Int. J. Nonlinear Anal. Appl. 13 (2022) 1, 3707-3721
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.6148

Improving completion time and execution time using FSMPIA: A case study

Mehran Mokhtari[a,*], Peyman Bayat[b]

[a]College of Skills and Entrepreneurship, Qaemshar Branch, Islamic Azad University, Qaemshar, Iran
[b]Department of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Iran

(Communicated by Javad Vahidi)

Abstract

Cloud computing (CC) has facilitated the use, access, and storage of resources via sharing them with customers of different organizations. To shorten the completion time and execution time, we introduced the fuzzy k-means (FKM) clustering method, which is based on the new fuzzy entropy. This method was combined with Greedy_SMPIA, Max-Min_SMPIA, Min-Min_SMPIA, GA_SMPIA and PSO_SMPIA to make virtual machines (VMs) smarter. The FKM clustering method was implemented for both Non-SMPIA and SMPIA. The simulation results in MATLAB showed that the improvement of completion time of tasks with GA_SMPIA was up to 88.43% more than other studied methods. Execution time was improved also further improved to 55.37% compared to the other methods studied. The fuzzy smart MPI approach (FSMPIA) performs better than Non-FSMPIA. Also, a comparison of both methods shows that the FSMPIA performance is 32.49% and 11.26% higher than that of the SMPIA in terms of competition time and resource utilization (RU), respectively.

Keywords: Cloud computing, completion time, execution time, resource utilization, SMPIA, fuzzy k-mean

1. Introduction

Task scheduling and resource allocation (TSRA) are key axes in cloud management [1, 2]. Cloud computing (CC) provides the users with computing resources such as software and hardware as a network service. Given the scale of modern data centers (DCs) and their dynamic resources, efficient scheduling techniques are needed to manage these resources [1, 2, 3, 4]. The smart message passing interface approach (SMPIA) can be used as a new idea and scheduling technique in cloud computing

*Corresponding author
Email addresses: mehrmokhtari@yahoo.com (Mehran Mokhtari), drbayat.iau@gmail.com (Peyman Bayat)

Received: June 2021 *Accepted:* October 2021

that has not yet been proposed in the research literature [1, 2]. Using SMPI in the cloud computing structure, the task scheduling process can be performed efficiently. MPI is a key component of high-performance computing (HPC) applications and is essential for improving the performance of HPC applications in the cloud [1, 2, 5, 6]. The performance of communication networks is a challenge and MPI communication methods with low latency are required for it [1, 2, 7]. Based on the findings [8], there are several future guidelines for consideration, which in case of lack of sufficient resources to fully review a request, it is not possible to do make. Also, they inteded to add performance evaluation of loads based on geographical distance to the data center. In addition, they scheduled different virtual networks (VNs) to review different resource allocation policies and to compare service performance for different client groups. Foundations of machine learning (FOML) is compared with the Min-Min, Max-Min, sufferage and enhancement heterogeneous earliest finish time (HEFT) algorithms [9]. The simulation results show that the Max-Min scheduling improved algorithm (MMSIA) is best with a large numbers of tasks and machines [10].

The main contributions of this paper are as follows:

(1) Its main difference is in the number of samples employed, i.e. $|U|$. Dividing the membership degrees by the number of samples cannot be very effective because the states are unclear and the number of states are determined using the fuzzy method. Where instead of dividing by $|U|$, it is divided by the sum of the membership degrees of all the values of the fitness function in all the samples.

(2) Combining smart message passing interface approach (SMPIA) [1] with extended of fuzzy k-means (FKM) based on fuzzy entropy. With the fuzzy smart MPI (FSMPIA), performance and maximum service execution time have been improved in this work. The motivation for this research is the implementation of workflow on the telecommunications transaction application in order to achieve proper processing time and manage cloud system. It also aims to improve the performance of algorithms, increase the quality of service (QoS) in MPI communications of the desired VNs and improved being premature convergence.

The need for this research is to delay MPI communication in the virtual MPI bus (VMPIB) [1]. Using FSMPIA on a telecommunications transaction application increases its efficiency.

The remainder of this paper is organized as follows: related work is presented in Section 2; FKM method in Section 3; FSMPIA performance evaluation in Section 4; Simulation results with FSMPIA in Section 5; Comparison analysis of FSMPIA and SMPIA in Section 6; conclusions and future work in Section 7.

2. Related Work

In [11], the results showed that the MMSIA has the shortest completion time among all virtual machines (VMs) compared to three algorithms such as Max-Min, Min-Min and round robin (RR). In [12], the proposed approach called optimal process placement found the best placement scheme compared to all mass communications in all message sizes. In [13], the multi-purpose workflow optimization strategy (MOWOS) uses the task division mechanism to divide large tasks into sub-tasks to reduce their scheduling time. The simulation results showed that MOWOS had better execution cost and completion time and also used better resources than the HSLJF [14] algorithms. In [15], a combination of the shortest tasks and RR is one of the most useful and powerful hybrids for solving starvation, in which we will used the performance of SJF in reducing turnover and RR in [1] reducing work expectation, but the quantum value of the task always prevents having a hybrid

optimal. In [19], a genetic algorithm-particle swarm optimization (GA-PSO) hybrid algorithm was proposed for assigning tasks to resources aimed at reducing costs, make span (MS), and load balancing tasks in cloud computing. The results showed that this algorithm decreased the total execution time (TET) of workflow tasks scheduling compared to GA, PSO, hybrid heuristic scheduling genetic algorithm (HSGA), workflow scheduling genetic algorithm (WSGA) algorithms and Min-Min based time and cost-trade-off (MTCT). In [20], the Min-Min and Max-Min algorithms were combined with the GA. Thus, with improved GA, MS has been minimized and utilization of resources has improved in comparison with the GA. In [21], a cuckoo PSO (CPSO) hybrid algorithm was proposed aims reduced MS, cost and deadline violation rate. In [22], the standard PSO easily got trapped into the local optimum solution, which results in improved being premature convergence. In this way, the improved PSO with a reduction in the linearly inertia weight, was able to have a strong public search in its initial repeats and, in subsequent reps, has got a strong local search too. In [23], the comparison of the improved PSO algorithm (in crossover and mutation) with PSO showed that improved PSO was not only converged faster, but also it was executed in a large scale faster than the other two algorithms. In addition, it caused reduced MS and better use of resources. In [24], an online incremental learning approach was provided by monitoring CPU, memory, and I/O resources, via which the workflow execution time was predicted and error rate was 29.89% of advanced methods. With the emergence of workflow as a service (WaaS) in the cloud, it is more challenging to predict workflow scheduling and estimate the runtime of tasks. Processing a large volume of data needs to predict real-time changes for the resource performance.

In [25], the authors present a time and energy-aware two-phase scheduling algorithm called best heuristic scheduling (BHS) for directed acyclic graph (DAG) scheduling on cloud data center processors.

In [26], the authors proposed a two-phase energy-aware load balancing (EALB) scheduling algorithm using the virtual machine migration through the Particle Swarm Optimization (PSO) algorithm to be applicable to dynamic voltage frequency scaling-enabled cloud data centers.

Dynamic voltage and frequency scaling (DVFS) has been proven to be a feasible solution to reduce processor power consumption in cloud data centers [27, 28]. By lowering processor clock frequency and supply voltage during some time slots, for example, idle or communication phases, large reductions in power consumption can be achieved with only modest performance losses.

In [29], the authors proposed an efficient method for ranking cloud services while accounting for uncertain user requirements. For this purpose, a requirement interval is defined to fulfill uncertain user requirements. Since there are a large number of cloud services, the services falling outside the requirement interval are filtered out.

3. FKM Method

To shorten the completion time and execution time, we introduced the k-means clustering method, which is based on the new fuzzy entropy. This method was combined with Greedy_SMPIA, Max-Min_SMPIA, Min-Min_SMPIA, GA_SMPIA and PSO_SMPIA [1] to make VMs smarter. The FKM clustering method was implemented for both Non-SMPIA and SMPIA. First, we implemented the FKM clustering method as follows. In the FKM, the data set of cloud management system configuration is employed, which is presented in Table 1. In total, 200 record are implemented as the training data in the fuzzy decision tree (FDT) method in the simulation. Several data records with the same training data format are used for testing. The training data are employed as the input to the training section in an Excel file called "Dataset.xlsx" for the simulation. In the fuzzy smart

MPI approach (FSMPIA), we first read the database data from an Excel file and, then, cluster the training data by the k-means clustering algorithm.

Table 1: Index attributes

Row	Attribute
1	DC ID
2	Server ID
3	CPU (Number)
4	CPU Freq (HZ)
5	Memory (MB)
6	Bandwidth (Mb/S)
7	VMs Count
8	Capacity

Each attribute contains sampes marked with and stored in the database. Table 2 (metadata) shows the attributes, samples, samples and integers used in the simulation of the proposed methodology. K-means is the most popular and the simplest partitional algorithm used for clustering [16].

Table 2: Metadata (mapping table)

Row	Attributes	Samples	Value
1	DC ID	DC 1	1
		DC 2	2
		DC 3	3
		DC 4	4
		DC 5	5
		DC 6	6
		DC 7	7
		DC 8	8
		DC 9	9
		DC 10	10
		DC 11	11
		DC 12	12
		DC 13	13
2	Server ID	Server 1	1
		Server 2	2
		Server 3	3
		Server 4	4
		Server 5	5
		Server 6	6
		Server 7	7
		Server 8	8
		Server 9	9
		Server 10	10
3	CPU (Number)	CPU 1	1
		CPU 2	2

Continue of table 2			
4	CPU Freq (HZ)	64	1
		128	2
		256	3
		512	4
		1024	5
5	Bandwidth (Mb/S)	512	1
		1024	2
6	VMs Count	Less than 20	1
		More than 20	2
7	Memory	512	1
		1024	2

In this algorithm, the number of clusters (groups) must already be specified. Data clustering is performed in the regions where the server is located. In the FSMPIA, We determining the number of clusters based on the areas, in which a specific number is used. Despite its simplicity, this is a basic method for many clustering methods. There are several forms for this algorithm, but they have iterative routines, and the following are estimated for a fixed number of clusters: (1) Obtaining points as the centers of the clusters; these points are in fact the mean points of each cluster. (2) Every data sample is assigned to the cluster, from the center of which has minimum distance.

In this method, a number of points are randomly selected in proportion to the required clusters. Then, the data are assigned to one of these clusters based on the extended of proximity (similarity). So, new clusters are calculated for them new clusters will be achieved by repeating the same procedure and averaging the data. Again, the data are assigned to new clusters. This process continues until there is no changed in the data. FI is considered as the fitness function in Eqution (3.1). Algorithm 1 (Table 3) is considered as the basic algorithm for the k-means algorithm.

$$FI = \sum_{o=1}^{z} \sum_{h=1}^{p} \|a_h^o - e_o\|^2 \tag{3.1}$$

$\|\|$ is the measure of distance the points, eo is the center of the o^{th} cluster, p is the sample number and z is the number of clusters, ah is a sample instance.

Table 3: The basic algorithm for the k-means

01	Start
02	z=10 centres are randomly selected. // Based on the number of servers
03	Repeat
04	Each sample is assigned to the closest center.
05	The centres are updated according to the clusters.
06	Until (there are not great changes in the clusters or the numbers are determined in advance.)
07	End

After clustering the training data, it is the time to train the program using the FDT. Data clustering is performed in the regions where the server is located. Learning condition in the FDT algorithm is that the data should be widespread. This step is performed by the k-means algorithm. The data are then stored in the matrix and sent to the FDT for training. Tree classification is done in the k-means algorithm; it is a two-step process as follows:

(1) Modeling: The training data set refers to a database containing the cloud configuration data with 10 active servers on the cloud which are stored in a matrix after discretization. The class tags associated with these samples are specified and every class is determined with the attribute in Table 2, which is called "class tag".

(2) Model usage: The class tag of each sample p of the database is predicted via function $y = f(p)$. This function is in the form of classification rules, decision trees or mathematical formulas.

In the FSMPIA, predictable data or testing data are inserted into the program by an Excel file with the same training data format. After being discretized by the k-means algorithm, they turn to classified rules by a function and are compared with the training data for decision making (providing predicted results). After the machine training phase, training data infer a pattern from this machine using the fuzzy algorithm. The testing data provide the predicted results after comparing with this pattern. In the FSMPIA, testing data include some tested attributes (server name, DC name, CPU number, CPU power, memory, bandwidth allocation, VM count). The capacity attribute (as the response attribute) is the result of the FDT adjustment which can be observed in the program output. According to the implementation, we stored the above data in the "Dataset.slsx" file. This file contains 200 records and 8 columns and, in accordance with the training data format, enters the simulation as input data by the "Create_Dictunary.m" file. Contents of the Excel file are stored in a matrix called "Train_Data". Each column of this matrix is inserted in the array to discretize the values for the training data. Next, a "For loop" will rotate as many times as of the number of the rows in this file and the values are discretized. Finally, the data are inserted in a matrix called "Train_Data" to be loaded by the "Main.m" file for learning the program. After executing "Create_Dictunary.m", the "Main.m" file should be executed. This file first loads "Train_Data", then inserts columns 1 to 7 in the "Train_Patterns" array, and the last column (answer column) is placed in the "Train_Targets" array. The response is tested or predicted using the "Test_Data.xlsx" file. Then, the data are inserted in the "Main.m" file in the program as the testing data, which is compared to the previous data (learning data), and the result is displayed as the output. Testing data containing 7 columns are also discretized. The response is unclear in this data, only the 7 columns with which the program learns how to predict the response are considered as inputs to the testing data. The data predict the response by FDT method and display it as the output. The "Test_Fuzzy" function is located in the "Test_Fuzzy.m" file and learns the response prediction based on the training data via the FDT method. An extended FDT is an ordinal decision tree (ODT), which uses fuzzy rules for classification. For this reason, in addition to the training data, it requires a precise definition of fuzzy sets for each attribute. But, its overall performance and what it does are similar to what an ODT does. One of the works that is done in a FDT, such as an ODT, is entropy calculation and information efficiency. However, their calculation formula is different and fuzzy. The entropy calculation is normally expressed in Equation (3.2).

$$Entropy(U) = -\sum_{h=1}^{r} p_h \log_2 p_h \tag{3.2}$$

Where U is the set of samples, r is the number of values that the fitness function can accept and p_h is a fraction of samples, in which the fitness function has a value of h. What is needed in this algorithm is the fuzzy entropy calculation. Equation (3.3) is employed to represent fuzzy entropy:

$$Entropyf(U) = -\sum_{h=1}^{r}\sum_{o=1}^{p}(\mu_{ho}/|U|)\log_2\sum_{o=1}^{p}(\mu_{ho}/|U|) \tag{3.3}$$

In Equation (3.3), $|U|$ is the number of members in the set U, p is the number of samples and r is the tag of fitness function classes. Equation (3.3) is an extended form of Equation (3.2); but, instead of putting a value of 0 or 1 for membership degree, we put a value between 0 and 1. The equation used to calculate the entropy in our algorithm is slightly different from the one mentioned above. Its main difference is in the number of samples employed, i.e. $|U|$. Dividing the membership degrees by the number of samples cannot be very effective because the states are unclear and the number of states are determined using the fuzzy method. For this reason, it is divided by the sum of membership degrees in all the samples. Accordingly, the extended entropy equation is presented as Equation (3.4).

$$\begin{gathered}(a)U = \{a_1, a_2, \cdots, a_p\} \text{ is a set of samples.}\\ (b)G = \{b_1, b_2, \cdots, b_r\} \text{ is a set of class tags.}\\ (c)\mu_{ho} \in [0,1]\\ (d)\sum_{h=1}^{r}\mu_{ho} = 1\\ (e)0 \le \sum_{o=1}^{p}\mu_{ho} \le p\\ (f)A = \sum_{o=1}^{p}\mu_{ho}/\sum_{z=1}^{p}\sum_{h=1}^{r}\mu_{hz}\\ -\sum_{h=1}^{r}A\log_2^A\end{gathered} \tag{3.4}$$

Where instead of dividing by $|U|$, it is divided by the sum of the membership degrees of all the values of the fitness function in all the samples. After training the data in the "Main.m" file of "Test_Fuzzy", we compared the trained data with the testing data and make predictions. The predicted data are used by the "Mainprogram.m" function. Also, the methodology of the PSO_FSMPIA is shown in Figure 3. According to Figure 3, in the second stage of FSMPIA, the appropriate alternative virtual machines (AVMs) for the next flows was selected according to the calculation of the completion time formula for all available AVMs.

SMPIA simulations were performed with real data from a homogeneous environment. 201535 Record of Iran Telecommunication transactions and tenders has been collected from 2011 to 2017. Simulations were performed in MATLAB. The program runs on a system with the following features: 1.83 GHz CPU, Core i7 4 GB RAM. The simulation parameters and their details are shown in Table 4 and Table 5.

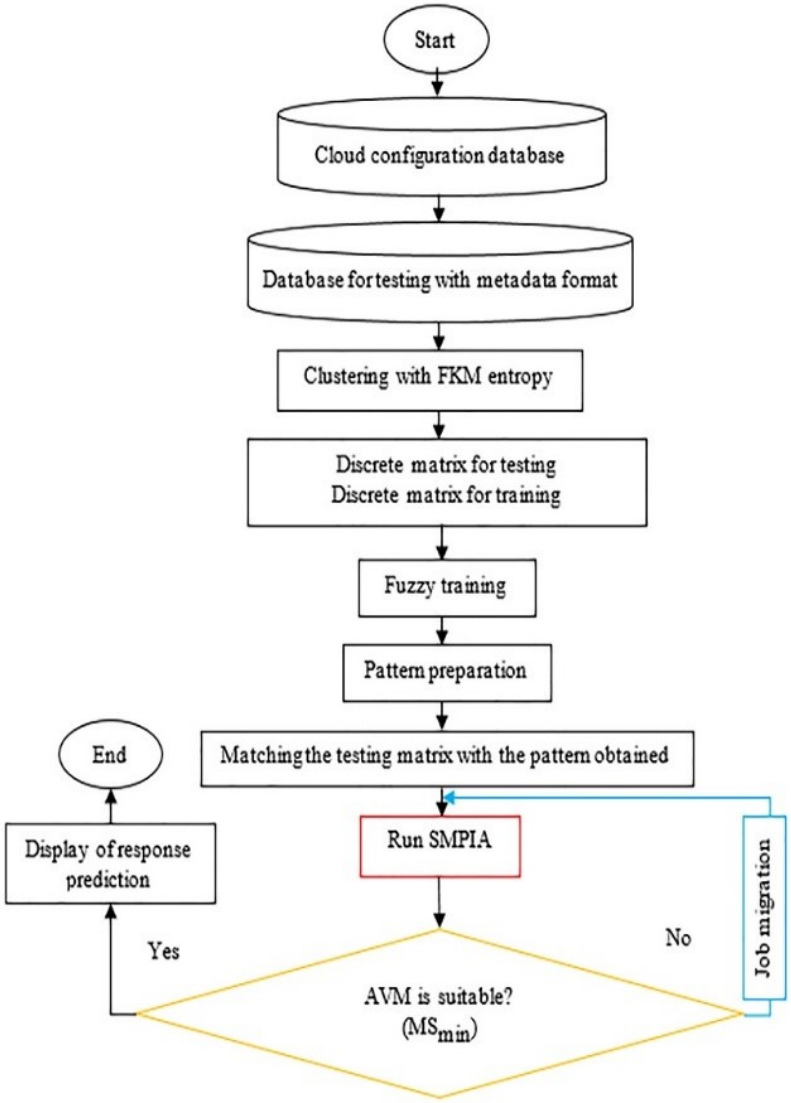

Figure 1: The SMPIA flowchart in FKM based on fuzzy entropy

4. FSMPIA Performance Evaluation

For evaluating the Non-FSMPIA and FSMPIA in the distributed system, Greedy_FSMPIA, Max-Min_FSMPIA, Min-Min_FSMPIA, GA_FSMPIA and PSO_FSMPIA were employed. For calculating the performance parameters, Equations (5.1)-(5.4) [1, 2] were employed for MS, TET, RU, and average RU, respectively. First, the Non-FSMPIA was implemented to each of the algorithms. To evaluate the performance of the FSMPIA, two approaches were executed in parallel using MPICH-3.0.4, and FSMPIA was applied to each algorithm. Any kinds of changes in the state of successor flows in successor VMs (changes in load, capacity, etc.) affected the next flows of the subsequent AVMs. The tests were conducted on the actual data in a homogenous environment including 13 DCs, 10 servers, 132 VMs, 132 flows, and 324 telecommunication equipment. Note that both parameters of the number of records and number of VMs were assumed to be constant. In the following, programs were executed at least 50 times in a system with identical specifications to compute the average of parameters. Ultimately, the mean values were recorded as well. To do so, 201535 records were collected on transactions (including deals and tenders) of the telephone company from 2011 to

Table 4: The simulation parameters [2]

Data Center ID	Server ID	CPU (Number)	CPU Freq (HZ)	Memory (MB)	BandWidth (Mb/S)	VM ID	Flow ID
1	1	4	1000	2000	100	12	12
1	2	2	2000	2000	50	9	9
1	3	1	1000	4000	100	9	9
1	4	4	1200	1000	100	3	3
1	5	2	1000	1000	100	9	9
1	6	1	1000	2000	100	6	6
1	7	4	1200	2000	100	6	6
2	8	2	1800	2000	50	9	9
2	9	2	2000	1000	50	3	3
2	10	2	1800	4000	20	6	6
2	11	2	1000	1000	50	6	6
2	12	1	1000	1000	100	3	3
2	13	1	1000	1000	100	3	3
2	14	1	1200	1000	100	9	9
3	15	4	1800	1000	50	3	3
3	16	4	1800	2000	20	6	6
3	17	1	1200	1000	20	3	3
3	18	1	1000	1000	20	6	6
4	19	2	1000	1000	20	6	6
4	20	1	1200	2000	100	6	6
4	21	1	1800	4000	50	3	3
4	22	2	1200	1000	100	6	6

Table 5: The details of simulation parameters [5]

Variable	Amount
Exec Speeds	63*132
Flow Count Some of the VMs have shared flow	63
Product	324*3 Cell
Server Count	22
DC	4
Transaction	201535*16 Cell
Trans Count	2o1535
VMs	132*4 Duble
VMs Count	132

2017. The simulation and implementation were performed in MATLAB. The experiments were done on a system with the following features: Windows 7, CPU 1.83 GHz, Core i7 4GB RAM. In this research, data sets of telecommunication transactions applications (deals and tenders) in ministry of communication and information technology of iran (MCITI) and workflows are used for simulation and clustering [1, 2].

5. Simulation Results with FSMPIA

5.1. Calculate the Total Execution Time

We have used Equation (5.1) to calculate the TET. As shown in Figure 5.1 and Table 6 and Table 7, the TET decreases at 132 cloud workloas with FSMPIA. The maximum percentage of TET improvement is 55.37% at 132 cloud workloads in FSMPIA performs better than the Non-FSMPIA. Other details are provided in Table 6 and Table 7.

$$TET = \frac{\text{The value of calculation load}}{\text{CPU execution speed}} = \frac{L_{CPU}}{ES_{CPU}} \tag{5.1}$$

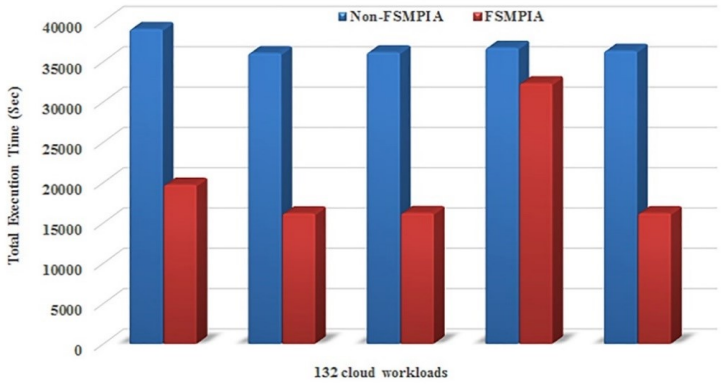

Figure 2: Effect of 132 cloud workloads on total execution time with Non-FSMPIA and FSMPIA

5.2. Calculate the maximum completion time

We have used Equation (5.2) to calculate of completion time. As shown in Figure 5.2 and Table 6 and Table 7, the completion time decreases at 132 cloud workloads with FSMPIA. The maximum recovery rate of fuzzy GA is 88.43% in 132 working times in FSMPIA, so that FSMPIA performs better than Non-FSMPIA. Other details are provided in Table 6 and Table 7.

$$ComplitionTime/MS = \max\left(\sum_{i=1}^{m}\sum_{j=1}^{t}\sum_{k=1}^{f}\left(\frac{(CC)_i}{(ES)_i}\right) \times S_{ijk}\right) \tag{5.2}$$

S_{ijk} is time spent by execution flowk of transactionj on VM_i, otherwise, it is zero. CC_i is the parameter determining the CC of flowk on VM_i. ES_i is the parameter determining the ES of flowk on VM_i.

5.3. Calculate resource utilization

We have used Equation (5.3) and 8 to caculate resource utilization (RU) and average RU, respectively. As shown in Figure 5.3 and Table 6 and Table 7, the RU increases at 132 cloud workloads with FSMPIA. The maximum RU progress rate at 132 cloud workloads in the fuzzy GA algorithm is 3%, but FSMPIA performs better than Non-FSMPIA. Other details are provided in Table 6 and Table 7.

$$RU_i = \frac{CP_i}{TPC_i} + \frac{MC_i - MCU_i}{MC_i} \tag{5.3}$$

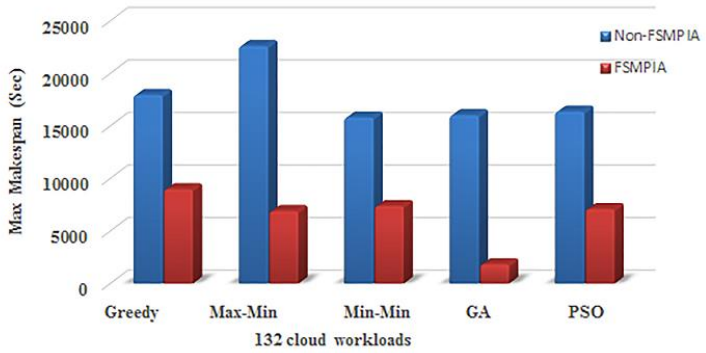

Figure 3: Effect of 132 cloud workloads on completion time with Non-FSMPIA and FSMPIA

$$AveRU = \frac{\sum_{i=1}^{m} RU_i}{m} * 100 \tag{5.4}$$

CP_i is the capacity processed in VM_i, TPC_i refers to the total processing capacity of VM_i, MC_i denotes the memory capacity of VM_i, MCU_i means the memory capacity used in VM_i.

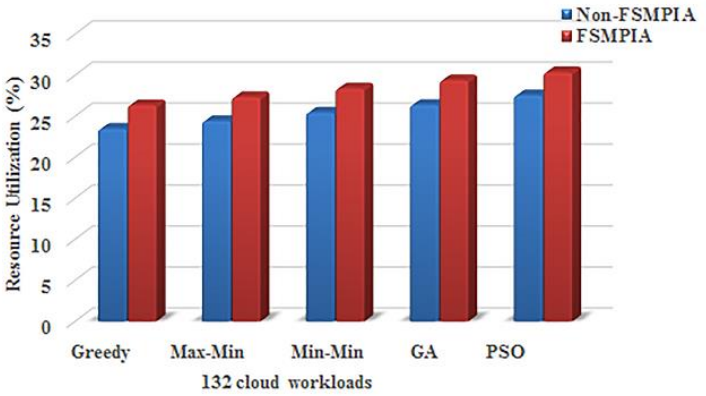

Figure 4: Effect of 132 cloud workloads on resource utilization with Non-FSMPIA and FSMPIA

5.4. The effect of total execution time on resource utilization

In Figure 5.4, at 23.47% involvement level and RU, the TET in Non-FSMPIA was 45.58% greater than in FSMPIA. However, at 30.38 involvement levels and RU, TET in FSMPIA was 45.58% smaller than the Non-FSMPIA. As a result, the performance of the FSMPIA was better than that of the Non-FSMPIA. Other details are provided in Table 6 and Table 7.

5.5. The effect of completion time on resource utilization

In Figure 5.5, at 23.47% involvement level and RU, completion time in Non-FSMPIA was 60.45% greater than the FSMPIA. However, at 30.38% involvement levels aof RU, completion time in FSMPIA was 60.45% smaller than the Non-FSMPIA. As a result, the performance of the FSMPIA was better than that of the Non-FSMPIA. Other details are provided in Tables 6 and 7.

6. Comparison Analysis of FSMPIA and SMPIA

Predictive models should be used to obtain effective forecasting results. One of the prediction techniques is machine learning based on k-means clustering. Execution time, completion time and

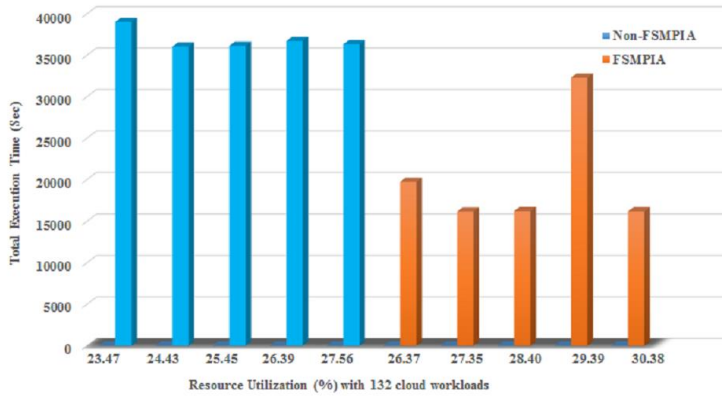

Figure 5: The effect of total execution time on resource utilization with FSMPIA and Non-FSMPIA

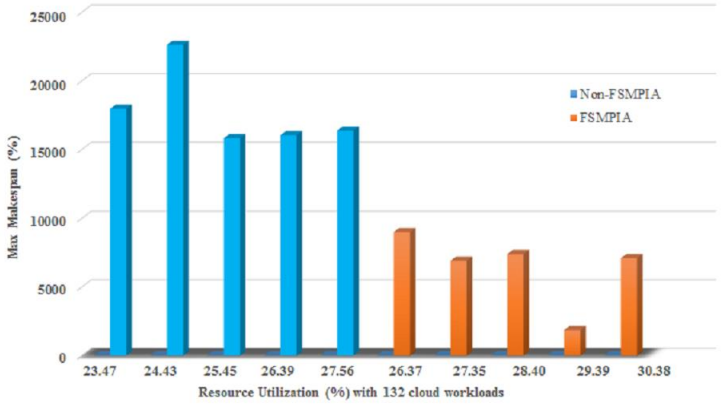

Figure 6: Effect of completion time on resource utilization with FSMPIA and Non-FSMPIA

RU are considered as three criteria of service efficiency and quality assurance [17]. Machine learning techniques are able to accurately predict results. They consider many parameters and learn the behavior of programs from the training of educational data sets [18]. In [18, 17], researchers separately number of requests, number of VMs, number of physical memory, RU, service level agreement (SLA) parameters, future resource demand, number of users, performance forecast, power consumption, execution time, response time, data, costs and data issues are considered.

A comparison of the results of the two methods in Table 6 and Table 7 shows that FSMPIA performs better than SMPIA. The use of SMPIA in both tables improved the performance parameters. By FSMPIA in Table 7, execution time improved by 55.37%, while by SMPIA in Table 6, execution time improved by 55.80%. Using FSMPIA in Table 7, the completion time improved to 88.43%, while with SMPIA in Table 6, the completion time improved to 55.94%.

In addition, RU improved by up to 3% by FSMPIA in Table 7, while RU by up to 14.26% improved by SMPIA in Table 6. To calculate RU using FSMPIA, prediction parameters were used as input through clustering, while to calculate RU using SMPIA, end-time parameters were used as input. Also, the performance parameters in SMPIA were calculated based on the average values obtained after 50 times of program execution. RU parameter shows the amount of RU and the degree of involvement of hardware resources in the cloud in percent. High and low percentages in this study do not indicate high or low quality. In the proposed method, resources became faster. Resource allocation was done with a maximum involvement rate and RU of less than 9%. This study is consistent with the results [17, 18], and prevents SLA violations and QoS decline.

Table 6: Comparison of performance parameters with Non-SMPIA and SMPIA

PSO	GA	Min-Min	Max-Min	Greedy	Algorithms	
143896	153665	143517	143172	155445	TET	Non-SMPIA
17563	20378	18921	17976	19668	MS	
76.79	75.76%	76.87%	75.72%	74.72%	RU	
63897	145544	63430	76008	75523	TET	SMPIA
8587	9000	8884	7919	12027	MS	
91.05	89.73%	88.73%	87.52%	87%	RU	
55.59%	5.28%	55.80	46.91	51.10	TET	Improvement (%)
51.10%	55.83%	53.04	55.94	38.84	MS	
14.26	13.97%	11.86%	11.80%	12.28%	RU	

Table 7: Comparison of performance parameters with Non-FSMPIA and FSMPIA

PSO	GA	Min-Min	Max-Min	Greedy	Algorithms	
36338	36716	36157	36051	39035	TET	Non-FSMPIA
16383	16043	15819	22624	17967	MS	
27.56	26.39%	25.45%	24.43%	23.47%	RU	
16217	32317	16240	16187	19739	TET	FSMPIA
7105	1855	7398	6913	9003	MS	
30.38	29.39%	28.40%	27.35%	26.37%	RU	
55.37	11.98	55.08	55.09	49.43	TET	Improvement (%)
56.63	88.43	53.23	69.44	49.89	MS	
2.82	3%	2.95%	2.92%	2.9%	RU	

7. Conclusion

In this work, a new predictive fuzzy method was combined with SMPIA to improve completion time and execution time.

With the new fuzzy entropy method, the completion time with GA_SMPIA algorithm was further improved than other algorithms. On the other hand, the execution time with PSO_SMPIA algorithm was more improved than other algorithms.

The results show that PSO_SMPIA is better than other methods studied in this study and is more suitable for scheduling and allocating telecommunication resources. FSMPIA performs better than Non-FSMPIA. The extended FKM algorithm was used to cluster the data set. Discrete matrices store the training and testing data. By training the machine via the fuzzy method, a pattern is inferred from this machine. In the fuzzy entropy, the membership degree is divided by the sum of membership degrees of all fitness function values in all the samples. "Test_Fuzzy" function learns the prediction of the response based on the training data using the FDT method. By matching the testing matrix with the obtained pattern, the predicted result is displayed. The FSMPIA was evaluated by simulation in the MATLAB environment. The best mode was PSO_FSMPIA, in which with FSMPIA, TET and MS were improved by 55.37% and 56.63%, respectively, and RU increased from 27.56% to 30.38% (2.82%). Due to the relative reduction of RU in this study, we propose the SMPIA optimization method using the O_EDF-VD algorithm for future work.

References

[1] M. Mokhtari, P. Bayat and H. Motameni, *Multi-objective task scheduling using smart MPI-based cloud resources*, Comput. Inf. 40(1) (2021).

[2] M. Mokhtari, P. Bayat and H. Motameni, *Solving the task starvation and resources problem using optimized SMPIA in cloud*, Comput. Syst. Sci. Engin. 42(2) (2022) 659–675.

[3] R.A. Al-Arasi and A. Saif, *Task scheduling in cloud computing based on metaheuristic techniques: A review paper*, EAI Endorsed Trans. Cloud Syst. 6(17) (2020).

[4] R. Jain and A. Nayyar, *A novel homomorphic RASD framework for secured data access and storage in cloud computing*, Open Comput. Sci. 10(1) (2020) 431–443.

[5] L. Espínola, D. Franco and E. Luque, *Mcm: A new mpi communication management for cloud environments*, Procedia Comput. Sci. 108 (2017) 2303–2307.

[6] H.M. Wei, J. Gao, P. Qing, K. Yu, Y.F. Fang and M.L. Li, *A framework for MPI runtime communication deadlock detection*, J. Comput. Sci. Technol. 35 (2020) 395–411.

[7] H. Hassanpour and M. Mokhtari, *Proposing a Dynamic Routing Approach to Improve Performance of Iran Data Network*, 2009.

[8] M.M.S. Maswood, C. Develder, E. Madeira and D. Medhi, *A stable matching based elephant flow scheduling algorithm in data center networks*, Comput. Networks 120 (2017) 186–197.

[9] B. Liang, X. Dong, Y. Wang and X. Zhang, *A low-power task scheduling algorithm for heterogeneous cloud computing*, J. Supercomput. 2020 (2020) 1–25.

[10] M.Y. Wu, W. Shu and H. Zhang, *Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems*, Proc. 9th Heterog. Computing Workshop (HCW 2000)(Cat. No. PR00556). IEEE, 2000.

[11] T.C. Hung, L.N. Hieu, P.T. Hy and N.X. Phi, *MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing*, Proc. 3rd Int. Conf. Machine Learn. Soft Comput. 2019.

[12] J. Zhang, J. Zhai, W. Chen and W. Zheng, *Wavelet-Based Adaptive Solvers on Multi-Core Architectures for the Simulation of Complex Systems*, Eur. Conf. Parall. Process. Springer, Berlin, Heidelberg, 2009.

[13] J.K. Konjaang and X. Lina, *Multi-objective workflow optimization strategy (MOWOS) for cloud computing*, J. Cloud Comput. 10(1) (2021) 1–19.

[14] P. Brucker, *Scheduling Algorithms*, 5th edition,, Berlin, Springer, 2006.

[15] S. Elmougy, S. Sarhan and M. Joundy, *A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique*, J. Cloud Comput. 6(1) (2017) 1–12.

[16] V. Singh and N.K. Verma, *An entropy-based variable feature weighted fuzzy k-means algorithm for high dimensional data*, arXiv preprint arXiv:1912.11209 (2019).
[17] Z. Xie, X. Shao and Y. Xin, *A scheduling algorithm for cloud computing system based on the driver of dynamic essential path*, PloS one 11(8) (2016) e0159932.
[18] N. Almezeini and A. Hafez, *An Enhanced Workflow Scheduling Algorithm in Cloud Computing*, CLOSER 2 (2016) 67—73.
[19] A. Manasrah ad H. Ba Ali, *Workflow scheduling using hybrid GA-PSO algorithm in cloud computing*, Wireless Commun. Mobile Comput. 2018 (2018).
[20] P. Kumar and A. Verma, *Scheduling using improved genetic algorithm in cloud computing for independent tasks*, Proc. Int. Conf. Adv. Comput. Commun. Inf. ACM, 2012, pp. 137-142.
[21] T.P. Jacob and K. Pradeep, *A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization*, Wireless Person. Commun. 109(1) (2019) 315–331.
[22] F. Luo, Y. Yuan, W. Ding and H. Lu, *An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing*, Proc. 2nd Int. Conf. Comput. Sci. Appl. Engin. ACM, vol. 142, 2018.
[23] L. Guo, S. Zhao, S. Shen and C. Jiang, *Task scheduling optimization in cloud computing based on heuristic algorithm*, J. Networks 7(3) (2012) 547.
[24] M.H. Hilman, M. A. Rodriguez and R. Buyya, *Task runtime prediction in scientific workflows using an online incremental learning approach*, Proc. IEEE/ACM 11th Int. Conf, Util. Cloud Comput., 2018, pp. 93-102.
[25] 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. Intell. Fuzzy Syst. 36(6) (2019) 5135–5152.
[26] J. Masoudi, B. Barzegar and H. Motameni, *Energy-aware virtual machine allocation in DVFS-enabled cloud data centers*, IEEE Access 10 (2021) 3617–3630.
[27] S. Fatehi, H. Motameni, B. Barzegar and M. Golsorkhtabaramiri, *Energy aware multi objective algorithm for task scheduling on DVFS-enabled cloud datacenters using fuzzy NSGA-II*, Int. J. Nonlinear Anal. Appl. 12(2) (2021) 2303–2331.
[28] Z. Peng, B. Barzegar, M. Yarahmadi, H. Motameni and P. Pirouzmand, *Energy-aware scheduling of workflow using a heuristic method on green cloud*, Sci. Programm. 2020 (2020).
[29] M.H. Nejat, H. Motameni, H. Vahdat-Nejad and B. Barzegar, *Efficient cloud service ranking based on uncertain user requirements*, Cluster Comput. 25(1) (2022) 485–502.