Journal of Intelligent Systems and Internet of Things
Vol. 15, No. 02, PP. 138-150, 2025
DOI: https://doi.org/10.54216/JISIoT.150210

Behavior of SPEA2 Algorithm to Resolve Scheduling Problem for IoT Cloud

Syed Mutiullah Hussaini[1], T. Abdul Razak[2], Muhammad Abid Jamil[3]

[1]Part-time Research Scholar, Jamal Mohammed College, Affiliated to Bharathidasan University, Trichy, Tamil Nadu State, India

[2]Associate Prof. in Dept., of Computer Science, Jamal Mohammed College, Affiliated to Bharathidasan University, Trichy, and Tamil Nadu State, India

[3]Asst. Prof., Department of Computer Science, Mustaqbal University, Buraydah, KSA

Emails: smhussaini@gmail.com; abdul1964@yahoo.com; majamil@uom.edu.sa

Abstract

The (SPEA2) Strength Pareto Evolutionary Algorithm 2 is a capable technique for managing multi-objective optimization problems. In IoT-cloud systems, this is particularly true with regard to task scheduling. Task scheduling and efficient resource allocation are necessary to improve performance and service quality as the Internet of Things (IoT) grows. SPEA2, which is especially helpful for cloud computing frameworks, is excellent at handling competing goals, such minimizing executing duration while increasing the usage of resources. The capacity of SPEA2 to keep a large collection of solutions allows for the exploration of various scheduling approaches in IoT-cloud scenarios, where tasks generated by several devices need to be handled effectively. In dynamic contexts where resource availability varies, this IoT-CS (IoT-Cloud_Scheduling) adaptability is essential. With SPEA2, researchers are able to create algorithms that enhance system responsiveness and dependability overall while also optimizing task scheduling. The management of resource distribution and task prioritizing difficulties is exemplified by the use of SPEA2 to scheduling problems in IoT-cloud infrastructures. Thus, by guaranteeing that computing resources are used efficiently while respecting performance limitations, SPEA2 makes a substantial contribution to the development of intelligent scheduling solutions that satisfy the changing requirements of IoT applications

Received: October 20, 2024 Revised: December 24, 2024 Accepted: January 21, 2025

Keywords: Cloud Computing; IoT things; IBEA; NSGA-II; SPEA2; IoT-CS; MOEA; Evolutionary Algorithm; Optimization; Scheduling Solution

1. Introduction

Effective work scheduling is becoming increasingly important in cloud computing environments because of the (IoT) volatile evolution. The immense amount of data generated by IoT devices presents additional issues for cloud computing systems, since it needs to be properly handled and stored. The use of cloud computing and the IoT has raised the need for effective job scheduling algorithms that may optimize several goals, including cost, energy usage, and completion time. IoT-based cloud computing environments are dynamic and complicated, making traditional scheduling methods unsuitable for handling them.

Researchers to address this challenge have investigated the use of evolutionary algorithms (EAs) to improve job scheduling in Internet of Things (IoT)-based cloud computing environments. The ability of EAs to generate a range of adaptive scheduling solutions that can respond to environmental changes has shown promising outcomes. Strength Pareto Evolutionary Algorithm 2 is one EA that demonstrates this (SPEA2). SPEA2 can efficiently search the space and produce a collection of non-dominated scheduling results while concurrently optimizing for multiple goals considering that it is a multi-objective evolutionary algorithm (MOEA). Adaptive scheduling solutions that can adjust to shifts in user requirements, work priorities, and resource availability can be generated via SPEA2 algorithm-based IoT-based cloud computing systems.

This study suggests a new model for SPEA2 algorithm-based scheduling of jobs optimization in an environment of computing in the cloud based on the Universal Internet of Things (IoT). The framework's goal is to produce flexible and varied scheduling options that can continue to operate at high levels of effectiveness and performance even when the demands and surroundings change. By enabling continuous runtime optimization of task scheduling, the suggested framework helps to design more dependable and effective IoT-cloud systems. This study can contribute to enhancing the overall robustness and performance of these systems by tackling the difficulties associated with job scheduling in dynamic IoT-based cloud computing environments [1] [2].

The term SPEA2_IoT-CS refers as framework to optimize the IoT cloud-scheduling problem. However, a large degree of unpredictability is often present in current SPEA2_IoT-CS, which expands the search field for every possible answer and makes the problem unsolvable. The precise search may not scale in runtime after this. Assigning different weights to different objectives can be challenging for researchers in some circumstances, while goal aggregation can be useful in others. This makes it more challenging for the whole set to maintain a range of possibilities [3].

When appropriately constructed, those algorithms can solve difficult IoT scheduling problems in a reasonable amount of time by providing approximate and nearly optimal solutions [4]. Additionally, it has been demonstrated that stochastic search works well for a number of real-time systems [5–7]. A single-objective evolutionary algorithm is frequently used to optimize IoT_CS after reducing an issue with several objectives to an accumulated single-objective problem [8].

In research [2, 4, 8], SPEA2 [9], a well-known multi-objective evolutionary algorithm (MOEA), has been used to enhance IoT_CS without requiring weighted aggregation. In the tradeoff space, the researchers demonstrated that MOEA could find congruent and different approaches as alternatives to optimization through objective aggregate.

The Internet of Things (IoT) is growing at a pace that is unparalleled, which has led to a surge in the quantity of data produced by devices that are connected. This has made effective processing and management strategies necessary. With its scalable resources and services, cloud computing has become a key solution for meeting the computational demands of Internet of Things applications. However, scheduling tasks and allocating resources becomes extremely difficult in dynamic IoT systems because of varying workloads and resource availability. The SPEA2 is an effective multi-objective optimization technique to address these problems. SPEA2 is made to manage complicated optimization issues by preserving a variety of solutions that strike a balance between conflicting goals, such maximizing resource usage and lowering execution time.

This feature is especially helpful in Internet of Things (IoT) cloud applications where jobs need to be properly scheduled to fulfill strict performance requirements while optimizing resource consumption. Notable is the way the SPEA2 algorithm behaves while tackling scheduling issues in IoT-cloud setups. It increases the system's overall responsiveness as well as the effectiveness with which tasks are completed. Through the utilization of SPEA2, scholars and professionals can create resilient scheduling plans that adjust to the ever-changing demands of Internet of Things workloads. This guarantees efficient resource distribution and preservation of service quality. This introduction lays the groundwork for investigating SPEA2's applicability to IoT-cloud scheduling issues and emphasizes how it may be used to maximize job management in settings that are getting more complicated and resource-constrained [33].

The essay's remaining sections are structured this way: The section 2 displays the related works. The discussion of certain cutting-edge works in this section serves as inspiration for our suggested methodology in section 3. Our suggested SPEA2 algorithm Proposed Approach-SPEA2-IoT-CS in section 4. Section 5 presents Experiments and Results, Discussion. Section 6 brings the paper concludes by discussing possible future initiatives.

2. Related Works

Our proposed approach is inspired by the discussions of certain cutting-edge research presented in this section. By maintaining copies on servers that are located far apart, RDM is a data privacy approach that minimizes the destruction of data and ensures availability of data [11, 12]. In addition, the RDM network needs to ensure that data that is distributed is neither lost nor corrupted while consuming the least amount of bandwidth possible. Real-time reconfiguration of the RDM can handle unknowns like missing or interrupted messages and network link problems. Along with operational costs that affect a regulating cost, each network link also has obviously great efficiency, delays, and breakage frequency. The overall efficacy and reliability of the RDM are evaluated using these metrics. If unanticipated issues arise, the RDM can modify its data mirroring techniques and network topology. Because it is so powerful and flexible, the RDM application can be thought of as a SPEA2_IoT-CS [13, 33].

[14] take a different approach, which asserts that improve quality while lowering execution complexity for IoT-CS optimizing by encoding an issue into MOEA utilizing the elitist chromosomal representation. When there are

few feasible alternatives, such a benefit is more inclined to be insignificant. Reliance-aware providers can effectively direct the search, leading to more diverse and convergent alternatives and improved IoT-CS optimization. However, dependency-aware operators may lose some of their effectiveness if they use it before first making sure that the adaptation method of choice is balanced.

The Proteus technique is a requirement-based runtime testing management system. Proteus is a framework that is specifically meant to carry out runtime operations related to testing, like online adaption and test execution. Proteus offers two test adaptability layers—test suite adaptability together with test suite value parameter adaptability—to accomplish this. Proteus framework is used to manage real-time assessment jobs, including the development and execution of scenario testing and evaluation programs. In order to help achieve this, Proteus eliminates two key jobs. The flexible job schedule for each IoT-CS setup is determined during design and is associated with a particular operational situation. Numerous test suites are included in these adaptive test programs. Every time a new IoT-CS setup is run, Proteus then conducts an evaluation phase, which may consist of several phases, each running a distinct test suite. A thorough explanation of each duty is then provided [15].

Our suggested method integrates the best features of evolutionary algorithms and IoT-CS. Particularly when using the SPEA2 approach, researchers can perform IoT-CS optimization using MOEAs even if they lack significant prior MOEA knowledge. In order to develop structured and understandable knowledge about the domain, evolutionary computation researchers can use the automatic features models transformation into a MOEA context to generate more creative findings in the IoT-CS field. Unlike the usual search-based software engineering challenges, our full technique increases the inner framework of MOEA by dynamically and automatically extracting IoT-CS domain knowledge. The forthcoming, SPEA2_IoT-CS will be extended to handle more conflicting goals and used in other IoT-CS domains.

3. SPEA2_IoT-CS as the Proposed Approach

3.1 Optimizing IoT Cloud Scheduling Problem Using SPEA2_IoT-CS Methodology

SPEA2-IOT_CS (Strength Pareto Evolutionary Algorithm 2 for Scheduling in IoT-Cloud Systems) is a suggested method that makes use of the advantages of the SPEA2 algorithm to improve job scheduling efficiency in IoT-cloud systems. This approach tackles the challenges of managing the many IoT devices that produce tasks and scheduling them while taking into account several goals, including meeting deadlines, saving execution time, and optimizing resource consumption.

SPEA2_IoT-CS's Working principal features are:

1. Multi-Objective Optimization: Utilizing SPEA2's multi-objective optimization capabilities, SPEA2_IoT-CS investigates a wide range of scheduling options. In order to ensure that no criterion is excessively favored at the expense of others, this enables a balanced approach to conflicting objectives.

2. Dynamic Adaptability: SPEA2_IoT-CS has mechanisms to adjust to the unpredictable characteristics of IoT-cloud systems, whereby task demands and resource availability might change quickly. To maintain responsiveness and peak performance in real-time applications, this flexibility is essential.

3. Allocation Efficiency: SPEA2_IoT-CS maximizes the utilization of computing resources to increase the general efficiency of the IoT-cloud platform. Ensuring efficient resource utilization and minimizing waste is crucial, especially in situations when tasks demand substantial processing power and storage capacity.

4. Cost and Deadline Awareness: The method incorporates cost and deadline awareness, which is essential in cloud computing settings. SPEA2_IoT-CS makes sure that projects are not only finished effectively but also within the allocated time and financial constraints by considering these factors.

5. Scalability: With the growing number of IoT devices and tasks, SPEA2_IoT-CS is made to scale well. This scalability guarantees that the scheduling algorithm will continue to function well as the system grows, meeting the increasingly demanding needs of Internet of Things applications.

To sum up, task scheduling for IoT-cloud systems has advanced significantly with the use of the SPEA2_IoT-CS technique. This approach attempts to address the complex needs of contemporary IoT applications while improving overall system performance by fusing the sturdy features of the SPEA2 algorithm with an emphasis on resource efficiency, dynamic adaptability, and multi-objective optimization.

A framework called SPEA2_IoT-CS is proposed created especially for carrying out testing tasks. An architecture called SPEA2_IoT-CS is used to manage real-time assessment operations, like modifying the setup of the IOT environment. To assist with this strategy, SPEA2_IoT-CS carries out two primary tasks. Multiple test suites comprise the design-time adaptable test approach for each IoT-CS setup, and each IoT-CS setup is linked to a

specific operational scenario. Second, SPEA2-IoT-CS performs an evaluation phase, which may include several iterations, each of which runs a distinct set of tests, whenever a new IoT-CS setup is run. All possible test suites for a particular operating system or group of platforms and environment-related aspects are included in an adaptive testing technique. For each IoT-CS setup, SPEA2_IoT-CS generates an adaptive test strategy and multiple intermediate, automatically generated test techniques in order to achieve this.

Let us use the IoT-CS framework to assess IoT setup as a flexible test strategy for a particular IoT-CS configuration in response to a specific group of operating conditions. Two goals must be addressed in this example: (1) the monetary value of tests and (2) the quantity of testing scenarios that require SPEA2 optimization. Based on the IoT-CS setups that are valid and invalid.

Using the SPEA2_IoT-CS technique, test suites are dynamically developed during runtime to provide optimum test adaption. The SPEA2_IoT-CS optimizer to dynamically look for combinations of IoT devices used a recursive loop. In the framework of dual objectives optimization, the encrypted recurrent cycle constantly modifications the IoT-CS to improve efficiency due to the constantly evolving setting.

A suggested choice looks for non-dominated approaches using a most appropriate adaption option. Optimizing does not mean taking into account the sequence in which a solution is implemented. In order to enforce the proper order of execution, it is therefore expected that the proposed method will assess the interdependence of IoT settings after identifying a viable and ideal solution. The evaluation process of objective functions rely on SPEA2_IoT-CS technique using defined parameter values.

If there are any issues with invalid test suites, the SPEA2_IoT-CS infrastructure will reactivate the valid test cases and reset the Internet of Things (IoT) setup before beginning a new evaluation phase. In order to optimize for several inefficient quality parameters, such as cost, timing of response, etc., the IoT-CS can modify characteristics at real-time. IoT-CS is often designed to use search techniques and dynamically search for characteristic sets that provide the best way to accomplish this goal.

Figure 1 depicts the SPEA2_IoT-CS architecture, which is divided into two primary components: The setup scenario that produces valid configurations is explained in the first section of the explanation. To adjust for a number of inactive indicators of quality, including cost and response time, the second section is responsible for an adaptive scenario that employs an evolutionary method and can modify characteristic sets at runtime.

Figure 1. SPEA2_IoT-CS Based Architecture

3.2. Conflicting Objectives and Dependent Designs in IoT-CS

Managing dependencies can be challenging, especially when the IoT-CS dataset includes a large number of category associations, because the majority of common precise and probabilistic search approaches (such as MOEA) are not designed to manage dependence requirements. Degradation in adaptation quality is possible if the restrictions are not managed properly [12]. When a huge number of configurations are generated by modern IoT-CS, they often show significant unpredictability, which leads to an expansion of search space. IoT-CS, decisions need to be taken often in order to simultaneously accomplish multiple competing quality objectives. According to a number of current methods [9], it is generally possible to accurately evaluate the relative importance of objectives as numerical weights, but this can be challenging in some circumstances [16]. Such characteristics usually negatively affect the finding procedure and result in an unfavorable, low-quality outcome when they are expressed and displayed wrongly. It is considerably more challenging to strike a reasonable balance between the objectives. Our suggested method, which successfully integrates a certain characteristic type with a MOEA, is driven by these problems.

3.3. Evolutionary Algorithm SPEA2 to Optimize IoT-CS (IoT Cloud Scheduling)

It can be difficult to identify specific IoT configurations and include them in the searching technique's description in a complex IoT-CS with numerous settings and factors. Therefore, configuration encoding may have a good or negative effect on a possible search algorithm's ability to search. This kind of description generates the necessary space for searching for the problem that has to be looked at in order to enhance the IoT-CS while it is functioning. Because it just encodes each attribute in binary form, prior study [16, 17] is unnecessary, even when various IoT-CS design elements might represent a similar feature of variation [18] or none at all.

3.4. IoT-CS Problem Description

Every time the Cloud-Fog computing system transmits queries from IoT-CS based Bag-of-Jobs (BoJ) apps to the Fog layer for processing, it separates them into distinct, controllable jobs. The different job's characteristics include the memory needed, the quantity of commands, and the overall size of both the input and the result files. The process frequently gets a set of n different jobs, as explained in our earlier work, assuming that J_c represents the c^{th} job.

$$J = \{J_1, J_2, J_3, \dots, J_n\} \quad (1)$$

The Cloud-Fog computing framework consists of distributed cloud networks and fog nodes, which are processors that utilize identical CPU rate, CPU usage charge, memory consumption expense, and bandwidth utilization cost. Cloud nodes are more costly to operate even if they frequently have more capacity than nodes embedded in fog. These nodes are part of the technique's c Cloud node and f Foggy nodes, as well which groups of n processors are:

$K(K = K_{cloud\ U} K_{Fog})$, It might be stated as

$$K = \{K_1, K_2, K_3, \dots, K_n\}, \quad (2)$$

where the i^{th} process node is shown by K_i.

All Jobs J_c

The processor $(J_c \in J_c\ jobs)$ is given $K_i (K_i \in Nodes)$, It is indicated as J_c^i.

$$K_i Jobs = \{J_x^i, J_y^i, \dots, J_z^i\} \quad (3)$$

The problem of job scheduling in a Cloud-Fog computing framework might be described as collection search.

$$NodeJobs = \{J_1^a, J_2^b, J_3^c, \dots, J_n^p\} \quad (4)$$

A node K_i needs the following execution time (EXT) to complete all of the jobs allocated for a group of jobs: (The time required for execution is supposed to be objectively -1).

Where $ExeTime(J_i)$

$$EXT(K_i) = \sum J_c^i \in K_i\ Jobs \qquad ExeTime(J_c^i) = \frac{\sum J_c^I \in K_i Jobs\ length(J_c)}{CPUrate(K_i)} \quad (5)$$

The time that node K_i processed J_k is expressed as ExeTime(J_k^i), and it may be calculated using:

$$ExeTime(J_c^i) = \frac{length(J_c^i)}{CPUrate(K_i)} \quad (6)$$

where $length(J_k^i)$ is how many instructions there are in job $J_k$and $CPUrate(K_i)$ is the node $K_{i`s}$ CPU rate.

Numerous variables, such as clocks rates, unit measure, concurrency at the instruction/path level, and others, influence this. The amount of time needed for the framework to complete each task, calculated from the moment the request is received until the last job is completed or the final device is used, is known as Makespan.

Makespan is calculated using a formula.

$$Makespan = Max[EXT((K_i)] \quad (7)$$

$$1 \leq i \leq m$$

The lower least Makespan should represent the Makespan limitation, or the shortest duration time that is required for the framework to do all jobs. To find the Minimal Makespan, it will be assumed that every node completes its assigned tasks at the same time.

$$MinimalMakespan = EXT(K_i) = \cdots = EXT((K_m)$$

Thus,

$$MinimalMakespan = \frac{\sum 1 \le k \le n\ length(J_c)}{\sum 1 \le i \le n\ CPUrate(K_i)} \quad (8)$$

In the Cloud-Fog system, processor, memory, and bandwidth expenses must be paid for each task that is finished. The following is the estimated cost for node Ki processing job J(c):

$$Cost(J_c^i) = cp(J_c^i)cm(J_c^i) + cb(J_c^i) \quad (9)$$

The following is the calculation of each cost using Equation (5).

Based on the cost of computation: (Cost of Computation Objective) -------2)

$$cp(J_c^i) = c1 * ExeTime(J_c^i) \quad (10)$$

where $ExeTime(J_c^i)$ is defined by the equation, and c1 is the node's cost of CPU usage per time unit. K_i (6).

The memory usage cost is as follows, in which c2 represents the memory utilize cost for each data unit in node K_i and $Mem(J_c)$ is the amount of memory required by job J_k:

$$cm(J_c^i) = c2 * Mem(J_c^i) \quad (11)$$

The node was in charge of the job J_K. The bandwidth needed by K_i is equal to the sum of each of the input and result file sizes, or $Bw(J_k)$. Let c3 be the rate of bandwidth utilizes cost for each data device, which has the following definition:

$$cb(J_c^i) = c3 * Bw(J_c^c) \quad (12)$$

The overall cost of completing each job, the following formula is used in the Cloud-Fog system:

$$TotalCost = \sum_{J_c^i \in NodeTasks} Cost(J_c^i) \quad (13)$$

The Cloud–Fog system determines MinTotalCost, the lowest cost necessary to complete a collection of jobs J, when each work is allocated to the least expensive node. The details about each node can be used to quickly determine the node executing job J_c at the lowest possible rate, $MinCost(J_c)$.

3.5. Using the SPEA2 Approach to Optimize the Goals

SPEA2 can concentrate its search approach by using a near-neighborhood estimation of density technique. Everyone's power or influence over others is taken into consideration by SPEA2's fitness attribution approach, and inversely as well [19]. To solve time and reliability problems during software project planning with better Pareto fronts for successful outcomes, Gueorguiev et al. employed the multi-objective genetic algorithm (MOGA) SPEA2 in their study [20].

The SPEA2 algorithm's pseudo code, which describes the population's fitness requirements for each individual, is explained below.

Step1. *Set SPEA2 algorithm's pseudo code parameters includes: Population Size (**Pop_N**), Size(**S_N**) with Max. Repetition (**$Max._R$**), Crossover% (**$Crossover_P$**), Mutation% (**$Mutation_P$**).*

Step2. *Make **Pop_N** random individuals as Pop*

Step3. ***For In= 1 to Pop_N***

Compute objective roles for Pop (with Cost, Risk, and Job assignment balance)

End

Step4. ***For R= I to $Max._R$***

i) ***For In = 1 to** Size(**S_N**)*

$S_{N\ (In)}$ Determines SPEA2 fitness values

End

ii) ***For In= 1 to Pop_N***

$Pop_{N\ (In)}$ Determines SPEA2 fitness values

End

*Replica non-dominated individuals of Population Size (**Pop_N**) to Size (**S_N**)*

***If** the number of individuals **size** > Size (**S_N**)*

Truncate the size considering truncation operator

Else *If the number of individuals **size** <= Size (**S_N**)*

Fill ***Size*** *with dominated individuals of* ***Pop***

iii) ***For c = 1 to*** $Pop_N \times$ ***Crossover***$_P$
Choose two chromosomes using binary tournament selection
Do Crossover
Compute objective roles
End

iv) ***For m = 1 to*** $Pop_N \times$ ***Mutation***$_P$
Choose two chromosomes using binary tournament selection
Do Mutation
Compute objective functions
End
Consider Crossover and Mutation as an individual as the Pop
End

Step 5. ***For n = 1 to*** *Size*(S_N)
If *size(n) not feasible*
Delete size (n)
End If
End

According to our suggested test, which has two primary goals (the cost and the timescale) that must be optimized, Figure 2 shows the SPEA2 workflow method. A sizable, mobile IoT-CS is the first step in the suggested process. Next, fitness functions are developed for the two objectives. The parameter values used by these functions to measure fitness are covered in Section 4. The process continues until the greatest outcomes are obtained. The procedure returns to the mating parameters and modification processes if the stop condition is not met. There, these values are assigned to the fitness parameters, and the following stages produce outcomes in the same way.

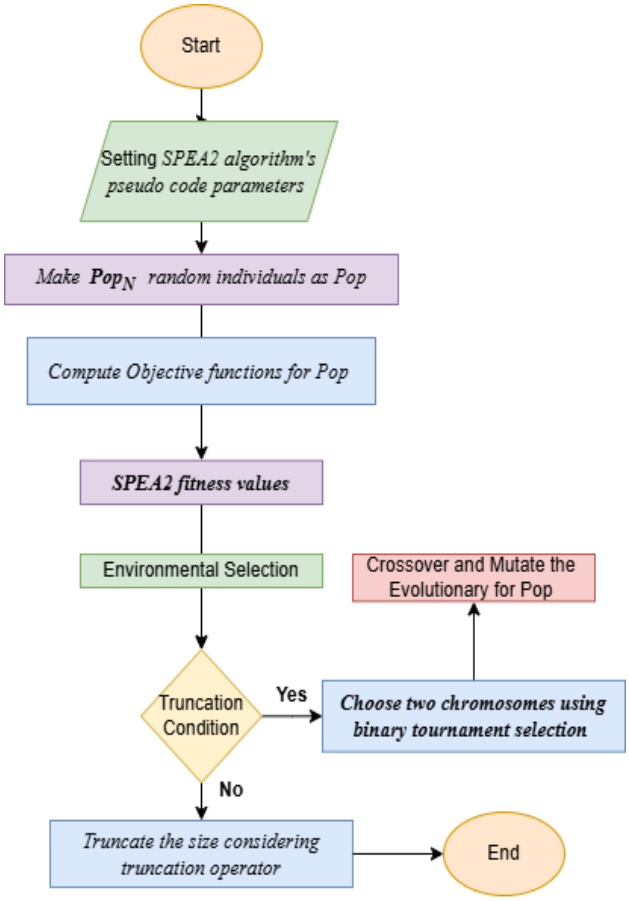

Figure 2. SPEA2 Workflow

4. Results and Experiments

The elite chromosomal representation makes it easier to specify the desired result (fit) of the operation and the input used for optimizing, as was stated earlier in the sections preceding this one. SPEA2_IoT-CS operates with a range of quantitative objectives for excellence, irrespective of how the true objective is formed, as the framework itself does not assume how those goals are organized inside.

Without assuming anything about how those objectives are internally organized, SPEA2_IoT-CS is adaptable enough to handle a variety of quantifiable quality criteria. The reason for this is that the framework itself is indifferent to the way the real goal is carried out. SPEA2_IoT-CS additionally allows for the consideration of more than two goals. We only do this in this post for the sake of a clearer explanation, even though the fundamental concept of multi-objective optimization applies to optimization with any number of objectives. The actual target functions that SPEA2_IoT-CS uses can be constructed using a variety of modeling approaches from other studies, provided that they align with the genes given by SPEA2_IoT-CS (e.g., derived from machine learning algorithms [18,21,22], analytical [23], and simulation-based [24]).

To prevent invalid offspring, the elicited gene sets and dependency networks are seamlessly included into the mutation process during real-time when the results are mutated. The edge mutant operator, which is on which the influence on each gene relies on its mutations percentage, is the foundation of our research in this work. The optional values of a gene are randomly selected according to the frequency of variation. Due to its relevance to optimization problems involving the selection of a value at random from an array of predefined values for an IoT-CS characteristic type, the operator known as mutation was chosen for evaluation. This work is very relevant to our proposed IoT-CS system optimization challenge. Additionally, it is a widely used operator. Because the violation of dependency is avoided every time a gene is changed, SPEA2_IoT-CS can be readily adjusted to function with different controllers, which modify gene sequences similar to the change operators.

It is necessary to eliminate invalid offspring when transferring solution components, just like in the mutation process. Using the integrated values structures and the acquired reliance connections, the crossing procedure in the MOEA may do this. Our studies involve two distinct genes from separate families that share a chromosomal position and may switch depending on crossover rates. The crossover strategy is the name given to this approach. Such homogeneous crossing regulators have been chosen because they reduce the problem of gene status. By reducing their sensitivity to the presence of highly dependent genes (characteristics) through order, this facilitates the relaxation of other IoT-CS design limitations. As long as every single set of swapped genes is consistently located at the identical spot in the encoded form, SPEA2_IoT-CS can be readily modified to function with a different operator, which is including mutation or crossover. This is because switching two genes stops the violation of dependencies.

In order to evaluate the SPEA2_IoT-CS technique's effectiveness using its modifications and state-of-the-art frameworks according to a number of criteria, we carried out extensive experiments utilizing the real functioning feature model IoT-CS. Specifically, we select the Pareto ratio and 1000 population size for SPEA2 [25].

The proposed approach can be analyzed using SPEA2_IoT-CS, as demonstrated by the suggested methodology. There are generally fewer workable solutions and a greater degree of complexity in the problem as a result of the feature model IoT-CS being under more strain due to the abundance of constraints. By using a two-objective optimization approach, the goal is to gradually reduce quality criteria.

Although SPEA2_IoT-CS can help concentrate the search for more effective approaches, we think that selecting a random adaptive solution from the remaining non-dominated choices has more benefits than avoiding the investigation of faulty IoT settings, which may be quite imbalanced for the various objectives. In our experiments, overhead costs along with quality factors either are set to standard values or have been tuned for the time of execution performance.

The IoT-Cloud-Scheduling features are shown in Table 1.

Table 1: IoT-Cloud-Scheduling features.

Factor(s)	**Configuration for the foggy clouds**
Cost of CPU utilize	[0.2,0.3]
CPU efficiency	[500,1500]
Cost of Memory Utilization	[0.02,0.03]
Cost of utilizing bandwidth	[0.02,0.03]
Frequency of Nodes	10

The IoT-Cloud-Scheduling features have an obligation to satisfy all of the demands voiced by the users. Every request generates a list of tasks that are then reviewed, and the amount of resources that are expected to be needed for each task is calculated. The number of jobs that are part of the set may differ somewhat from one another depending on the quantity of labor that each request requires. This led to the creation of 10 datasets for our investigation, each having 40–500 operations.

Table 2: SPEA2-IoT-CS Parameter sets

Factor(s)	**Figures**
Rate of crossing	50%
Rate of Mutation	50%
Count of generation	1000
Size of the population	400

The ideal parameter/factor configuration for the suggested issue is displayed in Table 2. In the meantime, the algorithm we propose for SPEA2-IoT-CS Parameter sets is based on (EAs) evolutionary algorithms.

Table 3 displays the scheduled period and computing costs of the SPEA2-IoT-CS algorithm. The two primary factors that influenced the role of fitness were the entire time and the total cost. Consequently, the SPEA2-IoT-CS algorithm demonstrated efficiency with respect to of time as well as cost on almost all datasets.

Table 3: Time and Cost of Execution for SPEA2-IoT-CS

	Execution Time	**Total Cost**
Number of Jobs	**SPEA2-IoT-CS**	**SPEA2-IoT-CS**
70	222.50*	835.58
100	395.55	1545.32
130	609.65	2565.47
170	822.67	3375.15
220	945.85	4035.35
260	1,285.95	4988.95
300	1,585.50	5288.65

Our proposed method outperformed the SPEA2-IoT-CS algorithm scheduled with respect of time required for execution. An overview of the particular problem characteristics for which particular MOEAs would be most effective.

The Pareto Ratio Metric is an effective way to evaluate the performance of MOEAs. This is how it influences outcomes and the reasons it is beneficial:

Impact on Outcomes

Analysis of Performance: the Pareto Ratio Metric measures the percentage of solutions that are near the actual Pareto front. Better performance is denoted by a larger Pareto ratio, which suggests that the algorithm is successfully approximating the ideal trade-off solutions. Convergence and Diversity: Two critical components of multi-objective optimization are revealed by the metric: convergence, which measures how near the solutions are to the genuine Pareto front, and diversity, which measures how effectively the solutions are distributed throughout the front [26].

Algorithm Comparison: It enables researchers and practitioners to compare various MOEAs quantitatively in order to determine which algorithm works best for a particular situation [27].

5. Beneficial for Description or Discussion

Clarity: The Pareto Ratio Metric makes it simpler to explain algorithm performance to stakeholders by providing a succinct and straightforward description. Making Decisions: It facilitates decision-making by offering a simple

metric to assess and choose the best algorithm for a given situation. Benchmarking: It ensures that achievements are judged objectively by acting as a baseline for assessing new algorithms or enhancements to current ones.

A summary comparing the computational features of the previously addressed multi-objective evolutionary algorithms (MOEAs) is given in Table 4.

Table 4: Summary of Multi-Objective Evolutionary Algorithms (MOEAs)

Algorithm	Type	Strengths	Weaknesses	Applications
IBEA	Indicator-based	Utilizes performance indicators; suitable for challenges with many objectives. [28]	For tiny populations, it could be slow.	Massive optimization
NSGA-II	Pareto-based	Effective non-dominated sorting with strong diversity preservation. [29]	Possibly difficult to solve complex problems.	Optimization and Engineering design
SPEA2	Pareto-based	Uses strength ranking and strong convergence.	More computational overhead.	Optimization with multiple objectives.

The suggested methodology produced outcomes that uphold an equitable trade-off among all goals. This is particularly interesting because SPEA2_IoT-CS focuses on scenarios in which the relative importance of IoT-CS 's conflicting goals is unclear and their measurement is excessively difficult. Figure 3. Reduces the quantity and costs associated with scenario testing by offering options. These thoughtfully balanced answers, which show a strong balance of approaches with respect to of two objectives, are probably found around the apparent curve. For each specific goal, these extreme solutions—also known as Pareto front solutions—often provide the poorest outcomes. Similar to searching for these possibilities in a non-dominated set, one can search for the appropriate answer or solutions that have the maximum overall proximity from the collection's excessive possibilities.

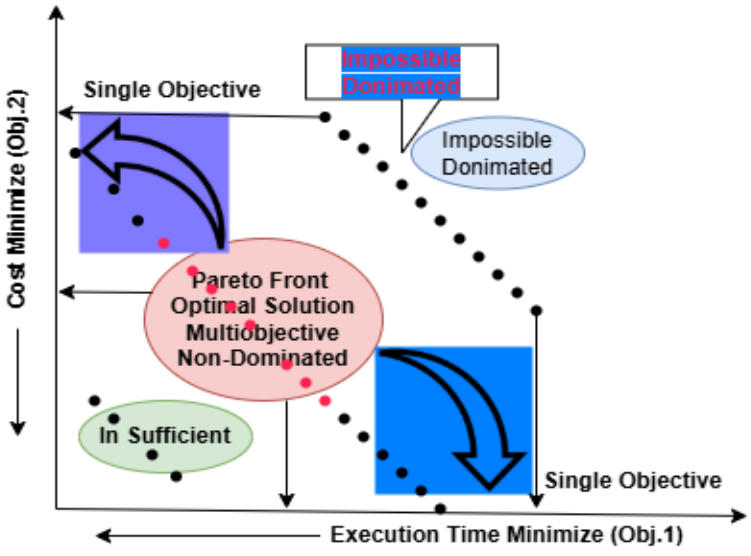

Figure 3. The Pareto Front Optimal Solutions (inimizing both are Default)

There exist Pareto optimal outcomes when the parameters for the constraint functions of objectives are modified. The concept of Pareto front, which is sometimes referred to as the set of the best alternatives in the space of objective operations, describes a group of solutions in multi-objective optimization problems (MOOPs) that perform better than the alternative options in the search space but are not dominated with each other.

Our approach involves using the SPEA2 method to select a distinct result from the MOEA results set. The feature-guided MOEA yields the final non-dominated set, which we see in Figure 4. With no reduction in generality, SPEA2_IoT-CS can function with multiple MOEAs. We run SPEA2_IoT-CS in this investigation using a single MOEA.

Multi-objective optimization techniques' performance in evolutionary algorithms is assessed using the Pareto Ratio Metric. Pareto Front: When none of the alternatives outperforms it for any given objective, the Pareto front is said to represent the collection of non-dominated solutions. The programs are it is employed to assess several multi-objective evolutionary algorithms (MOEAs) and identify the optimal strategy for a certain task [30]. It is made easier to appreciate the method's ability to approach the Pareto front, which is the collection of trade-off-optimal alternatives in an optimization issue with several objectives. The Pareto Ratio Metric determines the percentage of alternatives that are near the real Pareto front. Improved algorithm performance is shown by a greater Pareto ratio [31].

We used IoT models to balance the experiment's goal functions by monitoring the IOT_CS data for 300 intervals while subjecting it to a stochastic load. During the time of execution, the characteristics of sets are thereafter modified continuously and progressively. We also added a number of categories and numerical connections to the feature model. Using the Pareto Ratio Metric measure as a guide, we adjusted the variety level and throughput of specific concrete services step at each iteration to simulate volatility and dynamism in time-varying environmental conditions.

It raises the challenge of the task and decreases the number of feasible solutions. We may evaluate these competing targets for the composition as a whole and decide which ones to optimize and which to eliminate.

6. Conclusion and Future Work

This article demonstrates the effectiveness associated with the SPEA2 algorithm handles the complicated multi-objective optimization problems that arise in IoT cloud scheduling. The algorithm is especially well-suited to the dynamic nature of IoT workloads in cloud environments because it can maintain a diverse Pareto front while taking into account a number of conflicting objectives, such as resource utilization, energy consumption, reaction time, and cost.

When compared to conventional scheduling methods, the experimental outcomes confirm that SPEA2's improved fitness assignment scheme and environmental selection mechanism offer better results. Cloud service providers can choose from a variety of ideal scheduling options depending on their unique needs and priorities thanks to the algorithm's ability to effectively balance the trade-offs between various goals. In order to preserve solution variety and avoid early convergence, it has been particularly beneficial to incorporate density estimation approaches and fine-grained archiving algorithms.

Additionally, the suggested approach shows strong scalability traits, managing the growing complexity of IoT workloads with little to no decline in performance needed. This scalability is essential for real-world cloud-based applications where the number of IoT devices and service requests is increasing at an exponential rate. The adaptive nature of the algorithm ensures consistent performance across all operational contexts by enabling it to react efficiently to changes in workload behaviors and the availability of resources [32].

Yet, there are still chances for more study and advancement. Future research can examine adapting the approach for edge computing scenarios, integrating machine-learning techniques for predictive scheduling, and hybridizing with other meta-heuristic algorithms. Its practical usefulness could further be improved by integrating real-time modification capabilities and examining the algorithm's performance under more varied workload behaviors [34-36].

This study offers a solid, scalable, and effective solution to the IoT cloud scheduling issue, making a substantial contribution to the field of cloud resource management. According to the research, cloud service providers looking to maximize their resource allocation plans while preserving high quality of service standards for Internet of Things applications may discover that SPEA2-based techniques are useful resources.

References

[1] S. Ismail, K. Shah, H. Reza, R. Marsh, and E. Grant, "Toward management of uncertainty in self-adaptive software systems: IoT case study," *Computers*, vol. 10, no. 27, 2021.

[2] D. Arcelli, "Exploiting queuing networks to model and assess the performance of self-adaptive software systems: A survey," *Procedia Comput. Sci.*, vol. 170, pp. 498–505, 2020.

[3] A. J. Ramirez, D. B. Knoester, B. H. Cheng, and P. K. McKinley, "Plato: A genetic algorithm approach to run-time reconfiguration in autonomic computing systems," *Clust. Comput.*, vol. 14, pp. 229–244, 2011.

[4] Z. I. M. Yusoh and M. Tang, "Composite SaaS placement and resource optimization in cloud computing using evolutionary algorithms," in *Proc. IEEE 5th Int. Conf. Cloud Computing (CLOUD)*, Honolulu, HI, USA, 24–29 Jun. 2012, pp. 590–597.

[5] S. Hameed, Y. Elsheikh, and M. Azzeh, "An optimized case-based software project effort estimation using genetic algorithm," *Inf. Softw. Technol.*, vol. 153, p. 107088, 2023.

[6] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng, "SIP: Optimal product selection from feature models using many-objective evolutionary optimization," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, pp. 1–39, 2016.

[7] T. Chen and R. Bahsoon, "Self-adaptive trade-off decision making for autoscaling cloud-based services," *IEEE Trans. Serv. Comput.*, vol. 10, pp. 618–632, 2015.

[8] D. El Kateb, F. Fouquet, G. Nain, J. A. Meira, M. Ackerman, and Y. Le Traon, "Generic cloud platform multi-objective optimization leveraging models@ runtime," in *Proc. 29th Annu. ACM Symp. Appl. Comput.*, Gyeongju, Republic of Korea, 24–28 Mar. 2014, pp. 343–350.

[9] E. M. Fredericks, "Automatically hardening a self-adaptive system against uncertainty," in *Proc. 11th Int. Symp. Software Eng. Adaptive Self-Managing Syst. (SEAMS'16)*, Austin, TX, USA, 16–17 May 2016, pp. 16–27.

[10] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, pp. 712–731, 2017.

[11] C. G. Fidalgo, M. Sousa, D. Mendes, R. K. Dos Anjos, D. Medeiros, K. Singh, and J. Jorge, "Manipulating avatars and gestures to improve remote collaboration," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Shanghai, China, 25–29 Mar. 2023, pp. 438–448.

[12] K. Keeton, C. A. Santos, D. Beyer, J. S. Chase, and J. Wilkes, "Designing for DiIoT_CS ters," *FAST*, vol. 4, pp. 59–62, 2004.

[13] A. J. Ramirez, D. B. Knoester, B. H. Cheng, and P. K. McKinley, "Applying genetic algorithms to decision making in autonomic computing systems," in *Proc. 6th Int. Conf. Autonomic Comput.*, Barcelona, Spain, 15–19 Jun. 2009, pp. 97–106.

[14] T. Chen, K. Li, R. Bahsoon, and X. Yao, "FEMOSAA: Feature-guided and knee-driven multi-objective optimization for self-adaptive software," *ACM Trans. Softw. Eng. Methodol*, vol. 27, pp. 1–50, 2018.

[15] E. M. Fredericks and B. H. Cheng, "Automated generation of adaptive test plans for self-adaptive systems," in *Proc. IEEE/ACM 10th Int. Symp. Software Eng. Adaptive Self-Managing Syst.*, Florence, Italy, 18–19 May 2015, pp. 157–167.

[16] D. Han, Y. Cai, W. Chen, Z. Cui, and A. Li, "Timed-IOT_CS: Modeling and analyzing the time behaviors of self-adaptive software under uncertainty," *Appl. Sci.*, vol. 13, p. 2018, 2023.

[17] R. A. Sulaiman, D. N. Jawawi, and S. A. Halim, "Cost-effective test case generation with the hyper-heuristic for software product line testing," *Adv. Eng. Softw.*, vol. 175, p. 103335, 2023.

[18] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," *Inf. Syst.*, vol. 35, pp. 615–636, 2010.

[19] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK Report*, vol. 103, ETH Zurich, 2001.

[20] S. Gueorguiev, M. Harman, and G. Antoniol, "Software project planning for robustness and completion time in the presence of uncertainty using multi-objective search-based software engineering," in *Proc. 11th Annu. Conf. Genetic Evol. Comput.*, Montreal, QC, Canada, 8–12 Jul. 2009, pp. 1673–1680.

[21] L. Zhang, W. Li, and L. Zhang, "Adaptive QoS-aware resource allocation in cloud computing systems using multi-agent reinforcement learning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 5, pp. 1556–1569, Sept.-Oct. 2023.

[22] A. S. Tan, J. Sun, and X. Xu, "QoS-aware adaptive scheduling for cloud services based on evolutionary algorithms," *IEEE Trans. Serv. Comput.*, vol. 14, no. 3, pp. 856–867, May-Jun. 2021.

[23] N. Roy, A. Dubey, A. Gokhale, and L. Dowdy, "A capacity planning process for performance assurance of component-based distributed systems," in *Proc. 2nd ACM/SPEC Int. Conf. Perform. Eng.*, Delft, The Netherlands, 12–16 Mar. 2016, pp. 259–270.

[24] F. Fittkau, S. Frey, and W. Hasselbring, "CDOSim: Simulating cloud deployment options for software migration support," in *Proc. IEEE 6th Int. Workshop Maint. Evol. Service-Oriented Cloud-Based Syst. (MESOCA)*, Trento, Italy, 24 Sep. 2012, pp. 37–46.

[25] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257–271, 1999.

[26] J. Cao, Z. Yan, Z. Chen, and J. Zhang, "A Pareto front estimation-based constrained multi-objective evolutionary algorithm," *Appl. Sci.*, vol. 53, pp. 10380–10416, 2023.

[27] G. G. Yen and Z. He, "Performance metrics ensemble for multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 17, pp. 736–749, 2013.

[28] S. M. Hussaini and T. A. Razzak, "Optimizing scheduling problem IoT based cloud computing environment using EAs (IBEA algorithm)," *J. Data Acquis. Process*, vol. 38, no. 3, 2023, ISSN 1004-9037.

[29] Z. Movahedi, B. Defude, and A. M. Hosseininia, "An efficient population-based multi-objective task scheduling approach in fog computing systems," *J. Cloud Comput.*, vol. 10, no. 53, pp. 1–17, Oct. 2021.

[30] H. Gao, H. Nie, and K. Li, "Visualization of Pareto front approximation: A short survey and empirical comparisons," *arXiv: 1903.01768v1*, 5 Mar. 2019.

[31] D. Hadka, "MOEA Framework—A free and open source Java framework for multiobjective optimization," Version 2, 2015. Available online: http://moeaframework.org/ (accessed on 8 October 2022).

[32] M. A. Jamil, M. K. Nour, S. S. Alotaibi, M. J. Hussain, S. M. Hussaini, and A. Naseer, "Adaptive test suits generation for self-adaptive systems using SPEA2 algorithm," *Appl. Sci.*, vol. 13, no. 20, p. 11324, 2023.

[33] R. Wazirali, W. Alasmary, M. M. Mahmoud, and A. Alhindi, "An optimized steganography hiding capacity and imperceptibility using genetic algorithms," *IEEE Access*, vol. 7, pp. 133496–133508, 2019.

[34] M. A. Jamil and M. K. Nour, "Managing software testing technical debt using evolutionary algorithms," *Comput. Mater. Contin.*, vol. 73, pp. 735–747, 2022.

[35] A. S. Saini, V. Mishra, and S. Gupta, "A hybrid evolutionary approach for feature selection in software product line engineering," *J. Softw. Evol. Process*, vol. 35, no. 4, pp. 1–16, Apr. 2023.

[36] M. A. Jamil, M. K. Nour, S. S. Alotaibi, M. J. Hussain, S. M. Hussaini, and A. Naseer, "Adaptive test suits generation for self-adaptive systems using SPEA2 algorithm," *Appl. Sci.*, vol. 13, no. 20, p. 11324, Oct. 2023.