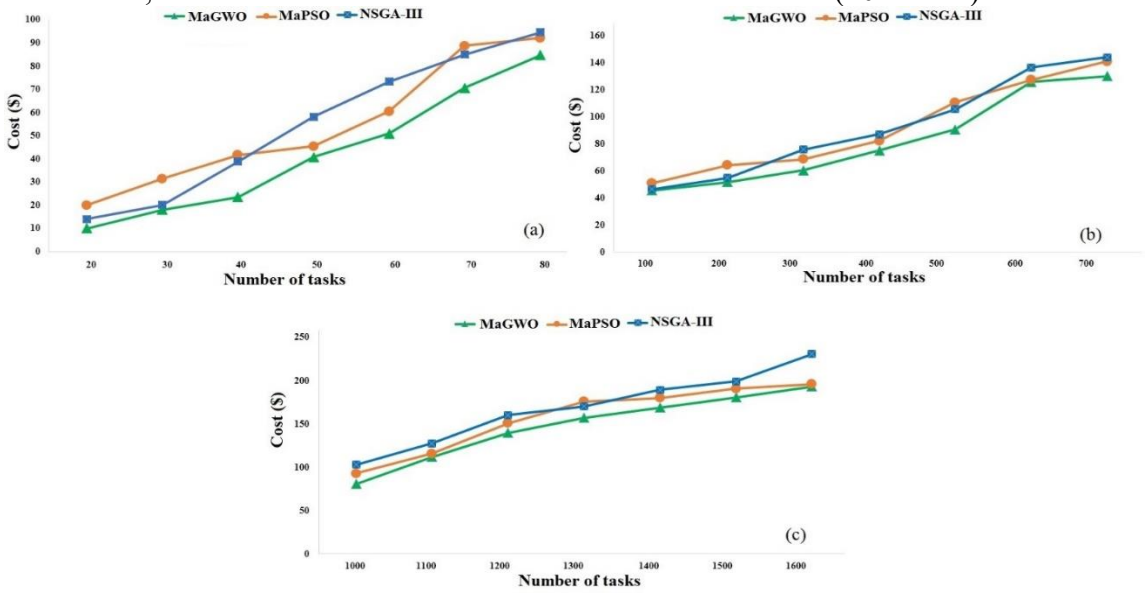

Figure 7
Task Execution Costs Based on Task Types:a- Number of Small-Sized Tasks; b- Number of Medium-Sized Tasks; c- Number of Large-Sized Tasks

The results indicate that the proposed algorithm incurs lower costs compared to the many-objective Genetic Algorithm and many-objective Particle Swarm Optimization. For the convergence test, the above algorithms were compared over 200 iterations. Figure 7 represents the convergence test over 200 iterations. The experiments demonstrate that the proposed model effectively balances the load by optimizing four objectives while maintaining the service level agreement. The performance of the proposed algorithm was compared with the many-objective Genetic Algorithm and many-objective Particle Swarm Optimization, achieving performance improvement under specified workload conditions. In summary, the simulation results of the proposed approach, with enhanced service quality and constraint compliance for fair resource utilization, outperform other algorithms.

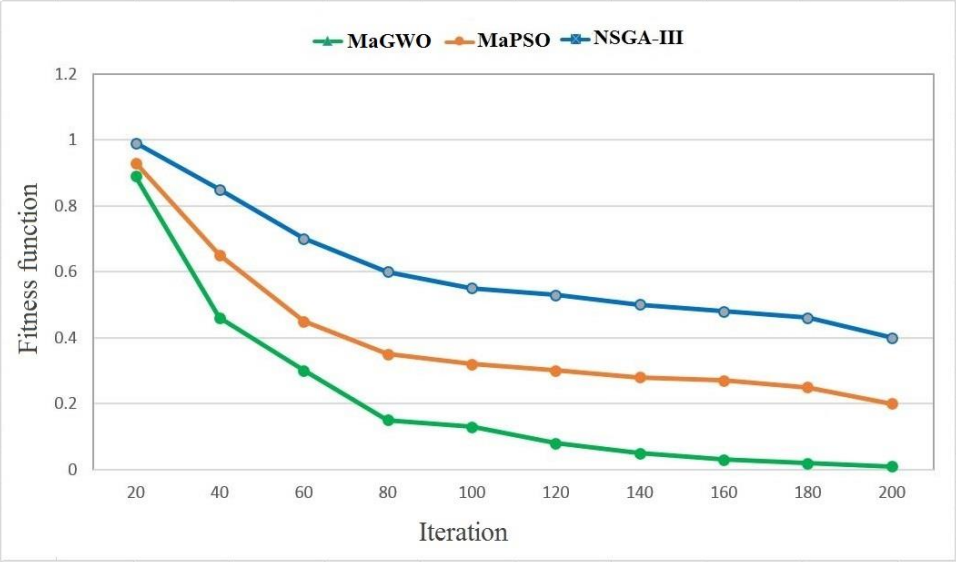

Figure 7

Convergence Chart Over 200 Iterations

The NSGA-III algorithm, with a large population size, has high computational complexity, which can lead to slow performance in large-scale problems. Additionally, NSGA-III may not perform well when conflicting objectives exist and often requires specific parameter tuning. The MaPSO algorithm is typically dependent on a fixed population and may get trapped in local optima if sufficient diversity is lacking. Moreover, MaPSO is sensitive to design parameters, and incorrect parameter selection can negatively affect its performance. The performance of NSGA-III and MaPSO in achieving convergence and maintaining Pareto-front-based solution diversity is a key consideration (Arya et al., 2024).

The MaGWO is simpler and faster to implement compared to NSGA-III and MaPSO. Due to its inherent nature, it has a strong ability to avoid getting trapped in local optima. The GWO algorithm generally performs better in complex many-objective problems and can adapt effectively to dynamic and changing conditions. It also provides a better balance between convergence and diversity in Pareto-front-based many-objective optimization problems.

Based on the obtained results, the MaGWO can effectively enhance the performance of the proposed model and achieve an optimal resource-to-task allocation strategy.

To statistically evaluate the solutions obtained from the proposed method and other optimizers, the Wilcoxon test was used. Given the non-parametric conditions due to the non-normal distribution of data and the ordinal nature of the data, the Wilcoxon test is applied. In the null hypothesis, it is assumed that there is no significant difference between the solutions obtained from the proposed optimizer and other optimizers. A significance level greater than

0.05 (5%) supports the null hypothesis, whereas a significance level less than 5% rejects it. Since the statistical tests were performed separately for each performance metric and algorithm pair, no multiple-comparison correction was applied.

The p-value presented in Table 7 indicates that the proposed algorithm achieved a significant improvement with a p-value of 0.00217. Furthermore, the improvement achieved by the MaGWO compared to the many-objective Genetic Algorithm and Particle Swarm Optimization was substantial. The null hypothesis for the solutions is rejected at the default 5% significance level. This indicates that the proposed approach effectively manages workload while satisfying service quality requirements.

To statistically validate the performance improvements, the Wilcoxon signed-rank test was employed due to the non-normal distribution of the simulation results. Each algorithm was executed multiple independent times under identical experimental conditions, and the resulting QoS metrics (response time, energy consumption, cost, and resource utilization) were collected. Pairwise statistical comparisons were conducted between the proposed method and each baseline algorithm for each performance metric.

Table 7
Wilcoxon Signed-Rank Test P-Values for Pairwise Comparison of QoS Metrics Between the Proposed Method and Other Algorithms

Comparison	P-values
MaGWO vs. MaPSO	0.00217
MaGWO vs. NSGA-III	0.00139

Unlike recent GWO-based and hybrid approaches that assume the existence of feasible Pareto solutions under fixed constraints, the proposed method maintains solution availability through workload tolerance, which directly contributes to higher task acceptance rates.

6. Conclusion

The QoS and load balancing in IoT environments can be formulated as many-objective optimization problems, as they involve the simultaneous satisfaction of multiple, often conflicting, user and service provider requirements. In this work, an SDN-enabled load-tolerant resource allocation framework was proposed, in which response time and cost are minimized for users, while energy consumption is minimized and resource utilization is maximized for service providers through a many-objective formulation.

To address this problem, a Pareto-based many-objective optimization method built upon the Grey Wolf Optimizer was developed. The core optimization process remains strictly Pareto-driven, while a weighted fitness function is employed solely as a post-Pareto decision-making mechanism to select admissible solutions. This design enables the effective integration of many-objective optimization outcomes into real-time SDN-based IoT control decisions.

While many-objective optimization typically produces large Pareto fronts that are difficult to interpret and utilize in practice, the proposed post-Pareto decision-making strategy facilitates the selection of operationally feasible solutions without compromising the diversity of trade-offs captured during optimization.

The proposed approach demonstrates how a Pareto-based MaGWO can be embedded within a load-tolerant resource allocation framework, thereby bridging the gap between many-objective optimization and practical SDN-based IoT deployment. Simulation results indicate improvements of 13.85% in resource allocation cost, a 17.2% reduction in response time, a 15.8% reduction in energy consumption, and a 10.25% improvement in resource utilization. Furthermore, the proposed load-tolerance mechanism increased the task acceptance rate by 8.4% compared to other algorithms under overloaded conditions.

Despite these promising results, the proposed approach has several limitations. As a metaheuristic-based optimization method, the framework incurs additional computational overhead due to iterative population-based search, which may affect scalability in large-scale or latency-critical deployments. Moreover, the performance of the algorithm depends on parameter settings that were fixed in this study and not dynamically adapted, which may limit robustness under highly dynamic workload conditions. In addition, handling and interpreting large Pareto sets in many-objective scenarios may increase decision complexity for SDN controllers, highlighting the importance of efficient post-optimization selection mechanisms. The experimental evaluation was conducted in a simulation environment, and real-world deployment aspects, such as controller failures, network dynamics, and real-time execution constraints, were not explicitly considered. Addressing these limitations forms an important direction for future research.

Based on the limitations identified in this study, several concrete directions for future work can be outlined. First, the proposed approach will be evaluated under real-world workloads to strengthen empirical validation beyond synthetic simulations. This includes assessing performance under highly dynamic traffic patterns and heterogeneous resource configurations.

Second, integrating the proposed framework into SDN controllers will enable the evaluation of runtime overhead, decision latency, and scalability under operational constraints.

Finally, extending the framework toward distributed and multi-controller SDN architectures represents a promising research direction. As network scale increases, centralized controllers may become performance bottlenecks, whereas distributed controller designs can improve scalability, fault tolerance, and QoS provisioning. Adapting the proposed many-objective load-tolerant optimization framework to such architectures will be investigated in future studies.

References

Alsadie, D. (2021). TSMGWO: Optimizing task schedule using multi-objectives grey Wolf Optimizer for cloud data centers. *IEEE Access, 9*, 37707-37725.

Amazon (2024). https://aws.amazon.com/ec2/instance-types/

Arya, A., Gunarani, G. I., Rathinakumar, V., Sharma, A., Pati, A. K., & Sethi, K. C. (2024). NSGA-III based optimization model for balancing time, cost, and quality in resource-constrained retrofitting projects. *Asian Journal of Civil Engineering, 25*(7), 5613-5625.

Badr, S. A., Gamal, M., Ali, K. A. E., & Abdel-Kader, R. F. (2024). Multi-objective improved particle swarm optimization for efficient offloading algorithm in fog-cloud collaboration. *Suez Canal Engineering, Energy and Environmental Science, 2*(2), 17-26.

Balicki, J. (2022). Many-Objective quantum-inspired particle swarm optimization algorithm for placement of virtual machines in smart computing cloud. *Entropy, 24*(1), 58.

Belgacem, A. (2022). Dynamic resource allocation in cloud computing: analysis and taxonomies. *Computing, 104*(3), 681-710.

Cao, B., Fu, Y., Sun, Z., Liu, X., He, H., & Lv, Z. (2021, October). A resource allocation strategy in fog-cloud computing towards the Internet of Things in the 5G era. In *2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (pp. 1-6). IEEE.

Cao, B., Sun, Z., Zhang, J., & Gu, Y. (2021). Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing. *IEEE Transactions on Intelligent Transportation Systems, 22*(6), 3832- 3840.

Figueiredo, E. M., Ludermir, T. B., & Bastos-Filho, C. J. (2016). Objective particle swarm optimization. *Information Sciences, 374*, 115-134.

Gohil, B. N., & Patel, D. R. (2022). Load balancing in cloud using improved gray wolf optimizer. *Concurrency and Computation: Practice and Experience, 34*(11), e6888.

Hashemi, M., Javaheri, D., Sabbagh, P., Arandian, B., & Abnoosian, K. (2021). A multi-objective method for virtual machines allocation in cloud data centres using an improved grey wolf optimization algorithm. *IET Communications, 15*(18), 2342-2353.

Hosseinzadeh, M., Haider, A., Rahmani, A. M., Gharehchopogh, F. S., Rajabi, S., Khoshvaght, P., ... Lee, S. W. (2025). SDN-Based NFV deployment for multi-objective resource allocation in edge computing: A deep reinforcement learning for iot workload scheduling. *Sustainable Computing: Informatics and Systems, 48*, 101218.

Hussaini, S. M., Razak, T. A., & Jamil, M. A. (2024). Multi-Objective evolutionary algorithm to optimize IoT based scheduling problem using (NSGA-II algorithm). *Journal of Intelligent Systems & Internet of Things, 12*(2).

Imene, L., Sihem, S., Okba, K., & Mohamed, B. (2022). A third generation genetic algorithm NSGAIII for task scheduling in cloud computing. *Journal of King Saud University-Computer and Information Sciences, 34*(9), 7515-7529.

Keshari, S. K., Kansal, V., & Kumar, S. (2021). A cluster based intelligent method to manage load of controllers in SDN-IoT networks for smart cities. *Scalable Computing: Practice and Experience, 22*(4), 247-257.

Khan, M. A., & ur Rasool, R. (2024). A multi-objective grey-wolf optimization based approach for scheduling on cloud platforms. *Journal of Parallel and Distributed Computing, 187*, 104847.

Makhadmeh, S. N., Al-Betar, M. A., Doush, I. A., Awadallah, M. A., Kassaymeh, S., Mirjalili, S., & Zitar, R. A. (2023). Recent advances in Grey Wolf Optimizer, its versions and applications. *IEEE Access, 12*, 22991-23028.

Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. D. S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106-119.

Mohammadi, R., Akleylek, S., & Ghaffari, A. (2023). SDN-IoT: SDN-based efficient clustering scheme for IoT using improved Sailfish optimization algorithm. *PeerJ Computer Science, 9*, e1424.

Nain, A., Sheikh, S., & Shahid, M. (2025). An efficient load distribution approach for optimizing resources in SDN-Based edge computing environment. *Concurrency and Computation: Practice and Experience, 37(*12-14), e70113.

Ramesh, D., Kolla, S. S., Naik, D., & Narvaneni, R. (2025). HGWO-MultiQoS: A hybrid grey wolf optimization approach for QoS-constrained workflow scheduling in IaaS clouds. *Simulation Modelling Practice and Theory, 142*, 103127.

Rostami, M., & Goli-Bidgoli, S. (2024). An overview of QoS-aware load balancing techniques in SDN-based IoT networks. *Journal of Cloud Computing, 13*(1), 89.

Saif, F. A., Latip, R., Hanapi, Z. M., & Shafinah, K. (2023). Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. *IEEE Access, 11*, 20635-20646.

Sandanasamy, A., & Charles, P. J. (2025). Dynamic load balancing through TOPSIS based optimal server selection and resource allocation in SDN IoT network. *OPSEARCH*, 1-24.

Sefati, S., Mousavinasab, M., & Zareh Farkhady, R. (2022). Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: Performance evaluation. *The Journal of Supercomputing, 78*(1), 18-42.

Singh, G., & Chaturvedi, A. K. (2024). Hybrid modified particle swarm optimization with genetic algorithm (GA) based workflow scheduling in cloud-fog environment for multi-objective optimization. *Cluster Computing, 27*(2),1947-1964.

Singh, S. P., Kumar, G., Ahirwar, U., Selvarajan, S., & Khan, F. (2025). Multi-objective quantum hybrid evolutionary algorithms for enhancing quality-of-service in internet of things. *Scientific Reports, 15*(1), 1-27.

Tyagi, V., Singh, S., Wu, H., & Gill, S. S. (2024). Load balancing in sdn-enabled wsns toward 6g ioe: Partial cluster migration approach. *IEEE Internet of Things Journal, 11*(18), 29557-29568.