sensors

MDPI

Article

Hyper-Angle Exploitative Searching for Enabling Multi-Objective Optimization of Fog Computing

Taj-Aldeen Naser Abdali [1], Rosilah Hassan [1,*], Azana Hafizah Mohd Aman [1], Quang Ngoc Nguyen [2] and Ahmed Salih Al-Khaleefa [1]

[1] Centre for Cyber Security, Faculty of Information Science and Technology (FTSM), Universiti Kebangsaan Malaysia, UKM, Bangi 43600, Malaysia; P94546@siswa.ukm.edu.my (T.-A.N.A.); azana@ukm.edu.my (A.H.M.A.); ahmed.salih89@siswa.ukm.edu.my (A.S.A.-K.)

[2] Department of Communications and Computer Engineering, Faculty of Science and Engineering, Waseda University, Tokyo 169-8050, Japan; quang.nguyen@aoni.waseda.jp

* Correspondence: rosilah@ukm.edu.my

Abstract: Fog computing is an emerging technology. It has the potential of enabling various wireless networks to offer computational services based on certain requirements given by the user. Typically, the users give their computing tasks to the network manager that has the responsibility of allocating needed fog nodes optimally for conducting the computation effectively. The optimal allocation of nodes with respect to various metrics is essential for fast execution and stable, energy-efficient, balanced, and cost-effective allocation. This article aims to optimize multiple objectives using fog computing by developing multi-objective optimization with high exploitive searching. The developed algorithm is an evolutionary genetic type designated as Hyper Angle Exploitative Searching (HAES). It uses hyper angle along with crowding distance for prioritizing solutions within the same rank and selecting the highest priority solutions. The approach was evaluated on multi-objective mathematical problems and its superiority was revealed by comparing its performance with benchmark approaches. A framework of multi-criteria optimization for fog computing was proposed, the Fog Computing Closed Loop Model (FCCL). Results have shown that HAES outperforms other relevant benchmarks in terms of non-domination and optimality metrics with over 70% confidence of the t-test for rejecting the null-hypothesis of non-superiority in terms of the domination metric set coverage.

Keywords: fog computing; task allocation; multi-objective optimization; evolutionary genetics; hyper-angle; crowding distance

check for updates

Citation: Naser Abdali, T.-A.; Hassan, R.; Mohd Aman, A.H.; Nguyen, Q.N.; Al-Khaleefa, A.S. Hyper-Angle Exploitative Searching for Enabling Multi-Objective Optimization of Fog Computing. *Sensors* **2021**, *21*, 558. https://doi.org/10.3390/s21020558

Received: 7 December 2020
Accepted: 8 January 2021
Published: 14 January 2021

1. Introduction

Internet of Things (IoT) has been used in several fields such as health care, environmental engineering, transportation, and safety [1,2]. The idea behind IoT is to connect physical items to the virtual world, so they can be controlled remotely and act as physical access points to Internet services [3]. These devices increased rapidly around the world and generate a huge amount of data, termed Big Data (BD) [4,5]. One of the fundamental challenges in IoT is the data transmissions [6,7] to the Cloud Computing (CC), which indicate to the infrastructure where both data storage and processing operate outside of the IoT devices [8,9].

CC data center is far from end-user, then causes high latency and affects the actual time constraints in many applications [10]. Therefore, CISCO [11] suggests the new paradigm Fog Computing (FC) to ensure reliable sending and receiving data between the Cloud and IoT devices [12]. Figure 1 gives a conceptual elaboration of the architecture of IoT, CC, and FC. The first layer is the IoT environment, this layer close to the user and the physical environment. It contains several devices such as mobile phones, sensors, smart cards, readers, and smart vehicles. The second layer fog layer this layer is located on the edge of the network means between IoT and cloud computing. This layer contains a huge number of fog nodes which generally including routers, gateways, switches, access points, base stations, and specific fog servers. The third layer is the cloud computing layer and consists of several effective servers and storage devices and provides various application services for smart homes, smart transportation, smart factories, and so on.

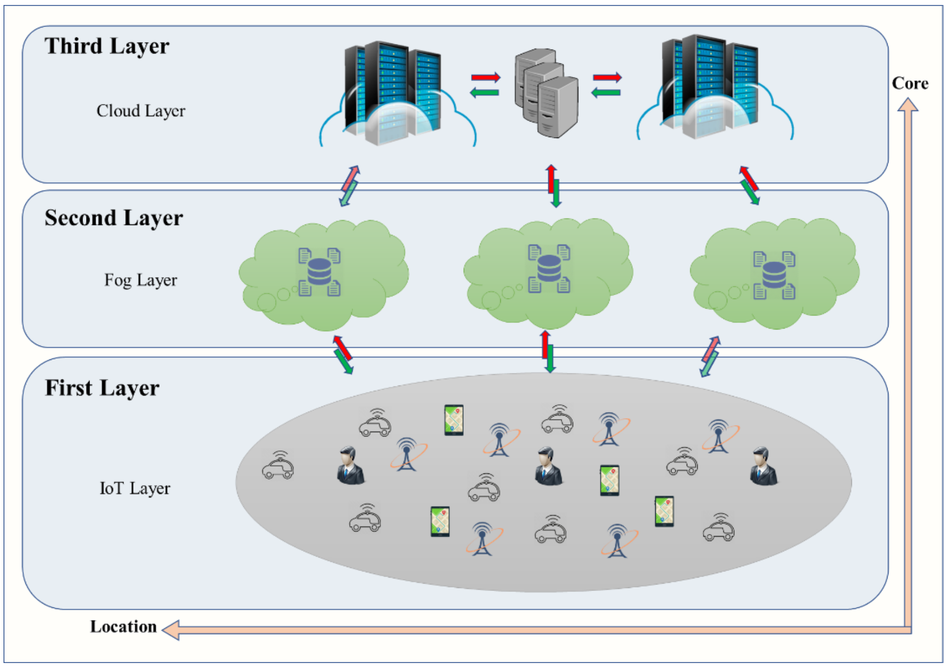


Figure 1. A basic conceptual framework of IoT, cloud computing, and fog computing.

The distributed nature of FC and the relatively limited computation, energy, and communication power of its nodes have motivated researchers to assure its load balancing aspect when various applications are required to be executed in FC. The load balancing of fog computing is accomplished by a set of methodological approaches named Task Allocation (TA) [13] in the literature. The term TA indicates allocating various network nodes optimally to execute a given task or application while maintaining various objectives. In the context of TA for FC, we are interested in dividing the given task into a set of sub-tasks with the independency aspect and dividing them on the network nodes with matching various constraints. Next, they will be presented with a mathematical model for calculating the various fog measures, including energy efficiency, cost-effectiveness, time latency, stability, and reliability. Having the ability to evaluate the candidate solution from the optimization and to provide its objectives values, we call Fog Computing Closed Loop (FCCL). This type of problem is regarded as a Non-Deterministic Polynomial Hard Problem (NP-hard) [14], which makes it a challenging optimization problem. This is due to the huge number of combinations of nodes' task allocation and the various conditions of the nodes and the tasks. Typical approaches for solving such a problem use a meta-heuristic family of optimization algorithms, and more specifically, the multi-objective type of meta-heuristic was enhanced to apply for fog computing to hold the huge number of tasks and set them based on their priority.

Multi-Objective Optimization (MOO) algorithms [15,16] aim at optimizing many objectives' functions using heuristic random searching in order to find a set of non-dominated solutions [17]. There is a high similarity between single objective [18] and multi-objective meta-heuristics [19,20] in the aspect of relying on a random pool of generated solutions, evaluating them, and selecting the best among them to generate off-spring. However, the essential difference between the single objective and multi-objective heuristic searching is the means of evaluating solutions. More specifically, in the multi-objective searching, the solutions are evaluated based on ranks that include a sub-set of non-dominated solutions instead of simple fitness value as in the single-objective optimization. Consequently, the goal of the MOO algorithm is to explore the solution space for finding maximum coverage of non-dominated solutions.

The goal of this article is to develop an optimization framework for computational fog computing. We aim to enable non-dominated optimization for fog computing by assuring high domination of the resulted decisions in terms of various performance metrics, which gives the decision-maker more flexibility as well as high achieved performance. Specifically, the integration of a novel hyper-angle exploitive searching optimization with the crowding distance of Non-Dominated Sorting Genetic Algorithm II (NSGA-II) in the context of fog computing optimization assists in providing more dominant solutions in terms of the fog measures that the decision-maker aims at optimizing. The article presents the following contributions.

- Proposing a fog computing optimization framework with multi-criteria perspectives. The multi-criteria cover the following metrics: Time Latency, Energy Consumption, Energy Distribution, Renting Cost, and Stability.
- Developing a novel optimization algorithm based on meta-heuristic genetic. The developed algorithm supports exploitive searching based on the hyper-angle indicator. We designate it as Hyper-Angle Exploitive Searching (HAES).
- Formulating a novel Fog Computing Closed Loop (FCCL) mathematical function and using HAES for optimizing it after discretization.
- Designing an adaptive objective partitioning by activating the sub-set of objectives at each iteration out of the entire objectives.
- Evaluating the developed HAES based on multi-objective optimization performance metrics and benchmarking mathematical functions and evaluating the optimized FCCL based on HAES, then analyzing its performance in comparison with other relevant optimization benchmarking algorithms.

2. Background and Literature Review

The article is focusing on multi-objective optimization for FC. Hence, the literature contains two phases. Firstly, the related work of the MOO algorithms is presented in Section 2.1, and Section 2.2 provides the related works of MOO fog computing optimization in.

2.1. MOO Algorithms

The studies on meta-heuristic-based MOO in the literature contain various approaches. Different criteria and techniques are used to generate the dominant Pareto Front (PF) and provide extensive exploration. In [21], a fitting function or interpolation method was applied from a finite set of objective values to calculate PF by selecting the individuals that have the shortest distance to the reference points based on the error matrix. The two algorithms, called MOGA/fitting and MOGA/interpolation, dealt with MOO without focusing on attaining the optimal solutions. Bao et al. [22] proposed Hierarchical NDS (HNDS), which focuses on reducing the number of comparisons in the search. HNDS initially sorts all the candidate solutions in ascending order, depending on their first objective. Next, HNDS compares the first solution with the rest of the candidate solutions, one by one, to make a speedy distinction by realizing different superiority solutions and then avoid the high number of unnecessary comparisons.

Other notable studies have extended the existing single-objective searching algorithms to multi-objective ones by introducing the concept of NSGA-II, which is fast NDS with crowding distance. This extension applies to Multi-Objective Vortex Searching (MOVS), which was proposed in [23]. MOVS uses the inverse incomplete gamma function with a parameter ranging from 0 to 1 to spread solutions over the PF. improved NSGA-II to make it more efficient and have better diversity by presenting a more efficient implementation of NDS, namely the dominance degree approach for NDS. Part and Select Algorithm (PSA) was also proposed to maintain diversity, and the entire algorithm after being integrated into NSGA-II was called Diversity DNSGA2–PSA. Additionally, several researchers have added a local search strategy to NSGA-II [24]. For example, the study in [25] proposed Heavy Perturbation (HP)-based NSGA-II. Two objectives, the size and total weight of a clique, were considered. In particular, the larger the size of a clique in terms of set inclusion is and the higher the total weight is, the better a solution is. HP-NSGA-II is then dedicated to the clique problem of a weighted graph with weights of vertices in which the perturbation is conducted by either improving a selected elite with a local search procedure or swapping its left and right parts.

Several types of research work also developed nature-inspired models for MOO. For instance, an improved method of GA based on an evolutionary computational model, namely the Physarum-Inspired Computational Model (PCM), was proposed in [26]. The initialization of the population used prior knowledge of PCM. Hill climbing was also used to improve the diversity of solutions, and the traveling salesman problem, which is one of the most classical NP-hard problems in combinatorial optimization, was utilized. Apart from improving the optimization of found solutions, several researchers have aimed at improving the searching speed. In the same context, [27] proposed an algorithm for MOO and compared it with four other competing algorithms on three different datasets to reduce the optimization complexity for a large number of objectives from $O\left(N \log^{M-1}\mathrm{N}\right)$ to $O\left(MN \ \log \ N + MN^2\right)$, where M denotes the number of objectives and N denotes the number of solutions. The algorithm removes unnecessary comparisons among solutions to improve the running time.

The work in [28] added the angle concept to crowd distance searching to balance the searching procedure among all angles. Other researchers have also used the framework of NSGA-II with different extensions. For example, [29] used a set of reference points while searching to maintain diversity. Then, from previous approaches, the concept of crowd distance, when combined with angle searching, achieves the extensive scope of the search. Specifically, authors in [30] have used range angle as a criterion to balance the search, then using it in finding criterion solutions as the goal of the study.

Overall, the previous research works that have focused on meta-heuristic for multi-objective aimed at incorporating various criteria for accomplishing exploration as well as exploitation. The crowding distance of NSGA-II is effective for exploration, while the angle searching was used in MOGA-AQCD as an additional base for the crowd-distance exploration. However, the angle usage for exploitation has not been explicitly considered and performed by the existing studies. This article then aims at tackling this aspect by proposing a novel MOO searching that incorporates angle searching for exploitation.

Particularly, the present paper proposes a MOO searching algorithm that uses crowding distance for exploration and angle searching for exploitation. The proposal optimizes the exploitation by selecting solutions from angular sectors that have the maximum found solutions. The crowding distance is also used for exploration; however, we aim at avoiding redundant operators for exploration. This goal is achieved by considering angle searching for exploitation, provided that the crowding distance has successfully played its role in the exploration process. To our knowledge, this is the first meta-heuristic searching algorithm for MOO that jointly considers and optimizes the angle criterion for exploitation and crowding distance for exploration at the same time. In the next section, we present the system models and the research background.

2.2. Fog Computing Optimization

Solving IoT challenges of data processing within real-time constraints have created the need to not rely on cloud network for processing. As a result, the concept of Fog Computing was first introduced by Cisco in 2012. However, congested networks, high latency in service delivery, and poor Quality of Service (QoS), non-stability, and increased cost have been experienced [31]. Such challenges have motivated researchers to focus on fog computing optimization.

The literature contains a significant amount of algorithmic works for fog computing optimization. Each work has focused on certain aspects of the fog network and followed a certain approach for optimization. While some work has tried to include more practical aspects of fog computing needs and nature, others were more simplified and ignored some crucial matters. In the work of [32], the authors have represented the fog computing optimization as a scheduling problem, where the algorithm has to assign tasks to nodes with assuring two objectives the stability and speed. Their model ignores energy and cost matters, which are considered to be crucial aspects of fog computing. On the other side, they used classical multi-objective optimization NSGA-II to solve their model without significant changes to explore the solution space more efficiently and find more dominant solutions. We find that other models have considered energy and cost like the work of [33]; however, there is no consideration of stability or reliability for finishing the work. Similarly, the work of [34] has included energy and latency while ignoring cost and reliability, while the work of [35] has included time latency and cost as objectives and it ignored energy and reliability.

A summary of the covered objectives of each model is given in Table 1. To the best of our knowledge, there is no developed model for fog computing optimization including four objectives: time latency, energy, cost, and reliability at the same time. Such inclusion implies more challenging multi-objective optimization. On the other side, all the previous works have applied NSGA-II and other similar non-dominated searching optimization without development in the searching aspect, which is needed because of the non-convex nature of the problem and a huge number of constraints resulting in the optimization surface non-linear and non-convex with NP-hard nature.

Table 1. Summary of the covered objectives in the fog computing model in the literature.

Authors/Objectives	Energy Consumption	Renting Cost	Stability	Time Latency	Energy Distribution
[32]	✗	✗	✓	✓	✗
[33]	✗	✗	✓	✗	✓
[34]	✓	✗	✗	✓	✗
[35]	✗	✓	✗	✓	✗
Proposed Model	✓	✓	✓	✓	✓

3. Proposed Methodology

This section presents the developed method for accomplishing the goal of the article. It starts with presenting the problem formulation of optimization and fog computing framework was provided in Section 3.1. Next, in Section 3.2, we provide the algorithm named hyper-angle exploitive searching. The fog computing closed-loop model is given in Section 3.3. Table 2 elaborates on the mathematical terms used in the article.

3.1. Problem Formulation of Optimization and Fog Framework

Assume that we have a tuple $x = (x_1, x_2, \ldots x_n) \in X$, where $X \subseteq R^n$ and a tuple $y = (y_1, y_2, \ldots y_m) \in Y$ where $Y \subseteq R^m$ in which the following constraints are held:

$$y_1 = f_1(x_1, x_2, \ldots x_n) \tag{1}$$

$$y_2 = f_2(x_1, x_2, \ldots x_n) \tag{2}$$

$$y_m = f_m(x_1, x_2, \ldots x_n) \tag{3}$$

In such a scenario: x is called the decision vectors; y is the objective vector. X is the solution space, and Y is the objective space to model a minimization problem, with two vectors a and b. We call b dominates a, denoted as $a \prec b\ iff$:

$$\begin{cases} \forall\, i\ \in \{1, 2, \ldots m\} : f_i(a) \leq f_i(b) \\ \exists\, j \in \{1, 2, \ldots m\} : f_j(a) < f_j(b) \end{cases} \tag{4}$$

The domination of b over a is applied when b is superior over a with at least one of the objectives j, and b is not worse than a in the remaining objectives i.

Table 2. Terms and symbols used for presenting the mathematical models.

Symbol	Meaning
$DG(V_t, E_t)$	Graph of tasks.
$V = \{t_1, t_2, \ldots t_m\}$	Tasks to be executed in the fog network.
$E = \{e_1, e_2, \ldots e_k\}$	The dependency relation between the tasks.
$e_i = (t_{m1}, t_{m2})$	A connection between task t_{m1} and t_{m2}.
$P = \{P_1, P_2, \ldots P_m\}$	Computation load of the task.
$L = \{L_1, L_2 \ldots . L_m\}$	Communication loads of the task.
$G = \{G_1, G_2, \ldots G_n\}$	Subsets of independent graphs of tasks (a task in any graph can be executed with any order comparing with other tasks in the same graph).
$V = \{v_1, v_2, \ldots v_n\}$	Speed of CPU of nodes in the network.
$UDG(V_n, E_n)$	Graph of nodes.
$V = \{n_1,\ n_2 \ldots n_n\}$	Nodes are available for service in the fog network.
$RC = \{r_1,\ r_2 \ldots r_n\}$	Renting cost of nodes.
$RR = \{rr_1,\ rr_2 \ldots rr_n\}$	Reliability of nodes.
E_{comp}	Energy consumption because of the computational load.
E_{comm}	Energy consumption because of communication.
B	The bandwidth of the connection's links between nodes that participate in executing the task.
E_σ	Energy balance is represented by the standard deviation of the energy
C	The cost, which is represented by the total rental cost.
S	The stability term is a measure of the reliability of the nodes that execute the task.
$d = \left[d_{ij}\right] = \left[d\left(n_i, n_j\right)\right]$	The distance information between every two nodes
$e = [e_i]$	The energy consumption rate of nodes in the network
P_0	The maximum computational load that can be given to a certain node
L_0	The maximum communication load that can be given to a certain node

3.2. *Hyper-Angle Exploitive Searching HAES*

This section presents a hyper angle exploitive searching HAES algorithm. Firstly, we present its working principle and the difference between HAES and MOGA-AQCD [30] in Section 3.2.1. Secondly, we present the objective partitioning in Section 3.2.2. Lastly, the algorithm of HAES in Section 3.2.3.

3.2.1. Working Principle and the Difference between HAES and MOGA-AQCD

Both the proposed HAES and MOGA-AQCD use the concept of angle quantization for searching, which is based on dividing the space into equal-angle sectors and building a histogram that calculates the number of solutions selected for each sector. However, HAES behaves differently from MOGA-AQCD in terms of the selection of the new solutions. MOGA-AQCD favors solutions located in the least angular sector in terms of the previously selected solutions when two solutions are non-dominated with each other. In contrast, HAES favors solutions located in the maximum angular sector in terms of the previously selected solutions. Typically, the MOGA-AQCD concept is to perform extensive exploration to yield substantial optimal solutions, whereas the HAES concept is that sectors that cover suitable solutions in the past are also likely to be rich in the future. We then provide an example to explain the critical difference between HAES and MOGA-AQCD regarding the searching concept.

The concept of HAES is depicted in Figure 2. The solution space is decomposed into a set of angular sectors. Each angular sector contains a set of solutions. The already found solutions are marked with black bullets and the candidate solutions are represented with white bullets. HAES selects the solutions that are located in the highest angular sector with respect to the number of solutions. We mark the selected solutions with yellow bullets and the ignored solutions with blue bullets.

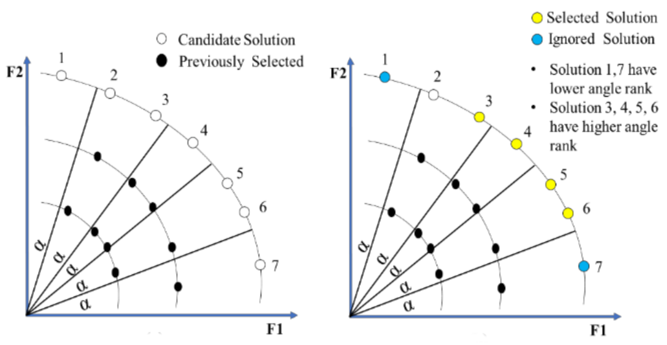


Figure 2. The selected solution in solution space by HAES.

3.2.2. Objectives Partitioning

The multi-objective optimization when working on a high number of objectives requires searching within a wide objective space, which makes it challenging to converge toward the boundary of the objective space. Hence, we do boundary searching mechanisms by activating the sub-set of objectives at each iteration out of the entire objectives. We name it objective partitioning; its role is to reach the boundary of the solution space with respect to the activated objectives. We select at each iteration of the optimization size $k < m$, where m denotes the number of objectives, and we use it for evaluating the solutions, sorting them, and selecting non-dominated ones. The sub-set of objectives is selected randomly at each iteration using a uniform distribution.

3.2.3. Algorithm of HAES

The general algorithm of HAES is presented in Algorithm 1. The algorithm takes the number of generations $NGen$, the number of solutions $NSol$, the sector range value $SectorRange$, and the set of objectives SoB as inputs, the size of objectives partitioning. The output of the algorithm is the Pareto front $ParetoFront$. As can be seen in Algorithm 1, the algorithm starts with the initialization of the first population in line 10, keeping it as a previous population in line 11, initialization of the counter of the population in line 12, initialization of the angle range rank in line 13, and initialization of crowding distance in line 14. Next, an iterative while loop is performed until the number of generations is finished. The loop is composed of calling for the evaluation of the solutions in the previous generation using the objective partitioning in function $selectSubSet$ (line 15) and the objective function calculation in the function $evaluate$ (line 16), updating the crowding distance using the function $updateCrowdingDistance$ (line 17), updating the ranges using the function $updateRanges$ (line 18), selecting the elites that are responsible for generating the off-spring using $selectElites$ (line 20), generating the off-spring using the function $geneticOperations$ (line 22), and the concatenation of the parents and their off-spring using the concatenation operator $||$ (line 23), and finally the new population is selected again from the resulted concatenated using the $electElites$ one more time (line 25). This process is repeated until the total iterations are finished, then the Pareto front of the last generated solution is the result of the algorithm, as presented in line 26.

The algorithm calls three essential functions: $updateCrowdingDistance()$, $updateRanges()$, and $selectElites()$. We provide the details of each of them in Algorithm 2, Algorithm 3, and Algorithm 4, respectively. For the $updateCrowdingDistance()$, the algorithm (detailed in Algorithm 2) takes the number of solutions $NSolutions$ and the objective values $objectiveValues$ as input, and provides the set of crowding distance $crowdingDistance$. The algorithm starts with the initialization of the set of the crowding distance with the size of solutions $NSolutions$ (line 7). Next, the two extreme solutions are assigned the value of infinity (line 8). Afterward, the algorithm sorts the solutions as the separated lists according to their objective values (line 9). Then, the algorithm updates the crowding distance in an accumulated way, corresponding to the difference between each objective of a solution and the value of its next solution in the sorted list (line 11).

Algorithm 1 Pseudocode of the HAES Algorithm

```
1.  Input:
2.      $N_{Gen}$                                   //Number of Generations
3.          $N_{Sol}$                               //Number of Solutions
4.      $SectorRange$                               //Sector Range
5.      $SoB = f_i$, where $i = 1, 2, \ldots, m$;   //Set of Objectives
6.      $K$                                         //size of objectives partitioning
7.  Output:
8.      $ParetoFront$                               //Found Pareto Front
9.  Start:
10.         $P_0$ = InitiateFirstPopulation $N_{Sol}$;     //generate first population randomly
11.         populationPrevious = $P_0$;            //first population is the previous population
12.         counterOfGeneration = 1;
13.         angleRangeRank = zeros (1, $2\pi/SectorRange$) //initialize the angle range rank
14.         while (CounterofGeneration < $N_{Gen}$)
15.             $SSoB = selectSubSet(SoB, k)$
16.             [solutionsRanks,objectiveValues] = evaluate (populationPrevious,$SSoB$)
17.             [$crowdingDistance$] = updateCrowdingDistance (populationPrevious,objectiveValues)
18.             [$angleRangeRank$] = updateRanges (populationPrevious,solutionsRanks,
19.             $SectorRange$,$angleRangeRank$, $SoB$) //select $N_{Sol}$ from the previous solutions
20.             selected Elites = $selectElites$
21.             ($P_0$,solutionsRanks,$angleRangeRank$,$crowdingDistance$, $N_{Sol}$)
22.             offSpring = geneticOperations (selected Elites)
23.             combinedPop = $selectedElites$ || offSpring sortedCombinedPop =
24.     NonNominatedSorting (combinedPop)
25.     $P_{New}$ = selectElites (sortedCombinedPop, $angleRangeRank$,NSol)
26.     populationPrevious = $P_{New}$;
27.     CounterofGeneration++;
28.     end while
29. End
```

Algorithim 2 Pseudocode of calculating the crowding distance

```
1.  Input:
2.      $N_{Sol}$
3.      $objectiveValues$
4.  Output:
5.          $CrowdingDistance$
6.  Start:
7.          $crowdingDistance$ = zeros ($N_{Sol}$);
8.          $crowdingDistance$ (1) = $crowdingDistance$ ($N_{Sol}$) = $\infty$
9.          for (each $i$ objective of $objectiveValues$) sortedSolutions = sort ($N_{Sol}$,$i$);
10.             for (solution $j$ from 2 to $N_{Sol}$)
11.                 $crowdingDistance(j) = crowdingDistance(j) + objectiveValues(i) - objectiveValues(i-1)$;
12.             end for
13.         end for
14. End
```

The $updateRanges()$ function is provided in Algorithm 3. It takes three variables: Solutions, $SectorRange$, and SoB, as input. Additionally, it gives $angleRangeRank$ as output. The approach of obtaining $angleRangeRank$ is based on performing an iterated loop in the input Solutions and updating the counter of each sector in the $SectorRange$ that contains the solution, as presented in the for loop from line 10 to line 13.

Algorithim 3 Pseudocode of updating the angle range rank

```
1.  Input:
2.      Solutions
3.      SectorRange
4.      SoB
5.  Output:
6.      angleRangeRank
7.  Start
8.          $L$ = length (Solutions)
9.          $angleRangeRank$ = zeros (360/$SectorRange$)
10.         for ($i$ = 1 to $L$)
11.             $A_i$ = angle (solution($i$))//angle of solution $i$
12.             angleRangeRank (j) = map ($A_i$, $SectorRange$) + $angleRangeRank$ ($j$)
13.         end for
14.         return $angleRangeRank$
15. End
```

The final procedure receives the pool of solutions Pool of Solutions, the rank of solution Rank, the angle range rank $AngleRangeRank$, the array of the crowding distance $CrowdingDistance$, and the number of solutions to be selected N as input and provides selected solutions (Algorithm 4). The procedure performs an iterated loop for N times, where it selects two solutions in each time and calculates three measures for each solution: rank, angle range rank, and crowding distance. Next, the selection function determines which one has a better rank (line 17), better angle range rank (line 19), and better crowding distance (line 21). Then, the selection process is applied by checking the condition (line 22–24) to identify which favors a solution that has a better rank. In the case that two solutions have the same rank, then the solution with better angle range rank is selected. If the two solutions both have the same values of rank and angle range rank, then the approach will select the solution that has better crowding distance. In addition, the definition of "better" is provided for rank in line 17, for angle range rank in line 19, and for crowding distance in line 21. The detail of the algorithm for selecting the elites is shown in Algorithm 4.

3.3. Fog Computing Closed Loop Model (FCCL)

This section presents our developed integrated objectives fog computing model FCCL. It is composed of five main sections. Section: 3.3.1 explains the first layer which is the fog interface. Section 3.3.2 is an overview of the task decomposer and task model. Next, Section 3.3.3 the task dispatcher. Then, Section 3.3.4 contains the network model, and lastly, Section 3.3.5 contains the optimization objectives.

From a fog computing perspective, our problem is formulated similarly. The fog has an interface that receives from the user a request of executing a computational task with the needed criterion for optimization. Next, it calls an optimization algorithm that provides a set of non-dominated solutions with respect to the provided criteria. The user will make a decision for selecting one among them. The criteria are denoted by vectors $y = (y_1, y_2, \ldots y_m)$, where $\{y_\mathrm{i}\}$ denotes a criterion for fog computing optimization. Without loss of generality, we consider five criteria, namely, Energy Consumption, Energy Distribution, Renting Cost, and Stability.

$y = (energy\ consumption,\ energy\ distribution,\ renting\ cost,\ and\ stability)$. The solutions that are provided to the user gives the selected fog nodes for the execution of the request; we are represented by vector $x = (x_1, x_2, \ldots x_n)$. The goal is to maximize the domination aspect of the provided solutions and their diversity. This gives the user more variety of choices. To elaborate more, we present Figure 3, which elaborates the user giving a request to the user interface and waiting for a set of non-dominated solutions to select one. The fog interface communicates with the task decomposer that decomposes the task that is requested by the user to execute in the fog network. The role of the task decomposer is to partition the task into subsets of independent subtasks; we call each subset a group. Each group is executable independently on the other task.

Algorithim 4 Pseudocode of selecting the elites

```
1.  Input:
2.      Pool of Solutions
3.      Rank
4.      AngleRangeRank
5       CrowdingDistance
6.      N                                   //number of the selected solutions
7.  Output:
8.      selected solutions
9.  Start:
10. for (solution = 1 to N)                 //number of the selected solutions
11.     Select two individuals A, B randomly for an individual
12.     Compute Non-domination rank (rank)
13.     Compute Crowding distance (distance)
14.     Compute Angle rank level (angle Range Rank)
15.
16.             //Compare Solutions
17.     betterRank = A_rank < B_rank
18.     sameRank = A_rank == B_rank
19.     betterAngleRangeRank = A_angleRangeRank > B_angleRangeRank
20.     sameAngleRangeRank = A_angleRangeRank == B_angleRangeRank
21.     betterCrowdingDiandstance = A_distance > B_distance
22.     if (betterRank)
23.       or (sameRank and betterAngleRangeRank)
24.       or (sameRank and sameAngleRangeRank and betterCrowdingDistance)
25.     then
26.             add A to the selected solutions
27.     else
28.             add B to the selected solutions
29.     end if
30.   end for
31. End
```

This aspect enables shorter execution time, which is one of the metrics to be optimized. The task decomposer communicates with the task dispatcher that is responsible for calling the mathematical functions of the fog criterion for calculating the objective function for any candidate solution. Obviously, the task dispatcher receives the needed information from the fog network and the task decomposition and specification before carrying the optimization. The optimization is carried using a multi-objective optimization algorithm named HAES.

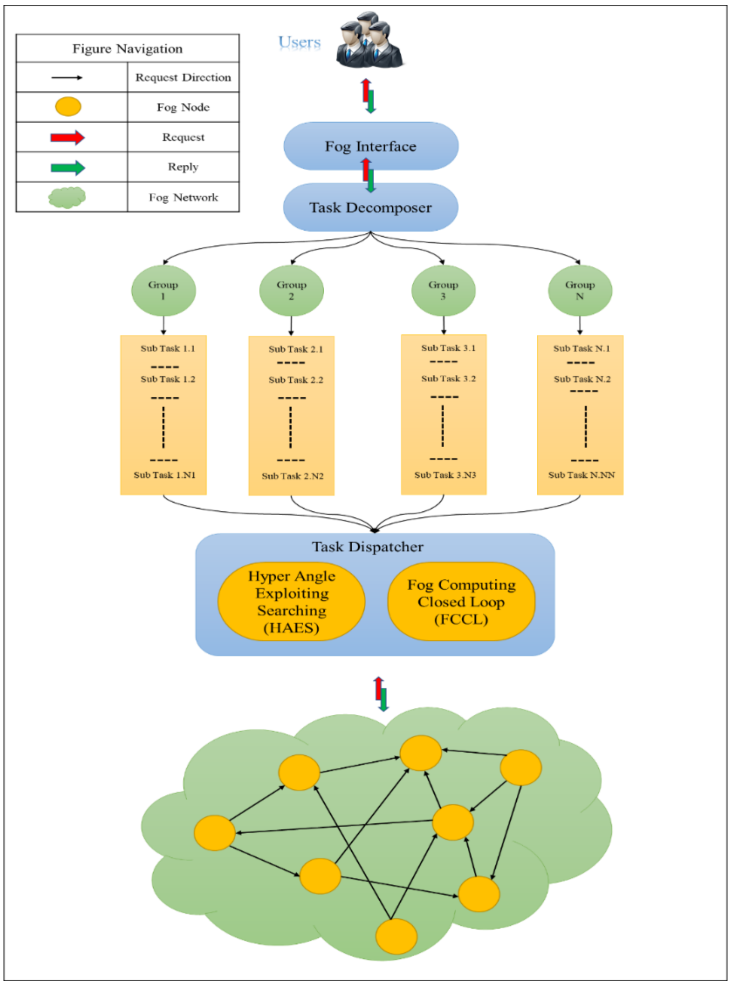


Figure 3. The framework of multi-criteria optimization for fog computing.

3.3.1. Fog Interface

The fog interface will accept from the user two inputs. The first one is the task, and the second one is the preference vector of the various objectives for optimizing the task. The vector of preference between the five objectives is the five components vector, given as $pre = [pr_1\ pr_2\ pr_3\ pr_4\ pr_5]$ with the constraint $\sum_{i=1}^{5} pr_i = 1$. The second input is the configuration input, which is also given by a vector named $conf = [itMax\ popSize]$, where $itMax$ denotes the maximum number of iterations, and $popSize$ denotes the size of the population. Assuming that there is more interest in the time execution (makespan) and stability, the second interest is in the cost, and the third interest in the energy consumption and the energy balance, then the value of $pre = \left(1 \times pr, 1 \times pr, \frac{1}{2} \times \mathrm{pr}, \frac{1}{3} \times \mathrm{pr}, \frac{1}{3} \times \mathrm{pr}\right)$. This implies, $1 \times pr + 1 \times pr + \frac{1}{2} \times \mathrm{pr} + \frac{1}{3} \times \mathrm{pr} + \frac{1}{3} \times \mathrm{pr} = 1$. Then, $pr = 6/19$.

3.3.2. Task Decomposer and Task Model

The logical decomposition of data fusion tasks is a fundamental process in the design of systems aiming at combining multiple and heterogeneous cues collected by sensors. In recent years, a relevant body of research has focused on formalizing logical models for multi-sensor data fusion in order to propose appropriate and general task decomposition. Therefore, we suggest a task decomposer, which is elaborated in Figure 4, to decompose the data and classify based on priority. The role of the task's decomposer is to decompose the tasks into a set of independent tasks; we denote them into groups $G = \{G_1, G_2, \ldots G_N\}$. Example 1 went particularly into decomposing and classifying the tasks.

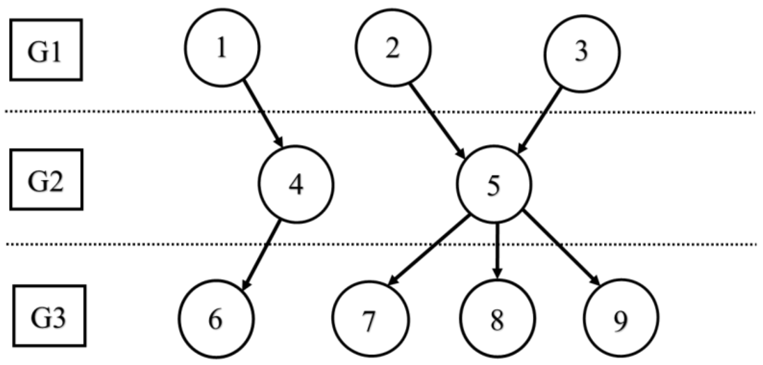


Figure 4. Task Decomposer.

This component has the role of accepting the task from the user. The task itself is modeled as a directed graph $DG(V,E)$, where $V_t = \{t_1, t_2, \ldots t_m\}$, $E = \{e_1, e_2, \ldots e_k\}$, where m denotes the number of tasks in the graph and k denotes the number of directed edges. Where each edge $e_i = (t_{m1}, t_{m2})$, it denotes that t_{m2} is dependent on t_{m1}. Another piece of information that is related to the task and has to be provided by the interface is the workload of tasks in terms of both computation and communication, where the computation is described by set $P = \{P_1, P_2, \ldots P_m\}$, where each P_i denotes the computation that is the number of a clock for the task P_i, and the communication load is described by the set $L = \{L_1, L_2 \ldots . L_m\}$, where L_i represents the communication loads, which describes the total length of data to be exchanged among selected nodes for executing the task.

Example 1:

Task decomposer will classify the nodes in the network into groups, and each group depends on the number of nodes in the fog network. In addition, its direct graph, which is the fog nodes, will forward the request to the next node. The result of the task decomposer is set of three groups as G_1 = {1,2,3}, G_2 = {4,5}, and G_3 = {6,7,8,9}. As we see, the tasks in each group are independent of each other, and they can be processed in any order.

3.3.3. Task Dispatcher

The task dispatcher is responsible for allocating certain nodes in the fog network for the execution of the sub-tasks that result from the task decomposer. It contains the optimization algorithm HAES, which was presented in Section 3.2.3. The fog computing closed loop is presented in Section 3.3.

3.3.4. Network Model

We assume that the network is an undirected graph $UDG(V_n, E_n)$, where $V = \{n_1,\ n_2 \ldots n_n\}$, where n denotes the number of nodes in the network. $E = \{(n_i, n_j)\}$ where $n_i, n_j \in V$. Assuming that the nodes have wireless connections between each other, then we are interested in the distance between every two nodes. Each node i has a rate of computational energy consumption e_i $[\frac{j}{sec}]$, and each two nodes $n_i, n_j \in V$, have distance between them, which is given as $d_{ij} = d(n_i, n_j)$. In addition, we assume that each node n_i has a speed for execution v_i. Furthermore, we assume that each node has a maximum capacity for executing computational load p_0 and maximum capacity for executing communication load l_0.

3.3.5. Optimization Objectives

We present in this section the equations of the optimization objectives. Our model has the aspect of integrating five objectives at the same time, which makes it distinguished from other models in the literature.

A. Time Latency

Time latency is an expression of how much time it takes for a packet of data to get from one designated point to another. It is sometimes measured as the time required for a packet to be returned to its sender, which is calculated by the following formula.

$$T = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} \tag{5}$$

$$t_{ij} = t_{ij}{}^{1} + t_{ij}{}^{2} \tag{6}$$

$$t_{ij}{}^{1} = \frac{P_{ij}}{v_i} \text{ computation time} \tag{7}$$

$$t_{ij}{}^{2} = \frac{l_{ij}}{B} + t_{ij}{}^{queue} \text{ communication time} \tag{8}$$

where $t_{ij}{}^{queue}$ denotes the queue waiting time. P_{ij} denotes the task computational load that is assigned to node i. The speed is v_i of the node i. l_{ij} denotes the communication load between i and j. Lastly, B denotes the bandwidth.

B. Energy Consumption

In order to send number packets from node A until node B, where the distance between the two nodes is $d(A, B) = d$, we calculate the consumed energy as Equation (9).

$$e(A, B) = e(d) = \begin{cases} (e_{elec} + \varepsilon_{amp} d^2) l(A, B) \; for\ transmit \\ e_{elec} l(A, B) \; for\ receive \end{cases} \tag{9}$$

where e_{elec} denotes energy consumption for operating the radio model for each bit in the data. d denotes distance between the two nodes A, B. The coefficient of transmit amplifier given by ε_{amp}. $l(A, B)$ denotes the number of bits to be sent from node A to node B.

Based on the term $e(A, B) = e_{A,B}$, $l(A, B) = l_{A,B}$, we can calculate the total energy consumption based on terms E_{comp}, E_{comm}, which represent the computation energy consumption and communication energy, respectively. The total energy is given in Equation (10), the computation energy is given in Equation (11), and the communication energy is given in Equation (12).

$$E = E_{comp} + E_{comm} \tag{10}$$

$$E_{comp} = \sum_{i\,=\,1}^{n} e_i t_i \tag{11}$$

where e_i denotes to the energy consumption because of execution in node i, t_i denotes to the time allocation of the node i, $e_{i,j}$ denotes the energy consumption because of communication between nodes i and j, and $l(i, j)$ number of bits transferred between nodes i and j.

$$E_{comm} = \sum_{i,j,\ i \neq j}^{m} e_{i,j} l_{i,j} \tag{12}$$

C. Energy Distribution

This term indicates the differences among the nodes in terms of the energy levels. The term is calculated as the standard deviation of the node's energy as it is given in Equation (13).

$$E_\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(E_i - \overline{E}\right)^2}{n-1}} \tag{13}$$

where E_i denotes the consumed energy of node i; $\overline{E}$ denotes the average consumed energy of all nodes. n denotes the total number of nodes.

D. Renting Cost

The renting cost is defined as the total cost of rent, which is the summation of node i rental rate r_i multiplied by the time of allocating the node according to Equation (14).

$$C = \sum_{i=1}^{n} t_i r_i \tag{14}$$

where r_i denotes the renting rate of the node i. t_i denotes the time of allocating the node i.

E. Stability

This term indicates the total stability of the task execution. It is calculated as the summation of the reliability percentage of a certain node rr_i multiplied by the time of allocating the nodes. The calculation is depicted in Equation (15).

$$S = \sum_{i=1}^{n} t_i rr_i \tag{15}$$

where rr_i denotes to the reliability rate of the node n_i, and t_i denotes the time of allocating the fog node i.

F. Constraints

Before assigning any given solution to the fog network, it is needed to assure that it meets the constraints. Basically, there are two types of constraints that should be satisfied. The first one is the connectivity constraint, which states that any sub-network is assigned an execution of a task; it should be connected in order to execute the task that is assigned to the sub-network. The second constraint is named the load constraint. It states that for a task T with computational load P and communication load L, it should be allocated at least N_0 for execution. The value N_0 is calculated based on Equation (16).

$$\begin{cases} N_0 = Max\left(\frac{L}{L_0}, \frac{P}{P_0}\right) \\ N \geq N_0 \end{cases} \tag{16}$$

4. Experimental Design and Parameters Setup

This section comprises three categories for presenting the evaluation of the proposed model and base benchmarks used in the evaluation. The first category, in Section 4.1, is the evaluation metrics of HAES and FCCL models. This section talks specifically about the most common and standard evaluation measures, which are hyper-volume, non-dominated solution, generational distance measure, inverse relative generational distance measure, delta metric measure, and set coverage measure. In addition, the parameters for HAES mode with base models. The second category, Section 4.2, is a dedicated section that presents the multi-objective mathematical functions that will test HAES and compare it with state-of-the-art approaches. The third, Section 4.3, presents the parameters for the FCCL model.

4.1. Evaluation Metrics of HAES and FCCL

This section presents the evaluation metrics that are used for evaluating our developed approaches, which are HAES and FCCL. Fog computing evaluation metrics are the same objectives that are used for optimization. We present the hyper-volume in sub-section A Next, we present the number of non-dominated solutions in sub-section B. Afterward, the generational distance is presented in sub-section C. Next, the inverse relative generational distance measure in sub-section D, and the delta metric is provided in sub-section E. Lastly, set coverage is giving in sub-section F.

A. Hyper Volume (HV) Measure

The hyper volume (HV) metric is widely used in evolutionary MOO to evaluate the performance of the searching algorithm [36]. It computes the volume of the dominated portion of the objective space related to the worst solution. This region is the union of the hypercube, with its diagonal as the distance between the reference point and a solution X from the Pareto Set (PS). High values of this measure present the desirable solutions. HV is presented by the following (Equation (17)):

$$HV = volume(\cup_{x \in P_s} Hyper\ Cube\ (x)). \tag{17}$$

B. Number of Non-Dominated Solutions (NDS)

The number of non-dominated solutions (NDS), which expresses the effectiveness of the optimization algorithm [37], can be calculated as the cardinality of PS as (18):

$$NDS(N) = |P_s|. \tag{18}$$

C. Generational Distance Measure (GDM)

This metric, also called the GD metric [38], is a measure to evaluate the performance of a found Pareto Set (PS) compared with a reference point set (a true Pareto set (PS)). This measure is based on the distance among obtained solutions and reference points, which is calculated as follows (Equation (19)):

$$GD(P_s, P_T) = \frac{\left(\sum_{i=1}^{|P_s|} d_i^2\right)^{\frac{1}{2}}}{|P_s|}. \tag{19}$$

D. Inverse Relative Generational Distance Measure (IRGD)

Inverse Relative Generational Distance Measure (IRGD)

Another metric that is used is the inverse Relative Generational Distance or IRGD, and it is given in Equation (20).

$$IRGD(P_s, P_T) = \frac{|P_s|}{\left(\sum_{i=1}^{|P_s|} d_i^2\right)^{\frac{1}{2}}}. \tag{20}$$

E. Delta Metric Measure

The delta or diversity metric Δ shows the extent to which it achieves the spread [14]. The delta measure receives the non-dominated set of solutions and provides the diversity metric, and can be computed according to the following equation:

$$\Delta = \frac{d_f + d_I + \sum_{i=1}^{N-1} |d_i - d^-|}{d_f + d_I(N-1)d^-} \tag{21}$$

where N is the number of solutions, d_f and d_i, the Euclidean detachments between the extreme and border solutions, and d is all the consecutive distances, d_i $(i = 1,\ 2, \ldots,\ N\ - 1)$. This measure is required to be slight and maintained to be less, because this measure indicates uniform distribution. In addition, it provides various selections to the decision-maker.

F. Set Coverage Measure

Set coverage measure [37], also called C metric, compares the Pareto sets *Ps1* and *Ps2* and can be identified by (22):

$$C(P_{s1}, P_{s2}) \ = \frac{|\{y \in P_{s2} | \exists x \in P_{s1} : y \prec x\}|}{P_{s2}} \tag{22}$$

C equals the ratio of nondominated solutions in P_{s2} dominated by non-dominated solutions in P_{s1} to the number of solutions in P_{s2}. Thus, when evaluating a set Ps, the value of $C(X;\ Ps)$ must be minimized for all Pareto sets X.

4.2. Multi-Objective Mathematical Functions

The algorithms are evaluated based on various relevant MOO mathematical functions. The formulas, optimization range, and true PF of each mathematical function are provided in Table 3. They have been used in most of the existing studies on MOO optimization as the benchmarking functions. The convexity is different for each function. Table 3 shows the bounds of the variables and the optimal solutions or PFs. In this way, our proposed approach can be validated against critical MMO measures. We selected three approaches, NSGA-II, NSGA-III, and MOGA-AQCD, which were presented in the background section, as the three relevant benchmarks to evaluate HAES.

To make the study quantitative, ten experiments are performed for each function using different seeds. This study also refers to the previous studies so that the same methodology of evaluation as of Multi-Objective Evolutionary Algorithms (MOEAs) is performed. The test function is chosen based on the well-known studies, including Fleming's study (FON) [39], Kursawe's study (KUR) [40], Poloni's study (POL) [41], and Schaffer's study (SCH) [42]. We then followed those guidelines and suggested six test problems, in which five of them are presented in Table 3, call ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. All problems have two objective functions, and none of these problems has any constraint. In addition, the number of variables, the bounds, the Pareto-optimal solutions, and the nature of the Pareto-optimal front for each problem.

Table 3. Mathematical functions for evaluating MOO measures.

Problem	***n***	**Variable Bounds**	**Objective Function**	**Optimal Solution**	**Remark**
FON	3	$[-4, 4]$	$f1\,(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(xi - \frac{1}{\frac{1}{\sqrt{3}}}\right)^2\right)$ $f2\,(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(xi - \frac{1}{\frac{1}{\sqrt{3}}}\right)^2\right)$	$x_1 = x_2 = x_3$	Non-convex
KUR	3	$[-5, 5]$	$f_1(x) = \sum_{i=1}^{n-1}\left(-10\, exp\left(-0.2\, \sqrt{x_i^2 + x_i^2 + 1}\right)\right)$ $f(x) = \sum_{i=1}^{n}\left(\lvert x_i\rvert^{\,0.8} + 5\sin x_i^3\right)$	[43]	Non-convex
POL	2	$[-\pi, \pi]$	$f_1(x) = \left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right]$ $f_2(x) = \left[(x_1 + 3)^2 + (x_2 + 1)^2\right]$	[43]	Non-convexDisconnected
SCH	1	$[10^{-3},\ 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$X\ \in [0,\ 2]$	Convex
ZDT1	30	$[0, 1]$	$f_1(x) = \sum_{i=1}^{n-1}(-10\ \exp(-0.2\ \sqrt{x_i^2 + x_{i+1}^2}$ $f_2\,(x) = \sum_{i=1}^{n}(\ \lvert x_i\rvert^{\,0.8} + 5\sin x_i^3)$	$x_1 \in [0,\ 1]$ $x_i = 0$ $i = 2,\ 3, \ldots, n$	Convex
ZDT2	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - (x_1/g_x)^2\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,\ 1]$ $x_i = 0$ $i = 2,\ 3, \ldots, n$	Non-convex
ZDT3	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\ \sin(10\,\pi\, x_1)\ \right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,\ 1]$ $x_i = 0$ $i = 2,\ 3, \ldots, n$	Convex, Disconnected

Table 3. *Cont.*

Prob-lem	n	Variable Bounds	Objective Function	Optimal Solution	Remark
ZDT4	10	$[0, 1]$ $[-5, 5]$	$f_1 = x_1$ $f_2 = g(x)\left[1 - (\frac{f_1}{g})^{0.5}\right]$ $g = 1 + 10(N-1) + \sum_{i=2}^{N}\left(x_i^2 - 10\cos(4\pi x_i)\right)$		Non-convex
ZDT6	10	$[0, 1]$	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)\left[1 - (\frac{f_1(x)}{g(x)})^2\right]$ $g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, 3, \ldots, n$	Convex, non-uniformly spaced

The implementation is conducted using MATLAB 2019b. The parameters for NSGA-II, NSGA-III, MOGA-AQCD, and HAES are given in Tables 4 and 5. The same number of solutions and generation was used for all the algorithms in order to have a fair comparison. An increase in the number of solutions and generations typically yields better performance results. The numbers of the population and generations are selected to be (100) and (500), respectively. The parameters of the crossover are determined based on two parts: fraction and ratio. The fraction is selected to be $2/n$, where n denotes the solution length and the ratio is selected to be 1,2. For the scale of the mutation, we selected the value of 0,1. These values are the default ones that are used by the MATLAB optimization package.

Table 4. Parameters of NSGA-II, MOGA-AQCD, and HAES.

Parameters		NSGA-II	MOGA-AQCD	HAES
No. of solution		100	100	100
No. of generation		500	500	500
Crossover option	Fraction	2/n	2/n	2/n
	Ratio	1,2	1,2	1,2
Mutation option	Fraction	2/n	2/n	2/n
	Scale	0,1	0,1	0,10
	Shrink	0.5	0.5	0.5
Quantification of angle space (α)		N/A	10^{-7} for all test except KUR 5×10^{-7}	10^{-7} for all test except KUR 5×10^{-7}

Table 5. Parameters of NSGA-III.

Parameters		NSGA-III
No. of Solution		100
No. of Generation		500
Crossover Percentage		0.5
Mutation Option	Mutation Percentage	0.5
	Mutation Rate	0.02
Number of Divisions		10

These selected numbers are to obtain the PF within a balanced time. However, increasing both or one of them yields highly dominated solutions, given extensive exploration will be conducted in the searching space.

4.3. HAES Evaluation Based on FCCL Model

The evaluation is done based on population size 200 and number of generations 200. We run the model on 10 experiments. Each experiment is conducted on a different value of the quantization, $\alpha = \{20, 23, 25, 28, 30, 33, 35, 37, 40, 45\}$. In addition, each experiment is repeated 10 times with different values of seed, which are given in Table 6. The results are decomposed into two sub-sections. The first one is the presentation of the results of the multi-objective mathematical functions, and the second one is the results of the evaluation of the fog computing closed-loop model.

Table 6. Table of parameters used for evaluation FCCL Model.

Parameter	Value
Population size	200
Number of generations	200
Number of random experiments	10
α	{20, 23, 25, 28, 30, 33, 35, 37, 40, 45}
Number of nodes	30
Number of tasks	6
Number of objectives	5
Crossover	1.2
Mutation	0.5, 1.5

5. Evolution and Enhanced Model Results

This part presents the results of the two models, HAES and FCCL, and discuss the experiment results comparing to the other models and their differences. Section 5.1 elaborates on the first phase which is the optimization of HAES with three benchmarks as follows NSGA-II, NSGA-III, and MOGA-AQCD; Section 5.2, the second phase, is the model of FCCL and the comparison of our model with the same benchmarks for phase one.

5.1. HAES Experimental Investigation and Results

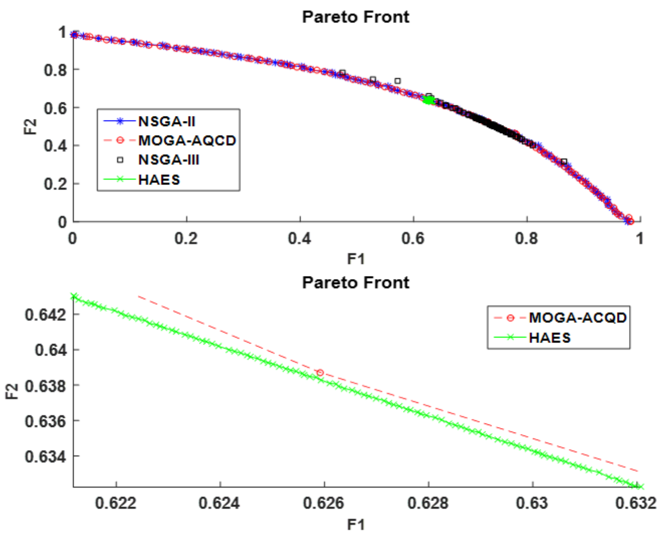
The evaluation of the HAES algorithm is performed firstly based on mathematical functions with a challenging MOO nature as follows: FON, KUR, POL, SCH, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. It presents the Pareto front, average hyper volume metric, average non-dominated solutions metric, an average of delta metric, and the average of generational distance metric, respectively, in each figure for HAES and other three benchmarks. As we observe in Figure 5, the Pareto front is plotted with two axes figures, because each of the mathematical functions has two objectives. Considering that HAES has an exploiting nature that enables the algorithm to each more dominant solution even if the regions of exploration were less, this has made it more capable of minimizing the values of the objectives.

In order to present this clearly, we show for each mathematical function two scales: the first one shows the general Pareto at the top and the second one shows the area of solutions found by HAES at the bottom. The Pareto front was lower for the functions FON, POL, SCH, ZDT1, ZDT2, ZDT3, and ZDT4, which is more domination with respect to these functions. The only function that has not achieved lower values of the Pareto front is KUR. However, HAES has achieved a more diverse Pareto front for KUR compared with the benchmarks. Figure 5 elaborate on the results for mathematical functions for each metric particularly.

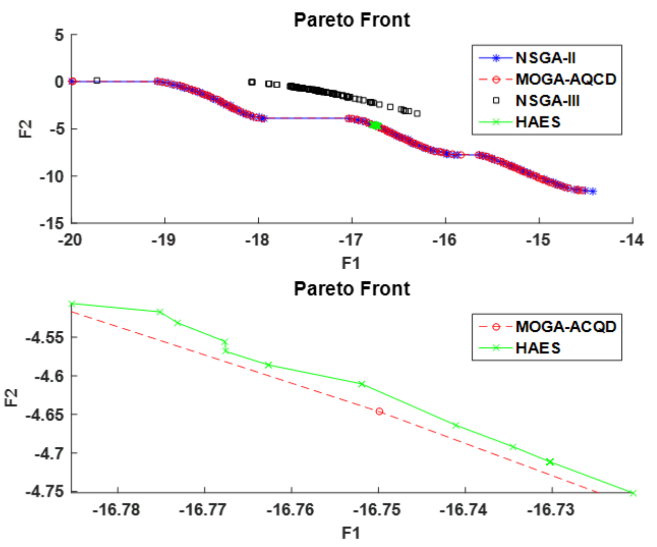
In order to identify the superiority in terms of domination, we provide two tables: the first one is showing the domination of the benchmarks over HAES in Table 7, and the second shows the domination of HAES over the benchmarks in Table 8. As can be seen, the values in Table 7 are higher than their corresponding values in Table 8, which means that HAES is more dominant over the MOGA-AQCD, NSGA-III, and NSGA-II.

Table 7. Average set coverage values of HAES compared to those of MOGA-AQCD, NSGA-III, and NSGA-II.

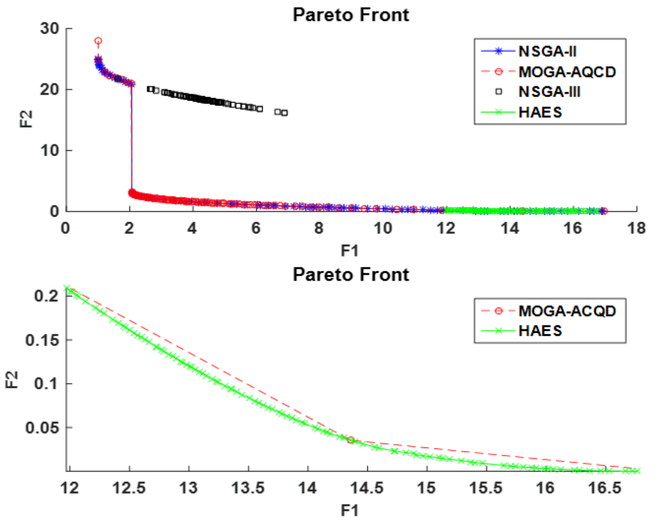
Functions	MOGA-AQCD	NSGA-III	NSGA-II
FON	1.100×10^{-2}	3.000×10^{-2}	1.500×10^{-2}
KUR	3.100×10^{-2}	2.290×10^{-1}	2.700×10^{-2}
POL	8.000×10^{-3}	1.000×10^{-3}	4.000×10^{-2}
SCH	2.000×10^{-3}	6.880×10^{-1}	2.000×10^{-3}
ZDT1	0.000×10^{-0}	0.000×10^{-0}	1.500×10^{-2}
ZDT2	0.000×10^{-0}	0.000×10^{-0}	0.000×10^{-0}
ZDT3	6.000×10^{-3}	0.000×10^{-0}	1.500×10^{-2}
ZDT4	4.815×10^{-2}	6.000×10^{-1}	9.000×10^{-2}
ZDT6	2.750×10^{-1}	0.000×10^{-0}	2.710×10^{-1}



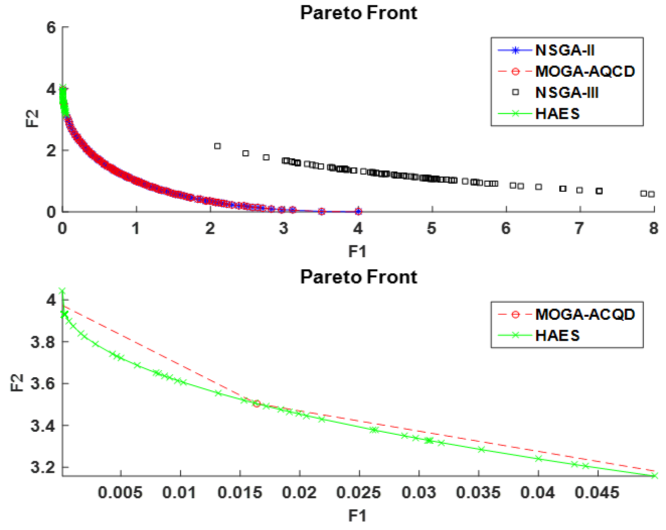
(**a**) FON



(**b**) KUR

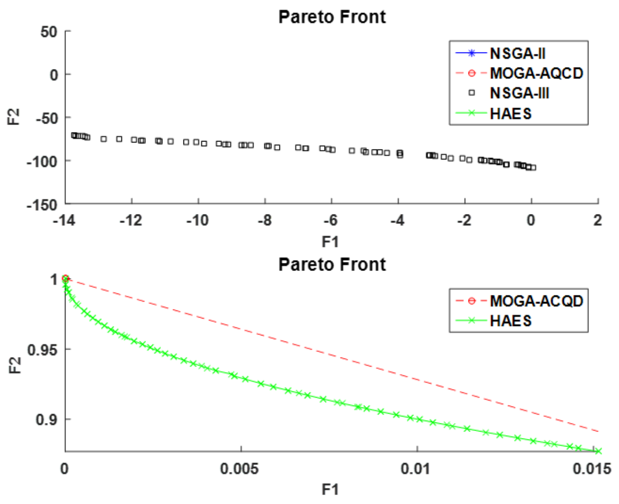


(**c**) POL

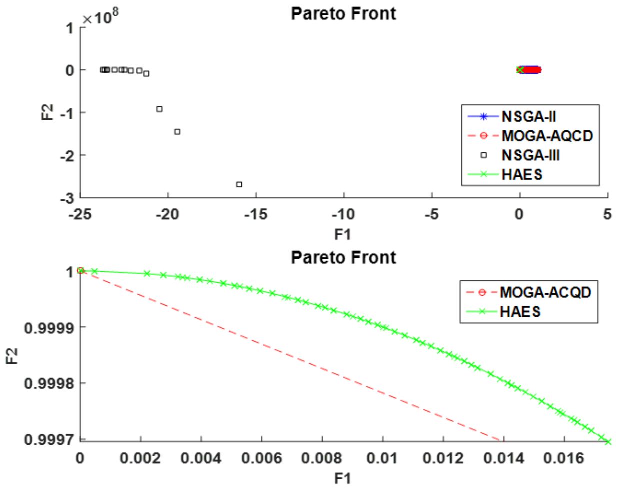


(**d**) SCH

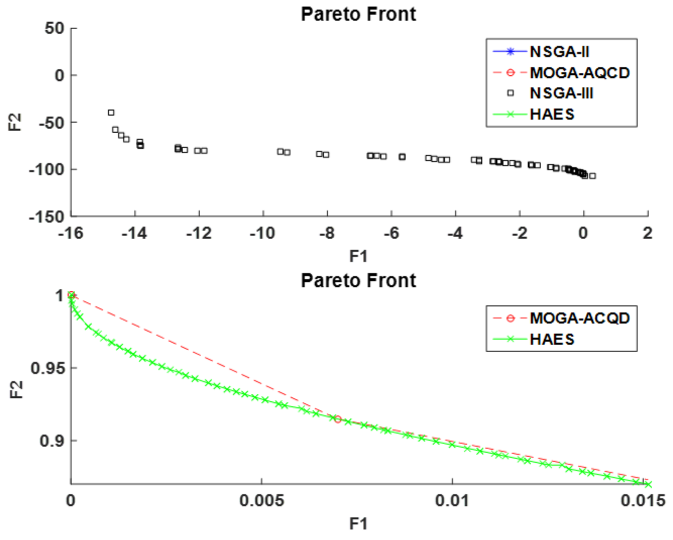
Figure 5. *Cont.*



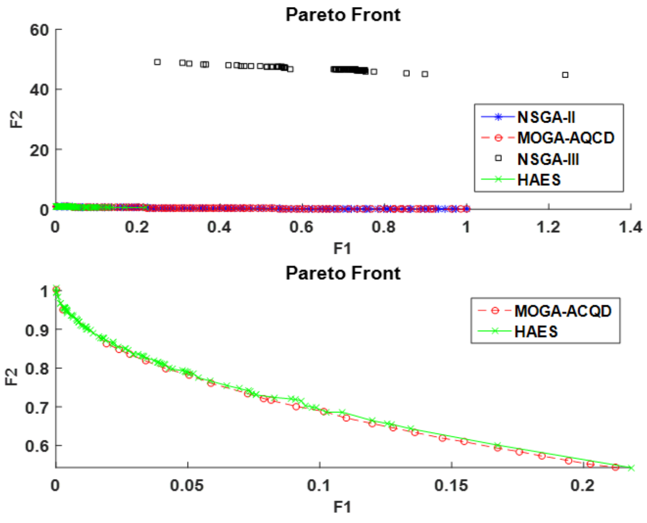
(**e**) ZDT1



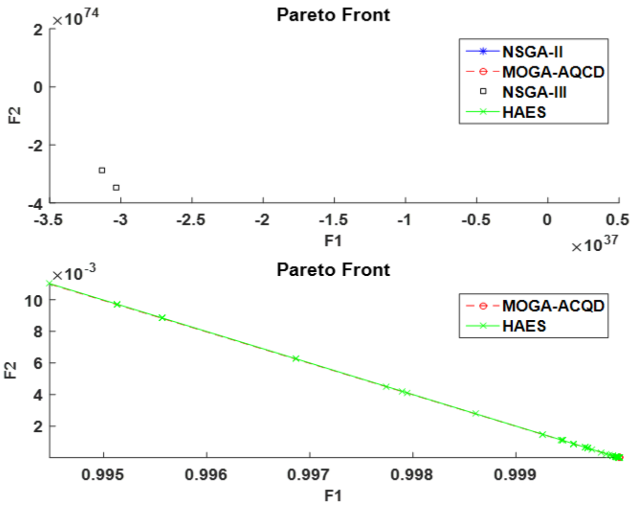
(**f**) ZDT2



(**g**) ZDT3



(**h**) ZDT4



(**i**) ZDT6

Figure 5. Pareto front with two scales: sub-figure HAES with MOGA-AQCD for FON, KUR, POL, SCH, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6.

Table 8. Average set coverage of MOGA-AQCD, NSGA-III, and NSGA-II compared to that of HAES.

Functions	MOGA-AQCD	NSGA-III	NSGA-II
FON	0.000×10^{-0}	0.000×10^{-0}	0.000×10^{-0}
KUR	8.140×10^{-3}	7.488×10^{-3}	1.279×10^{-2}
POL	1.000×10^{-3}	0.000×10^{-0}	1.300×10^{-2}
SCH	2.000×10^{-3}	0.000×10^{-0}	2.000×10^{-3}
ZDT1	0.000×10^{-0}	1.000×10^{-0}	3.000×10^{-3}
ZDT2	0.000×10^{-0}	1.000×10^{-0}	0.000×10^{-0}
ZDT3	0.000×10^{-0}	1.000×10^{-0}	0.000×10^{-0}
ZDT4	0.0481209	0	0.1139833
ZDT6	0	1	0

In order to assess the performance of HEAS in terms of the richness of the found solutions compared with the benchmarks, we present the hyper-volume. As it is shown in Table 9, ZDT6 has accomplished high hyper-volume only for KUR and ZDT6, while it was less for the other functions. This is interpreted as more domination of solutions that was accomplished for HAES compared with the benchmarks. This makes it more challenging to obtain high hyper-volume compared with MOGA-AQCD, NSGA-II, and NSGA-III, which has generated a lower dominant Pareto front.

Table 9. Average of MOO metrics for benchmarking mathematical functions.

Problems	Evaluation Measure	HAES	MOGA-AQCD	NSGA-III	NSGA-II
FON	Average of Hyper Volume	**5.685**	0.298	0.089	0.297
	Average Non-Dominated Solutions	100	100	100	100
	Delta Metric	0.991	**0.196**	1.011	0.281
	Average Generational Distance	**0.00109**	0.001199	0.001483	0.001199
KUR	Average of Hyper Volume	15.85	25.66	2.316	**25.67**
	Average Non-Dominated Solutions	61.8	**100**	**100**	**100**
	Delta Metric	0.8695	**0.3695**	1.035	0.4129
	Average Generational Distance	0.01893	0.006606	0.07131	**0.006420**
POL	Average of Hyper Volume	0.4963	368.2	17.45	**369.1**
	Average Non-Dominated Solutions	100	100	100	100
	Delta Metric	**0.9289**	1.308	1.026	0.9444
	Average Generational Distance	**0.001193**	0.007846	0.204	0.008936
SCH	Average of Hyper Volume	0.02784	13.26	**17.45**	13.26
	Average Non-Dominated Solutions	100	100	100	100
	Delta Metric	1.057	**0.6812**	1.021	0.6812
	Average Generational Distance	0.001227	**0.0008915**	1.15	**0.0008915**
ZDT1	Average of Hyper Volume	0.0012	0.6591	**187.1**	0.6579
	Average Non-Dominated Solutions	**100**	**100**	66	**100**
	Delta Metric	0.9863	**0.4984**	0.9223	0.6562
	Average Generational Distance	7.92×10^{-4}	$\mathbf{4.18} \times 10^{-4}$	10.9096	5.02×10^{-4}
ZDT2	Average of Hyper Volume	1.6993	0.3274	0.3247	**2.1159**
	Averages Non-Dominated Solutions	**100**	**100**	13.8	**100**
	Delta Metric	0.9985	**0.3258**	1.295	0.6794
	Average Generational Distance	0.0011	$\mathbf{5.06} \times 10^{-4}$	2.31×10^{11}	5.31×10^{-4}
ZDT3	Average of Hyper Volume	0.0012	0.7763	341.5	0.7771
	Average Non-Dominated Solutions	**100**	**100**	39.1	**100**
	Delta Metric	0.9915	0.7661	0.9718	**0.7541**
	Average Generational Distance	$\mathbf{5.55} \times 10^{-4}$	6.81×10^{-4}	14.3872	6.60×10^{-4}

Table 9. *Cont.*

Problems	Evaluation Measure	HAES	MOGA-AQCD	NSGA-III	NSGA-II
ZDT4	Average of Hyper Volume	0.2211	0.6407	**0.829**	0.6119
	Average Non-Dominated Solutions	87.3	**100**	67.6	**100**
	Delta Metric	1.014	0.4384	1.013	**0.3854**
	Average Generational Distance	**9.05** $\times 10^{-4}$	0.0012	7.9171	09.05 $\times 10^{-4}$
ZDT6	Average of Hyper Volume	**0.4746**	0.2646	0	0.2636
	Average Non-Dominated Solutions	59.8	**100**	1.4	**100**
	Delta Metric	1.214	**0.635**	0.9666	**0.7989**
	Average Generational Distance	0.0363	3.35 $\times 10^{-4}$	3.47 $\times 10^{85}$	**3.20** $\times 10^{-4}$

In addition to hyper-volume, we generated an NDS measure that indicates the number of found solutions in the Pareto front. A higher value of NDS is equivalent to better performance in general. However, it is important to read NDS as a secondary metric after domination. We observe that HAES has accomplished competing values of NDS to the benchmarks for FON, POL, ZDT1, ZDT3, ZDT4, and ZDT6. Hence, it is considered a good performing algorithm from the perspective of not only domination, but also NDS.

The delta metric shows how much the solutions were equally distributed on the resultant Pareto front. A lower value of the delta metric implies a more equal distribution of the found solutions on the Pareto front. Considering that HAES's focus is to search in an exploiting way, it provides lower distributed solutions in the Pareto front, which makes its value higher compared with the benchmarks and in general closer in order to the value of delta metric of NSGA-III. On the other side, we observe that NSGA-II and MOGA-AQCD have lower values of delta metric.

Another metric that is used to evaluate the performance of MOO is GD, which is preferred to be lower. It shows that HAES has accomplished lower GD for FON, POL, SCH, ZDT1, ZDT2, ZDT4, and ZDT6. We also observe that NSGA-III has suffered from relatively higher values of GD compared with the other approaches. It is important to point out that GD is not always correlated with the percentage of domination due to the change of scales between one objective and the other.

5.2. FCCL Investigation and Results

This section presents the evaluation of implementing HAES on the fog computing closed-loop model. Three main measures are presented for each of the provided configurations in the experimental design, namely, IRGD, which represented the inverse of the relative generational distance, HV, which represents the hyper volume, and NDS, which denotes the number of non-dominated solutions. The evaluation measures are presented with the different configurations in Figure 6. Looking at the figure, we observe that HAES was capable of accomplishing full IRGD and NDS for configurations 23, 25, 33, and 45. Additionally, we observe that HAES' different configuration was not able to bring HV to its maximum value.

For a more quantitative comparison of the difference in the performance between HAES and other benchmarks, we generated the results of the t-test in Figure 7 for three metrics: IRGD, HV, and set coverage. Their values reveal that HAES has outperformed other benchmarks with respect to set coverage with a confidence of more than 70%, and with respect to IRGD with a confidence of more than 90%. However, HAES was less superior with respect to HV, with a confidence of more than 90%.

Looking at the hyper-volume as a secondary measure after the domination and considering that reaching more optimal solutions might limit their spread in the objective space, we interpret that hyper-volume of HAES has not outperformed the hyper-volume of the benchmarks. However, we could have accomplished more optimal solutions with HAES compared with the benchmarks, as both IRGD and set-coverage of HAES have outperformed their corresponding values in the benchmarks.

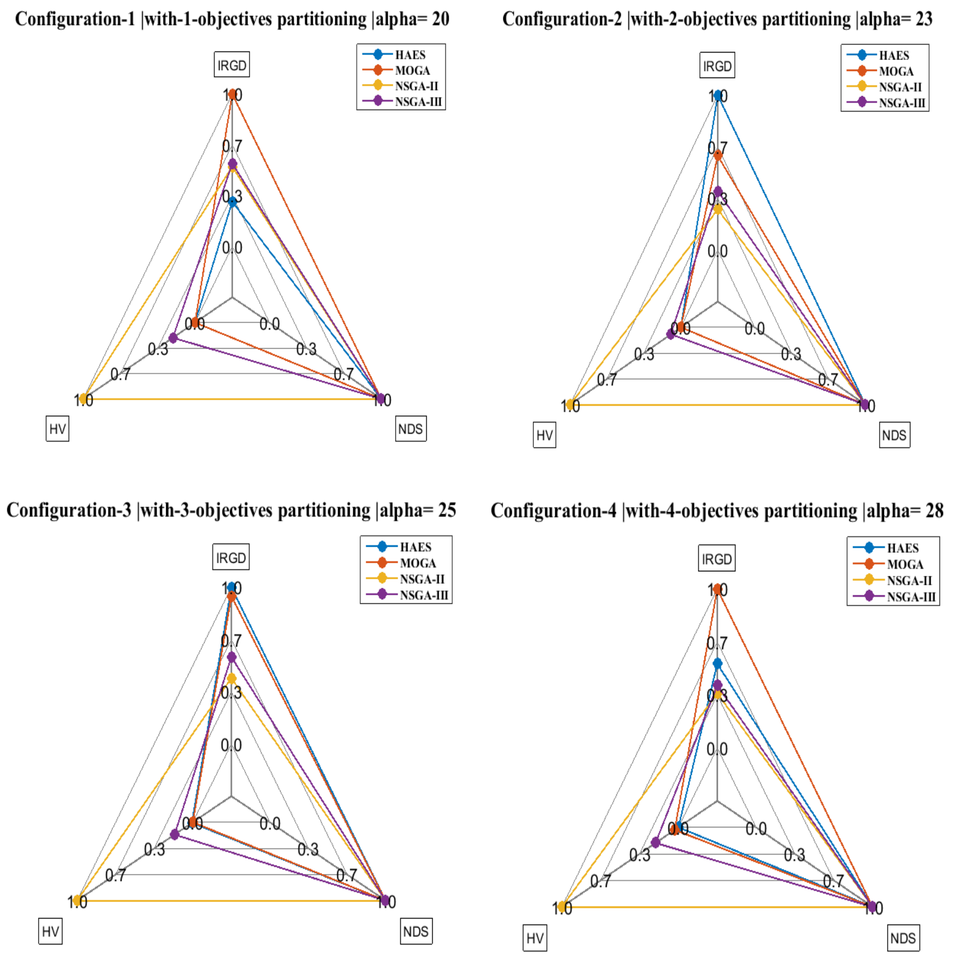


Figure 6. *Cont.*

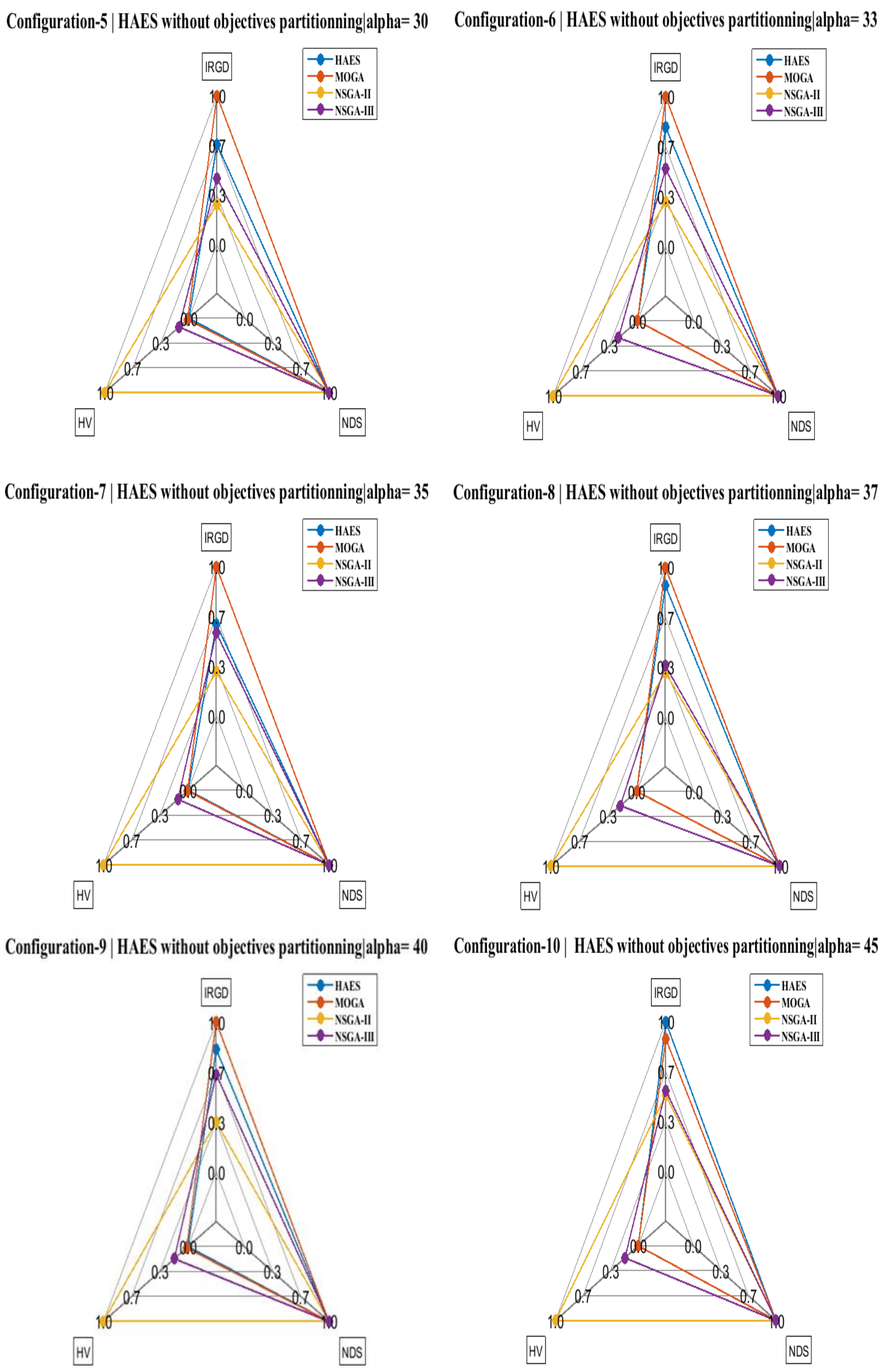


Figure 6. Comparison between HAES different configurations in terms of alpha and the other algorithms.

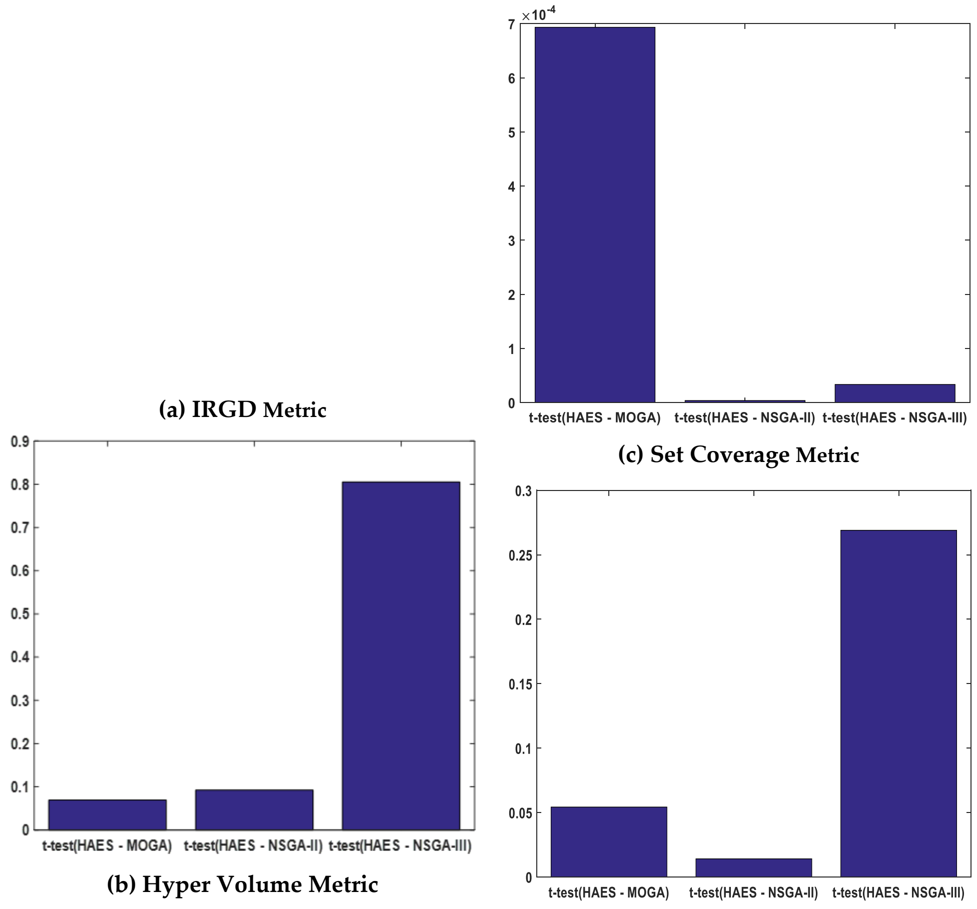


Figure 7. *t*-test to compare the performance of HAES and MOGA-AQCD, NSGA-II, and NSGA-III.

6. Conclusions and Future Works

This article has presented a novel formulation of the problem of fog computing optimization with a multi-objective perspective. The covered objectives are the time latency, the energy consumption with the energy distribution, the renting cost, and stability. The multi-objective and the conflicting nature of the problem require adopting meta-heuristic searching for solving it. However, due to the relatively high number of objectives, different from the relevant existing studies in literature, this research has proposed a novel hyper-angle genetic optimization. The role of the hyper angle is to prioritize solutions within the same rank based on their best-accomplishing rank, which gives the algorithm more exploitive capability. In addition, the article has adopted the concept of objective decomposition by evaluating the approach on various sizes of sub-set of objectives for the objective's decomposition. Objective decomposition enables exploring the boundary of the objective space before going to the intermediate region while searching. Such an approach is crucial for the relatively large number of objectives. Furthermore, various values of angle resolutions were used for the evaluation. It was found that the number of sub-set of objectives while performing the objectives decomposition as well as the value of the angle play an important role in the overall performance. The approach is limited in its dependency on static parameters for both. Hence, our planned future work is to enable an adaptive number of objectives, in which the value of the angle is investigated.

Author Contributions: Supervision: R.H.; validation: A.H.M.A.; visualization and writing—original draft: T.-A.N.A.; review and editing: A.S.A.-K. and Q.N.N. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported under grant Fundamental Research Grant Scheme FRGS/1/2018/TK04/UKM/02/17 and Dana Impak Perdana UKM DIP-2018-040.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, [RH], upon reasonable request.

Acknowledgments: The authors would like to acknowledge the support provided by the Network and Communication Technology (NCT) Research Groups, FTSM, UKM in providing facilities throughout this paper. The authors would also like to thank the Editor and the anonymous reviewers for their valuable comments and suggestions.

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

References

1. Badii, C.; Bellini, P.; Difino, A.; Nesi, P. Sii-Mobility: An IoT/IoE architecture to enhance smart city mobility and transportation services. *Sensors* **2019**, *19*, 1. [CrossRef]
2. Wu, F.; Wu, T.; Yuce, M.R. An internet-of-things (IoT) network system for connected safety and health monitoring applications. *Sensors* **2019**, *19*, 21. [CrossRef] [PubMed]
3. Ibrahim, M.Z.; Hassan, R. The Implementation of Internet of Things using Test Bed in the UKMnet Environment. *Asia Pac. J. Inf. Technol. Multimed* **2019**, *8*, 1–17. [CrossRef]
4. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep learning for IoT big data and streaming analytics: A survey. *Ieee Commun. Surv. Tutor.* **2018**, *20*, 2923–2960. [CrossRef]
5. Sadeq, A.S.; Hassan, R.; Al-rawi, S.S.; Jubair, A.M.; Aman, A.H.M. A Qos Approach For Internet Of Things (Iot) Environment Using Mqtt Protocol. In Proceedings of the 2019 International Conference on Cybersecurity (ICoCSec), Negeri Sembilan, Malaysia, 25–26 September 2019; pp. 59–63.
6. Jia, M.; Yin, Z.; Li, D.; Guo, Q.; Gu, X. Toward improved offloading efficiency of data transmission in the IoT-cloud by leveraging secure truncating OFDM. *Ieee Internet Things J.* **2018**, *6*, 4252–4261. [CrossRef]
7. Aman, A.H.M.; Yadegaridehkordi, E.; Attarbashi, Z.S.; Hassan, R.; Park, Y.-J. A survey on trend and classification of internet of things reviews. *Ieee Access* **2020**, *8*, 111763–111782. [CrossRef]
8. Stergiou, C.; Psannis, K.E.; Kim, B.-G.; Gupta, B. Secure integration of IoT and cloud computing. *Future Gener. Comput. Syst.* **2018**, *78*, 964–975. [CrossRef]
9. Hassan, R.; Jubair, A.M.; Azmi, K.; Bakar, A. Adaptive congestion control mechanism in CoAP application protocol for internet of things (IoT). In Proceedings of the 2016 International Conference on Signal Processing and Communication (ICSC), Noida, India, 26–28 December 2016; pp. 121–125.
10. Iyer, G.N. Evolutionary games for cloud, fog and edge computing—A comprehensive study. In *Computational Intelligence in Data Mining*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 299–309.
11. Cisco Systems. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*; White paper; Cisco Systems: Cisco San Jose, CA, USA, 2015.
12. Shabisha, P.; Braeken, A.; Steenhaut, K. Symmetric Key-Based Secure Storage and Retrieval of IoT Data on a Semi-trusted Cloud Server. *Wirel. Pers. Commun.* **2020**, *113*, 1–17. [CrossRef]
13. Zhu, C.; Tao, J.; Pastor, G.; Xiao, Y.; Ji, Y.; Zhou, Q.; Li, Y.; Ylä-Jääski, A. Folo: Latency and quality optimized task allocation in vehicular fog computing. *Ieee Internet Things J.* **2018**, *6*, 4150–4161. [CrossRef]
14. Bjerkevik, H.B.; Botnan, M.B.; Kerber, M. Computing the interleaving distance is NP-hard. *Found. Comput. Math.* **2019**, *20*, 1–35. [CrossRef]
15. Wang, H.; Olhofer, M.; Jin, Y. A mini-review on preference modeling and articulation in multi-objective optimization: Current status and challenges. *Complex Intell. Syst.* **2017**, *3*, 233–245. [CrossRef]
16. Han, D.; Li, Y.; Song, T.; Liu, Z. Multi-Objective Optimization of Loop Closure Detection Parameters for Indoor 2D Simultaneous Localization and Mapping. *Sensors* **2020**, *20*, 1906. [CrossRef] [PubMed]
17. Mayer, M.J.; Szilágyi, A.; Gróf, G. Environmental and economic multi-objective optimization of a household level hybrid renewable energy system by genetic algorithm. *Appl. Energy* **2020**, *269*, 115058. [CrossRef]
18. Albadr, M.A.; Tiun, S.; Ayob, M.; AL-Dhief, F. Genetic Algorithm Based on Natural Selection Theory for Optimization Problems. *Symmetry* **2020**, *12*, 1758. [CrossRef]
19. Abdali, T.-A.N.; Hassan, R.; Muniyandi, R.C.; Mohd Aman, A.H.; Nguyen, Q.N.; Al-Khaleefa, A.S. Optimized Particle Swarm Optimization Algorithm for the Realization of an Enhanced Energy-Aware Location-Aided Routing Protocol in MANET. *Information* **2020**, *11*, 529. [CrossRef]
20. Mai, Y.; Shi, H.; Liao, Q.; Sheng, Z.; Zhao, S.; Ni, Q.; Zhang, W. Using the Decomposition-Based Multi-Objective Evolutionary Algorithm with Adaptive Neighborhood Sizes and Dynamic Constraint Strategies to Retrieve Atmospheric Ducts. *Sensors* **2020**, *20*, 2230. [CrossRef]
21. Han, C.; Wang, L.; Zhang, Z.; Xie, J.; Xing, Z. A multi-objective genetic algorithm based on fitting and interpolation. *Ieee Access* **2018**, *6*, 22920–22929. [CrossRef]
22. Bao, C.; Xu, L.; Goodman, E.D.; Cao, L. A novel non-dominated sorting algorithm for evolutionary multi-objective optimization. *J. Comput. Sci.* **2017**, *23*, 31–43. [CrossRef]

23. Arslan, H.D.; Özer, G.; Özkiş, A. Evaluation of Final Product Integrated with Intelligent Systems in Architectural Education Studios. *Online J. Art Des.* **2017**, *5*, 119.
24. Qu, D.; Ding, X.; Wang, H. An improved multiobjective algorithm: DNSGA2-PSA. *J. Robot.* **2018**, *2018*, 9697104. [CrossRef]
25. Cai, D.; Gao, Y.; Yin, M. NSGAII with local search based heavy perturbation for bi-objective weighted clique problem. *Ieee Access* **2018**, *6*, 51253–51261. [CrossRef]
26. Chen, X.; Liu, Y.; Li, X.; Wang, Z.; Wang, S.; Gao, C. A new evolutionary multiobjective model for traveling salesman problem. *Ieee Access* **2019**, *7*, 66964–66979. [CrossRef]
27. Roy, P.C.; Islam, M.M.; Deb, K. Best order sort: A new algorithm to non-dominated sorting for evolutionary multi-objective optimization. In Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, Denver, CO, USA, 20–24 July 2016; pp. 1113–1120.
28. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Ieee Trans. Evol.. Comput.* **2002**, *6*, 182–197. [CrossRef]
29. Mkaouer, W.; Kessentini, M.; Shaout, A.; Koligheu, P.; Bechikh, S.; Deb, K.; Ouni, A. Many-objective software remodularization using NSGA-III. *Acm Trans. Softw. Eng. Methodol. (Tosem)* **2015**, *24*, 1–45. [CrossRef]
30. Metiaf, A.; Wu, Q.; Aljeroudi, Y. Searching with direction awareness: Multi-objective genetic algorithm based on angle quantization and crowding distance MOGA-AQCD. *Ieee Access* **2019**, *7*, 10196–10207. [CrossRef]
31. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 103–130.
32. Sun, Y.; Lin, F.; Xu, H. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. *Wirel. Pers. Commun.* **2018**, *102*, 1369–1385. [CrossRef]
33. Liu, L.; Chang, Z.; Guo, X.; Mao, S.; Ristaniemi, T. Multiobjective optimization for computation offloading in fog computing. *Ieee Internet Things J.* **2017**, *5*, 283–294. [CrossRef]
34. Cui, L.; Xu, C.; Yang, S.; Huang, J.Z.; Li, J.; Wang, X.; Ming, Z.; Lu, N. Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things. *Ieee Internet Things J.* **2018**, *6*, 4791–4803. [CrossRef]
35. Zahoor, S.; Javaid, S.; Javaid, N.; Ashraf, M.; Ishmanov, F.; Afzal, M.K. Cloud–fog–based smart grid model for efficient resource management. *Sustainability* **2018**, *10*, 2079. [CrossRef]
36. Rakshit, P. Memory based self-adaptive sampling for noisy multi-objective optimization. *Inf. Sci.* **2020**, *511*, 243–264. [CrossRef]
37. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *Ieee Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
38. Kleeman, M.P.; Seibert, B.A.; Lamont, G.B.; Hopkinson, K.M.; Graham, S.R. Solving multicommodity capacitated network design problems using multiobjective evolutionary algorithms. *Ieee Trans. Evol. Comput.* **2012**, *16*, 449–471. [CrossRef]
39. Fonseca, C.M.; Fleming, P.J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *Ieee Trans. Syst. Man Cybern.-Part A Syst. Hum.* **1998**, *28*, 26–37. [CrossRef]
40. Kursawe, F. A variant of evolution strategies for vector optimization. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Edinburgh, Scotland, 17–21 September 2016; pp. 193–197.
41. Poloni, C. *Hybrid GA for Multi Objective Aerodynamic Shape Optimisation*; Genetic Algorithms in Engineering and Computer Science; Winter, G., Periaux, J., Galan, M., Cuesta, P., Eds.; 1997; pp. 397–414.
42. Lin, J.C.-W.; Zhang, Y.; Zhang, B.; Fournier-Viger, P.; Djenouri, Y. Hiding sensitive itemsets with multiple objective optimization. *Soft Comput.* **2019**, *23*, 12779–12797. [CrossRef]
43. Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2001; Volume 16.