Int. J. Nonlinear Anal. Appl. 12 (2021) No. 2, 2303-2331
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2020.21625.2283

Energy aware multi objective algorithm for task scheduling on DVFS-enabled cloud datacenters using fuzzy NSGA-II

Saeed Fatehi[a], Homayun Motameni[b,*], Behnam Barzegar[a], Mehdi Golsorkhtabaramiri[a]

[a]*Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran*
[b]*Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran*

(Communicated by Madjid Eshaghi Gordji)

Abstract

Nowadays, energy consumption is curtailed in an effort to further protect the environment as well as to avoid service level agreement (SLA) breach, as critical issues in task scheduling on heterogeneous computing centers. Any reliable task scheduling algorithm should minimize energy consumption, makespan, and cost for cloud users and maximize resource utilization. However, reduction of energy consumption leads to larger makespan and decreases load balancing and customer satisfaction. Therefore, it is essential to obtain a set of non-domination solutions for these multiple, conflicting objectives, as a non-linear, multi-objective, NP-hard problem. This paper formulates the energy efficient task scheduling in green data centers as a multi-objective optimization problem so that fuzzy Non-dominated Sorting Genetic Algorithm 2 (NSGA-II) has been applied using the concept of Dynamic Voltage Frequency Scaling (DVFS). In this procedure, we adopted fuzzy crossover and mutation for optimal convergence of initial solutions. For this purpose, the binary variance function of gene values and the mean variance function of objective values are proposed for fuzzy control of mutation rate, increasing the variation in the optimal Pareto front as well as the correct frequency variance function of the processors engaged in scheduling to control the crossover rate. This serves to add the objective of indirect load balancing to the optimization objectives, thereby to replace the three-objective optimization process with four-objective optimization. In the experiments, the proposed NSGA-II with fuzzy algorithm is compared against the NSGA-II algorithm, involving three scheduling strategies namely Green, Time and Cost Oriented Scheduling Strategy. The simulation results illustrate that the newly method finds better solutions than others considering these objectives and with less iteration. In fact, the optimal Pareto solutions obtained from the proposed

*Corresponding Author
Email address: motameni@iausari.ac.ir (Homayun Motameni)

Received: February 2020 *Accepted:* November 2020

method improved the objectives of makespan, cost, energy and load balance by 4%, 17%, 1% and 13%, respectively.

Keywords: Green Computing, Multi Objective Optimization, Pareto solutions, DVFS, Task scheduling.

1. Introduction

Energy management has always been one of the major concerns in cloud when it comes to supporting the rapid growth of data centers and computing. Since fossil fuels are the main source of energy production, energy consumption not only increases the cost of electricity for service providers but also plays an important role in intensifying CO_2 emissions and greenhouse gases resulting in environmental pollution [1]. For instance, medium-sized data centers, such as university centers, consume about 8,000 KW of electricity [2]. Large companies use computing nodes with DVFS capability, in an attempt to minimize their dependence on conventional electricity generation [3]. Furthermore, the emergence of cloud computing has led to vast development of computing resources, where the maintenance of large-scale infrastructures has been assigned to cloud data centers through centralized hardware and software management practices, as the technique becomes widespread, cloud computing moves toward commercialization, where computing and non-computing owners provide resources to the cloud system with certain financial incentives. In this scenario, remote clients and users will be able to access resources under the pay-as-you-go model [4]. Basically, the market-based model is known as one of the crucial economic cloud models. In this model, resources hold prices set either by their owners or on a supply/demand basis offered to the cloud system.

The efficient execution of tasks on cloud depends in turn on proper scheduling and load balancing on such a system. Load balancing is a technique for distribution of workload equitably among computing resources in the cloud environment. This serves to achieve optimal resource efficiency, maximize Throughput, reduce task completion time (makespan) and avoid excess overload on resources in the cloud system is necessary [5]. Therefore, the proposed scheduling technique on data centers with DVFS-enabled processors is expected to incur lowest cost at minimal time and energy consumption for task completion, while satisfying the load balance.

The objectives of minimum cost, minimum execution time, minimum energy consumption and maximum load balancing are inconsistent so that fast computing resources incur the highest cost to complete tasks and vice versa. Moreover, the load distribution must be equitably distributed among computational resources so as to achieve the objective of maximum load balancing. This exacerbates the energy consumption while increasing the load balance, and reducing the execution time of tasks. Since the problem of task scheduling on cloud data center processors is of NP Hard, and given the multiple inconsistent objectives, the adoption of single-objective optimization algorithms can eliminate scheduling information and deprive users of the right to choose [6]. Therefore, it is recommended to employ multi-objective optimization scheduling methods that offer users multiple solutions in the form of optimized Pareto fronts.

In the proposed method, due to the multiple conflicting objectives, we adopt a genetic algorithm with non-dominated sorting. Moreover, we equipped the proposed algorithm with mutation and crossover fuzzy operators based on variance function, so that the Pareto front is produced at higher quality and less iterations. Meanwhile, the included fuzzy operators enhance the intelligence of the algorithm.

The rest of this paper has been organized as follows. Section 2 reviews related works. Section 3 describes the problem description. Section 4 describes the proposed method Based on fuzzy NSGA-II and mathematical models and the multi-objective estimation scheduling model. Results of evaluation experiments compared with other algorithm are discussed in Section 5. Finally, Section 6 concludes this paper.

2. Literature Review

Task scheduling in cloud data centers consuming a substantial amount of energy, has remained as an inconclusive problem in research, because achievement of an ideal state still requires further investigation. Therefore, the efforts made by numerous researchers have focused on reduction of energy consumption, makespan and cost, while enhancing resource efficiency and other parameters. In most previous efforts, the explorations to solve this problem have been single-objective, making decisions on optimization prior to the optimization operation. These approaches adopt decisions regardless of possible choices about solutions. On the other hand, it is not easy to build a single objective function by combining several objective functions, because there might be conflicting, contradictory objectives in such problems. In modern approaches, multi-objective optimization algorithms have replaced single-objective ones. Pareto's optimal solutions in multi-objective optimization allow making a decision and choosing the right solution from among several optimal solutions available. Genetic algorithm is one of the most efficient optimization algorithms to cover the problem of task scheduling in heterogeneous systems. Therefore, multi-objective genetic algorithm is preferable over other alternatives. One of the most efficient types of multi-objective genetic algorithms is known as NSGA-II.

Barzegar et al. [1] proposed a dual-phase, time- and energy-aware scheduling algorithm dubbed EATDCDA to schedule directed acyclic graphs on DVFS-capable processors at the cloud. In the first phase of the new method, a smart combination of duplication and clustering strategies focus on reducing makespan and energy consumption while maintaining a set limit for throughput. After determining the critical path and specifying the non-critical task in the second phase, the slack time of each non-critical task was calculated and then the frequencies of DVFS-capable processors were scaled down without prolonging the task execution times. The authors in [7] suggested NSGA-II together with controlled elitism for task scheduling with dual-objective optimization. The first objective is to minimize makespan while the second objective is to minimize flowtime. One of the key points of this procedure is the control of elitism. Even though it does leave a significant effect on the quality of solutions, the adoption of multi-objective optimization in the problem of task scheduling in heterogeneous computational systems is considered a new venture. The authors in [8] investigated the problem of scheduling in distributed systems using a non-dominated multi-objective particle swarm optimization algorithm, where the two conflicting objectives of makespan and flowtime are optimized simultaneously. In [9], the authors intended to solve the problems of resource load balancing in a hybrid technique, which combines two algorithms of SA and GA. This provides an optimal strategy for scheduling independent tasks in grades based on the asexual Genetic-Simulated Annealing Clonal Algorithm (GSACA). The results of the newly proposed method have demonstrated desirable performance compared to the genetic algorithm and simulated annealing algorithm. In [10], the authors examined a hybrid approach to scheduling tasks while maintaining a load balance in a distributed environment. This approach mainly attempts to achieve the task assignments with minimum execution time and maximum node productivity and a good load balance among the nodes. As representatives of both classes, the FCFS algorithm has been combined with the genetic algorithm to cooperate. Jin et al. [11] proposed a polynomial solution that was called Speed Scaling (SS) for task scheduling with polynomial complexity on restricted parallel computing nodes. Piatek et al. [12] proposed an energy model that can be effectively used to estimate performance metric and energy consumption with DCworms simulation framework. Ding et al. [13] proposed algorithm with deadline constraint, EEVS and develop new VM scheduler to reduce energy consumption. A taxonomy of resource management techniques with Performance evaluation parameters, evaluation platforms and design goals was presented in [5, 14] by considering conflicting metrics includes energy aware, SLA-aware, network load aware, load balancing, revenue handling and hybrid cloud. A green renewable energy scheduling algorithm in the data center when considering the environment respect, system cost and the energy crisis was proposed by Lei et al. [15] to increase the utilization of renewable energy as well as the task satisfaction rate and reduce the system cost in a cloud data center. Sathya and GaneshKumar [16] proposed a multi-objective evolutionary model to regulate the frequency and voltage of the VM so that NSGA-II was used together with DVFS to achieve a set of non-domination solutions. Furthermore, the proposed model predicted the VM based on the properties of tasks and processors using Artificial Neural Network (ANN). Sathya and GaneshKumar [16] also investigated the impact of simultaneous using NSGA-II and ANN as a further discussion.

Literature also reveals works on the usage of DVFS for proposing task slack time algorithms [17, 18]. A comprehensive review on how to utilize the DVFS technique in cloud data centers has been published in [19]. In this work, the opportunity of applying DVFS technique in task scheduling problems towards reduction of energy consumption without violating the user's SLA was discussed. A hybrid discrete particle swarm optimization (HDPSO) algorithm for solving scientific workflow scheduling on heterogeneous platforms has been presented [20]. The notable procedure of HDPSO is to call Hill Climbing algorithm randomly during execution of main DPSO.

3. Problem description

Scheduling tasks on computational data centers with a market-based cloud economic model pursues several key objectives. Firstly, it intends to minimize the time required to complete the set of tasks delegated to the processors, known as makespan, which depends on the nature of the distributed computing systems [21, 22]. In conventional methods, this objective is covered by default. The second objective involves the cost of executing user tasks in the cloud. In a market-based economic cloud model, resource owners pre-set their service prices as per the amount of resources consumed by users. Thus, the cost of task execution is of utmost importance in the cloud computing with a market-based economic model. There is another objective known as load balancing considered in optimization of the task scheduling problem. In fact, load balancing is addressed as one of the critical objectives of the problem. Moreover, one of the key green objectives is the amount of energy consumed by cloud data center processors. The noteworthy point is that the task scheduling problem in mainstream methods is optimized through an objective function. Given the multiple, inconsistent objectives, new approaches to multi-objective optimization are necessary. After all, integration of inconsistent objectives into a single one becomes difficult and gives rise to potential fault when the objectives do not vary at identical rates.

In the discussion of cost, time, and energy, considered in cloud computing with a market-based economic model, the higher the cost of task execution, the lower the computational time, and higher the energy consumption. Consequently, the longer the computation time, the less expensive the processors dedicated to execution, and thus the cost of computing and power consumption will be mitigated. This is an instance of conflicting objectives with non -identical variation rates. In an effort to increase time and reduce cost, for example, we may find a solution where there is insignificant cost reduction at longer times. In this scenario, the user will experience high execution cost and time. As for load balancing, the amount of energy consumption and execution time, greater load balancing implies that tasks are distributed evenly on the processors. As a result, the amount of energy consumed by the processors increase, whereas proper load balancing leads to shorter task execution time on cloud data center processors. Therefore, this problem is resolved through multi-objective optimization algorithms attempting to generate multiple solutions.

GA is one of the best possible strategies to tasks scheduling in clouds, since its efficiency has been proven for solving the scheduling and dynamic load balancing problems in distributed parallel systems. Due to the multiple, conflicting objectives, we adopt a multi-objective genetic algorithm known as NSGA-II, which can achieve Pareto front [21, 23]. In addition to optimizing price, time, energy and load balance, the proposed algorithm enables users to choose one or more alternatives available by providing optimal or near-optimal various solutions. Additionally, we equipped the proposed algorithm with fuzzy genetic operators so as to generate a higher quality Pareto front with less iteration. In the proposed approach, we also consider cloud data centers consisting of DVFS-enabled processors. This allows them to operate with different voltage-frequency pairs, which in turn reduces power/energy consumption.

4. Proposed Method Based on Fuzzy NSGA-II

NSGA-II is a multi-objective evolutionary optimization algorithm which finds Pareto solutions from initial population through consecutive iterations [24]. Generally the objective parameters have conflicts within the fitness function. This algorithm relies on the principles of elitism to maintain variation, emphasizing non-dominated solutions and forming the Pareto front as optimal Pareto solutions. NSGA-II adopts two effective strategies, including elite preservation and variation preservation over generations. The variation preservation, i.e. niching technique, is used to assign variation rankings to all subjects engaged in a non-dominated front. The members falling in any non-dominated fronts in a lower density area are assigned higher rankings. The crowding distance criterion is adopted to calculate the mean distance between two solutions on each side of the particular solution is achieved along each objective vector. Therefore, NSGA-II has a unique advantage over other multi-objective optimization algorithms [7].

In optimizing the problem of scheduling standalone tasks in a market-based cloud computing, we first optimize the objectives of makespan, price, and energy through NSGA-II. We also propose the idea of I AM HERE!!!!I AM HERE!!!!mutation and crossover fuzzy operators and compare it against fixed rate operators in an effort to generate Pareto optimal solution at faster rate, minimal iteration and higher quality. For this purpose, we propose several functions based on problem objectives in order to achieve fuzzy control of mutation and crossover rates. Then, load balancing is indirectly added to the optimization objectives. The fuzzy function inputs at this stage represent the variance between the fitness's of population subjects as well as the variance of subject genes indicating the level of difference between the values of genes. This serves to increase variation in the optimal Pareto front and its faster formation. The output of this fuzzy function determines the mutation rate.

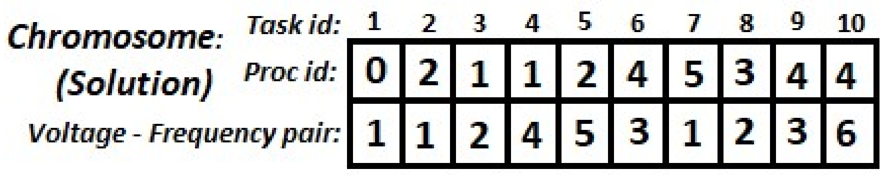
In the next stages, the load balancing is indirectly considered to further improve the algorithm's efficiency. Using the NSGA-II and variance-based fuzzy crossover operator, the four-objective optimization is replaced by a three-objective optimization process. For this purpose, a function is designed to indirectly optimize load balancing for fuzzy crossover rate control. The fuzzy function inputs at this stage include the variance between subject fitnesses together with the variance of the number of processors engaged in scheduling. The first input is used to increase variation in the Pareto optimal front while the second input is used together with the makespan objective to indirectly apply load balance. The output of this fuzzy function determines the crossover rate in the population of each generation.

4.1. Solution Encoding

Each solution is represented as a two-dimensional vector with a length equal to the number of tasks. For n-task and m-processor scheduling problem, we consider a chromosomes of length n. The values in the first row of the chromosome range from 0 to m-1, indicating the processor index assigned to the task corresponding to that gene. The values of each gene in the second row represent the voltage-frequency level of the processor assigned to the task. This encoding strategy has been illustrated in Figure 1, while the voltage-frequency levels of the processor equipped with the DVFS technique can be viewed in Table 1.

Table 1: Voltage-frequency pairs of AMD Athlon-64 processors [3]

Level	Frequency (GHz)	Voltage (V)	Speed (MIPS)	Relative speed	Power
0	0.8	0.9	4000	40	2.03
1	1.0	1.0	5000	50	3.85
2	1.2	1.1	6000	60	4.84
3	1.4	1.2	7000	70	5.95
4	1.6	1.3	8000	80	6.35
5	1.8	1.4	9000	90	7.2
6	2	1.5	10000	100	8.4



Chromosome: (Solution)										
Task id:	1	2	3	4	5	6	7	8	9	10
Proc id:	0	2	1	1	2	4	5	3	4	4
Voltage - Frequency pair:	1	1	2	4	5	3	1	2	3	6

Figure 1: Schematic overview of the chromosome used for the scheduling problem

For example, as shown in Figure 1, the 6th task is assigned to the 4th computing node with 1.2 V and 1.4 GHz, where the corresponding processing speed and relative processor speed are 7000 MIP and 70%, respectively.

In order to generate the initial population to a number of k, a random number between 0 and m-1 is assigned for each gene from the first row of the chromosome, while a random number between 0 and L-1 is assigned for each gene from the second row of the chromosome. Then, the k number of chromosomes this process is iterated until the initial population is generated.

4.2. Mathematical modeling of the system

This section provides formal definitions for the system's architecture model, task model, resource model, energy model, and energy-aware multi-objective task scheduling model in green cloud data centers used in implementing our proposed method. The notations have been summarized in Table 2.

Table 2: Definitions of notations

Notation	Definition
t_i	The task ith
DC	Data Center
N or n	The number of tasks (nodes) in DAG
MIPS	Million Instruction Per Second
MI	Million Instruction
$et(t_i, p_i)$	The execution time of task ith on processor jth
P	Power consumption
$P_{dynamic}$	Dynamic power consumption
P_{static}	Static power consumption
E	Energy consumption
k	Number of initial population
T_i	Size of ith task in terms of MI
et	The communication time to transfer message d_{ij} between task t_i and t_j
T_{exe}	Execution time of task i on processor j
SP_j	Processing speed of processor j in terms of MIPS
j	Processor index
pr_j	Source unit price j per second
Price(j)	Cost of executing task i on source j
PF	Pareto Front set of members
$D(k)$	Then number of k^{th} member genes with values different from the corresponding genes in the best number
σ^2	Mean variance of objective values
A_1	Input membership function for binary variance of gene values
A_2	The input membership function for the correct mean variance of sources involved
B	Input membership function for mean variance of objective values
PXover	Output membership function for crossover rate
α	Expected mean value of each processor frequency
L_i	Frequency of each processor in scheduling
A_j	Set of task indices assigned to processor j
$\|j\|$	Number of computing nodes
$\|k\|$	Number of virtual machines in each computing node
$\|m\|$	Number of processors in each virtual machine
(v_j, f_j)	Voltage and frequency pairs of processor
v_{high_j}	Highest voltage of jth processor
f_{high_j}	Highest frequency of jth processor

4.2.1. Proposed system architecture model

This architecture consists of batch tasks, a data center with m number of heterogeneous, independent computing nodes, and a global scheduler. At any given time, an application is subdivided into multiple tasks and sent to the cloud data center global scheduler. After optimizing the task scheduling problem based on scheduling information and preset objectives, the cloud scheduler offers the user different schedules. Being informed of the energy consumed, the price and time to complete a task in the proposed schedules, the user or decision-maker selects one execution method.

Then, the user-selected scheduling is executed and the tasks are assigned to one of the processors equipped with DVFS-enabled, which enables processors to operate with multiple voltage-frequency pairs. Each processor in computing nodes has a local queue. Computing nodes receive tasks for execution from their local queue. The post-completion results of execution are integrated and delivered to the user by the cloud scheduler. Upon approved payment of task execution cost, the user will receive the result of application execution at the time set in scheduling (Figure 2).

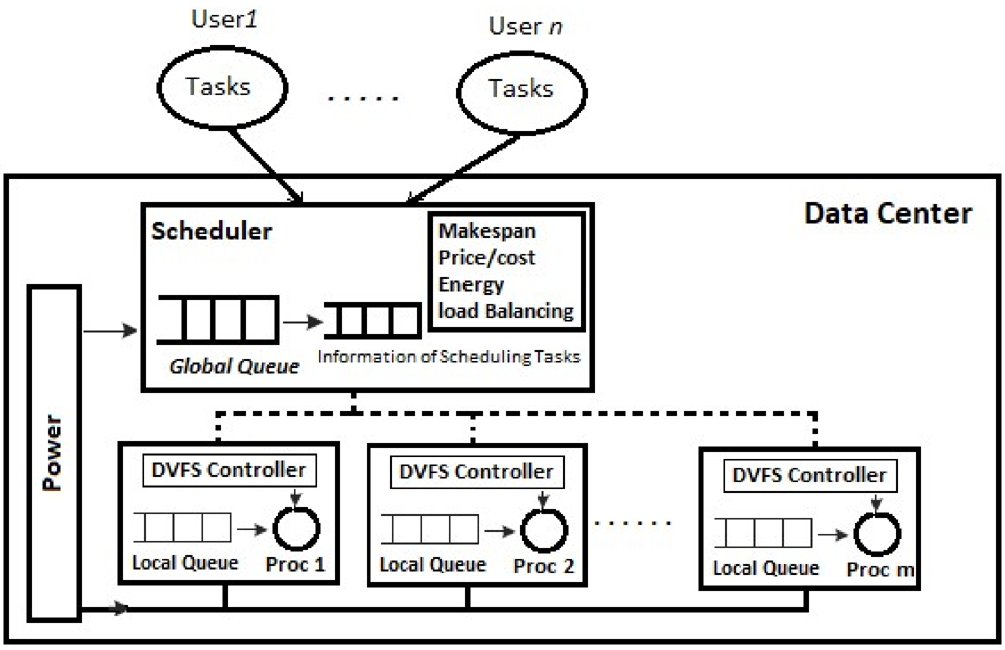


Figure 2: Architecture model of the proposed system for Tasks scheduling in heterogeneous cloud data center

4.2.2. Task model

Each application is composed of several independent and heterogeneous tasks represented by $T = \{t_1, t_2, \cdots, t_j, \cdots, t_N\}$. Each task is executed uninterruptedly on only one processor. The size of tasks is selected randomly within one range (minimum ... maximum) so that the distribution of task sizes will be uniform.

4.2.3. Computational Resource Model

Each cloud computing node is a single processor with four features based on economic cloud computing principles as follows:

- The processing speed in millions of instructions per second,
- The price of each processor in terms of money per second,
- The specific tasks on each processor available prior to the scheduling execution. Once the task is completed, it begins to complete a new task.
- The number of voltage-frequency pairs per processor along with the amount of power consumed per level.

The current speed, price and load of each processor are each determined randomly within a specific interval. For this purpose, a range (minimum ... maximum) is set for each factor and then produced uniformly. The prices of processors are directly proportional to their processing speeds. In other words, the higher the processing speed of a processor, the higher the price. Considering the speeds and generating random prices, they are arranged in an ascending order to be directly proportional to each other.

4.2.4. Energy model

Generally cloud data centers incorporates CMOS circuits based processors with dynamic and static power consumption based on Equation (4.1).

$$P = P_{dynamic} + P_{static}. \tag{4.1}$$

Execution of tasks consume processors computation energy which can be calculated solely with dynamic power consumption through Equation (4.2).

$$P_{dynamic} = ACv^2 f \tag{4.2}$$

DVFS-enabled processors are exploited to in HPC systems.

The new approaches are used in reduction of energy consumption such as DVFS-enabled processors exploited in HPC systems. The execution of DVFS-enabled processors are based on two factors namely; voltage and frequency (v_j, f_j).

Table 1 shows an example for the voltage-frequency pairs of AMD Athlon-64 processors at seven DVFS levels.

The voltage and frequency of processor j which has k number of DVFS levels is calculated by Equation (4.3).

$$(v_j, f_j) = \left\{(v_{low_j}, f_{low_j}) = (v_{1j}, f_{1j}) < (v_{2j}, f_{2j}) < \cdots < (v_{kj}, f_{kj}) = (v_{high_j}, f_{high_j})\right\}. \tag{4.3}$$

The maximum power consumption of processor $P_{proc.highest}$ occurs when it operates at maximum voltage $v_{highest}$ and frequency $f_{highest}$. Therefore, it can be concluded that the active power consumption for a processor under voltage and frequency set (v_j, f_j) is calculated through Equation (4.4).

$$\begin{aligned} P_{procj} &= P_{proc.highest} \times \frac{v_j^2 \times f_j}{v_{highest}^2 \times f_{highest}} \\ P_{proc.highest} &= ACv_{highest}^2 f_{highest} \end{aligned} \tag{4.4}$$

Given the scheduling n tasks on DVFS-enabled processors, the total energy consumption can be calculated through Equation (4.5).

$$\begin{aligned} P_{procs.active} &= \sum_{i=1}^{n} P_{proc.highest} \left(\sum_{j=1}^{k} \frac{v_j^2 \times f_j}{v_{highest}^2 \times f_{highest}} \right) \\ E_{procs.active} &= \sum_{i=1}^{n} P_{proc.highest} \left(\sum_{j=1}^{k} \frac{v_j^2 \times f_j}{v_{highest}^2 \times f_{highest}} \times et\big(t_i, p_m(v_j, f_j)\big) \right) \end{aligned} \tag{4.5}$$

4.2.5. Multi-objective scheduling model

This paper mainly focuses on the four objectives of reducing total energy consumed, reducing the time required to execute tasks, reducing the cost, and increasing the load balance. In fact, the proposed method considers the efficiency of processor utilization. The objectives of the proposed model have been shown below.

4.2.5.1. Total energy consumption. Energy consumption should be minimized in cloud data centers so as to save energy. Since one of the main objectives of this paper is to reduce the energy consumed by processor during the execution of tasks, this has been formulated through Equation (4.6).

$$F_1(X) = \min\{E_{procs.active}\} = \min\left\{\sum_{i=1}^{n} P_{proc.highest}\left(\sum_{j=1}^{k}\frac{v_j^2 \times f_j}{v_{highest}^2 \times f_{highest}} \times et\big(t_i, p_m(v_j, f_j)\big)\right)\right\}. \tag{4.6}$$

4.2.5.2. Makespan of tasks. Referring to the longest time to complete tasks on cloud data center processors, makespan is a parameter involved in scheduling as an objective function. Suppose T_i represents the size of ith task in terms of MI and processing speed of each processor may turn low or high (SP_j is the processing speed of processor j in terms of MIPS). Therefore, the execution time of task i on processor j is obtained by Equation (4.7).

$$t_{exe}(t_i, Proc_j) = \frac{T_i}{SP_j(v_{kj}, f_{kj})} \tag{4.7}$$

Suppose 10 tasks are assigned to 5 heterogeneous processors, each operating at a voltage-frequency pair level. The tasks and processors come with sizes and speeds shown in Tables 3 and 4, respectively. For each processor, there will be a time to complete the tasks assigned accordingly. Figure 3 shows the completion time of each processor.

Table 3: Example of tasks and their sizes

Tasks	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Size (MI)	60	36	54	46	62	34	32	58	40	24

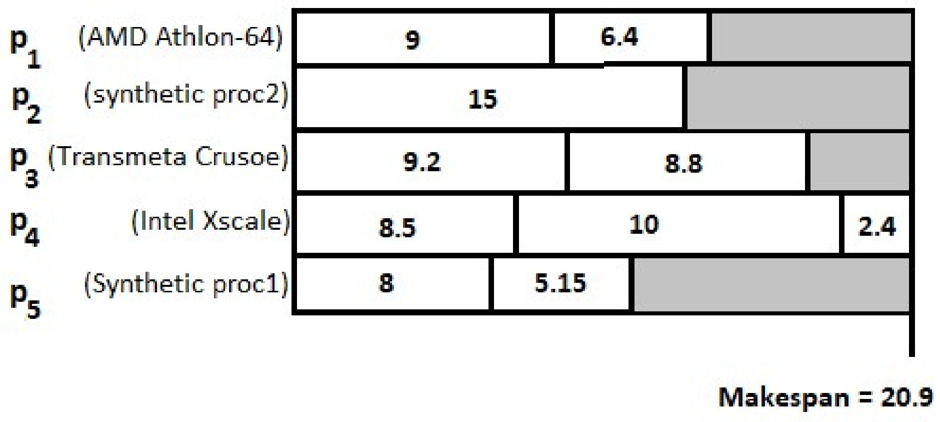


Figure 3: Example of task completion times on processors

Table 4: Example of processors and their properties [?]

Name of processor	AMD Athlon-64 (No.1)	AMD Turion MT-64 (No.2)	AMD opteron 2218 (No.3)	Intel core i3-540 (No.4)	Synthetic (No.5)
Voltage (V)	0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5	0.9, 0.95, 1, 1.05, 1.25	1.2, 1.225, 1.35, 1.5, 1.6	0.75, 1, 1.3, 1.6, 1.8	0.9, 1, 1.05, 1.1, 1.15, 1.2
Frequency (GHz)	0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2	0.3, 0.4, 0.5, 0.9, 1	0.3, 0.4, 0.533, 0.6, 0.667	0.15, 0.4, 0.6, 0.8, 1	0.5, 0.6, 0.7, 0.8, 0.9, 1
Power (w)	2.03, 3.85, 4.84, 5.95, 6.35, 7.2, 8.4	0.97, 1.64, 2.05, 3.29, 5	1.3, 1.9, 3.5, 4.2, 5.3	0.08, 0.17, 0.4, 0.9, 1.6	2.03, 3, 3.85, 4.84, 5.95, 7.2
Speed (MIPS)	4000, 5000, 6000, 7000, 8000, 9000, 10000	3000, 4000, 5000, 6000, 7000	4000, 5000, 6000, 7000, 8000	4000, 5000, 6000, 8000, 10000	2000, 3000, 4000, 5000, 6000, 7000

The execution time of each task on the corresponding dedicated processor will be obtained through Equation (4.7) according to Figure 3 below:

$$\begin{array}{lll} t_{exe}(1,2,1) = 60/4 = 15, & t_{exe}(2,5,5) = 36/7 = 5.15, & t_{exe}(3,1,2) = 54/6 = 9, \\ t_{exe}(4,3,1) = 46/5 = 9.2, & t_{exe}(5,3,3) = 62/7 = 8.8, & t_{exe}(6,4,0) = 34/4 = 8.5, \\ t_{exe}(7,5,2) = 32/4 = 8, & t_{exe}(8,1,5) = 58/9 = 6.4, & t_{exe}(9,4,0) = 40/4 = 10, \\ t_{exe}(10,4,4) = 24/10 = 2.4 & & \end{array}$$

Before calculating makespan, we need to specify the time required to complete tasks on each processor so that the longest ones can be considered as makespan. Hence, we obtain the time to complete tasks on each processor using Equation (4.8):

$$t_{complete}(j) = \frac{\sum_{k\in A_j} T_k}{SP_j(v_{kj}, f_{kj})}, \qquad 1 \le j \le m \tag{4.8}$$

Where, $A(j)$ is a set of task indices assigned to processor j. Now we can calculate makespan:

$$F_2(X) = Makespan = Min\left\{Max\left\{t_{complete}(j)\right\}\right\} \qquad 1 \le j \le m \tag{4.9}$$

In Figure 7, for example, T_6, T_9 and T_{10} tasks are assigned to processor P_4. Using Equation (4.9), the time taken to complete tasks on the processor equals:

$$\begin{aligned} t_{complete}(1) &= 9 + 6.4 = 15.4 \\ t_{complete}(2) &= 15 \\ t_{complete}(3) &= 9.2 + 8.8 = 18 \\ t_{complete}(4) &= 8.5 + 10 + 2.4 = 20.9 \\ t_{complete}(5) &= 8 + 5.15 = 13.15 \end{aligned}$$

According to Equation (4.18), we obtain the makespan, the value of which is 20.9 in the previous example.

$$Makespan = Max\left\{15.4, 15, 18, 20.9, 13.15\right\} = 20.9$$

One of the optimization objectives is to minimize Equation (4.9), i.e. to complete the tasks assigned to the processors within the shortest possible time.

4.2.5.3. Price minimization . The third objective function is the total cost of executing the tasks that must be minimized. Since the cloud-based payment service model is pay-as-you-go, Suppose Pr_j is the unit price of processor j per second of usage. Therefore, the cost of executing t_i on the P_j is obtained through Equation (4.10), while the total cost for scheduling, which represents the cost of executing a chromosome in the population, is obtained through Equation (4.11).

$$Price(j) = t_{complete}(j) \times Pr_j \tag{4.10}$$

$$F_3(X) = Total\ Cost = \sum_{j=1}^{m} Price(j) \tag{4.11}$$

4.2.5.4. Maximization of load balance. Load balancing refers to a fair distribution of loads on data center processors, maximizing efficiency and minimizing makespan. To achieve this, the distribution of load on all processors must be equal, requiring that the load difference between a processor with the heaviest and lightest loads should be minimized. In order to calculate the load balance, the expected productivity of each processor is first calculated by dividing the time taken by each processor to complete the task in accordance with Equation (4.12).

$$Proc_u(j) = \frac{t_{complete}(j)}{Makespan} \qquad 1 \leq j \leq m. \tag{4.12}$$

The mean high processor efficiency does not always imply an optimal load balance [24]. Hence, we need to calculate the mean processor efficiency according to Equation (4.13).

$$\overline{P} = \frac{(\sum_{j=1}^{m} Proc_u(j))}{m}. \tag{4.13}$$

By minimizing the squared deviations from the mean $P_u(j)$, we can improve the load balancing of processors through Equations (4.12) and (4.13) [24]. Hence, the fourth objective function serves to maximize the load balance obtained by minimizing Equation (4.14).

$$F_4(X) = P_{msd} = \sqrt{\frac{\sum_{j=1}^{m}(Proc_u(j) - \overline{P})^2}{m}} \tag{4.14}$$

4.3. Fuzzy Genetic Operators

Based on the simulations, high mutation rate optimizes objectives with larger values more than objectives with smaller values. Furthermore, the low mutation rate focuses more on the optimality of objectives whose values are smaller than the others. Since the newly proposed method has four objectives with different values of magnitude, it is a desired to optimize all objectives.

In this paper, we propose a fuzzy method to determine mutation rates, which include low mutation rates for focusing on objectives with smaller magnitudes and high mutation rates for focusing on larger magnitudes. Furthermore, the mutation rate must be directed not to undermine the quality of the solutions. In addition to a useful and uniformly optimal concentration on all objective functions with very different values, it will lead to rapid converge to the optimal Pareto solutions. We therefore define functions whose values have a significant impact both on the convergence of the algorithm and on the generation of higher quality, varied solutions.

Furthermore, the fuzzy crossover rate based on the variance of the number of processors involved in scheduling can help optimize makespan to direct load balancing toward optimality and save more on power consumption. In fact, when we optimize load balance indirectly, we need to search for and discover new areas that, although not based on the three intended objectives, can provide load balancing solutions that are better than in other points. We carried this out by proposing variance-based functions for fuzzy system inputs and applying its output as crossover rate.

4.3.1. Fuzzy mutation operator

In this paper, we adopt a standard mutation with bit mutations of one solution based on the probability of bit mutation. The probability of mutation in the population, i.e. the number of genes selected for mutation as well as the probability of bit mutation, is obtained from the output of the fuzzy system designed for that purpose.

Two functions are designed to calculate variance. The first function calculates the binary variance between the values of genes on different chromosomes in order to enhance variation and consider all modes of task assignment to processors. This function captures the chromosomes on the Pareto front as input, selects the best one for mean fitness, and then compares the genes of other chromosomes against those of the best chromosome. Meanwhile, it generates the binary variance between the values of chromosome genes as output within interval [0,1]. The second function inputs are also members of the Pareto Front. This function calculates the variance of the mean fitness of chromosomes and ensures variation among the members based on their objective values. This function also produces an output within interval [0,1]. With these two inputs at hand, we design a fuzzy system that, based on fuzzy rules, produces a suitable output to determine the probability of mutation in the number of chromosome genes.

An example of the mutation operator used in this paper has been illustrated in Figure 4. This operator randomly selects a task from the task set (a gene from the set of chromosome genes), and then randomly assigns a new processor and a pair of new active voltage voltages to the selected task.

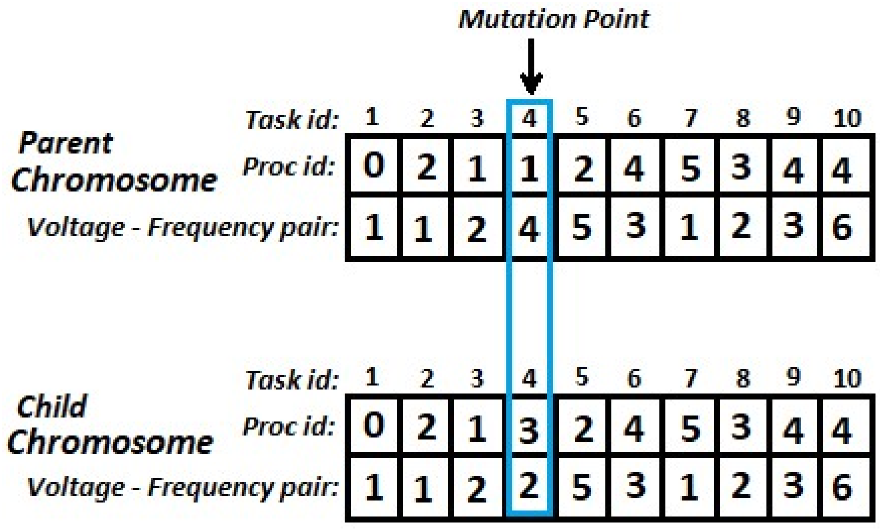

Mutation Point
Parent Chromosome
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 1 2 4 5 3 4 4
Voltage - Frequency pair: 1 1 2 4 5 3 1 2 3 6
Child Chromosome
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 3 2 4 5 3 4 4
Voltage - Frequency pair: 1 1 2 2 5 3 1 2 3 6

Figure 4: An example of mutation operator

4.3.1.1. Binary variance function of gene values. In order to create more variation among members of the population, as chromosomes move toward uniformity, mutation rates can be prevented by increasing mutation rates. In contrast, when chromosome variation is high, large mutation rates may decrease such variation. By reducing the mutation rate, high variation among population members can be retained. To this end, we can control and direct mutation rates by defining a function that represents the variance of gene values. The function designed in the new method receives members of the Pareto front as inputs and gives as outputs the binary variance of the gene values. Therefore, the best member must be selected in terms of mean fitness. Since the magnitudes of optimization objectives are not identical, we first normalize them through Equations (4.15), (4.16), (4.17) and (4.18). Where, PF represents the set of members in the Pareto Front.

$$NormalMakespan(k) = \frac{Makespan(k) - Makespan^{\min}}{Makespan^{\max} - Makespan^{\min}} \quad k \in PF \tag{4.15}$$

$$NormalTotalEnergy(k) = \frac{TotalEnergy(k) - TotalEnergy^{\min}}{TotalEnergy^{\max} - TotalEnergy^{\min}} \quad k \in PF \tag{4.16}$$

$$NormalTotalCost(k) = \frac{TotalCost(k) - TotalCost^{\min}}{TotalCost^{\max} - TotalCost^{\min}} \quad k \in PF \tag{4.17}$$

$$NormalP_{msd}(k) = \frac{P_{msd}(k) - P_{msd}^{\min}}{P_{msd}^{\max} - P_{msd}^{\min}} \quad k \in PF \tag{4.18}$$

Then, we obtain their mean values through Equation (4.19). The chromosomes with the best mean of objective values I AM HERE!!!!I AM HERE!!!!are selected and used to calculate the variance of other samples (Equation (4.20)).

$$AvgObjectives(k) = \frac{NormalMakespan(k)+NormalTotalEnergy(k)+NormalTotalCost(k)+NormalP_{msd}(k)}{4} \; \forall\, k \in PF \tag{4.19}$$

$$BestMember = \min\{AvgObjectives(PF)\} \tag{4.20}$$

Then, for each member of the Pareto front, the number of genes with different values compared with the corresponding gene value in the best member should be calculated. For this purpose, each gene from each chromosome is compared against the genes in the best chromosome. The calculation is done for the rest of the members in the Pareto front accordingly. Figure 5 shows that there are 6 unequal genes compared with the best chromosome.

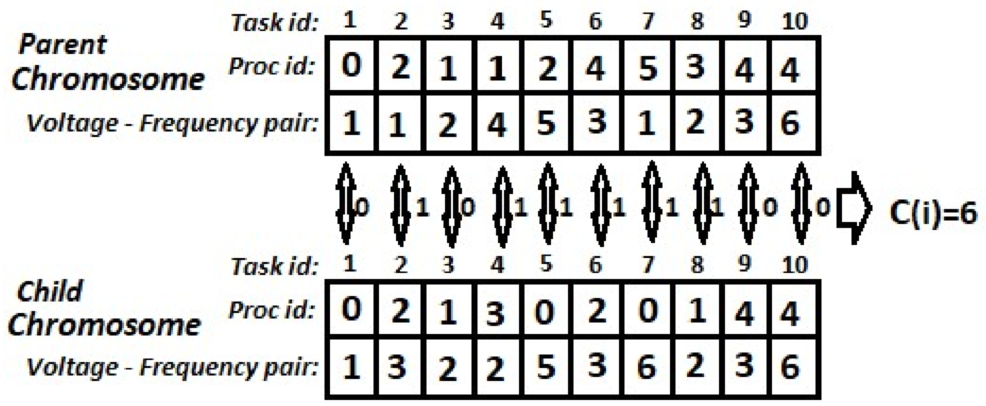

Parent Chromosome
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 1 2 4 5 3 4 4
Voltage - Frequency pair: 1 1 2 4 5 3 1 2 3 6
0 1 0 1 1 1 1 1 0 0
C(i)=6
Child Chromosome
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 3 0 2 0 1 4 4
Voltage - Frequency pair: 1 3 2 2 5 3 6 2 3 6

Figure 5: An example of how binary variance is calculated for subject genes

After the C value is calculated for all chromosomes, according to Equation (4.21), the binary variance of the gene values for each chromosome is obtained, which is between zero and one. The closer this value is to one, the better the mean of objective values for the chromosome. Then, according to Equation (4.22), the mean of these variances is calculated as the input of the fuzzy system.

$$GeneBinVariance(k) = \frac{C(k)}{n} \tag{4.21}$$

$$MutFISinput1 = \frac{\sum_{k \in PF} GeneBinVariance(k)}{P-1} \tag{4.22}$$

Where, n is the number of tasks or genes available in a chromosome, P-1 is the number of chromosomes found in the Pareto front except for the best member, and C(k) is the number of genes in the kth member whose value is different from the corresponding genes in the best member.

The large value for the correct variance indicates a high degree of variation between the chromosome genes belonging to the Pareto front, whereas the small value indicates low variation between the chromosome genes and low variation between the chromosomes themselves. Therefore, there is an inverse correlation between mutation rate and the value of binary. Since these concepts are expressed in fuzzy terms, we employ a fuzzy system to control the mutation rate.

4.3.2. Mean variance function of objective values

This function similarly receives members of the Pareto front as input in each generation and produces the mean variance of objective values as output. In this case, we first normalize the values of the optimization objectives, and then obtain their mean values. There is now a vector where every element is the mean of the corresponding objective values in each member of the Pareto front. In order to calculate the variance, we need to obtain the mean value from those means. The variance of mean values for the objectives of individual members are obtained by Equations (4.23) and (4.24).

$$\mu = \frac{\sum_{k\in PF} AvgObjectives(k)}{P} \tag{4.23}$$

$$\sigma^2 = \frac{\sum_{k\in PF}(AvgObjectives(k) - \mu)^2}{P}. \tag{4.24}$$

This variance indicates the difference between the mean of objective values in different members of the Pareto front, so it has an inverse effect on the mutation rate, because when the mean variance of the objective values is low, the variation in the objective values also low. Hence, the mutation rate must be increased to enhance the variation of objective values.

Similarly, when the variance is high, it indicates good variation in the mean objective values among the Pareto front members. As the mutation rate decreases, the variation is maintained at a desirable level. In order for the output of this function to be used in the fuzzy system decision-making, it must have specific upper and lower bounds. Using Equation (4.25), we normalize this variance before insertion into the fuzzy system.

$$MutFISinput2 = \frac{\sigma^2 - \sigma^2_{\min}}{\sigma^2_{\max} - \sigma^2_{\min}}. \tag{4.25}$$

In this equation, as in (4.15) to (4.18), we used maximum and minimum values for normalization. Therefore, the output value of this function will eventually be between zero and one.

4.3.2.1. Designing a fuzzy system to determine fuzzy mutation rate. After calculating the fuzzy system, two membership functions must be designed for these inputs. Table 5 shows the input and output membership functions for mutation rate. Then, the set of fuzzy rules must be defined for this purpose (Table 6). This fuzzy inference system produces values ranging from zero to 0.5 for the mutation rate.

In order to calculate the output of fuzzy system, the Mamdani method and the center of gravity technique are adopted. The output of the designed fuzzy system yields a fuzzy adaptive mutation rate, which varies according to the features considered from the Pareto front solutions.

Table 5: Input and output membership functions for determining the mutation rate.

Variable	Fuzzy values of membership functions	Number of membership functions	Type of variable
VGV	very low, low, medium, high, very high	5	Input
VAOV	low, medium, high	3	Input
PM	very low, low, medium, high, very high	5	Output

Table 6: Fuzzy rule for the fuzzy mutation rate system

If	GV	is very low	and	VAOV is low	then	PM is very high
If	GV	is very low	and	VAOV is medium	then	PM is high
If	GV	is very low	and	VAOV is high	then	PM is medium
If	GV	is low	and	VAOV is low	then	PM is high
If	GV	is low	and	VAOV is medium	then	PM is medium
If	GV	is low	and	VAOV is high	then	PM is medium
If	GV	is medium	and	VAOV is low	then	PM is high
If	GV	is medium	and	VAOV is medium	then	PM is medium
If	GV	is medium	and	VAOV is high	then	PM is low
If	GV	is high	and	VAOV is low	then	PM is medium
If	GV	is high	and	VAOV is medium	then	PM is low
If	GV	is high	and	VAOV is high	then	PM is very high
If	GV	is very high	and	VAOV is low	then	PM is medium
If	GV	is very high	and	VAOV is medium	then	PM is medium
If	GV	is very high	and	VAOV is high	then	PM is very low

4.3.3. Fuzzy crossover operator

This operator has been designed to reduce computations in four-objective optimization so as to indirectly solve the problem of load balancing in the computing cloud without any optimization computation, and with only optimizing three other objectives.

The chosen parent chromosomes are provided by that crossover operator in order to produce the offspring chromosomes. The chosen parent chromosomes are provided by that crossover operator in order to produce the offspring chromosomes. This similarly belongs to a grouping crossover operator. In this procedure the operator initially matches up all chromosome pairs in the population. On a random basis, the k th ($1 \leq k \leq m$) node is selected by the operator as the crossover point. The crossover operator copies the allocation information of every single task from the first parent chromosome initially. Then, the allocation information of the task related to the latter km nodes in the second parent is updated based on the second parent chromosome. The second offspring is constructed accordingly so that the second parent is responsible for providing allocation information of all tasks in second offspring and the allocation information of the tasks related to the former k nodes in the first parent chromosome is copied from the first parent chromosome. An example of the crossover operator has been illustrated in Figure 6.

The second offspring is constructed accordingly so that the second parent is responsible for providing allocation information of all tasks in second offspring and the allocation information of the tasks related to the former k nodes in the first parent chromosome is copied from the first parent chromosome. An example of the crossover operator has been illustrated in Figure 6.

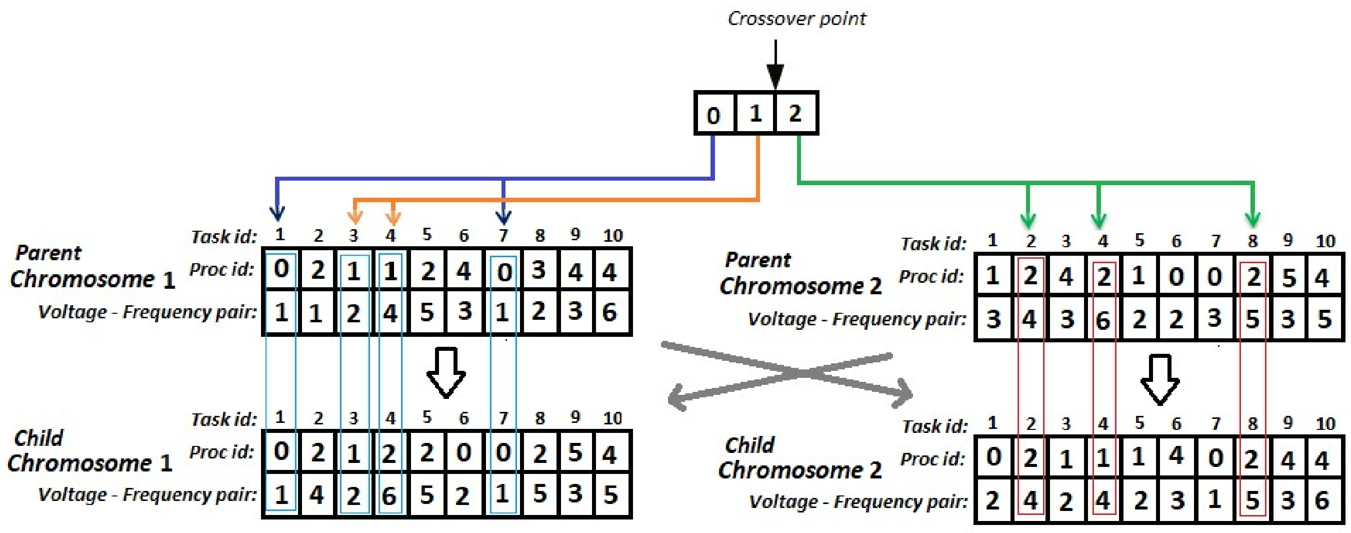

Crossover point
0 1 2
Parent Chromosome 1
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 1 2 4 0 3 4 4
Voltage - Frequency pair: 1 1 2 4 5 3 1 2 3 6
Child Chromosome 1
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 2 2 0 0 2 5 4
Voltage - Frequency pair: 1 4 2 6 5 2 1 5 3 5
Parent Chromosome 2
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 1 2 4 2 1 0 0 2 5 4
Voltage - Frequency pair: 3 4 3 6 2 2 3 5 3 5
Child Chromosome 2
Task id: 1 2 3 4 5 6 7 8 9 10
Proc id: 0 2 1 1 1 4 0 2 4 4
Voltage - Frequency pair: 2 4 2 4 2 3 1 5 3 6

Figure 6: An example of crossover operator

The fuzzy system of this operator utilizes two effective inputs to create variation and load balancing, and ultimately determine the probability of crossover. Two functions are used to calculate the variance. One is the mean variance function of the objective values and the other is the function of calculating the correct variance of the number of processors involved in scheduling.

4.3.3.1. Correct variance function for the frequency of processors involved in scheduling. The inputs of this function in each generation include Pareto front members. Since each processor is assigned multiple tasks. In simulation of the newly proposed algorithm with four objectives considering load balance indirectly, this function calculates the correct variance of the frequency of processors, designed to distribute tasks to processors in an attempt to reduce makespan and achieve load balance.

The correct variance function of the frequency of processors involved in scheduling is defined through dividing the number of tasks by the number of processors. This calculates the expected mean value of the frequency of each processor in scheduling, which is denoted by . Then, the frequency of each processor is calculated in the current scheduling. Given the data, we calculate the correct variance of the frequency of processors. In this method, the sum of the absolute magnitude of the difference between the data and the mean value is divided by the number of tasks. The output will be a number between zero and one, so that values close to zero represent an almost equitable distribution. Whereas the values close to one indicate extremely unequal distribution of tasks to processors. Therefore, this value should have a direct impact on the crossover rate, so that large level of this variance will increase the crossover rate, thus providing more search in the solution space to discover better solutions.

On the other hand, the new algorithm optimizes the makespan, so the results of the two indirectly satisfy the load balance. In other words, by minimizing the makespan as well as distributing tasks to processors relatively evenly, the distribution of tasks will be such that tasks with smaller volumes are given to processors with lower computing power, and vice versa. Equations (4.26) and (4.27) are employed to calculate the variance, is the frequency of each processor in scheduling and n and m the number of tasks and processors.

$$F(k) = \frac{\sum_{i=1}^{m} |L_i - \alpha|}{n} \qquad \forall\, k \in PF \tag{4.26}$$

$$XoverFISinput1 = \frac{\sum_{k \in PF} F(k)}{P} \tag{4.27}$$

We first calculate the mean of total frequency of processors for each Pareto front member, and then obtain the mean of these values through Equation (4.27) to determine the mean of the frequency of processors on the entire Pareto front. Ranging from zero and one, this value will be used as the first input of the crossover fuzzy system and second input is the mean variance of the objective values to maintain the variability between the solutions (Equation (4.28)).

$$XoverFISinput2 = MutFISinput2 \tag{4.28}$$

4.3.3.2. Designing a fuzzy system for crossover rate. Table 7 shows the two membership functions designed for fuzzy system inputs and the output for determination of crossover rate. Then, a set of rules are defined for the fuzzy crossover rate as shown in Table 8. This fuzzy inference system produces values I AM HERE!!!!I AM HERE!!!!between 0.5 and 1 for the crossover rate.

Table 7: Input and output membership functions for determining the crossover rate

Variable	Fuzzy values of membership functions	Number of membership functions	Type of variable
VFIP	very low, low, medium, high, very high	5	Input
VAOV	low, medium, high	3	Input
pXover	very low, low, medium, high, very high	5	Output

Mamdani method and the center of gravity technique are adopted for output of fuzzy adaptive crossover rate, which varies according to the features considered from the Pareto front solutions. This fuzzy system is designed so that, after numerous iterations, the makespan objective optimization process can help achieve a proper load balance between the processors in all optimal solutions generated through three-objective optimization.

Table 8: Fuzzy rule database for the fuzzy crossover rate system

If	VFIP	is very low	and	VAOV is low	then	pXover is medium
If	VFIP	is very low	and	VAOV is medium	then	pXover is low
If	VFIP	is very low	and	VAOV is high	then	pXover is very low
If	VFIP	is low	and	VAOV is low	then	pXover is medium
If	VFIP	is low	and	VAOV is medium	then	pXover is low
If	VFIP	is low	and	VAOV is high	then	pXover is very low
If	VFIP	is medium	and	VAOV is low	then	pXover is high
If	VFIP	is medium	and	VAOV is medium	then	pXover is medium
If	VFIP	is medium	and	VAOV is high	then	pXover is low
If	VFIP	is high	and	VAOV is low	then	pXover is very high
If	VFIP	is high	and	VAOV is medium	then	pXover is high
If	VFIP	is high	and	VAOV is high	then	pXover is medium
If	VFIP	is very high	and	VAOV is low	then	pXover is very high
If	VFIP	is very high	and	VAOV is medium	then	pXover is high
If	VFIP	is very high	and	VAOV is high	then	pXover is medium

5. Efficiency analysis with simulation

In this section, we simulate and evaluate the efficiency of the proposed green scheduling algorithm in an economic cloud. The proposed method schedules independent tasks on the cloud processors. After optimizing the problem objectives, it will offer several solutions to the user. Each optimal solution has a good load balance but has a specific feature. For example, a solution may have low makespan but high energy consumption and cost, or vice versa. There are also solutions that are ideal in every aspect. In this case, the users decide on their own terms. If the user finds price or energy insignificant, they can choose a solution with the least amount of makespan to get the tasks done as quickly as possible. We installed and used MATLAB on a PC with Intel Core i7-A540UP 2.4 GHz processor with 8 cores and 4 GB of memory. Then, we simulated task scheduling in the economic cloud computing and evaluated the results of the proposed method.

5.1. Simulation of NSGA-II optimization algorithm with fuzzy crossover and mutation operators involving three objectives of energy, makespan and price through indirect load balance optimization

In the simulations, we explored the effect of mutation rate on the Pareto front belonging to large and small objectives as well as the effect of crossover rate on the search into the solution space to satisfy certain objectives indirectly via another objective. In the first case, we adopted a fuzzy mutation rate to resolve one defect where the small mutation rate places the optimal focus on objectives with larger magnitudes while the large mutation rate places the optimal focus on the larger magnitude objectives. In fact, we create a focus on all objectives so that all objectives with different values were optimized. In the second case, we managed to indirectly satisfy the load balancing objective through fuzzy crossover rate together with makespan optimality and apply it to all solutions and user choices. With four-objective optimization, however, only a few of the solutions achieved good load balancing and the user might, regardless of this objective, have chosen one of the solutions with good price, energy consumption and makespan, but undesirable load balancing.

In the proposed green approach, we intend to solve the task scheduling problem on cloud data centers by integrating the fuzzy crossover and mutation rates, direct optimization of makespan, energy and price objectives. We also expect to discover a Pareto front delivering the best solution in terms of each objective against fixed crossover and mutation rates. We employed the functions defined in the previous section to simulate the NSGA-II equipped with fuzzy operators and compare it against a conventional algorithm. In particular, the two functions of correct frequency variance of processors involved in scheduling and the mean variance of objective values are used to generate fuzzy system inputs and produce a fuzzy adaptive crossover rate. Furthermore, two functions of binary variance of gene values and mean variance of objective values are used to generate fuzzy mutation system inputs and produce mutation rates. The settings of simulation parameters have been shown in Tables 9 and 10.

The quality of Pareto optimal solutions obtained from NSGA-II with fuzzy crossover and mutation rates as well as NSGA-II with fixed rate have been shown in Tables 11 to 18. In addition, we analyzed the proposed method and compared it against fixed rates for the problem objectives separately. Evidently, the quality of Pareto optimal solutions obtained by the fuzzy method is almost better than all other fixed rates. The ranges of crossover and mutation probability change in the fuzzy system are [0.5 to 1] and [0.1 to 0.5], respectively. In the fixed-rate systems of [0.5 to 1] and [0.1 to 0.5], the steps are 0.1 and 0.05, respectively. In order to precisely determine the effect of fuzzy genetic operators on the quality of solutions, as shown in Table 11, we implemented the algorithm and compared the results for different but fixed rates over the same intervals, with steps 0.1 and 0.05 for crossover and mutation. The results demonstrated that the quality of Pareto optimal solutions

Table 9: Simulation parameters

Parameter	Vale (fixed) - (variable)
Population size	100
Number of generations	100
Number of tasks	1000
Task size	([7200, 144000])-([0, 2400], [2400, 7200], [7200, 144000]) (MI)
Number of processors	128 – (16, 32, 48, 64, 96, 128)
Type of processors	5
Price range of processors	1-5 (G$PS)
Speed range of AMD Athlon-64	4000-10000 (MIPS)
Speed range of AMD Turion MT-64	3000-7000 (MIPS)
Speed range of AMD optero 2218	4000-8000 (MIPS)
Speed range of Intel core i3-540	4000-10000 (MIPS)
Speed range of Synthetic	2000-7000 (MIPS)
Power consumption range of AMD Athlon-64	2.03, 3.85, 4.84, 5.95, 6.35, 7.2, 8.4(wat)
Power consumption range of AMD Turion MT-64	0.97, 1.64, 2.05, 3.29, 5 (wat)
Power consumption range of AMD optero 2218	1.3, 1.9, 3.5, 4.2, 5.3 (wat)
Power consumption range of Intel core i3-540	0.08, 0.17, 0.4, 0.9, 1.6 (wat)
Power consumption range of Synthetic	2.03, 3, 3.85, 4.84, 5.95, 7.2 (wat)
Number of simulation tests	10
Number of optimization objectives	4
Number of direct optimization objectives	3
Number of indirect optimization objectives	1

Table 10: Genetic operators for comparison of Pareto front

Parameters Algorithms	NSGA-II with fixed rate	NSGA-II with fuzzy operate rate
Probability of crossover	0.5 - 1 (with step 0.1)	pXover Table 7
Probability of mutation	0.15 - 0.4 (with step 0.05)	PM Table 5
Probability of mutation in genes	0.15 - 0.4 (with step 0.05)	PM Table 5

obtained from NSGA-II in fuzzy system method for crossover rate and mutation rate is better than that of the fixed-rate method for all problem objectives have been shown in Table 19. Evidently, the quality of solutions obtained by the fuzzy method is almost better than all other fixed rates.

As the results of simulation suggested, we managed to solve the defect of fixed rates in genetic operators using the proposed method. In the first phase of testing, despite the optimized solutions that were minimal in price, energy, and makespan but did not achieve good load balance, it was difficult for the user to make an optimal choice. The user may choose a solution desirable in terms of price, energy and makespan regardless of the load balancing on the cloud data processors, while the selected solution does not provide the appropriate load balance. By indirectly satisfying the load balance, we intend to indirectly create an optimal 3D Pareto front that can only meet the user's objectives, while all possible choices may have a proper load balance as well. In fact, the user satisfies the load balance no matter what optimal solution is chosen from the Pareto front. Since the Pareto front at this stage was not of good quality due to the fixed rate of mutation, the optimum focus was shifted to a specific objective. At this stage, by employing fuzzy genetic operators, we achieved a three-dimensional optimized Pareto front better than the others and also an appropriate load balance across this Pareto front.

Table 11: Best makespan values in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	125	125	125	125	125	125	125
20	112	122	113	121	122	119	123
40	118	121	113	113	121	116	120
60	107	121	113	112	121	116	119
80	107	119	113.5	113	121	115	111
100	106	116	113	113	119	115.5	111

Table 12: Mean of makespan values for Pareto Front members in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	363	355	365	346	354	398	356
20	310	302	332	315	322	365	311
40	265	256	275	292	307	341	271
60	227	218	229	271	288	312	245
80	212	209	221	258	271	301	232
100	209	206	217	229	252	275	221

Table 13: Best price values in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	48562	48750	48855	48530	48785	48980	48590
20	31875	46756	39785	43564	45643	43575	41252
40	28657	44560	38223	38523	41255	36450	37560
60	27510	41565	36451	33565	37258	34520	35895
80	27012	37525	35675	34565	35565	32525	35678
100	26565	35452	33860	33878	32592	32138	32834

Table 14: Mean of price values for Pareto front members in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	123565	123875	123750	123559	123255	123855	123455
20	101525	112569	107565	105567	104565	102455	102457
40	91454	102850	100455	98455	92258	92358	92255
60	85250	96557	94455	94565	86675	87253	88515
80	76510	92478	89175	92195	79653	83457	84251
100	73560	92585	88575	83254	80347	83289	82595

Table 15: Best energy consumption values in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	98016	97564	97875	96578	97489	97835	96985
20	86016	93696	86921	92928	93696	91392	94464
40	90624	92928	86884	86784	92764	89088	92160
60	83187	92928	86784	86016	92345	89712	91392
80	82176	91392	86951	86893	92142	88320	86278
100	81408	89088	86784	85182	91392	84572	85248

Table 16: Mean of energy consumption values in testing fuzzy genetic rates

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	162624	159040	159488	178304	158592	155008	159488
20	138880	135296	139328	163520	144256	141120	148736
40	118720	114688	121408	152768	137536	130816	123200
60	101696	97664	109760	139776	129024	121408	102592
80	94976	93632	103936	134848	121408	115584	99008
100	93632	92288	99008	123200	112896	102592	97216

Table 17: Best squared deviations from the mean for resource efficiency

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	0.031	0.0325	0.0335	0.0351	0.0312	0.0325	0.0351
20	0.0291	0.0312	0.0315	0.0301	0.0289	0.0315	0.0289
40	0.0215	0.03	0.0265	0.0281	0.0275	0.031	0.0265
60	0.0196	0.0281	0.0255	0.0261	0.025	0.0256	0.0245
80	0.0193	0.0251	0.025	0.0245	0.0245	0.0245	0.0238
100	0.0191	0.022	0.023	0.0228	0.0233	0.0228	0.0237

Table 18: Mean of squared deviations from the mean for resource efficiency

Mutation/Crossover Rate Number of iterations	Fuzzy (Proposed method)	0.15/0.5	0.2/0.6	0.25/0.7	0.3/0.8	0.35/0.9	0.4/1
0	0.0382	0.0341	0.0367	0.0357	0.0315	0.0336	0.0367
20	0.0351	0.0325	0.0361	0.0309	0.0385	0.0328	0.0332
40	0.0265	0.0315	0.0345	0.0387	0.0376	0.0316	0.0315
60	0.0236	0.0296	0.0325	0.0271	0.0261	0.0257	0.03295
80	0.0215	0.0275	0.0321	0.0265	0.0254	0.0257	0.0288
100	0.0196	0.028	0.032	0.0257	0.0239	0.027	0.026

Table 19: Pareto optimal solutions for different choices

Best Objective	Best makespan				Best price				Best energy				Best load balance
Objectives / Rate Operators	Make-span	Price	Energy	Squared deviation from the mean for efficiency	Make-span	Price	Energy	Squared deviation from the mean for efficiency	Make-span	Price	Energy	Squared deviation from the mean for efficiency	Squared deviation from the mean for efficiency
Fuzzy operator rate	106	35564	181733	0.0192	309	26565	190733	0.0196	312	49517	81408	0.0197	0.0191
0.15 & 0.5	116	35680	183733	0.024	316	35452	194745	0.028	321	55316	89088	0.029	0.022
0.2 & 0.6	113	40194	192732	0.026	317	33860	192745	0.03	309	53650	86784	0.032	0.023
0.25 & 0.7	113	42542	194787	0.0231	329	33878	192345	0.025	325	53654	85182	0.0245	0.0228
0.3 & 0.8	119	38214	185737	0.0249	352	32592	191145	0.0238	328	52932	91392	0.027	0.0233
0.35 & 0.9	115.5	36781	184984	0.03	375	32138	191876	0.027	367	52237	84572	0.0265	0.0228
0.4 & 1	111	39234	190722	0.029	321	32834	191854	0.026	318	52345	85248	0.028	0.0237

5.2. Simulation of the new fuzzy NSGA-II algorithm with different parameters

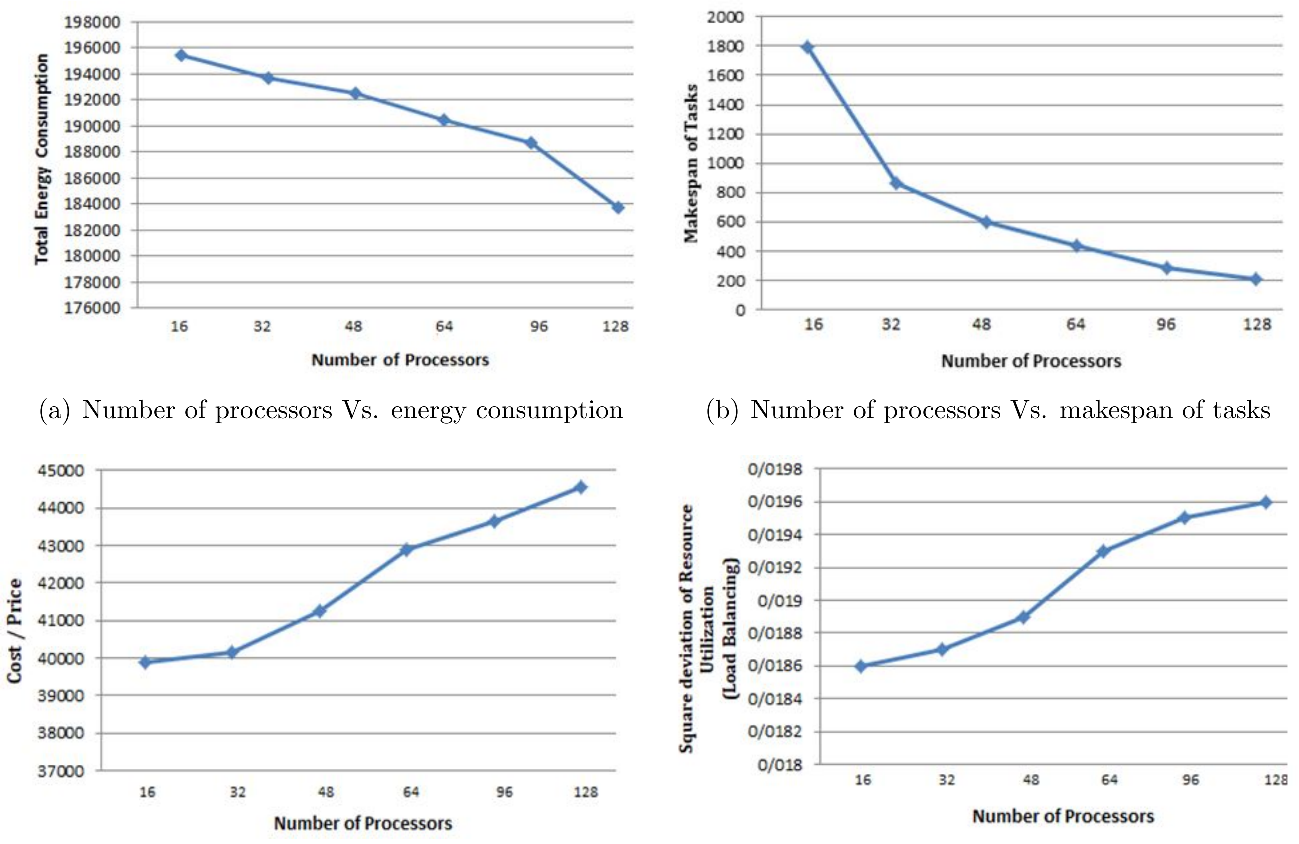
This section investigates the measurement criteria on the proposed method. The values in the following tables are the mean values of non-dominant solution objectives in the simulations.

5.2.1. Effect of the number of processors on simulation objectives

The effect of increasing the number of processors from 16 to 128 on objectives are investigated in Table 20 and Figure 7. The calculated results in Table 20 and Figure 7 confirm that as the number of processors increases, the total energy consumption and the price/cost increases, while the makespan of tasks decrease.

Table 20: The test results for the effect of the number of processors on the objective functions

Number of processors	F_1	F_2	F_3	F_4
16	195440	1793	39875	0.0186
32	193720	867	40150	0.0187
48	192520	598	41245	0.0189
64	190480	439	42895	0.0193
96	188720	289	43652	0.0195
128	183760	209	44560	0.0196



(a) Number of processors Vs. energy consumption (b) Number of processors Vs. makespan of tasks

(c) Number of processors Vs. cost (d) Number of processors Vs. load balancing

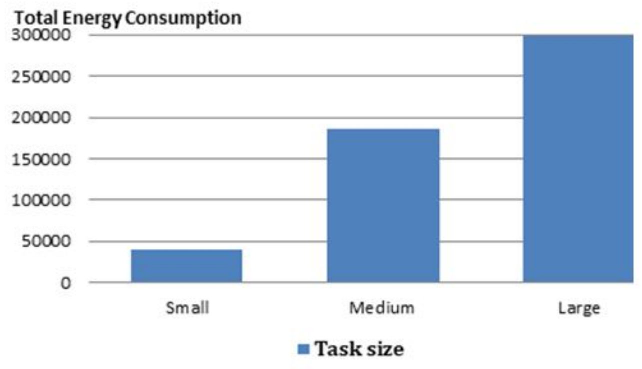
Figure 7: Effect of the number of processors on simulation objectives (energy consumption, makespan, cost, load balance)

5.2.2. Effect of the number of tasks on simulation objectives

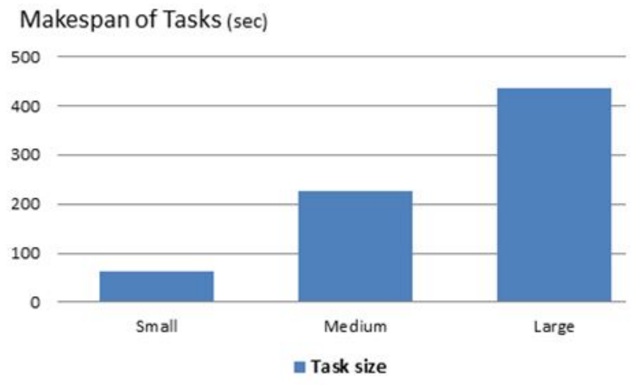
Table 21 categorizes the task size into three groups namely; small, medium, and large. Figure 8 shows how application of the proposed method improves the measurement criteria so that makespan of tasks and cost/price increase when the size of tasks becomes larger than the total energy consumption. In contrast, the load balancing rate of tasks for small and large tasks decrease compared to medium tasks.

Table 21: Testing results for the effect of task size on the mean of objective function values

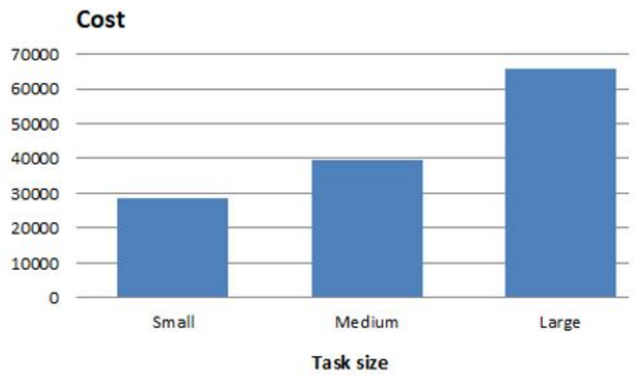
Task size	F_1	F_2	F_3	F_4
Small	39851	62	23521	0.0192
Medium	186752	227	87168	0.0194
Large	298680	437	151781	0.0193



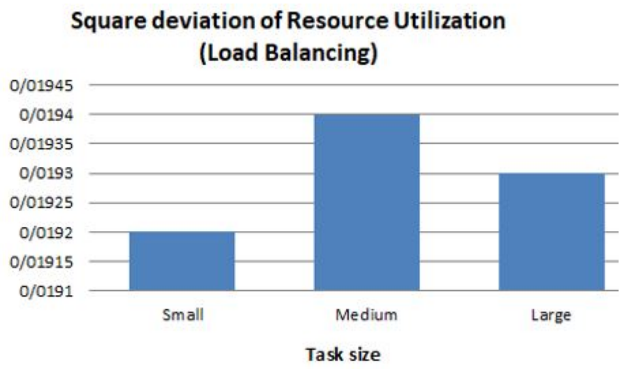
(a) total energy consumption



(b) makespan of tasks



(c) price/cost



(d) load balancing

Figure 8: Effect of task size on simulation objectives (energy consumption, makespan, cost, load balance)

6. Conclusions and future work

This paper proposed a multi-objective optimization method to solve task scheduling on a market-based green cloud with DVFS-enabled processors. Thus, fuzzy NSGA-II is proposed with the aim of both maximizing and minimizing conflicting objectives. Extensive experimental results have clearly demonstrated the superiority of proposed fuzzy NSGA-II than NSGA-II with fixed rates, and three considered scheduling strategies – GOSS, TOSS and COSS. Moreover, it was shown how increasing the number of processors affects objectives. For future works we intend to handle the task scheduling on a green cloud data center system partially powered by the renewable energy.

References

[1] B. Barzegar, H. Motameni and, A. Movaghar, *EATSDCD: A green energy-aware scheduling algorithm for parallel task-based application using clustering, duplication and DVFS technique in cloud datacenters*, J. Intel. Fuzzy Syst. 36(6) (2019) 5135–5152.

[2] L. Wang, K. Su, D. Chen, J. Kolodziej, R. Ranjan, C.Z. Xu and A. Zomaya, *Energy-aware parallel task scheduling in a cluster*, Future Gener. Comput. Syst. 29(7) (2013) 1661—1670.

[3] H. Lei, R. Wang, T. Zhang, Y. Liu and Y. Zha, *A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center*, Comput. Oper. Res. 75 (2016) 103–117.

[4] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu and R. Buyya, *Energy-aware virtual machine allocation for cloudwith resource reservation*, J. Syst. Softw. 147 (2019) 147-161.

[5] S. Mustafa, B. Nazir, A. Hayat and S.A. Madani, *Resource management in cloud computing: taxonomy, prospects, and challenges*, Comput. Electr. Eng. 47 (2015) 186—203.

[6] B. Barzegar, A.M. Rahmani and K. Zamanifar, *Gravitational emulation local search algorithm for advanced reservation and scheduling in grid computing systems*, 2009 Fourth Int. Conf. Comput. Sci. Conver.gence Inf. Tech. (2009) p.1240–1245.

[7] G. Subashini and M.C. Bhuvaneswari, *NSGA - II with controlled elitism for scheduling tTasks in heterogeneous computing systems*, Int. J. Open Prob.lems Compt. Math. 4(1) (2011) 1998–6262.

[8] G. subashini, and M.C. Bhuvaneswari, *Non dominated particle swarm optimization for scheduling independent tasks on heterogeneous distributed environments*, Int. J. Advance. Soft Comput. Appl. 3(1) (2011).

[9] H. PENG and Q. LI, *One kind of improved load balancing algorithm in grid computing*, Int. Conf. Network Comput. Inf. Secur. 2011.

[10] Y. Li, Y. Yang, M. Ma and L. Zhou, *A hybrid load balancing strategy of sequential tasks for grid computingenvironments*, Future Gener. Comput. Syst. 25 (2009) 819–828.

[11] X. Jin, F. Zhang, L. Fan, Y. Song and Z. Liu, *Scheduling for energy minimization on restricted parallel processors*, J. Parallel Distrib. Comput. 81 (2015) 36—46.

[12] W. Pitek, A. Oleksiak and G. Da Costa, *Energy and thermal models for simulation of workload and resource management in computing systems*, Simul. Model.Pract. Theory 58 (2015) 40–54.

[13] Y. Ding, X. Qin, L. Liu and T. Wang, *Energy efficient scheduling of virtual machines in cloud with deadline constraint*, Future Gener. Comput. Syst. 50 (2015) 62-–74.

[14] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A.A. Alelaiwi and F. Li, *Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms*, Future Gener. Comput. Syst. 86 (2018) 836–850.

[15] H. Lei, T. Zhang, Y. Liu, Y. Zha and X. Zhu, *SGEESS: smart green energy-efficient scheduling strategy with dynamic electricity price for data center*, J. Syst. Softw. 108 (2015) 23—38.

[16] A. Sathya Sofia, P. GaneshKumar, *Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II*, J. Netw. Syst. Manag. 26 (2018) 463—485.

[17] C.M. Wu, R.S. Chang and H.Y. Chan, *A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters*, Future Gener. Comput. Syst. 37 (2014) 141—147.

[18] Y. Hu, C. Liu, K. Li, X. Chen, K. Li, Slack allocation algorithm for energy minimization in cluster systems, Future Gener. Comput. Syst. 74 (2017) 119-–131.

[19] M. Hosseini Shirvani, A.M. Rahmani and A. Sahafi, *A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges*, J. King Saud University-Computer Inf. Sci. 32(3) (2020) 267–286.

[20] M. Hosseini Shirvani, *A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems*, Engin. Appl. Artif. Intel. 90 (2020) 1–20.

[21] A. Sathya Sofia and P. GaneshKumar, *Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II*, J. Netw. Syst. Manag. 26 (2018) 463–0485.

[22] Z. Peng, B. Barzegar, M. Yarahmadi, H. Motameni, P. Pirouzmand, *Energy-Aware Scheduling of Workflow Using a Heuristic Method on Green Cloud*, Scientific Programming, 2020.

[23] H. Kumar and S.P. Yadav, *NSGA-II based fuzzy multi-objective reliability analysis*, Int. J. Syst. Assur. Eng. Manag. 8 (2017) 817-–825.

[24] R. Salimi, H. Motameni and H. Omranpour, *Task scheduling using NSGA II with fuzzy adaptive operators for computational grids*, J. Par. Distr. Comput. 74(5) (2014) 2333–2350.