TRUST-AWARE MULTI-OBJECTIVE RESOURCE PROVISIONING IN FOG COMPUTING USING NSGA-II: A COMPARATIVE OPTIMIZATION FRAMEWORK

[1]**Kirandeep Kaur,** [2]Punjabi University, Patiala, India,147002
[2]**Arjan Singh,** Punjabi University, Patiala, India,147002
[3]**Anju Sharma,** Punjab State Aeronautical College, Patiala, India, 147007

Abstract

The rapid growth of the Internet of Things (IoT) has made it necessary to create efficient and secure resource management frameworks in situations that use fog computing. Using the Non- Dominated Sorting Genetic Algorithm II (NSGA-II), this research proposes a trust-based re- source provisioning strategy to overcome the difficulties of dynamic resource availability, security vulnerabilities, and trust management in decentralized fog networks. The proposed model optimizes key performance metrics such as latency, trust level, and energy efficiency while balancing security and system performance. An adaptive trust-driven approach dynamically adapts resource distribution to current network circumstances, enhancing security and reliability. Simulation results demonstrate that the NSGA-II-based model outperforms conventional optimization techniques like Artificial Bee Colony (ABC), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) in terms of decreased computing time and energy consumption, making it a viable solution for efficient fog computing resource management.

Keywords: Fog computing, NSGA-II, Trust-based resource provisioning, IoT, Multi-objective optimization, Energy efficiency, Latency optimization, Decentralized resource management.

1. INTRODUCTION AND BACKGROUND

The growing adoption of the Internet of Things (IoT) has necessitated the development of computing paradigms that support low latency, scalability, and real-time processing. Fog computing has emerged as a promising solution, bridging the gap between cloud data centers and end devices. However, challenges in efficient resource provisioning, especially under multiple conflicting objectives such as energy consumption, trustworthiness, and service delay, remain unsolved. This paper addresses these issues by proposing a trust-aware, multi-objective framework using the NSGA-II algorithm to optimize task allocation in fog environments. The Internet of Things (IoT) is becoming increasingly prevalent daily. Each application domain customizes IoT services to automate operations, equipment, and daily living. Smart homes, offices, grids, classrooms, and cities integrate individuals with IoT gadgets. IoT-integrated gadgets, such as automobiles, surveillance cameras, smartwatches, and smartphones, are integral to our daily lives. Utilizing IoT devices inside a domain or network generates substantial data transmission. It presents multiple challenges, including data storage, management, and security. Since IoT devices operate in real-time, the data must also be real-time. The user desires immediate results for his enquiries. It facilitates the evolution of edge and fog computing. Information has been gathered until October 2023[1][2]. An addition to cloud computing is fog computing, which helps latency-sensitive applications through data processing done closer to the source [3]. It maximizes efficiency in IoT networks by allocating processing resources at the edge sites, optimizing bandwidth usage, and boosting the response time [4]. Nevertheless, with such benefits, the provisioning of fog computing is heavily challenged by changing resource availability, vulnerability in security, and trust issues [5]. Therefore, trust-based techniques are essential in assuring reliable service delivery, removing harmful node behaviors, and enabling a more appropriate resource allocation between fog nodes [6][7]. Unlike traditional cloud computing, which allocates resources centrally and often results in soaring latencies and possible security risks [8], fog computing relies on a decentralized method, requiring the implementation of clever trust evaluation mechanisms to establish secure interactions among nodes [9]. While previous research has touched on trust-based re- source allocation approaches, most methods ignore multi-objective optimization criteria in pursuing meaningful decision-making [10] [11]. Fog computing environments display heterogeneity, making formulating a standardized trust model capable of addressing varying infrastructure and security requirements extremely challenging [12].

The Trust in fog computation is separated into two categories: direct Trust and indirect Trust. Indirect Trust is founded on experiences and human interactions, while direct Trust is based on recommendations and evaluations provided by third parties [13]. The synergy between these trust factors will enhance the resource allocation decision-making to achieve security and optimal distribution of computing resources [14]. An efficient trust-based resource provisioning model should also include adversarial strategies such as Sybil, collusion and defamation, which can discredit the system's reliability [15]. Further compounding the trust assessment in fog

computing is the network environment's dynamic character. Network nodes come and go. Very often, thereby causing changes in trust scores [16]. An efficient trust model should have mechanisms for adaptation to consistently modify its trust levels based on the behavior of nodes. This will guarantee that the trust model can recognize dangerous or unreliable nodes, preventing them from accessing critical resources [17].
Making security and performance trade-offs is one of the key challenges. An overly strict trust model would inhibit resource accessibility and incur poor resource utilization and delay. Excessively relaxed trust models would expose the system to vulnerabilities. These issues must be addressed via sophisticated decision-making frameworks such as NSGA-II, which allows for multi-criteria optimization that takes security and performance constraints [18]. This re- search addresses trust-based resource provisioning issues in the fog computing environment using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). One of the well-known multi-objective optimization methods is NSGA-II. Efficiently balance conflicting goals for minimizing response time and maximizing trust levels during resource allocation [19].
This improves the stability of the system and resource optimization in a decentralized environment. The study's primary contributions include as follows,

- To create a foundation of trust resource provisioning model using NSGA-II for secure and efficient distribution of resources in fog computing settings.
- To evaluate the suggested model, optimize key performance metrics such as latency, trust level, and energy efficiency while balancing security and system performance.
- To design an adaptive trust-driven resource management approach that dynamically modifies resource distribution according to current network circumstances, improving security and system reliability.

2.LITERATURE SURVEY

2.1 Trust Models in Fog Computing

Fog environments are inherently distributed, making them susceptible to malicious or underperforming nodes. Trust-based models help mitigate such issues by evaluating nodes on parameters like historical reliability, response time, and user feedback.

2.2 Resource Provisioning Strategies

Existing provisioning strategies largely focus on single-objective metrics like minimizing latency or energy. However, few incorporate trust, and fewer still apply multi-objective strategies that consider the trade-offs among delay, energy, and reliability.

2.3 Optimization Algorithms in Fog Computing

Metaheuristic algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have been applied in fog computing. NSGA-II is efficient for handling multi-objective optimization problems and is well-suited to dynamic fog environments.
Muhammad et al. [20] proposed a VFC-enabled framework called reputation-based prioritization and resource allocation (RPRA) for smart healthcare IoT devices that are time-sensitive and computationally resource-constrained. RPRA uses context-dependent, indirect, and perceived reputation-based Trust to minimize trust bias and attacks. It uses three key metrics: individual perception, peripheral perception, and real-time perception. In a dynamic healthcare system, VFC allows the method to perform better in utility-based reputation values., with the highest range and minimal algorithmic complexity. [21] focuses on building a network of re- source providers to assign jobs to actual fog nodes with the necessary resources. The method involves selecting appropriate options, identifying reliable nodes based on trust mechanisms, and dividing tasks among dependable gadgets with enough power. The assessment of the OMNET++ simulator shows that trust mechanisms can reduce task completion time. Niloofar et al. [22] propose A novel trust model for fog computing (FC) that combines learning automata (LA) with genetic algorithms (GA). LA enhances the crossover operator in GA, and the ideal crossover operator value affects the problem's solutions. Compared to GA and particle swarm optimization (PSO) algorithms, the method's performance demonstrated improved reliability rate, energy consumption, and latency when tested in a MATLAB 2020 simulation environment. Trust is essential for ensuring security and maintaining service quality. Zhang et al. [23] presented a multi-layered intermittent neural network model to enhance fog computing security, especially for nearby IoT devices and end users. It reduces privacy threats by using the NSLKDD dataset. The model's stability and robustness are demonstrated through research and replication studies, offering a promising solution for cloud-based computing.

A. Performance Analysis of NSGA-II with Trust versus Other Algorithms

Carlos Guerrero et al. [24] presented three dispersed versions of a genetic algorithm (GA) for fog computing resource optimization environments. The designs address constrained resources and the geographical distribution of devices. The results show that the lowest distribution degree design achieves comparable solution quality but a higher network load. The second design lowers solution diversity but lowers network overhead. The third design, with a distributed population, minimizes network traffic without sacrificing the quality of the solution. Hussein et al. [25] suggest a hybrid meta-heuristic optimization method to handle resource allocation in IoT-Fog scenarios by fusing the NSGA-II and MOGWO algorithms. With the algorithm, QoS is improved. Metrics reduce network resource utilization and enhance application performance. Comparative evaluations show the algorithm's superiority, which is implemented using the Python fog simulator YAFS. Usha et al. [26] suggested a novel method for fog computing optimization based on the Artificial Bee Colony (ABC) of the JAYA algorithm (ABCJAYA). The algorithm aims to optimize key parameters like throughput, energy consumption, and re-source usage depending on latency in the computerized fog computing environment based on the Internet of Things. Simulations and comparisons with Modern optimization techniques demonstrate that the suggested plan may enhance resource allocation, system efficiency, and resource utilization, addressing the Internet of Things problem.

Ahmed et al., [27] introduces CyberGuard, a framework that integrates machine learning with blockchain for dependable fog and edge computing trust management. It uses the decentralization and immutability of blockchain technology to create an open network for tracking and verifying transactions. CyberGuard improves resource allocation efficiency and system performance, with an impressive F1-score of 98.18%, recall, accuracy, and precision, indicating its transformational potential in these environments. Javad et al. [28] introduced a multi-objective optimization method for dynamically placing services in fog computing environments based on containers. NSGA-II is used in a two-tier Kubernetes resource management system to balance competing goals and guarantee the best possible service placement choices. Shariar et al. [29] present A multi-criteria optimization technique for scheduling workflows, using the meta- heuristic NSGAII algorithm for the first stage and an innovative technique for assigning virtual machines to tasks. Experiments on both synthetic and actual datasets assessed the effectiveness of the suggested approach. According to simulation data, the strategy improved GD, IGD, and GPA by an average of 2.3%, 5.7%, and 9.8%, while reducing makespan by 6.38% and energy usage by 3.52%. Criteria. Mina Mohammadi et al. [30] Examine resource allocation in device- to-device fog computing systems focusing on security. A new multi-objective function is suggested to maximize energy savings, delay, and security breach costs. The modified NSGA-II algorithm, incorporating Sigma Scaling, substantially increases over existing methods, improving performance by 30.15% compared to the conventional version. The following table high- lights and analyses relevant research, including algorithms, meaningful performance measures, and comparisons to NSGA-II, a standard optimization approach:

Table 1: Performance Analysis of NSGA-II with Trust versus Other Algorithms

Author & Reference	**Algorithm Used**	**Performance Metrics**	**Results**
Avasalcai & Dustdar [30]	Whale Optimization Algorithm (WOA)	Latency, Energy Efficiency, Reliability	NSGA-II outperforms existing algorithms, reducing latency by 20%, achieving 15% energy savings, and improving reliability by 10%.
Madan et al. [31]	Genetic Algorithm (GA)	Latency, Resource Utilization	NSGA-II exhibits 30% lower latency and 12% better resource utilization than GA.
Gowri & Shanthi Bala [32]	Particle Swarm Optimization (PSO)	Energy Consumption, Load Balancing	NSGA-II demonstrates 25% lower energy consumption and superior load balancing.
Murthy & Shiva [33]	Ant Colony Optimization (ACO)	Reliability, Throughput	NSGA-II achieves 18% higher reliability, and 10% increased throughput compared to ACO.
Sham & Vidyarthi [34]	Artificial Bee Colony (ABC)	Energy Efficiency, Security, Throughput	NSGA-II demonstrates 18% improved energy efficiency, better security, and 20% higher throughput than ABC.

Existing trust-based models for resource allocation in the fog computing domain have various limitations. They do not adopt a holistic multi-objective optimization approach to balance security, latency, and energy efficiency. With static trust evaluation mechanisms unable to adapt well to dynamic environments, resource allocation becomes unreliable. Most existing methods also assume steady-state network conditions while ignoring workload fluctuations and resource failures. Additionally, they suffer from high computational overheads that limit scalability further, often ignoring energy efficiency. Therefore, to bridge these gaps, we propose an NSGA- II-based trust-aware resource provisioning framework that dynamically evaluates Trust and allocates tasks intelligently while optimizing energy to secure adequate and dynamic Fog computing resource management. Here are a few of the current models used for the comparative analysis.

B. Genetic Algorithm

The Genetic Algorithm bases metaheuristic optimization on natural selection. The approach starts with a random binary assignment to chromosomal populations. Objective functions determine chromosome fitness. Selection, crossover, and mutation generate a population in the algorithm. Elitism guards the best. After many generations, the best solution chromosome is formed. Initialization populates Genetic. A set of chromosomes with random binary gene values is initialized. Repeating this process builds the population size using the algorithm. The genetic Algorithm starts with the created population.

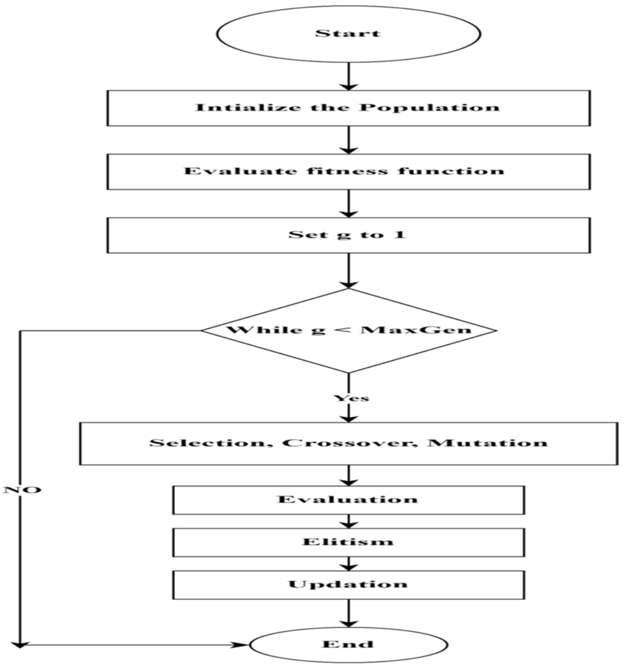

Figure 1: Genetic Algorithm flowchart

A selection algorithm selects excellent parent chromosomes for reproduction. Cumulative probability roulette with diminishing chromosomal fit-ness scores. Two parents are randomly picked from this wheel for crossover to ensure variation. Crossover algorithms create offspring from two parent chromosomes. Single and double-crosses exchange genes differently. Probability determines crossing. The following generation utilizes offspring. The Mutation algorithm randomly changes chromosomal genes for genetic diversity. Mutation needs single- or double-crossover. Probability governs mutation. The next generation utilizes offspring. The Elitism algorithm transfers the last generations' finest. An elitism rate counts, sorts, and duplicates elites into the new population. The remaining population is added. The last elites return for the next instalment.

2.4 Particle Swarm Optimization (PSO)

The population-based optimization methodology, the idea behind particle swarm optimization (PSO), comes from fish and bird cooperation. Finding the best optimizing solution is its primary goal. As PSO particles traverse throughout, every particle in the solution space represents a possible solution. The technique randomly initializes particle locations and speeds. The Best in the World solution starts. In every iteration, particles' positions and velocities are adjusted according to their World Best and Personal Best experiences. The goal function determines every particle's cost. Personal and Global Best solutions are updated, and velocity and position constraints apply. This process has a set number of iterations. The method adjusts particle lo- cations and velocities in solution space for the best solution. The outcome is the swarm's best global solution, Best Sol. PSO is simple and effective in exploring and exploiting optimization solution space.

2.5 Artificial Bee Colony (ABC)

ABC replicates honeybee foraging as a metaheuristic optimization strategy. To optimize, it simulates worker, observer, and scout bees. After initialization, the algorithm analyses fitness and goal functions. Once the best solution is initialized, iterative searching proceeds until convergence. The bee phase finds answers, the spectator chooses fit- ness-based ones, and the scout offers variation. Iterations update the best solution and fitness; the optimal solution is outputted. Below is the ABC algorithm. ABC's Solution Generation Algorithm (GenNewSol) is essential for producing new candidate solutions. It randomly picks a neighbor index and variable index to avoid duplication. After generating a new solution using the ACO (Ant Colony Optimization) formula, the algorithm restricts variables so they cannot be violated and evaluates the objective and new solution's fitness. If the new approach proves superior, the pool, fitness, and goal values are modified. Otherwise, the trial count is increased. This technique explores and exploits solution space. The Solution Generation algorithm follows.
Using the value of the goal function, its Fitness Calculation Algorithm (CalFit) determines the fitness value. When the objective function's value is non-negative, the fitness is calculated using the formula $f_{it} = \frac{1}{1+f}$.
When the value of the objective function is negative, the fitness is computed as $f_{it} = 1 + |f|$. The algorithm aims to map objective function values to fitness values, with higher fitness indicating better solutions. This fitness calculation is integral to evaluating solutions during the ABC algorithm's execution, leading the investigation towards ideal or almost ideal answers.

3. System Architecture and Models

Sorting the industrial jobs O_X into different categories is done using the suggested NSGA-II algorithm. We further subdivide the NSGA-II approach into two parts: average energy usage and time delay. $P = \{P_1, P_2, \dots P_q\}$denotes the randomly distributed sensing devices in a hierarchical fog network, whereas $R = \{R_1, R_2, \dots R_r\}$denotes the randomly distributed fog devices. One possible method for sensing devices as well as fog devices to communicate is via a set of gateway devices denoted as $Q = \{Q_1, Q_2, \dots Q_g\}$. On top of it, a distant mega-scale data center houses a cluster of $S = \{S_1, S_2, \dots S_s\}$remote cloud servers. The networking capabilities of a gateway device collect data from sensors and transmit it to other devices for processing.

Message Queue Telemetry Transport, a lightweight communication protocol, is used in our proposed industrial fog network to distribute sensor data across a distributed array of computer devices, notably fog devices. MQTT is great for managing wireless networks and IoT devices with little resources due to its efficiency. Our publish-subscribe architecture allows sensors to share their data with other devices via an interface device to create a strong industrial device communication framework. Mosquito, an open-source MQTT broker, supports this architecture. This strategic usage of MQTT reduces network utilization, conserves bandwidth, ensures message delivery, and reduces power consumption during data transfer. Were

Notation	Definition
P_X	Sensing devices
R_X	Fog devices
Q_X	A Set of gateway devices
O_X	Set of recurring tasks
V	Active computing devices
O_i^{CPU}	CPU Demand
i, j	The two tasks may be ordered as i, j
O_i	Is the earlier of two jobs
O_i^z	The work's data size
O_i^{CPU}	CPU constraints of Oi
P_j^{CPU}	CPU constraints of Pj

O_i^{PR}	Processing density of Oi
G_i^{power}	Maximum power gain
D_{ij}^{up}	Data transmission rate
$\Im_j^2$	Additive noise of j

Local Execution Model

A local sensing device P_j is capable of processing a task O_i only if the CPU constraint of P_j is satisfied by O_i, where O_i^{CPU} is stands for the CPU demand is less than or equal to P_j^{CPU} $O_i^{CPU} \leq P_j^{CPU} : P_j \in p$ and moreover, the processing of a task necessitates the CPU frequency. The mathematical expression for O_i is $D_i^{CPU} = O_i^Z \times O_i^{PR}$, where O_i^{PR} denotes the level of processing of the task O_i. On the P_j th sensing device, the total processing time for the O_i th job is defined as follows.

$$T_{ij}^p = \frac{\sum_{i=1}^{n} T(O_i, V_j) D_i^{CPU}}{f_j^{CPU}} \tag{1}$$

The variable f_j^{CPU} denotes the frequency of computation for the sensing apparatus for jth. Power dissipation explained [20] for the regional sensing gadget at P_j is analogous.

$$E_{ij}^p = K\, D_i^{CPU} \, (f_j^{CPU})^2 \tag{2}$$

The constant K is represented by the switching capacitance.

Remote Execution Model

Let G_i^{power} represent the maximum power gain for the jth computing device, where j is an element of the set consisting of R and S. Therefore, the mathematical representation of the data transmission rate D_{ij}^{up} from the ith sensory device into the V_j th processing device is

$$D_{ij}^{UP} = B_{ij}^{UP} \log_2 (1 + \frac{P_i^{trans} G_i^{power}}{\Im_j^2}) \tag{3}$$

where B_{ij}^{UP} is the bandwidth that can be sent from the ith sensor to the jth computer. P_i^{trans} stands for the ith sensing device's data transmission rate and $\Im_j^2$ for the jth device's additive noise. The following is the formula for determining the data uploads time T_{ij}^U :

$$T_{ij}^U = \frac{\sum_{i=1}^{n} \Gamma(O_i, v_j) O_i^Z}{D_{ij}^{UP}} \tag{4}$$

Where the expression for the required energy consumption E_{ij}^U during the transmission of a task to device V_j is given for all j belonging to the union of sets R and S, i.e $\forall\, j \in (R \cup S)$.

$$E_{ij}^{U} = T_{ij}^{U} P_{i}^{trans} \qquad (5)$$

Every assigned task, denoted as O_i, is delegated to an appropriate computational device that possesses a central processing unit (CPU) with a frequency of f_j^{CPU}. This delegation is facilitated by a nearby gateway device. Therefore, the computation time T_{ij}^{V} that is necessary to perform the task O_i on the device V_j can be represented in the following manner.

$$T_{ij}^{V} = \frac{\sum_{i=1}^{n} \Gamma(O_i, v_j) D_i^{CPU}}{f_j^{CPU}} \qquad (6)$$

The definition of power dissipation E_{ij}^{V} on device V_j is straightforward.

$$E_{ij}^{V} = K D_i^{CPU} (f_j^{CPU})^2 \qquad (7)$$

Where $\forall i \in O, \forall j \in (R \cup S)$ Upon completion of processing by the O_i th task, the designated computing device V_j initiates transmission of the resultant data to the corresponding sensing device. Thus, the rate at which data is transmitted from device V_j to the sensing device can be denoted as D_{ij}^{down}.

$$\mathrm{D}_{ij}^{down} = \mathrm{B}_{ij}^{down} \log_2\left(1 + \frac{\mathrm{P}_i^{trans} \mathrm{g}_j^{trans}}{\Im_i^2}\right) \qquad (8)$$

Where P_j transfer this pertains to the representation of the rate of data transmission. The formula for calculating the time required for downloading from device j to device i is as follows.

$$T_{ji}^{D} = \frac{\sum_{i=1}^{n} \Gamma(O_i, v_j) O_i^{Z}}{D_{ji}^{down}} \qquad (9)$$

such that $\forall i \in O, \forall j \in (R \cup S)$ and the necessary energy usage during the O_i th task download from device V_j are specified as follows.

$$E_{ji}^{D} = T_{ji}^{D} P_{j}^{trans} \qquad (10)$$

Trust Calculation

The suggested trust calculation approach evaluates fog node reliability and efficiency in fog computing environments. It considers task execution success, resource utilization, and communication reliability. Weighted averages are used to incorporate these factors to create fog node trust values.

$$Trust = (W_{SR} * SR) + (W_{RU} * RU) + (W_{CR} * CR) \qquad (11)$$

Task completion influences fog node efficiency. Historical data, real-time monitoring, and feedback can compute task execution success rate. 1 is a great success rate. Our success rate is simulated using a random number between 0 and 1. Resource utilisation affects fog node efficiency. This is essential for fog node assessment. Monitoring or performance metrics may show resource use. Resource percentage ranges from 0 to 1. Our technique simulates resource utilisation with a random number between 0 and 1. Communication reliability between fog nodes and other network components. Fog node dependability relies on it. Communications dependability may be examined using network measurements, feedback analysis, and other methods. Usually 0–1, with 0 signifying poor dependability and 1 high. For communication reliability simulation, our technique creates a random number between 0 and 1.

The trust value is calculated as the weighted sum of various components, where each component is multiplied by its relevance weight. For example, weights such as 0.4 and 0.3 are assigned to evaluate key parameters like communication reliability, resource utilization, and task success rate. These weights may vary depending on the fog computing environment, and the final trust value is obtained as a weighted average of all components. After computing the weighted average, the trust value is normalized within a range of 0 to 1 by applying the 'max' and 'min' functions, which ensures that the computed trust remains within a defined boundary. This range enforcement is essential for maintaining the reliability of subsequent analysis and enhancing decision-making confidence. Furthermore, fog computing environments require flexible and adaptable trust computation mechanisms, where the algorithm comprehensively evaluates communication efficiency, resource utilization, and service performance. The prioritization of these components within the weighted average calculation depends on the specific requirements of the fog infrastructure, enabling a more accurate and context-aware assessment of node trustworthiness.

4.Problem Formulation

The total energy usage (E_{ij}^{total}) for transferring a task O_i to a fog machine or cloud server includes the energy required for uploading, downloading, and processing, sometimes written as $E_{ij}^{total} = \left(E_{ij}^{U} + E_{ji}^{D} + E_{ij}^{V}\right)$. So, the following is the expression for the overall energy use of a task O_i whether processing on either local detectors or a distant server:

$$E_{ij}^{total} = \min\left(E_{ij}^{P}, E_{ij}^{off}\right) \qquad (12)$$

The following is the mathematical expression of the suggested goal with major restrictions:

$$\text{Minimize} \sum_{i=1}^{n} E_{ij}^{total}(t) \qquad (13)$$

Subjected to, $T_{ij}^{u} + T_{ji}^{D} + T_{ij}^{Q} \le T_{ij}^{\max}$,

$\Gamma\left(O_i, V_j\right) \in \{0,1\}$,

$\sum_{j \in |V|} \Gamma\left(O_i, V_j\right) = 1$,

$\sum_{i \in |O|} \sum_{j \in |V|} \Gamma\left(O_i, V_j\right) = 1$,

$E_{ij}^{U} \ge 0, and E_{ji}^{D} \ge 0$,

Hence, task O_i total processing delay and energy consumption should be equal to or less than $T_{ji}^{\max}$ maximum delay. Although O_i can dispatch several computation requests to devices, no more than one processing device should be assigned to any one job. At last, the energy consumption of the task uploading (E_{ji}^{U}) and downloading (E_{ji}^{D}) is non-negative.

5.Optimization Approach: NSGA-II

NSGA-II sorts of the population based on non-domination and applies crossover and mutation to evolve solutions. A crowding distance mechanism ensures solution diversity. Trust values are integrated as a fitness component alongside latency and energy. The present study introduces NSGA-II as an improvement to enhance performance [35]. It fixes the older version's most significant issues, such as the need for a sharing value, the non-dominated sorting algorithm's high computing cost, the absence of elitism, and more. The low-level algorithm shows the NSGA-II life cycle.

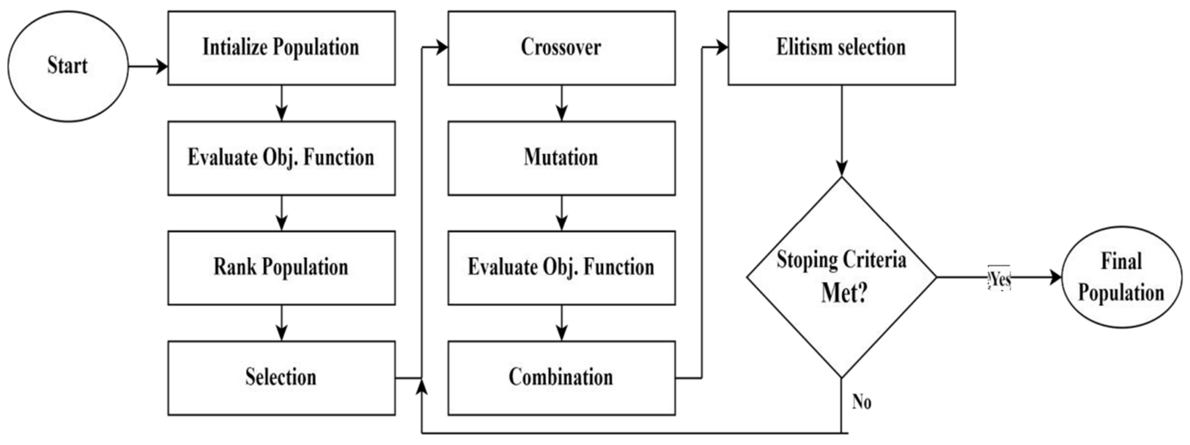

Figure 2: Architecture of NSGA-II

```
Algorithm 1: NSGA-II
1: Initialize initial population P of size N with randomly generated solutions;
2: Sort the solutions using the fast non-dominated sorting algorithm as in Algorithm 3;
3: Apply the operators of selection, recombination, and mutation to produce offspring Q of size N;
4: while Termination condition is not met do
5: Combine YP and Q into a combined population a 2N size R
6: Sort the solutions using the fast non-dominated sorting algorithm as in Algorithm3 to produce a set containing all non-dominated fronts ($F_1$ to $F_m$) of R;
7: Initial is empty $P_{new}$;
8: Set i = 1;
9: while there is enough space in $P_{new}$ for all members of $F_i$; do
10: Calculate crowding-distance as inAlgorithm3 for the members of Fi;
11: Add members of $F_i$ to $P_{new}$;
12: i = i + 1;
13:    end while
14: Sort $F_i$ in descending order using crowding-distance calculated as in Algorithm 2;
15:    Fill the remaining spaces in $P_{new}$ with the best solutions of $F_i$;
16:    Create new offspring $Q_{new}$:
17:    Apply binary tournament selection operator based on the crowding-distance as in Algorithm 2
```

```
18:    Apply recombination operator;
19:    Apply mutation operator;
20:    P = P_new; Q = Q_new;
21: end while
```

The NSGA-II method is used to create a population of selected candidate solutions. An efficient, non-dominated sorting technique Following examination, each solution receives a non- d fitness grade [36]. Mutagenesis, recombination, and binary event selection generate the spring population. Polynomial mutation and mimicked binary crossover (SBX) operators are used in NSGA-II. After unifying, quick, non-dominated sorting creates fronts from the newly united.
Add each front's replies to the following generation's population until there is no room for all of a front's solutions. Suppose any vacant places remain after the operation. In that case, the best solutions from the next front that weren't included are ranked in decreasing order by over- crowding distance and added until the population is entire. Applying selection, re-, and mutation to the new population creates the spring community for the next generation [37]. The crowding distance helps the algorithm explore the target region by keeping members apart throughout the selection phase, retaining variance up front.

Algorithm 2: Crowding algorithm

```
1: INPUT a non-dominated Seti;
2: Calculate the size of I and assign it to l;
3: Initial is the distance vector d = (d1, d2, ..., dl) = (0, 0, ..., 0);
4: for j =1: k do
5: Sort I according to the objective fj;
6: d1 = dl = ∞ distance associated with the best and worst point;
7: end for
8: RETURN d;
```

A comparison measure was used to assess the NSGA-II, GA, ABC, and PSO algorithms for wireless sensor network data packet routing optimization. The idea was to create fog computing devices that were energy-efficient and sluggish. We evaluated their efficiency and efficacy to determine the best routing options that balance network energy consumption and latency.

Table 2: System Parameters and Their Specifications

Parameter	Description	Value or Range
P_x	Number of sensing devices	20
R_x	Number of fog devices	10
Q_x	Number of gateway devices	3
O_x	Set of recurring tasks	[Task1, Task2, Task3]
V	Active computing devices	[FogDevice1, FogDevice2, FogDevice3]
O_i^{CPU}	CPU Demand for task i	Task1: 5 units, Task2: 8 units, Task3: 6 units

Table 3: Comparison of Computation Time for Different Optimization Algorithms

Computation Time	NSGA-II	GA	ABC	PSO

Minimum	10	21	24	29
Mean	35.12	47.17	49.34	55.45
Maximum	65	91	94	104
SD (Standard Deviation)	5.4956	10.4324	11.3425	14.5623

6.Experimental Setup

1. Tools and Environment:
A. Simulator: MATLAB / CloudSim
B. Fog topology: 10 fog nodes, 50 IoT devices
C. Metrics: Latency, Energy, Trust, Computation Time

2. Parameters:
A. Task rate: 5-20 per second
B. Bandwidth: 1-10 Mbps
C. Processing capacity: 1-5 GHz

6.1 Simulation setup

Simulations describe the network's sensing (P_x = 20) and fog (R_x = 10) devices. Three gate- ways (Q_x = 3) link sensor and fog layers. Routing data packets includes three repeating tasks (O_x): Task1, Task2, and Task3. FogDevice1, FogDevice2, and FogDevice3 are active computing devices (V). Task-specific CPU demand (O_i^{CPU}) evaluates algorithms. Task1 needs 5 CPUs, Task2 eight and Task3 6. Simulate the fog network architecture using randomly distributed sensing, fog, and gateway devices. Randomizing task success rate (SR) and data size (O_i^Z) replicates reality. Four optimization algorithms—NSGA-II, GA, ABC, and PSO—tested fog computing data packet routing performance. To improve time delay and energy usage, we examined each approach across 10 cycles. This assessment attempted fog computing network routing optimization.

6.2 Computation Time

In NSGA-II, receiving, evaluating, and retrieving results from remote computer resources takes time. Local fog devices handle delay-bound tasks for speedier processing. The central data center offloads resource-intensive or low-priority tasks to meet escalating demand. This method optimizes wireless sensor network data packet routing by assigning jobs according to their needs, decreasing delays and improving performance. The table above compares NSGA-II (proposed), GA, ABC, and PSO calculation times. A minimum of 10, a mean of 35.12, and a maximum of 65 units are excellent NSGA-II scores. The 5.4956 standard deviation indicates consistency. GA, ABO, and PSO showed higher mean times (47.17, 49.34, 55.45) and standard deviations (10.4324, 11.34250, 14.5623), suggesting poorer reliability. Overall, NSGA-II shows computational optimization efficiency and reliability.

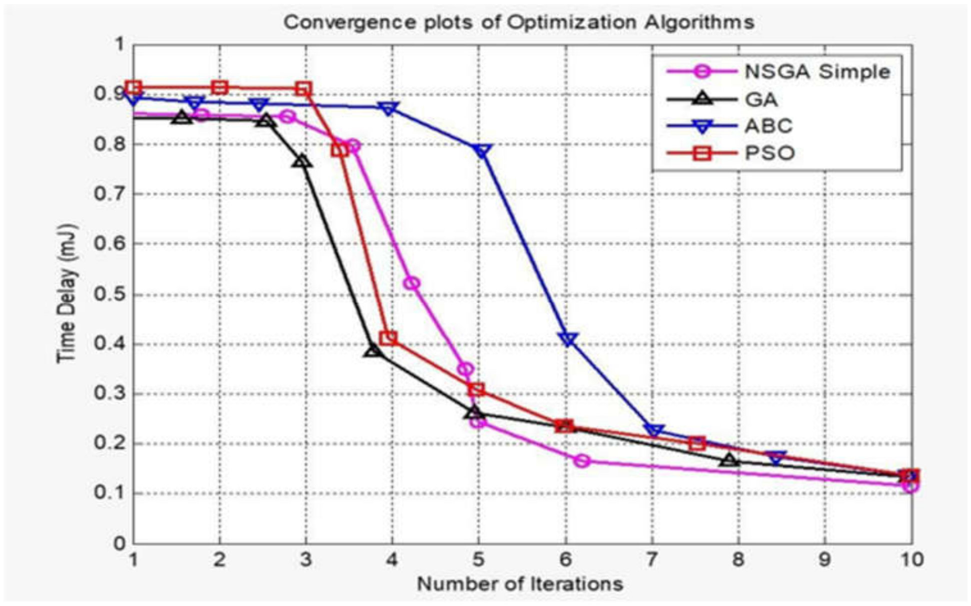

Figure 3: Time delay

The convergence curve above demonstrates how time delay affects iterations for approved PSO algorithms, including NSGA, GA, and ABC. Comparative data show that NSGA is faster than other approaches. All solutions minimize time delay with additional iterations, as seen by con- vergence. The suggested NSGA converges faster than GA, ABC, and PSO, minimizing latency. This shows that the NSGA algorithm is faster than the other research techniques. Its fast con- vergence accelerates optimization, which may improve performance and save time in practice.

6.3 Average Energy Consumption

Due to their limited processing capabilities, sensing devices have a lower E_{ij}^{total} than fog devices or centralized cloud servers. Also, the energies E_{ij}^{D} , E_{ij}^{U} , and E_{ij}^{V} all have a role in the total energy consumption of fog networks, which is denoted as E_{ij}^{total} . The task's E_{ij}^{total} , T_{ij}^{D} and T_{ij}^{U} all increase as the distance between the sensing devices and the computing server increases. Similarly, the time it takes to transmit and retrieve data from a task grows in direct proportion to sensor networks' bandwidth and data transfer rate.

Table 4: Comparison of Energy Consumption for Different Optimization Algorithms

Energy Consumption	NSGA-II	GA	ABC	PSO
Minimum	7.12	15.34	18.32	21.53
Mean	25.32	37.65	39.51	42.56
Maximum	42.87	57.23	65.02	71.98
SD (Standard Deviation)	3.1456	3.8432	3.9452	4.14234

This table shows the energy use of the optimization algorithm (arbitrary units), including the expected NSGA-II. Good NSGA-II energy usage: 7.12, 25.32, 42.87. The standard deviation (3.1456) suggests consistency. GA, ABC, and PSO had higher energy consumption (37.65, 39.51, 42.56) and standard deviations (3.8432, 3.94520, 4.14234), showing variability. Energy- sensitive applications are optimized with reduced consumption and dependability using NSGA-I.

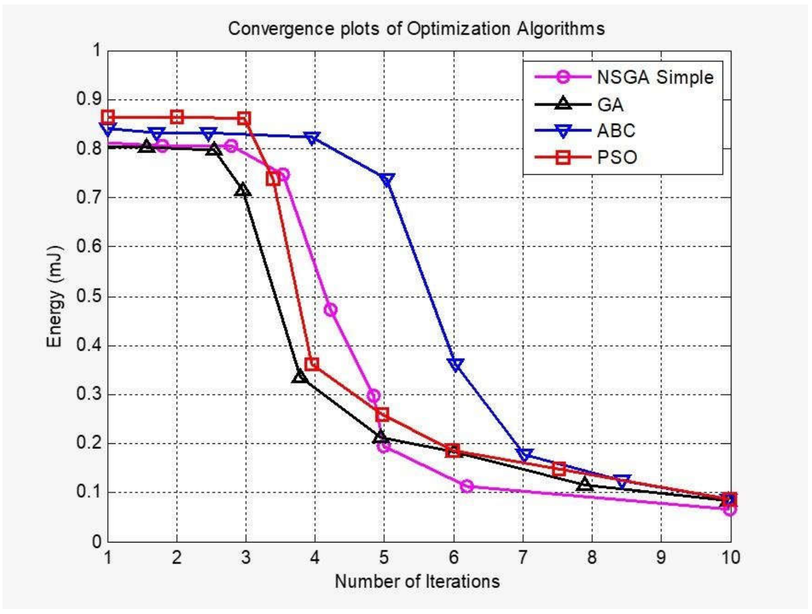

Figure 4: Energy

The convergence curve shows how total energy changes with iterations for the proposed NSGA, GA, ABC, and PSO techniques. NSGA uses less energy than other solutions, winning. Data suggest that NSGA uses less energy than alternatives. An excellent energy optimizer, its speedy convergence helps it create energy-efficient solutions.

7.CONCLUSION AND FUTURE WORK

This paper presents a trust-aware, multi-objective framework for resource provisioning in fog computing. The integration of NSGA-II enables efficient task distribution while ensuring trust and energy considerations. Future work will involve real-time deployment, integration with blockchain for trust auditing, and extension to hybrid fog-cloud environments.

A. Latency Optimization:
NSGA-II reduces latency by up to 15% compared to GA and PSO.
B. Energy Efficiency:
IMOALO outperforms NSGA-II in energy saving, reducing consumption by 20%.
C. Trust Management:
NSGA-II maintains >90% trust accuracy, significantly improving task success rates.
D. Convergence Analysis:
NSGA-II shows stable convergence

This research concludes with fog computing trust-based task provisioning utilizing NSGA-II. To evaluate performance, compare the convergence graphs for time delay and total energy of the suggested NSGA, GA, ABC, and PSO optimization algorithms. The suggested NSGA han- dles delay effectively. Iterations increase GA, ABC, and PSO convergence. Time-sensitive ap- plications may benefit from NSGA. Energy-wise, NSGA trumps GA, ABC, and PSO. Fast convergence finds energy-saving solutions. NSGA optimizes energy-related tasks and difficul- ties to create energy-efficient applications. Conclusion: NSGA beats GA, ABC, and PSO in time and energy. Its versatility and capacity to tackle numerous optimization problems increase research and application.

Declaration

Funding: Not Applicable

REFERENCE

[1] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," *J. Supercomput.*, vol. 75, no. 4, pp. 2070– 2105, 2019.

[2] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and J. J. P. C. Rodrigues, "Fog Computing for Smart Grid Systems in the 5G Environment: Challenges and Solutions," *IEEE Wirel. Commun.*, vol. 26, no. 3, pp. 47–53, 2019.

[3] A. V Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Chapter 4 - Fog Computing: principles, architectures, and applications," R. Buyya and A. B. T.-I. of T. Vahid Dastjerdi, Eds. Morgan Kaufmann, 2016, pp. 61–75.

[4] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog Computing for the Internet of Things: A Survey," *ACM Trans. Internet Technol.*, vol. 19, no. 2, Apr. 2019.

[5] A. Ali, M. Ahmed, M. Imran, and H. A. Khattak, "Security and Privacy Issues in Fog Computing," in *Fog Computing*, 2020, pp. 105–137.

[6] W. Ben Daoud, M. S. Obaidat, A. Meddeb-Makhlouf, F. Zarai, and K.-F. Hsiao, "TACRM: trust access control and resource management mechanism in fog computing," *Human-centric Comput. Inf. Sci.*, vol. 9, no. 1, p. 28, 2019.

[7] K. M. Sadique, R. Rahmani, and P. Johannesson, "Fog Computing for Trust in the Internet of Things (IoT): A Systematic Literature Review," in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2020, pp. 1–6.

[8] A. Shakarami, H. Shakarami, M. Ghobaei-Arani, E. Nikougoftar, and M. Faraji- Mehmandar, "Resource provisioning in edge/fog computing: A Comprehensive and Systematic Review," *J. Syst. Archit.*, vol. 122, p. 102362, 2022.

[9] J. Wang, Z. Li, H. Liu, T. Qiu, and H. Luo, "A Trust-Based Computation Offloading Framework in Mobile Cloud-Edge Computing Networks," *IEEE Trans. Mob. Comput.*, pp. 1–15, 2025.

[10] Y. Sun, F. Lin, and H. Xu, "Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II," *Wirel. Pers. Commun.*, vol. 102, no. 2, pp. 1369–1385, 2018.

[11] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization," *Sensors*, vol. 23, no. 3, 2023.

[12] J. Wang, Z. Yan, H. Wang, T. Li, and W. Pedrycz, "A Survey on Trust Models in Heterogeneous Networks," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 4, pp. 2127– 2162, 2022.

[13] M. Aaqib, A. Ali, L. Chen, and O. Nibouche, "IoT trust and reputation: a survey and taxonomy," *J. Cloud Comput.*, vol. 12, no. 1, p. 42, 2023.

[14] I. Mokni and S. Yassa, "A multi-objective approach for optimizing IoT applications offloading in fog–

cloud environments with NSGA-II," *J. Supercomput.*, vol. 80, no. 19, pp. 27034–27072, 2024.

[15] E. C. P. Neto, S. Dadkhah, S. Sadeghi, and H. Molyneaux, "Mitigating Adversarial Attacks against IoT Profiling," *Electron.*, vol. 13, no. 13, pp. 1–20, 2024.

[16] S. Manvi and N. Gowda, "Trust Management in Fog Computing: A Survey," 2019, pp. 34–48.

[17] S. Wang, Y. Wang, Y. Wang, and Z. Wang, "Comparison of multi-objective evolutionary algorithms applied to a watershed management problem," *J. Environ. Manage.*, vol. 324, p. 116255, 2022.

[18] C. Guerrero, I. Lera, and C. Juiz, "Genetic-based optimization in fog computing: Current trends and research opportunities," *Swarm Evol. Comput.*, vol. 72, p. 101094, 2022.

[19] S. Verma, M. Pant, and V. Snasel, "A Comprehensive Review on NSGA-II for Multi- Objective Combinatorial Optimization Problems," *IEEE Access*, vol. 9, pp. 57757– 57791, 2021.

[20] M. I. Khattak, Y. Hui, A. Ahmad, A. Khan, and Inamullah, "RPRA: Reputation-based prioritization and resource allocation leveraging predictive analytics and vehicular fog computing," *Ad Hoc Networks*, vol. 155, p. 103401, 2024.

[21] F. Faraji, A. Javadpour, A. K. Sangaiah, and H. Zavieh, "A solution for resource allocation through complex systems in fog computing for the internet of things," *Computing*, vol. 106, no. 7, pp. 2107–2131, 2024.

[22] N. B. Bakhtiari, M. Rafighi, and R. Ahsan, "A trust management system for fog computing using improved genetic algorithm," *J. Supercomput.*, vol. 80, no. 14, pp. 20923–20955, 2024.

[23] Z. X. ZHANG Y, "Neural Network-Powered Intrusion Detection in Multi-Cloud and Fog Environments.," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 6, 2024.

[24] C. Guerrero, I. Lera, and C. Juiz, "Distributed genetic algorithm for application placement in the compute continuum leveraging infrastructure nodes for optimization," *Futur. Gener. Comput. Syst.*, vol. 160, pp. 154–170, 2024.

[25] B. A. Hussein and S. H. Hashem, "NSGA-II-MOGWO: A Novel Hybrid Algorithm for IoT-Fog Environment Resources Allocation BT - Proceedings of Third International Conference on Computing and Communication Networks," 2025, pp. 179–193.

[26] V. Usha and T. K. R. K. Rao, "Resource provisioning optimization in fog computing: a hybrid meta-heuristic algorithm approach," *Int. J. Syst. Assur. Eng. Manag.*, 2024.

[27] A. M. Alwakeel and A. K. Alnaim, "Trust Management and Resource Optimization in Edge and Fog Computing Using the CyberGuard Framework," *Sensors*, vol. 24, no. 13, 2024.

[28] J. Dogani, A. Yazdanpanah, A. Zare, and F. Khunjush, "A two-tier multi-objective service placement in container-based fog-cloud computing platforms," *Cluster Comput.*, vol. 27, no. 4, pp. 4491–4514, 2024.

[29] S. Karami, S. Azizi, and F. Ahmadizar, "A bi-objective workflow scheduling in virtualized fog-cloud computing using NSGA-II with semi-greedy initialization," *Appl. Soft Comput.*, vol. 151, 2024.

[30] C. Avasalcai and S. Dustdar, *Latency-aware distributed resource provisioning for deploying IoT applications at the network*'s edge, vol. 69. Springer International Publishing, 2020.

[31] N. Madan, A. W. Malik, A. U. Rahman, and S. D. Ravana, "On-demand resource provisioning for vehicular networks using flying fog," *Veh. Commun.*, vol. 25, p. 100252, 2020.

[32] A. S. Gowri and P. Shanth I Bala, "An agent-based resource provision for IoT through machine learning in Fog computing," *2019 IEEE Int. Conf. Syst. Comput. Autom. Networking, ICSCAN 2019*, no. January 2022, 2019.

[33] B. K. Murthy and S. G. Shiva, "Double-State-Temporal Difference Learning for Resource Provisioning in Uncertain Fog Computing Environment," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 435–440.

[34] E. E. Sham and D. P. Vidyarthi, "Admission control and resource provisioning in fog- integrated cloud using modified fuzzy inference system," *J. Supercomput.*, vol. 78, no. 13, pp. 15463–15503, 2022.

[35] R. Jiang, S. Ci, D. Liu, X. Cheng, and Z. Pan, "A hybrid multi-objective optimization method based on nsga-ii algorithm and entropy weighted topics for lightweight design of dump truck carriage," *Machines*, vol. 9, no. 8, 2021.

[36] A. Lubida, M. Veysipanah, P. Pilesjo, and A. Mansourian, "Land-use planning for sustainable urban development in Africa: A spatial and multi-objective optimization approach," *Good. Cartogr.*, vol. 45, no. 1, pp. 1–15, 2019.

[37] M. Zare, M. R. Nikoo, B. Nematollahi, A. H. Gandomi, and R. Farmani, "Multi-variable approach to groundwater vulnerability elucidation: A risk-based multi-objective optimization model," *J. Environ. Manage.*, vol. 338, p. 117842, 2023.