mathematics

MDPI

Article

An Evolutionary Algorithm for Task Clustering and Scheduling in IoT Edge Computing

Adil Yousif [1,*], Mohammed Bakri Bashir [2,3] and Awad Ali [1]

[1] Department of Computer Science, College of Science and Arts-Sharourah, Najran University, Sharourah 68378, Saudi Arabia; aaomar@nu.edu.sa
[2] Department of Mathematics, Turubah University College, Taif University, Taif 26571, Saudi Arabia; safi@tu.edu.sa
[3] Department of Computer Science, Faculty of Computer Science and Information Technology, Shendi University, Shendi 41601, Sudan
* Correspondence: ayalfaki@nu.edu.sa; Tel.: +966-123-986-9

Abstract: The Internet of Things (IoT) edge is an emerging technology of sensors and devices that communicate real-time data to a network. IoT edge computing was introduced to handle the latency concerns related to cloud computing data management, as the data are processed closer to their point of origin. Clustering and scheduling tasks on IoT edge computing are considered a challenging problem due to the diverse nature of task and resource characteristics. Metaheuristics and optimization methods are widely used in IoT edge task clustering and scheduling. This paper introduced a new task clustering and scheduling mechanism using differential evolution optimization on IoT edge computing. The proposed mechanism aims to optimize task clustering and scheduling to find optimal execution times for submitted tasks. The proposed mechanism for task clustering is based on the degree of similarity of task characteristics. The proposed mechanisms use an evolutionary mechanism to distribute system tasks across suitable IoT edge resources. The clustering tasks process categorizes tasks with similar requirements and then maps them to appropriate resources. To evaluate the proposed differential evolution mechanism for IoT edge task clustering and scheduling, this study conducted several simulation experiments against two established mechanisms: the Firefly Algorithm (FA) and Particle Swarm Optimization (PSO). The simulation configuration was carefully created to mimic real-world IoT edge computing settings to ensure the proposed mechanism's applicability and the simulation results' relevance. In the heavyweight workload scenario, the proposed DE mechanism started with an execution time of 916.61 milliseconds, compared to FA's 1092 milliseconds and PSO's 1026.09 milliseconds. By the 50th iteration, the proposed DE mechanism had reduced its execution time significantly to around 821.27 milliseconds, whereas FA and PSO showed lesser improvements, with FA at approximately 1053.06 milliseconds and PSO stabilizing at 956.12 milliseconds. The simulation results revealed that the proposed differential evolution mechanism for edge task clustering and scheduling outperforms FA and PSO regarding system efficiency and stability, significantly reducing execution time and having minimal variation across simulation iterations.

Keywords: edge computing; clustering; resource management; firefly algorithm; particle swarm optimization; differential evolution

MSC: 68T20

Citation: Yousif, A.; Bashir, M.B.; Ali, A. An Evolutionary Algorithm for Task Clustering and Scheduling in IoT Edge Computing. *Mathematics* **2024**, *12*, 281. https://doi.org/10.3390/math12020281

Academic Editors: Takfarinas Saber and Aman Singh

Received: 10 November 2023
Revised: 6 January 2024
Accepted: 12 January 2024
Published: 15 January 2024

1. Introduction

The massive increase in Internet of Things (IoT) sensor and actuator devices led to the transfer of large amounts of data to cloud servers over the Internet [1,2]. Generally, IoT devices have poor performance due to the limited storage and processing capabilities. To handle this issue, IoT sensors and actuator devices transfer the processing of tasks to

cloud providers. Cloud computing consists of high-performance servers aiming to satisfy the high demand for storage and computational power [3,4]. Due to the massive number of IoT sensor and actuator devices, the cloud network suffers from congestion, and cloud provider nodes become overwhelmed. The processing of real-time programs on IoT devices suffers from a significant decrease in execution efficiency due to the considerable latency between the edge device and the cloud providers. IoT edge computing was introduced to handle the latency concerns related to cloud computing data management, as the data are processed closer to their point of origin. While edge computing addresses the problem of IoT resource limitations, clustering and scheduling of edge task issues arise.

In the edge computing systems, the IoT tasks are clustered at the edge device layer to perform the offloaded IoT edge tasks in parallel and distributed schemes. Enhancing the performance of IoT edge resource-limited devices is a crucial process. This process is accomplished by classifying, clustering, and scheduling IoT edge tasks based on their characteristics and resource requirements.

As shown in Figure 1, the IoT edge computing architecture consists of three layers: the edge things layer, the edge device layer, and the cloud layer. The edge things layer includes various IoT devices such as sensors and actuators. The IoT devices collect and produce massive amounts of IoT data from the system environment. The second layer is the edge device layer, consisting of numerous edge devices, such as smartphones, laptops, tablets, and personal computers. These edge devices are generally distributed near IoT layer devices and have crucial processing capabilities. The edge devices in this layer are considered as middle computational units that preprocess the data generated by sensors and actuators in the things layer. The edge layer aims to reduce and optimize the latency of cloud server processing. The edge device layer enables fast decision making and supports real-time processing by handling data closer to the source. The third layer is the cloud servers' layer at the topmost layer. This layer contains high-performance servers and data centers. Cloud data centers maintain large-scale data processing and storage. The edge device layer sends refined data to cloud servers for a further analysis and long-term storage.

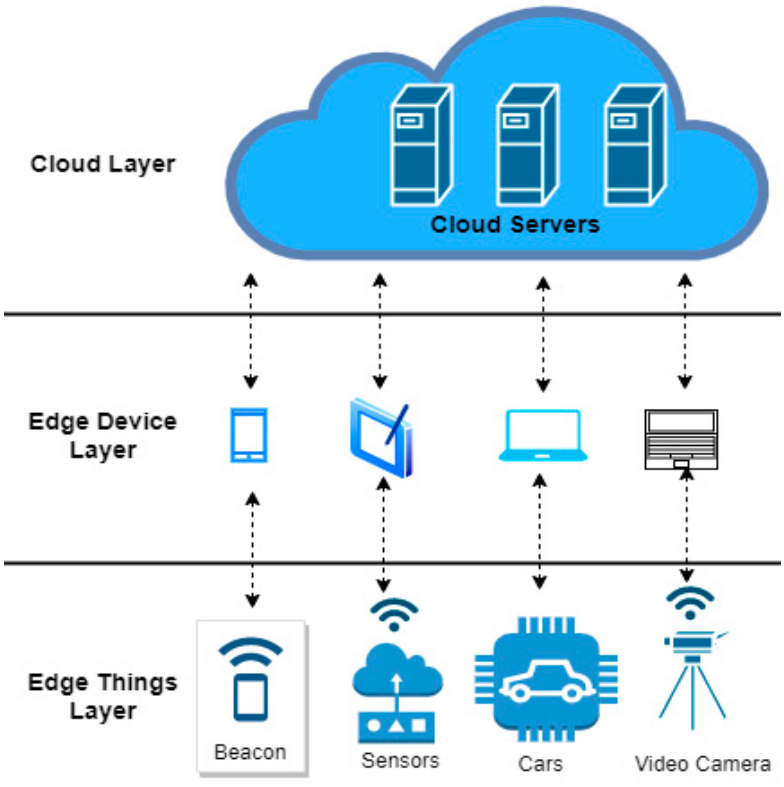

Figure 1. IoT edge computing architecture.

IoT edge task scheduling is the allocation of submitted tasks to suitable resources [5,6]. Task scheduling mechanisms aim to minimize the task execution times and optimize the IoT edge system to work efficiently [7,8]. Clustering IoT edge tasks is grouping IoT edge tasks into classes of similar tasks and discovering suitable categories based on some similarity measures. An IoT edge task cluster is defined as a group of edge computing tasks similar to other tasks inside the cluster and dissimilar to the tasks in different clusters [9,10]. Clustering tasks in IoT edge computing is considered a significant challenge due to the diverse nature of tasks within edge systems. Moreover, IoT edge resources belong to different administrative domains, each applying distinct management policies [11,12]. Additionally, due to IoT edge task characteristics, the scheduling tasks on IoT edge computing are identified as an NP-complete problem [13,14].

Considering the varying attributes of IoT edge tasks and the diverse nature of resources within a dynamically evolving system, clustering IoT edge tasks is considered a significant process for improving the IoT edge task scheduling process. This paper presents the DE mechanism that aims to cluster the IoT edge tasks based on the level of similarity of task characteristics to improve the task scheduling process. While there are various mechanisms for IoT edge task clustering and scheduling in the literature, the proposed differential evolution mechanism distinguishes itself by handling IoT edge environments in a real-time and dynamic manner. By considering task characteristics and resource attributes in the clustering procedure, the proposed DE mechanism aims to reduce the execution time of IoT edge tasks, which is a vital contribution to the field.

The proposed DE mechanism aims to optimize the task allocation process in IoT edge computing by adapting the differential evolution (DE) algorithm. The DE mechanism focuses on the unique challenges of IoT edge computing, such as resource constraints and diverse task characteristics. The proposed mechanism includes optimizing task clustering based on critical characteristics such as task length and resource requirements. Furthermore, the proposed mechanism develops an effective scheduling method to assign IoT edge tasks appropriate resources. This is aimed at minimizing the total execution time of the submitted tasks, consequently improving IoT edge responsiveness and throughput. The adaptation of DE optimization in this context addresses the discrete optimization requirements of IoT task scheduling. The study's main goal is to fill an essential gap in the IoT edge system, presenting a scalable and efficient mechanism for IoT edge task clustering and scheduling.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 describes the task clustering and scheduling issues in the IoT. Section 4 describes the details of the proposed evolutionary algorithm for IoT edge task clustering and scheduling. Section 5 illustrates the performance evaluation of the proposed algorithm. Section 6 presents the paper conclusion.

2. Related Works

The optimization of IoT edge task clustering and scheduling has been reviewed extensively due to the critical need for efficient task processing and optimized task execution times [15–17]. This section reviews the state-of-the-art literature on IoT edge task clustering and scheduling mechanisms. Due to the diverse characteristics of IoT edge tasks and resources, clustering and scheduling tasks in IoT edge computing are considered a challenging problem. This problem has been extensively investigated using various heuristics and metaheuristics mechanisms.

The IoT task clustering process is significant in IoT edge task scheduling by grouping tasks into clusters using task characteristics and similarity measurements. IoT edge task clustering helps organize historical data from IoT edge devices and enhances task scheduling within the IoT edge architecture. Several IoT edge task clustering mechanisms are presented in the literature. Generally, IoT edge task clustering mechanisms are classified into partitioning clustering based on the clustering approach applied. Additionally, based on the structure of the outcome of the clustering process, the IoT edge clustering mechanisms are classified into hierarchical and non-hierarchical methods [9,18]. In the IoT

edge automatic clustering mechanism, the set of edge tasks is gathered into several clusters, and the IoT edge clustering mechanism determines the number of clusters.

Evolutionary and bio-inspired algorithms optimize the IoT edge task clustering and scheduling in the relevant literature. The authors of [19] introduced an evolutionary mechanism for IoT edge task clustering and scheduling to optimize the cost and energy consumption. Similarly, the study in [20] developed a Particle Swarm Optimization (PSO) mechanism for edge device task distribution. While important, these related works do not address the unique constraints and real-time issues present in IoT edge task scheduling, a gap the proposed DE mechanism aims to fill.

The study in [21] extended the capabilities of the computational cloud to the IoT edge computing as it introduced a multi-cluster edge layer framework. This multi-cluster edge layer framework minimizes the latency and delays of IoT edge tasks. The research in [21] applied Particle Swarm Optimization to distributed load processing, representing the application of bio-inspired mechanisms in the management of semi-autonomous edge clusters. The exploration of the Particle Swarm Optimization mechanism presented in [21] complements the proposed mechanism that uses evolutionary algorithms for task clustering and scheduling, showcasing a diverse range of bio-inspired mechanisms to optimize task efficiency in IoT edge computing.

The task management in IoT edge computing, mainly the edge micro-cluster platforms, is vital for efficiently tackling IoT applications that demand Quality of Service (QoS) [20]. The study in [20] extends previous research on task management by proposing a linear-based model coupled with a metaheuristic Particle Swarm Optimization (PSO) approach. The study aims to optimize the makespan time and the task allocation overhead for heterogeneous edge workload management [20]. This study complements and informs the proposed research by providing insights into the scalability and efficiency of various allocation mechanisms in micro-cluster IoT edge computing.

Recent studies, such as in [22,23], have begun considering both task and resource characteristics in their IoT edge task clustering mechanisms. Yet, these clustering mechanisms do not consider the evolutionary features that can adapt to IoT edge dynamic environments. The mechanism in [24] that applied task characteristics to the IoT edge scheduling process does not consider the optimization potential of DE. Several studies, such as [25,26], have considered task clustering with constraints in IoT edge computing, such as energy consumption and delay. Even so, these mechanisms focus on static task clustering approaches that do not evolve with the system's state. This limitation is addressed by the proposed dynamic DE clustering and scheduling mechanism. The use of similarity measures in IoT edge task clustering has been crucial in recent research such as in [27–29]. These studies focus on static parameter similarity measures [23,30]. The proposed DE mechanism extends this by employing a dynamic similarity function that accounts for varying task lengths, priorities, and resource requirements tailored for the IoT edge computing domain.

Differential evolution has been applied successfully in various optimization domains, such as parameter optimization [6,15,16,31,32] and feature selection [33,34]. Nevertheless, DE applications in IoT edge task clustering and scheduling are less explored. The research in [35] applied a DE mechanism for energy-efficient task scheduling; nonetheless, the study does not handle the key multidimensional task characteristics in IoT edge computing.

In summary, while the existing literature provides several mechanisms for IoT edge task clustering and scheduling, the proposed differential evolution mechanism distinguishes itself by addressing IoT edge task clustering and scheduling in a real-time, dynamic, and multi-faceted nature.

3. Task Clustering and Scheduling in IoT Edge

The growing field of IoT edge computing raised the critical need for the efficient task clustering and scheduling mechanism to optimize edge computing performance. As the numbers and complexity of IoT edge tasks increase, minimizing task execution times and optimizing resource utilization become increasingly a challenging issue. This section is

divided into two subsections, each addressing an essential aspect of the IoT edge task clustering and scheduling problem. Section 3.1, "Problem Formulation," illustrates the foundational framework, presenting a formal structured approach to the IoT edge task clustering and scheduling problem. Section 3.1 defines the mathematical framework and constraints crucial to formulating the IoT edge clustering and scheduling problem. The section focused on clustering IoT edge tasks based on task characteristics' similarity and carefully allocates these clusters to the suitable IoT edge resources. Section 3.2, "Standard Differential Evolution Optimization," describes the basic concepts related to the standard differential evolution optimization.

3.1. Problem Formulation

The problem under this study involves scheduling tasks on a set of IoT edge computing resources to minimize the total execution time [20,21]. The IoT edge tasks are divided into clusters based on their similarities, and each cluster is assigned to a resource for execution [23,36]. The main objective is to find the optimal task-cluster-resource assignment that minimizes the total execution time [1,37].

Let T be the set of IoT edge tasks, C be the set of task clusters, and R be the IoT edge resources. Define X[i][j][k] as a variable representing the assignment of task i to cluster j and resource k. X[i][j][k] equals 1 if task i is assigned to cluster j and resource k and 0 otherwise. The main objective is to minimize the total execution time, which is the sum of the execution times of IoT edge tasks allocated to their respective clusters and resources. Each IoT edge task can only be assigned to one cluster and one edge resource. Each cluster can only be assigned to one IoT edge resource. The total number of IoT edge tasks allocated to edge resources should not exceed its capacity. E[i][j][k] represents the execution time of IoT edge task i in cluster j on resource k. $Capacity[k]$ represents the capacity of the computational power of IoT edge resource k.

The allocation of IoT edge tasks to clusters and edge resources can be represented as a permutation in the differential evolution mechanism. Representing the IoT edge computing task clustering and scheduling problem as a mathematical model helps apply optimization mechanisms such as differential evolution to find an optimal solution that minimizes the total execution time.

Suppose there is a finite set N of n IoT edge tasks and a finite set V of m variables describing characteristics of each IoT edge task, $t \in N$, by the value $M(t)$. Suppose that Rv is the range of the variable $x \in M$.

Mathematically, Grabmeier [9] defined clustering as $C = \{C_1, C_2, \ldots, C_t\}$, such that C is a subset of the set of all subsets of N, such that C elements are disjoint and each task $t \in N$ is included in one and only one of $\{C_1, C_2, \ldots, C_t\}$, where C_i represents an IoT edge task cluster. There are two types of clustering mechanisms, hard clustering mechanisms, in which each IoT edge task t is assigned to one and exactly one cluster as in Equation (1).

$$N = \{x_1, x_2, \ldots, x_n\} = \bigcup_{i=1}^{t} C_i \text{ and } C_i \cap C_j = \varnothing \text{ for all } i \neq j \quad (1)$$

The second type of IoT edge task clustering is soft clustering, in which the clusters are permitted to overlap. The proposed DE for IoT edge clustering considers hard clustering. Using hard clustering, the proposed DE for the IoT edge clusters n tasks into t disjoint and non-empty clusters. The number of possible IoT edge task clusters is calculated using $\binom{n}{t}$, which is called the Stirling number of the second kind, which fulfills Equation (2).

$$one\binom{n}{t} = t\binom{n-1}{t} + \binom{n}{t-1} \quad (2)$$

The proposed DE for the IoT edge clustering process focuses on clustering where the IoT edge tasks of each cluster are as similar as possible, and the clusters are as dissimilar as possible.

3.2. Standard Differential Evolution Optimization

Differential evolution optimization (DE) was introduced by Price and Storn [38,39] as a reliable, adaptable, and effective optimization scheme for solving NP-complete problems. The DE optimization process starts with generating a random initial population of feasible candidate solutions and a set of random integer numbers, and each solution represents a chromosome. Each chromosome in the DE initial population is an integer vector indexed with a number from 1 to NP, where NP is the population size.

The next step in the DE optimization process is improving the population chromosomes as follows: For each chromosome r, randomly select three other chromosomes, r_1, r_2, and r_3, where r_1, r_2, r_3 are not equal. Determine the difference between r_1 and r_2 and multiply it with the parameter F to scale it. Add the scaled result to the chromosome r_3 to generate the chromosome y. Then, the chromosome m is produced via the crossover of y and r. The final step is to compare the fitness of the trial chromosome m with the fitness of the chromosome r. The chromosome with the better fitness value is included in the next population. These steps are repeated for several iterations until a termination condition is met. In DE optimization, the population size does not change during the optimization process [38]. Algorithm 1 provides a detailed pseudocode for the DE optimization.

Algorithm 1 describes the pseudo code for the standard DE algorithm as described in [39].

Algorithm 1 Pseudo code for Standard Differential Evolution (DE) Algorithm

```
Begin
Initialize a population with NP random chromosomes.
Define fitness function f
setScalingFactor(F)
Define the termination condition for the algorithm.
Main DE algorithm loop
while (not termination condition) {
    for (i = 1; i <= NP; ++i) {
        r1, r2, r3 = selectRandomDistinctIndices(NP, i)
        donor vector y = population[r3] + F × (population[r1] − population[r2])
        trial vector m = crossover(population[r], donor vector y)
        if (f(trial vector m) <= f(population[r])) {
                population[r] = trial vector m
    end for
end while
    Optionally, evaluate the overall population fitness and perform additional operations.
    evaluatePopulationFitness(population)
    Check and update termination conditions if necessary.
    updateTerminationCondition()
After the loop ends, the population contains the optimized solutions.
End
```

4. The Proposed Differential Evolution for IoT Edge Task Clustering and Scheduling

This section presents the details of the proposed differential evolution (DE) optimization mechanism for IoT edge task clustering and scheduling.

To measure the IoT task similarity and dissimilarity in the proposed DE model, this paper uses the job characteristics described in Section 4.2. The proposed DE mechanism mainly focuses on the job length and the requirements of the resources for each job. The proposed differential evolution clustering and scheduling mechanism consists of two main layers. The first layer maintains the received IoT tasks and divides them into clusters. The second layer is responsible for scheduling the group of received jobs into suitable resources using differential evolution optimization.

To design the proposed DE mechanism for IoT edge task clustering and scheduling, this paper applies the job information knowledge discovery method described in [40]. In

this mechanism, the IoT edge devices submit the tasks to the edge broker; the edge broker has a receiver that maintains the IoT edge task information as historical data in a database. When a new IoT edge task arrives at the broker, the task characteristics and information are compared with the historical data to allocate the task to the appropriate cluster.

4.1. Mathematical Representation of DE for IoT edge Task Clustering and Scheduling

The typical DE optimization uses a real-coded chromosome representation. This study is based on a genetic representation that focuses on discrete variables or task clusters. In the proposed DE mechanism, the chromosome values are between 1 and k and correspond to the cluster to which the IoT edge task is assigned.

The proposed DE mechanism process starts with generating a random initial population of task clustering solutions. Each solution represents a chromosome in the population. The proposed DE mechanism encodes chromosomes as a discrete string, $X_r(t)$, such that $X_r(t) = \{x_{r,1}(t), x_{r,2}(t), \ldots, x_{r,n}(t)\}$ and $x_{r,i}(t) \in \{1, 2, \ldots, k\}$, $r = 1, 2, \ldots,$ NP where NP is the population size, and $i = 1, 2, \ldots, n$ where n is the number of tasks. For instance, if the number of IoT edge tasks to be clustered is 6, the number of clusters = 3 and the size of the population is 3; then, a population can be $X_1(t) = \{2, 1, 2, 3, 2, 1\}$, $X_2(t) = \{3, 1, 1, 3, 2, 2\}$, and $X_3(t) = \{1, 2, 2, 3, 1, 2\}$. In this example, each $X_i(t), i = 1, 2, 3$ is a candidate solution (chromosome) and $\bigcup_{r=1}^{3} X_r(t)$ represents the population. In the first chromosome, the first, third, and fifth IoT edge tasks are assigned to the second cluster, and the second and the last tasks are assigned to cluster one, while the fourth task is allocated to the third cluster.

Parameters and Variables:

- T: Set of tasks, where $T = \{T_1, T_2, \ldots, T_n\}$.
- C: Set of clusters, where $C = \{C_1, C_2, \ldots, C_k\}$.
- R: Set of IoT edge computing resources.
- $X[i][j][k]$: A binary decision variable representing the assignment of task T_I to cluster C_j and resource R_k.
- $X[i][j][k] = 1$ if T_i is assigned to C_j and resource R_k.
- $X[i][j][k] = 0$ otherwise.
- $E[i][j][k]$: Execution time of task T_I in cluster C_j on resource R_k.
- $Cap[k]$: Capacity of resource R_k.
- Objective function:

The objective is to minimize the total execution time of the tasks assigned to clusters and resources as in Equation (3):

$$Minimize\ Z = \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{r} X[i][j][k] \times E[i][j][k] \tag{3}$$

- Constraints:

Each task is assigned to exactly one cluster and one resource as in Equation (4):

$$\sum_{k=1}^{k} \sum_{r=1}^{r} X[i][j][k] \times E[i][j][k] = 1 \ \ \forall_i \in T \tag{4}$$

Each cluster is assigned to only one resource as in Equation (5):

$$\sum_{r=1}^{r} X[i][j][k] \times E[i][j][k] = 1 \ \forall_j \in C \tag{5}$$

- Capacity constraint:

The total number of tasks assigned to a resource should not exceed its capacity as in Equation (6):

$$\sum_{i=1}^{n} \sum_{j=1}^{k} X[i][j][k] \leq Cap[k] \tag{6}$$

4.2. Measures of Similarity and Dissimilarity

Define a similarity function $S(\mathrm{T_m}, \mathrm{T_n})$ for IoT edge tasks T_m and T_n based on the given properties. The properties are

- Task_Length
- Task_FileSize
- Num_PE (number of CPUs)
- Task_priority

For each property p of an IoT edge task T, denote its value as $p(\mathrm{T})$.

Using Equation (3) ensures the similarity between two tasks is 1 when the property values of the tasks are identical and it decreases as the variation between IoT edge task property values increases.

Then, the proposed mechanism calculates the overall similarity $S(T_m, T_n)$ by finding the average of the similarities across the remaining properties as in Equation (7):

$$S(T_m, T_n) = \frac{1}{4}(S_{tasklength}(T_m, T_n) + S_{taskfilesize}(T_m, T_n) + S_{taskNuPE}(T_m, T_n) + S_{taskpriority}(T_m, T_n)) \quad (7)$$

The proposed DE mechanism uses the similarity in Equation (3) to ensure that using tasks with similar characteristics will have a high $S(\mathrm{T_m}, \mathrm{T_n})$ value, while those with dissimilar properties will have a low $S(\mathrm{T_m}, \mathrm{T_n})$ value. The proposed DE uses these measurements to maintain the clustering process in IoT edge computing.

For each property (p) of an IoT edge task, (T) denotes its value as $(\mathrm{p(T)})$. The similarity for each property (p) between tasks $(\mathrm{T_m})$ and $(\mathrm{T_n})$ is defined as in Equation (8):

$$S_p(T_m, T_n) = \frac{1}{1 + |p(T_m) - p(T_n)|} \quad (8)$$

Using Equation (5), the exact similarity measures for each of the characteristics are stated in Equations (9)–(12):

$$(S_{tasklength}(T_m, T_n) = \frac{1}{1 + |Task\_Length(T_m) - Task\_Length(T_n)|}) \quad (9)$$

$$(S_{taskfilesize}(T_m, T_n) = \frac{1}{1 + |Task\_FileSize(T_m) - Task\_FileSize(T_n)|}) \quad (10)$$

$$(S_{numpe}(T_m, T_n) = \frac{1}{1 + |Num\_PE(T_m) - Num\_PE(T_n)|}) \quad (11)$$

$$(S_{taskpriority}(T_m, T_n) = \frac{1}{1 + |Task\_priority(T_m) - Task\_priority(T_n)|}) \quad (12)$$

4.3. The Architecture of the Proposed DE Mechanism

The architecture of the proposed IoT edge task clustering mechanism is described in Figure 2. The architecture consists of three main layers. Furthermore, each layer of the proposed method architecture is illustrated.

The IoT edge task clustering process uses the task's length and the resource required information to divide the submitted tasks to the suitable cluster. The proposed clustering mechanism aims to map the task with suitable workload resources using differential evolution optimization. The appropriate allocation of IoT edge tasks to edge resources improves the overall performance of IoT edge computing. For example, if an IoT edge task is mapped to a high-task load cluster, then it may be scheduled to a low-workload IoT edge resource, and if the IoT edge task is mapped to a low-task load cluster, then it may be scheduled to a high-workload cluster.

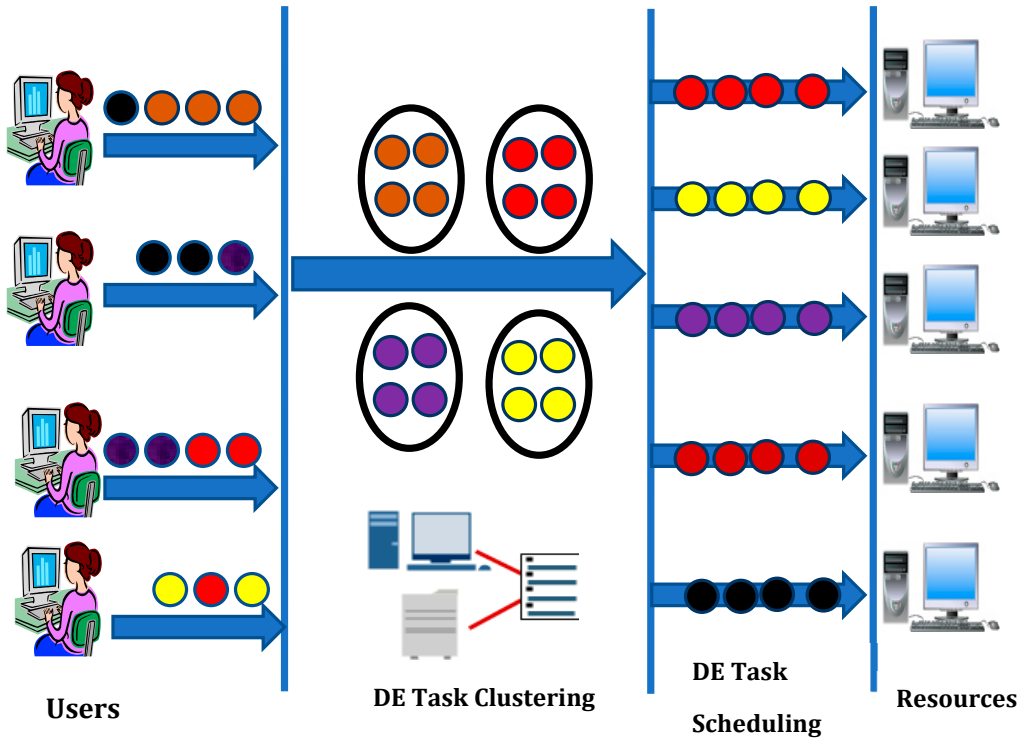

Figure 2. The architecture of the proposed DE for IoT edge task clustering and scheduling.

Figure 2 describes the proposed evolutionary algorithm for task clustering and scheduling in IoT edge computing architecture. The following is a detailed description based on the components and the process flow of the architecture:

4.3.1. Task Submission

The process starts with tasks being submitted to the IoT edge system. The attributes of the submitted IoT edge tasks include task length and resource requirements. As shown in Figure 2, the submitted IoT edge tasks are represented using different colors, indicating different task lengths and resource requirements.

4.3.2. Clustering Process

The first step of the proposed DE mechanism involves clustering the submitted tasks based on their characteristics, such as task length and resource requirements. The proposed DE mechanism divides the submitted tasks into clusters based on their characteristics. The clustering process in the proposed DE mechanism aims to efficiently identify IoT edge tasks with similar load characteristics to allocate them to suitable resources.

4.3.3. Scheduling Process

The next step after clustering the task is the scheduling process. The scheduling process aims to map each IoT edge task cluster to the appropriate resource. The scheduling mechanism considers the workload of resources to minimize the task execution times. The IoT edge tasks with high workloads are allocated to resources with lower current workloads and vice versa.

4.4. The Objective Function

The set of tasks $T = \{T_1, T_2, \ldots, T_n\}$ are divided into non-overlapping clusters $C = \{C_1, C_2, \ldots, C_k\}$ where C_i is the cluster number i, and the k value is determined using the proposed DE clustering mechanism. The IoT edge task clustering mechanism has three main features:

1. There is no IoT edge task belonging to more than one cluster, that is, $C_i \cap C_j = \emptyset$ for $\forall i \neq j$
2. Each IoT edge task is allocated to a cluster, that is, $\bigcup_{i=1}^{k} C_i = T$
3. There is no empty cluster, that is, $C_i \neq \emptyset$ for $\forall i \in \{1, 2, \ldots, k\}$.

4.5. Pseudo Code of the Proposed DE Algorithm

The pseudo code of the proposed DE for IoT edge task clustering and scheduling describes in detail the steps of the proposed mechanism to achieve its goals (refer to Algorithm 2).

Algorithm 2 The proposed DE for IoT edge Task clustering and Scheduling

```
Begin
num_clusters = <number of clusters to form>
num_tasks = <number of tasks to cluster>
pop_size = <population size>
F = <weighting factor>
CR = <probability of crossover>
Initialize population
for i = 1 to pop_size do
    population[i] = <randomly generate a clustering solution>
Evaluate objective function for initial population
    fitness[i] = <evaluate objective function for population[i]>
end for
for iter = 1 to max_iter do
    Generate offspring
    for i = 1 to pop_size do
        Select parents
        a, b, c = <select three distinct solutions from population>
                Generate mutant vector
        mutant = a + F × (b − c)
                Perform crossover
        for j = 1 to num_tasks do
            if rand(0, 1) < CR or j == rand(1, num_tasks) then
                offspring[i][j] = mutant[j]
            else
                offspring[i][j] = population[i][j]
                 Evaluate objective function for offspring
        end for
        if offspring[i][j] < lower_bound or offspring[i][j] > upper_bound then
        offspring[i][j] = <correct to nearest feasible value within the bounds>
        end if
        offspring_fitness[i] = <evaluate objective function for offspring[i]>
         Select best-performing solution from population and offspring
    end for
    for i = 1 to pop_size do
        if offspring_fitness[i] < fitness[i] then
population[i] = offspring[i]
            fitness[i] = offspring_fitness[i]
        Display progress
    end for
    if iter % display_interval == 0 then
        print("Iteration", iter, "best fitness", min(fitness))
end for
Return best-performing solution from population
return population[argmin(fitness)]
```

4.6. The Proposed DE Mechanism Configurations

The experiments were conducted using a custom-built software application running on the simulation. The simulations were conducted on a DELL server equipped with an Intel(R) Core (TM) i9-10900 CPU featuring a 2.80 GHz clock speed, nine cores, and 32 GB of memory. The simulation experiments focus on IoT edge task execution time as the key performance metric. The simulation environment mimicked a private IoT edge computing framework supported by two processing servers: AMD and Intel.

The proposed DE mechanism has applied a total of 50 iterations for each set of parameters and conditions. This number of repetitions was chosen to balance computational feasibility with the need for statistical robustness. By conducting multiple runs of experiments, the proposed DE was able to mitigate the effects of randomness and variability inherent in the mechanism. This allows the simulation results to capture a more accurate and representative performance measure of the proposed DE mechanism.

In the simulation experiments of the proposed DE for task clustering and scheduling in the IoT edge within this study, specific parameter configurations were applied to improve the simulation performance. The mutation scaling factor, represented as F, was set at 0.5, and the crossover rate, CR, was established at 0.7. The selected values were based on preliminary simulation experiments that showed their effectiveness in balancing the exploratory and exploitative capabilities of the proposed DE mechanism. The selected mutation scaling factor of the simulation experiments falls within the widely recommended range of 0.4 to 0.9, while the crossover rate aligns with the commonly suggested range of 0.5 to 0.9.

The population size of the proposed DE mechanism was set to 50. This population size was determined to be sufficient for achieving a comprehensive search of the scheduling space without incurring excessive computational costs.

To tackle the stochastic nature of the differential evolution (DE) employed in the proposed mechanism, this study acknowledges that stochastic characteristics can generate varying results with each run of the experiments. To address this challenge and guarantee the stability and reliability of the proposed mechanism for task clustering and scheduling, the experiments conducted multiple iterations of the proposed mechanism under identical conditions. The experiments executed the proposed DE mechanism 100 times for each simulation scenario, maintaining consistent parameter configurations across all runs. This approach permitted the proposed DE mechanism to capture the inherent variability of the DE optimization and measure its performance comprehensively. The results presented in this study represent the aggregate findings of these multiple executions, thus providing a more accurate and consistent evaluation of the proposed DE mechanism for IoT task clustering and scheduling.

5. Performance Evaluations

To assess the efficacy of the proposed "Evolutionary Algorithm for Task Clustering and Scheduling in IoT Edge Computing", this study conducted several simulation experiments. The proposed DE mechanism was benchmarked against well-established optimization methods: the Firefly Algorithm (FA) and Particle Swarm Optimization (PSO). FA and PSO were selected due to their proven track records in solving task clustering and scheduling in IoT edge computing. Execution time, measured in milliseconds, was the main metric for the performance evaluation of the proposed DE mechanism. Execution time compared the computational cost of each of the three mechanisms under study. The experiment process was conducted in a controlled test environment that simulated an IoT edge computing scenario. This simulation configuration was important for proving the relevance of the proposed DE performance outcomes to the IoT edge environment. Parameter configuration for the three mechanisms was carefully selected based on a combination of the literature review and preliminary experiments and tests to ensure a fair comparison. IoT edge resources were standardized for all experiments to ensure a consistent comparison platform for the three mechanisms. The resource configuration included fixed processor speed,

memory, and other resource characteristics that affect execution time. The software and hardware specifications used in the simulation experiments are described in Section 5.1. Each experiment was repeated under identical circumstances to minimize variance and ensure the accuracy of the simulation results. Considering the proposed DE mechanism adaptability and scalability across different loads, two distinct types of loads, designated as 'lightweight' and 'heavyweight', were integrated into the IoT edge simulation environment. The lightweight load represented scenarios with fewer IoT edge tasks. In contrast, the heavy workload simulated circumstances requiring significant computational power similar to real-world tasks running on the edge of IoT networks. By including these different loads, the simulation results comprehensively reflected the proposed DE mechanism performance spectrum and potential adaptability to real-world IoT edge environments.

5.1. The First Scenario: Lightweight Workload

In this scenario, the simulation experiments considered a lightweight workload trace with the number of 10 jobs, 13 resources, and 100 iterations.

5.1.1. Test Case 1: Execution Times of the Proposed DE Algorithm for 100 Iterations

The results of the experiments of 100 iterations are described in Figure 3.

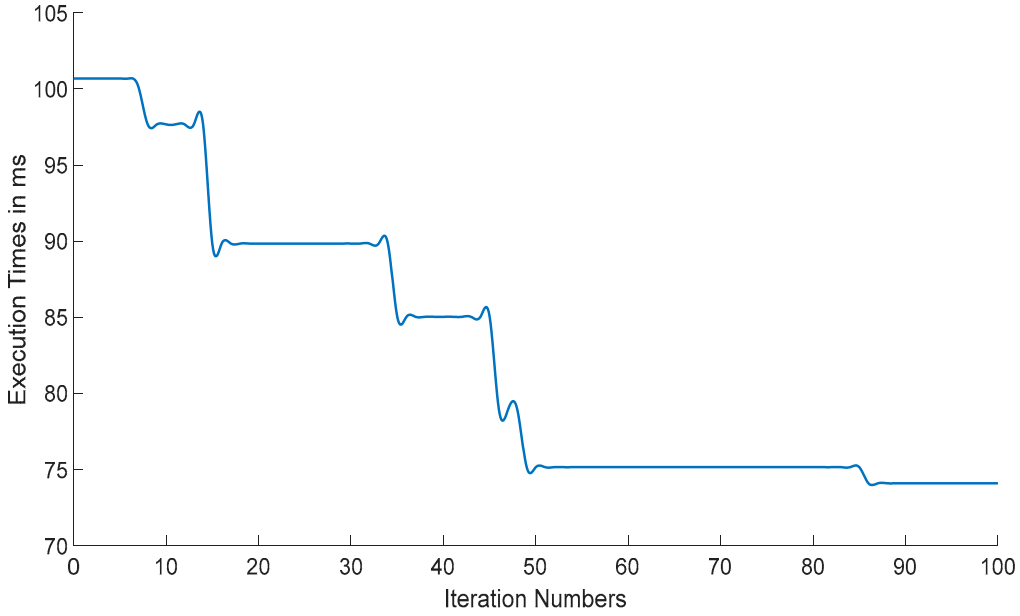

Figure 3. The execution times of the proposed DE algorithm for 100 iterations in Lightweight Workload Scenario.

Figure 3 shows the results of the proposed DE for task clustering and scheduling over 100 iterations. The experiment results provide insight into the behaviors and the efficiency of the proposed DE task for IoT edge task clustering and scheduling for lightweight loads. The execution time of the proposed DE mechanism started at 100.68 milliseconds. Considering iterations 7–14, there is a minor decrease as the execution time drops to 100.11 milliseconds, and then more markedly, the execution times decrease to 97.66 milliseconds. For iterations 15–34, the execution time of the proposed DE mechanism decreased to 89.84 milliseconds and then remained constant for 20 iterations. In the following iterations 35–46, the simulation results show that the execution time decreased to 85.03 milliseconds, and then was sustained over 12 iterations. Then, the proposed DE mechanism execution time has another drop to around 78.94 milliseconds, indicating scheduling enhancement. From iteration 47 onward, the proposed DE mechanism execution time significantly reduces to and stabilizes at 75.15 milliseconds. This indicates that the proposed DE mechanism has found an optimal or near-optimal schedule for the submitted IoT edge task, and this opti-

mization is consistent for a considerable number of simulation iterations. In the final stages (iterations 92–100), the DE mechanism execution time drops slightly to 74.09 milliseconds.

The simulation results of the lightweight load revealed that the proposed DE task clustering and scheduling mechanism exhibits multiple optimization stages. Starting with an initial high execution time, the proposed DE mechanism experiences various refinement phases, each with a consistent execution time. The noticeable stability of the proposed DE mechanism at certain execution times demonstrates that the proposed DE mechanism has reached temporary plateaus or local optima before moving to a better schedule.

5.1.2. Test Case 2: The Execution Times of the Proposed DE Compared to FA

The second test case compares the results of the execution times of the proposed DE for IoT task clustering and scheduling with FA and PSO, as described in Table 1 and Figure 4.

Table 1. The execution times of the proposed DE algorithm Compared to FA and PSO in Lightweight Workload Scenario.

Iteration No.	FA	PSO	DE
1	139.93	104.91	100.68
10	121.06	82.614	97.66
20	118.94	82.614	89.84
30	118.94	82.614	89.84
40	118.94	82.614	85.03
50	118.94	82.614	75.15

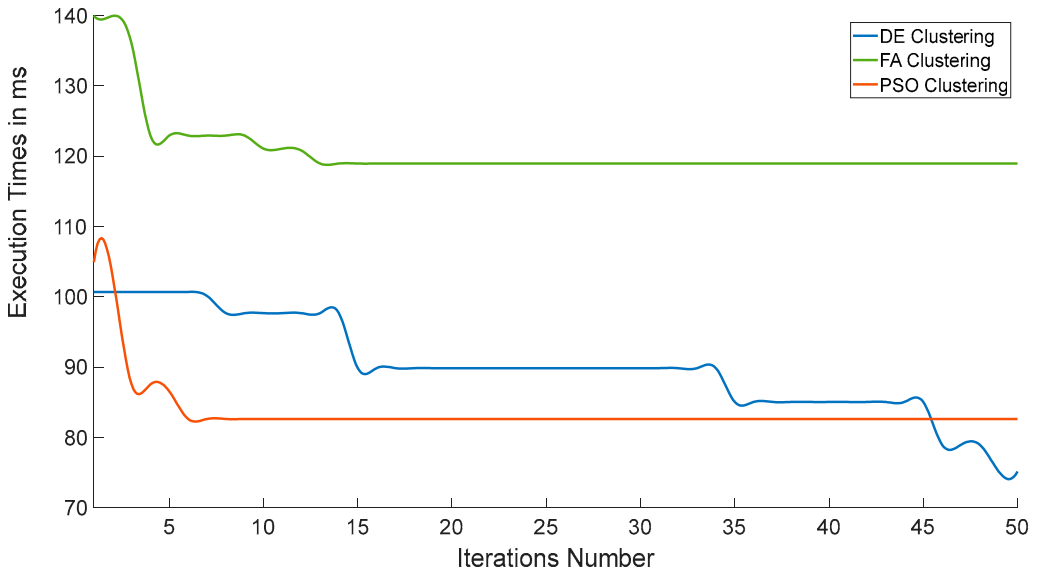

Figure 4. The execution times of the proposed DE algorithm compared to FA and PSO in Lightweight Workload Scenario.

Figure 4 and Table 1 describe the performance evaluation of the proposed differential evolution (DE) mechanism for task clustering and scheduling in IoT edge computing under a lightweight load, compared to the Firefly Algorithm (FA) and Particle Swarm Optimization (PSO). DE demonstrates superior efficiency at the first iteration with the lowest execution time of 100.68, followed closely by PSO at 104.91, while FA lags at 139.93. As the simulation iterations progress, mainly noticeable from the 10th to the 40th, the proposed DE mechanism steadily improves its performance, achieving an execution time of 85.03 by the 40th iteration. In contrast, the PSO mechanism results reach execution time

plateaus early, preserving a constant at 82.61 from the 10th iteration onwards. FA execution times reveal some improvement; however, it remains the least efficient throughout the simulation iterations, stabilizing at an execution time of 118.94. At the final stage at the 50th iteration, the proposed DE mechanism has achieved the shortest execution time of 75.15, while the PSO mechanism execution time remains unchanged, and the FA mechanism continues to trail at 118.94.

5.2. The Second Scenario: Heavyweight Workload

The simulation experiments considered heavyweight workload traces with 100 iterations in this scenario.

5.2.1. Test Case 3: Execution Times of the Proposed DE Algorithm for 100 Iterations

The results of the experiments of 100 iterations are described in Figure 5.

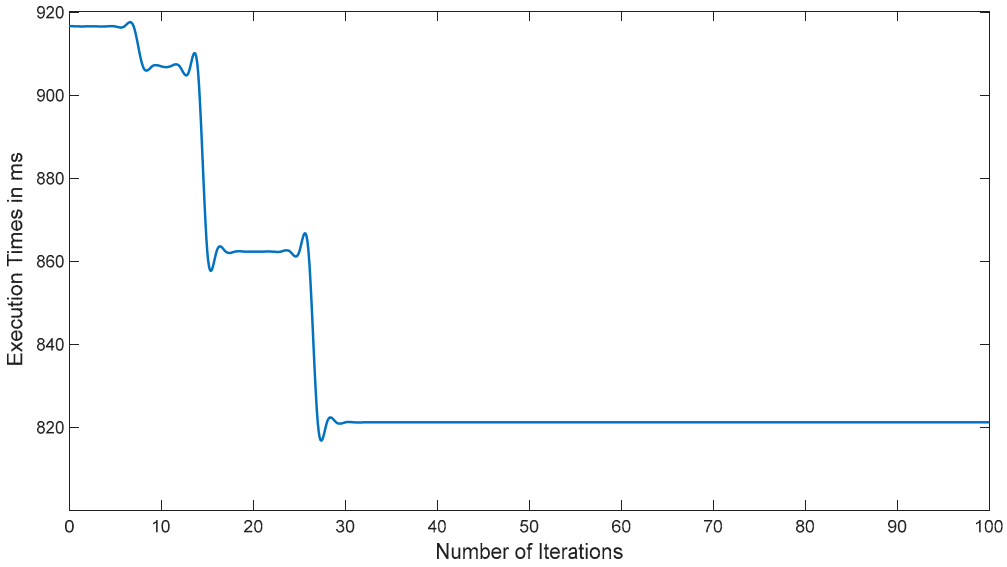

Figure 5. The execution times of the proposed DE algorithm for 100 iterations in Heavyweight Workload Scenario.

Figure 5 presents the results of the execution time of the proposed DE mechanism over 100 iterations for a heavy load. At the first stage of the simulation, the execution time results show an obvious plateau at 916.60 milliseconds, which appears consistently over several iterations. Following this, the results show a reduction in the proposed DE mechanism execution time to 906.98 and 905.65 milliseconds, respectively. The most significant improvement in the simulation execution time of the proposed DE mechanism was observed when the times decreased to 862.38 milliseconds, and finally, the execution time stabilized at 821.26 milliseconds for most of the subsequent iterations. This trend reveals the optimization phase in the early iterations, where the proposed DE mechanism refines its clustering method, achieving more efficient results as iterations progress.

In this result, the repetitive occurrence of identical execution times indicates periods of algorithmic convergence, where the proposed DE consistently finds optimal or near-optimal schedules. In some cases, the constant execution times of the proposed DE mechanism are because the proposed DE mechanism reaches a local optimum and struggles to find a pathway to better schedules.

In summary, the simulation results demonstrate that the proposed DE-based clustering mechanism shows promise, particularly in optimizing the edge resource scheduling process. The decreasing results of execution times signify optimization over iterations, crucially leading to faster decision-making procedures in real-world IoT edge computing scenarios.

5.2.2. Test Case 4: The Execution Times of the Proposed DE Compared to PSO and FA

This test case compares the results of the execution times of the proposed DE for IoT task clustering and scheduling with PSO and FA.

Table 2 and Figure 6 illustrates the execution times of the proposed DE algorithm compared to FA and PSO.

Table 2. The execution times of the proposed DE algorithm Compared to FA and PSO in Heavyweight Workload Scenario.

Iteration No.	FA	PSO	DE
1	1092.00	1026.09	0916.61
10	1053.06	956.12	0906.99
20	1053.06	956.12	862.38
30	1053.06	956.12	821.27
40	1053.06	956.12	821.27
50	1053.06	956.12	821.27

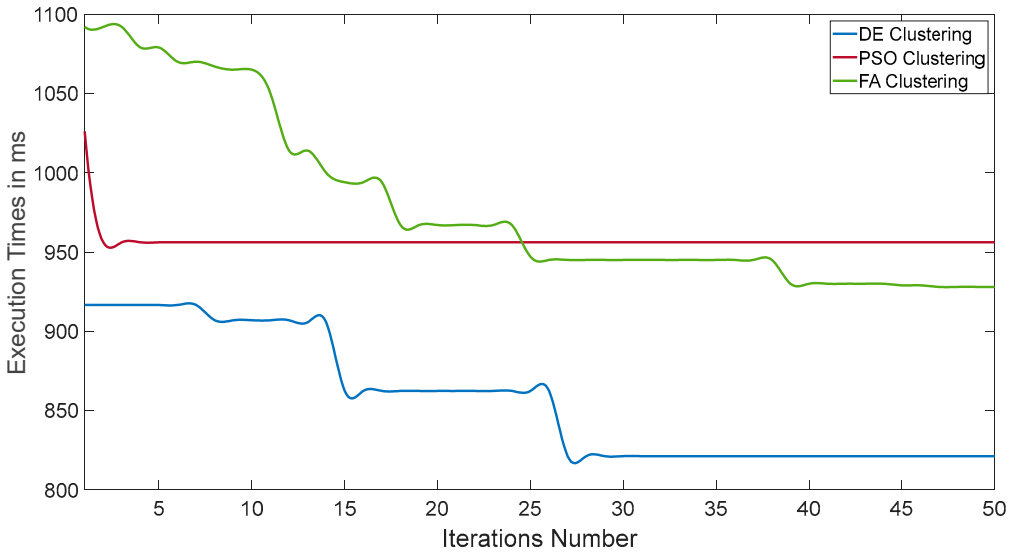

Figure 6. The execution times of the proposed DE algorithm compared to FA and PSO in Heavyweight Workload Scenario.

Figure 6 and Table 2 display the execution times of the proposed differential evolution (DE) clustering and scheduling mechanism compared to Particle Swarm Optimization (PSO) and Firefly Algorithm (FA) clustering and scheduling mechanisms. The simulation results for the three mechanisms are calculated through a series of iterations, from 1 to 50. The proposed DE clustering and scheduling mechanism (depicted in blue) has the lowest execution times across all iterations compared to FA and PSO. The proposed DE mechanism starts at 916.61 milliseconds, experiences a gradual decrease, and then stabilizes after around 15 iterations. The PSO clustering and scheduling mechanism (in green) starts with the highest execution times at the beginning of the simulation (above 1000 milliseconds). In the second stage, the execution time for the PSO mechanism then decreases rapidly and stabilizes at 956.12 milliseconds for the remainder of the simulation iterations. This indicates that the PSO clustering mechanism has a longer initialization time; nonetheless, the execution time becomes more stable as the simulation iterations progress. The FA clustering and scheduling mechanism (in red) begins with execution times above 1000 milliseconds. FA remains relatively flat throughout the simulation

iterations, representing steady performance with a small change in the execution time as the simulation iterates.

In summary, the proposed DE clustering and scheduling mechanism is the most efficient mechanism compared to FA and PSO regarding execution time. Despite the fact that the PSO mechanism starts with higher execution times, it decreases and stabilizes reasonably quickly. The FA mechanism revealed consistent scheduling performance but at higher execution times than DE. Overall, the simulation results demonstrate that if execution time is a critical performance metric for the edge system, the proposed DE is the suitable mechanism. The proposed DE mechanism for task clustering and scheduling shows both low execution time and minimal variation in that time as the simulation progresses.

5.3. Discussion and Comparison with Related Works

This section presents a comparative analysis of the proposed differential evolution (DE) mechanism against other recent works in the field, such as the Firefly Algorithm (FA) and Particle Swarm Optimization (PSO). For instance, Alhaizaey et al. [20] demonstrated the application of PSO in IoT environments, achieving an average execution time of 967.78 ms, which contrasts with the proposed DE mechanism time of 858.29 ms in similar conditions. Similarly, Yousif et al. [6] applied FA in a comparable context, but with an execution time of 1059.55 ms, underscoring the proposed DE's efficiency.

The study also explored the practical applications of the proposed DE mechanism for IoT task clustering and scheduling in real-world scenarios. The proposed DE mechanism reduced application execution times and showed adaptability in various load conditions, which make the proposed DE mechanism particularly suitable for real-time data processing application in IoT edge computing. These enhancements to IoT edge computing performance could extend its applications in smart city infrastructure, healthcare monitoring systems, and automated industrial processes, where efficient task scheduling and processing is critical.

Further research will investigate the integration of DE in different computational environments, such as cloud computing and fog computing. Furthermore, future research will validate the DE mechanism against other fast-converging and efficient optimization algorithms such as the Arithmetic Optimization Algorithm (AOA) and the Archimedes Optimization Algorithm (AHA). This can provide deeper insights into its adaptability and potential as a general optimization mechanism.

6. Conclusions and Future Works

This study proposed an evolutionary algorithm for task clustering and scheduling in IoT edge computing. The proposed mechanism utilized differential evolution optimization to enhance task clustering and scheduling progress. The proposed mechanism has been comprehensively assessed against benchmark algorithms FA and PSO. The experimental results demonstrate that the proposed DE mechanism exhibits the highest performance, with the lowest execution times and consistent behavior throughout the iterative simulation progress. The proposed DE mechanism's robustness in handling diverse IoT edge task loads and scalability within the IoT edge computing makes it a valuable contribution to the field. Furthermore, the proposed DE adaptability to different IoT edge task constraints and loads promises significant implications for future IoT edge computing. The proposed DE-based mechanism provides a feasible solution for real-time IoT edge task clustering and scheduling challenges. This facilitates the efficient utilization of IoT edge resources in the growing IoT field. Future work will focus on refining the proposed DE mechanism performance with complex, real-world datasets and extending its application to other domains within the IoT ecosystem.

Author Contributions: Conceptualization, A.Y.; methodology, A.Y.; original draft preparation, A.Y.; review and editing, M.B.B. and A.A.; visualization, M.B.B. and A.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research at Najran University, under the Distinguish Research Funding Program (grant code: NU/DRP/SERC/12/22).

Data Availability Statement: All data underlying the results are available as part of the article, and no additional source data are required.

Acknowledgments: The authors are thankful to the Deanship of Scientific Research at Najran University for funding this work, under the Distinguish Research Funding Program (grant code: NU/DRP/SERC/12/22).

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

References

1. Ullah, I.; Youn, H.Y. Task classification and scheduling based on K-means clustering for edge computing. *Wirel. Pers. Commun.* **2020**, *113*, 2611–2624. [CrossRef]
2. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource scheduling in edge computing: A survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2131–2165. [CrossRef]
3. Zain, A.M.; Yousif, A. Chemical reaction optimization (CRO) for cloud job scheduling. *SN Appl. Sci.* **2020**, *2*, 53. [CrossRef]
4. Suliman, Y.M.; Yousif, A.; Bashir, M.B. Shark Smell Optimization (SSO) Algorithm for Cloud Jobs Scheduling. In Proceedings of the First International Conference on Computing, ICC 2019, Riyadh, Saudi Arabia, 10–12 December 2019; pp. 71–80.
5. Yousif, A. An Enhanced Firefly Algorithm for Time Shared Grid Task Scheduling. *Appl. Artif. Intell.* **2021**, *35*, 1567–1586. [CrossRef]
6. Khaleel, M.I. Efficient job scheduling paradigm based on hybrid sparrow search algorithm and differential evolution optimization for heterogeneous cloud computing platforms. *Internet Things* **2023**, *22*, 100697. [CrossRef]
7. Hassan, M.E.; Yousif, A. Cloud Job Scheduling with Ions Motion Optimization Algorithm. *Eng. Technol. Appl. Sci. Res.* **2020**, *10*, 5459–5465. [CrossRef]
8. Abdel-Basset, M.; Mohamed, R.; Abd Elkhalik, W.; Sharawi, M.; Sallam, K.M. Task Scheduling Approach in Cloud Computing Environment Using Hybrid Differential Evolution. *Mathematics* **2022**, *10*, 4049. [CrossRef]
9. Grabmeier, J.; Rudolph, A. Techniques of cluster algorithms in data mining. *Data Min. Knowl. Discov.* **2002**, *6*, 303–360. [CrossRef]
10. Lorpunmanee, S. Adaptive Intelligent Grid Scheduling System. Ph.D. Thesis, Universiti Teknologi Malaysia, Skudai, Malaysia, 2010.
11. Yousif, A.; Abdullah, A.H.; Latiff, M.S.B.A.; Bashir, M.B. A Taxonomy of Grid Resource Selection Mechanisms. *Int. J. Grid Distrib. Comput.* **2011**, *4*, 107–118.
12. Foster, I.; Kesselman, C.; Nick, J.; Tuecke, S. The physiology of the grid. In *Grid Computing: Making the Global Infrastructure a Reality*; John Wiley & Sons Ltd.: Chichester, UK, 2003; pp. 217–250.
13. Liu, H.; Abraham, A.; Hassanien, A.E. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Gener. Comput. Syst.* **2010**, *26*, 1336–1343. [CrossRef]
14. Izakian, H.; Tork Ladani, B.; Zamanifar, K.; Abraham, A. A novel particle swarm optimization approach for grid job scheduling. In *Information Systems, Technology and Management, Proceedings of the Third International Conference, ICISTM 2009, Ghaziabad, India, 12–13 March 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 100–109.
15. Lyu, C.; Shi, Y.; Sun, L.; Lin, C.-T. Community detection in multiplex networks based on evolutionary multi-task optimization and evolutionary clustering ensemble. *IEEE Trans. Evol. Comput.* **2022**, *27*, 728–742. [CrossRef]
16. Mousa, M.H.; Hussein, M.K. Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization. *PeerJ Comput. Sci.* **2022**, *8*, e870. [CrossRef]
17. Kim, M.-S.; Han, J. A particle-and-density based evolutionary clustering method for dynamic networks. *Proc. VLDB Endow.* **2009**, *2*, 622–633. [CrossRef]
18. Alguliev, R.; Aliguliyev, R. Evolutionary Algorithm for Extractive Text Summarization. *J. Intell. Inf. Manag.* **2009**, *1*, 128–138. [CrossRef]
19. Pan, L.; Liu, X.; Jia, Z.; Xu, J.; Li, X. A multi-objective clustering evolutionary algorithm for multi-workflow computation offloading in mobile edge computing. *IEEE Trans. Cloud Comput.* **2021**, *11*, 1334–1351. [CrossRef]
20. Alhaizaey, Y.; Singer, J.; Michala, A.L. Optimizing Heterogeneous Task Allocation for Edge Compute Micro Clusters Using PSO Metaheuristic. In Proceedings of the 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 12–15 December 2022; pp. 1–8.
21. Azimi, S.; Pahl, C.; Shirvani, M.H. Particle Swarm Optimization for Performance Management in Multi-cluster IoT Edge Architectures. In Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020), Online, 7–9 May 2022; pp. 328–337.
22. Imteaj, A.; Thakker, U.; Wang, S.; Li, J.; Amini, M.H. A survey on federated learning for resource-constrained IoT devices. *IEEE Internet Things J.* **2021**, *9*, 1–24. [CrossRef]
23. Guo, T.; Yu, K.; Aloqaily, M.; Wan, S. Constructing a prior-dependent graph for data clustering and dimension reduction in the edge of AIoT. *Future Gener. Comput. Syst.* **2022**, *128*, 381–394. [CrossRef]

24. Han, J.; Wang, H.; Wu, S.; Wei, J.; Yan, L. Task scheduling of high dynamic edge cluster in satellite edge computing. In Proceedings of the 2020 IEEE World Congress on Services (SERVICES), Beijing, China, 18–24 October 2020; pp. 287–293.
25. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2348–2359. [CrossRef]
26. Bouet, M.; Conan, V. Mobile edge computing resources optimization: A geo-clustering approach. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 787–796. [CrossRef]
27. Irani, J.; Pise, N.; Phatak, M. Clustering techniques and the similarity measures used in clustering: A survey. *Int. J. Comput. Appl.* **2016**, *134*, 9–14. [CrossRef]
28. Saxena, A.; Prasad, M.; Gupta, A.; Bharill, N.; Patel, O.P.; Tiwari, A.; Er, M.J.; Ding, W.; Lin, C.-T. A review of clustering techniques and developments. *Neurocomputing* **2017**, *267*, 664–681. [CrossRef]
29. Iglesias, F.; Kastner, W. Analysis of similarity measures in times series clustering for the discovery of building energy patterns. *Energies* **2013**, *6*, 579–597. [CrossRef]
30. Dautov, R.; Distefano, S. Automating IoT data-intensive application allocation in clustered edge computing. *IEEE Trans. Knowl. Data Eng.* **2019**, *33*, 55–69. [CrossRef]
31. Song, Y.; Wu, D.; Deng, W.; Gao, X.-Z.; Li, T.; Zhang, B.; Li, Y. MPPCEDE: Multi-population parallel co-evolutionary differential evolution for parameter optimization. *Energy Convers. Manag.* **2021**, *228*, 113661. [CrossRef]
32. Mohamed, A.W.; Sabry, H.Z.; Abd-Elaziz, T. Real parameter optimization by an effective differential evolution algorithm. *Egypt. Inform. J.* **2013**, *14*, 37–53. [CrossRef]
33. Abd Elaziz, M.; Ewees, A.A.; Ibrahim, R.A.; Lu, S. Opposition-based moth-flame optimization improved by differential evolution for feature selection. *Math. Comput. Simul.* **2020**, *168*, 48–75. [CrossRef]
34. Faris, M.; Mahmud, M.N.; Salleh, M.F.M.; Alsharaa, B. A differential evolution-based algorithm with maturity extension for feature selection in intrusion detection system. *Alex. Eng. J.* **2023**, *81*, 178–192. [CrossRef]
35. Wang, C.; Yu, X.; Xu, L.; Wang, W. Energy-efficient task scheduling based on traffic mapping in heterogeneous mobile-edge computing: A green IoT perspective. *IEEE Trans. Green Commun. Netw.* **2022**, *7*, 972–982. [CrossRef]
36. Liu, X.; Yu, J.; Wang, J.; Gao, Y. Resource allocation with edge computing in IoT networks via machine learning. *IEEE Internet Things J.* **2020**, *7*, 3415–3426. [CrossRef]
37. Lin, Y.; Zhang, Y.; Li, J.; Shu, F.; Li, C. Popularity-aware online task offloading for heterogeneous vehicular edge computing using contextual clustering of bandits. *IEEE Internet Things J.* **2021**, *9*, 5422–5433. [CrossRef]
38. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
39. Price, K.V.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
40. Lorpunmanee, S.; Sap, M.; Noor, M.; Abdullah, A.H. Adaptive intelligence job online scheduling within dynamic grid environment based on gridsim. *J. Teknol. Mklm.* **2008**, *20*, 173–189.