Paper

# Ganging of Resources via Fuzzy Manhattan Distance Similarity with Priority Task Scheduling in Cloud Computing

S. Sharon Priya[1], K. M. Mehata[2], and W. Aisha Banu[1]

[1] *Department of Computer Science and Engineering, B. S. Abdur Rahman University, Chennai, Tamil Nadu, India*
[2] *School of Computer, Information and Mathematical Sciences at B. S. Abdur Rahman University, Chennai, Tamil Nadu, India*

https://doi.org/10.26636/jtit.2018.108916

**Abstract—This paper proposes a fuzzy Manhattan distance-based similarity for gang formation of resources (FMDSGR) method with priority task scheduling in cloud computing. The proposed work decides which processor is to execute the current task in order to achieve efficient resource utilization and effective task scheduling. FMDSGR groups the resources into gangs which rely upon the similarity of resource characteristics in order to use the resources effectively. Then, the tasks are scheduled based on the priority in the gang of processors using gang-based priority scheduling (GPS). This reduces mainly the cost of deciding which processor is to execute the current task. Performance has been evaluated in terms of makespan, scheduling length ratio, speedup, efficiency and load balancing. CloudSim simulator is the toolkit used for simulation and for demonstrating experimental results in cloud computing environments.**

***Keywords—clustering, pre-processing, quality of service, resource allocation.***

## 1. Introduction

In recent years, cloud computing has been used for developing a manner for providing assorted services and computational resources to the customers [1]. The Internet and central remote servers were used in cloud computing to produce scalable services for its customers. There is a requirement to establish an autonomic resource management approach that enhances both QoS targets: resource centric (reliability, availability, utilization) and user centric (budget, implementation time) [2]. Task scheduling is performed by the cloud computing platform. Firm resources, computational power and allocated tasks are used to restrict the task node. The job management node has various subtasks when a cloud computing task is assigned and stored in the task pool [3].

The process of resource allocation in cloud computing may be divided into two categories. The first one is static allocation. In this type, the cloud customer needs to create a prior demand for the resources. In this scenario the customer recognizes which resources are essential and in how many cases the resources are required. All that is performed prior to employing the system. Such an approach leads to overutilization or underutilization of resources, depending on the time of application. This is one of the disadvantages of using static allocation [4].

The other type is dynamic allocation. Here, cloud resources are required by the cloud customer when there is a demand for application. At this point, overutilization and underutilization may be prevented [5]. The service provider has to allocate resources from an alternative cloud information center [6], [7]. Cloud computing performs most important tasks between virtual machine (VM) placements, which is considered to be one of the most severe problems. This method will select the perfectly suited physical hosts with respect to energy efficiency and resource consumption – from the point of view of the cloud provider.

Calculations concerned with the capacity of each resource node are commonly unstable in heterogeneous or homogeneous multi-cluster and multi-resource environments [8]. Therefore, cloud computing scenarios need a consistent job scheduling scheme and a suitable resource management model [9], [10].

Scheduling algorithms which are commonly run on dedicated and homogeneous resources, in parallel, and on distributed systems, cannot perform well in the new cloud computing scenarios [11]. Transmission delay is a significant issue that disturbs the scheduling algorithm in cloud task scheduling [12], [13]. The creation and full usage of resources are also significant factors during scheduling [14]–[17].

Gang scheduling is used in distributed systems, and it is a very useful function [18]. The scheduling algorithm is necessary for assigning processors to the current activities in a distributed system. A scheduling algorithm assigns the functions that need to be performed simultaneously in the case of similar events, meaning mainly in communication operations. It is a powerful method for scheduling such activities, and it depends on time-space sharing [19]. The major role of the scheduling policy of gang scheduling algorithms is to collect the functions of a parallel job, and to perform them instantaneously on various proces-

sors. So, the threat of having wait for a function that is not currently working on any processor is eliminated [20]. In light of the above, one may notice that the quantity of functions within a gang cannot surpass the number of accessible processors.

The main objective of this paper is to address an efficient resource allocation aspect of cloud computing for selecting proper resources from a specific gang. It supports task scheduling with the aim of reducing the selection time. In this paper, a novel fuzzy Manhattan distance similarity-based ganging of resources (FMDSGR) technique, developed for gang formation of resources, with its characteristics aiming to attain better resource scheduling results in cloud computing, is discussed. Then, gang priority scheduling (GPS) is proposed to schedule tasks within a gang of processors.

The subsequent parts of the paper are ordered as follows. Section 2 reviews the various related works. Section 3 presents proposals for effective pre-processing of resources and task scheduling. The results are presented and discussed in Section 4. Lastly, the relevant research work is described in Section 5.

## 2. Related Work

In manufacturing relying on cloud systems, there is a considerable amount of assets which have comparable useful qualities, as indicated by the requirements of the manufacturing task at hand. Step-by-step instructions to choose the ideal assets series or assets combinations to finish the manufacturing task with a number of distributed subtasks was a key consideration in the investigation of cloud manufacturing. Cao *et al*. [21] had developed a multivariate process capability indicator to assess the manufacturing process with the unwavering quality of information on cloud scheduling. As indicated by the fuzzy quality theory, the strength level of intuitionistic fuzzy esteem was considered in decision making while selecting the ideal asset chain. Another technique was proposed for ideal determination of assets based on the multivariate process indicator and on the predominance level of intuitionistic fuzzy esteem. A practical contextual investigation was relied upon to outline the proposed strategy and technique.

Due to the burstiness of VM in computing clouds, real application workloads are characterized by spikes that are more often characterized by periodic occurrence, low frequency and a brief span. Zang *et al*. [22] explored the burstiness-mindful server solidification issue from the viewpoint of asset reservation, i.e. holding an accurate measure of additional assets on every physical machine (PM) to maintain a strategic distance from live movements. The creator first modeled the asset necessity pattern of each VM as a two-state Markov chain to catch burstiness, then an asset reservation system was intended for every PM, based on the stationary distribution of the Markov chain.

With the promotion and advancement of cloud computing, loads of logical computing applications have been developed in cloud scenarios. A general elastic resource provisioning and task scheduling mechanism performing the logical work in the cloud was used to address this issue. Shi *et al*. [23] had proposed elastic resource provisioning to perform logical work process employments in the cloud environment. The issue of elastic resource provisioning was displayed as a problem that consisted of elastic resource provisioning algorithms related to VM occupancy.

Saraswathi *et al*. [24] had proposed a technique that concentrated on the assignment of VM to the customer. This work sets a fundamental standard for low priority occupations (due date of the employment was high), without postponing the execution of high priority occupations (due date of the employment was low) and progressively allocated VM assets to the customer who was able to complete the work within the deadline.

Cloud computing offers the ability to share resources over various geographical destinations. Cloud resource scheduling has been a task that needs to be performed on a repetitive basis, as the issue of finding the best match for the workload needs to be tackled. Proficient management of the dynamic nature of resources should be possible with the assistance of cloud workloads. Very few effective resource scheduling strategies for energy, cost and time-imperative cloud workloads have been accounted for in writing. Singh *et al*. [25] had proposed an effective cloud workload administration framework, in which cloud workloads have been recognized, dissected and clustered through K-means on the premise of weights relegated and their QoS necessities. The execution of the proposed algorithm had been assessed with the existing scheduling strategies through the CloudSim toolkit. While clustering the cloud workloads utilizing K-means, diverse preparatory allotments can bring various final clusters, and it does not work well with clusters of different sizes and clusters prone to minimization.

Sfrent and Pop [26] dealt with the problem of scheduling a set of jobs across a set of machines and specifically analyzed the behavior of the system at very high loads, which was specific to big data processing. Under certain conditions, they could easily discover the best scheduling algorithm, prove its optimality and compute its asymptotic throughput. Vasile *et al*. [27] proposed a scheduling algorithm for different types of computation requests: independent tasks, like a bag of tasks model, or tasks with dependencies modeled as directed acyclic graphs, and they will be scheduled for execution in a cloud data center. The tasks in the requests are scheduled based on the available resources using the scheduling algorithm suitable for each request.

Vasile *et al*. in [28] proposed a resource-aware hybrid scheduling algorithm for different types of applications, such as batch jobs and workflows. The proposed algorithm considers hierarchical clustering of the available resources into groups in the allocation phase. Task execution was performed in two steps. In the first phase, tasks were assigned to groups of resources and in the second phase, a classical scheduling algorithm was used for each group

of resources. The proposed algorithm was suitable for heterogeneous distributed computing, especially for modern high-performance computing systems in which applications were modeled with various requirements (both IO and computationally intensive), with a particular emphasis on data from multimedia applications.

Li [29] approach was to design and analyze the performance of heuristic algorithms based on the equal-speed method. Pre-power and post-power determination algorithms were developed for both energy and time-constrained scheduling of precedence-constrained parallel tasks on multiple-core processors with continuous or discrete speed levels.

Rimal and Maier in [30] proposed a new cloud-based workflow scheduling policy for compute-intensive workflow applications in multi-tenant cloud computing environments, which helps minimize the overall workflow completion time, tardiness, cost of execution of the workflows, and utilize idle cloud resources in an effective manner. The proposed algorithm was compared with the state-of-the-art algorithms, i.e. first come first serve (FCFS), easy backfilling, and minimum completion time (MCT) scheduling policies to evaluate the performance levels.

Keshanchi *et al*. [31] proposed a powerful and improved genetic algorithm to optimize task scheduling solutions. The proposed algorithm uses the advantages of evolutionary genetic algorithm and heuristic approaches.

Taking into consideration the work mentioned above, we can say that there is a need to develop a resource management technique that optimizes both QoS targets: user-centric (cost and execution time) and resource-centric (reliability, availability, and utilization). Due to unavailability of the required resources at a point in time, some sub-tasks are stuck in a deadlock condition and continue to struggle for resources, which further leads to customer dissatisfaction (increasing execution time and cost). The QoS-aware resource management technique needs to be decentralized. In centralized distributed systems, it is tough to manage large numbers of user requests in multiple service queues, which further leads to performance degradation (decreased reliability and scalability). Due to the difficulty in predicting the behavior (regarding QoS requirements) and demand (in terms of resources) of the workload/application, there is a need for an effective QoS-aware resource management technique that can easily make the right decision concerning the dynamic scaling of resources.

## 3. Proposed Resource Pre-processing and Task Scheduling

The proposed work is divided into in two stages: pre-processing and task scheduling. Fuzzy clustering is done in the first stage, where resource characteristics, such as processing performance, average communication ability, maximum transmission capacity, network position, and many links are grouped into a gang of resources. Resources with a similar computing capability are grouped into one gang

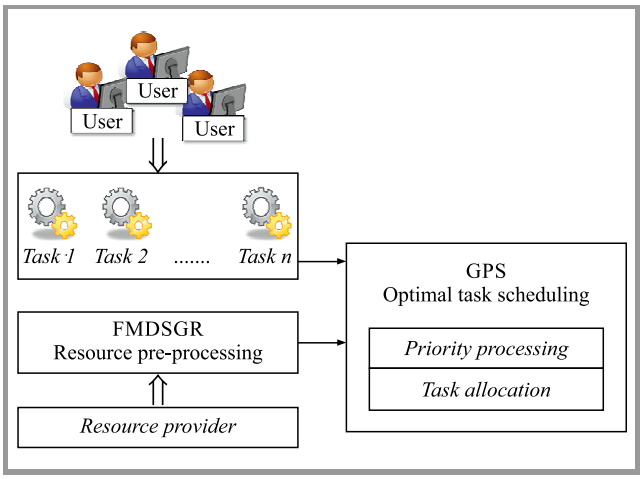


***Fig. 1.*** Architecture of the proposed scheme.

by separating all resources into many gangs based on the Manhattan distance similarity. The resources in one gang have a similar data transfer rate, as they share a similar network for communicating with each other. In the second stage, the tasks from the users are scheduled based on gang priority scheduling in order to minimize the scheduling length or makespan. The architecture of the proposed work is given in Fig. 1.

### *3.1. Resource Pre-processing using FMDSGR algorithm*

There are many typical workflow task scheduling algorithms which are deployed in a heterogeneous environment. Those methods are based on quantitative characteristics of the task and do not consider the service-oriented resources. Moreover, it is hard to describe exactly the task's demand for resources. Meanwhile, resource attributes cannot be described accurately. The fuzzy theory provides various effective means to solve the uncertain problems in the real world. Therefore, fuzzy clustering is used to divide the resources, and the Manhattan distance is used to group the resources into gangs based on similarity to improve the efficiency of task scheduling.

Also, computation and communication ability of resources allocated by the task affects the completion of the subsequent task. Therefore, distinguishing the performance of resources is conductive to choosing the appropriate resources for workflow task scheduling. However, it is difficult to describe the attributes of resources accurately, and there are no attributes to distinguish the processing units strictly. Fuzzy clustering is an effective method to divide resources.

**Design objective**: The purpose of the resource pre-processing method is to minimize the distance vector of the processor's characteristics to achieve effective utilization of resources. We can dynamically group the resources as gangs during scheduling in cloud systems, partitioning the given data by minimizing the distance objective function:

$$f(x) = \min(M_t) \in \varepsilon\,. \tag{1}$$

We have to gang the processing unit with minimum Manhattan distance $M_t$ of the $n$-th characteristic feature $t$ dependent upon the threshold value $\varepsilon$.

**Initialization of resources**: Consider the original dataset $U$ with the processing unit $\{p_1, p_2, \ldots, p_M\}$ in which each processor has its separate characteristics $u = \{t_1, t_2, \ldots, t_N\}$. In which, $p_M$ is the $x$-th processor of the $M$-th process unit and $t_N$ is the $y$-th characteristic value in the $N$-th characteristic elements. The performance of the scheduling process is improved by forming the resources in the cloud system as a gang. Qualitative characteristics refer to qualities or properties of cloud computing resources.

**Data standardization and extreme standardization**: We obtain the standardization data $U'$ to deal with the data $U$ in the target system using mean and standard deviation. The standardized value $u_{ik}$ of each data is equal to:

$$u'_{xy} = \frac{u_{xy} - \mu_{xy}}{\sigma_{xy}}, \tag{2}$$

where $u_{ik}$ denotes the $k$-th eigenvector of original data. In the Eq. 2, $\mu_{xy} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} u_{xy}$ is the mean value of $u_{xy}$ and $\sigma_{xy} = \sqrt{\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}(u_{xy} - \mu_{xy})^2}$ is the standard deviation of $u_{xy}$. The final value of the standardized data $u'_{xy}$is not in the range $[0, 1]$. Therefore, in order to normalize the $U'$ to $U''$ an extremely standardized method is presented. The extreme standardized method is defined as:

$$u''_{xy} = \frac{u'_{xy} - u'_{y\min}}{u_{y\max'} - u'_{y\min}}, \tag{3}$$

where $u'_{y\min}$ is the minimum value in $u'_{1y}, u'_{2y}, \ldots, u'_{Ny}$ and $u'_{y\max}$ is the maximum value in $u'_{1k}, u'_{2k}, \ldots, u'_{Nk}$.

**Fuzzy similarity matrix**: The correlation coefficient between the elements in the fuzzy matrix includes the similar index coefficient method. The similar index coefficient can be calculated as:

$$I_{xy} = \frac{1}{N}\sum \mathrm{e}^{\frac{3(u_{xk}-u_{yk})^2}{4\sigma''^2_{xk}}}, \tag{4}$$

where $n$ denotes the number of characteristic features. The values of $x$ and $y$ belongs to the processing units. The value of $k$ fits with the characteristic feature.

Consequently, a fuzzy similarity matrix is obtained as a result of the correlation coefficient between the elements and is expressed as:

$$F = [I_{xy}]_{2(n+1)} = \begin{bmatrix} I_{11} & I_{12} & \cdots & I_{1(n+1)} \\ I_{21} & I_{22} & \cdots & I_{2(n+1)} \\ \vdots & \vdots & \vdots & \vdots \\ I_{(n+1)1} & I_{(n+1)2} & \cdots & I_{2(n+1)} \end{bmatrix}_{2(n+1)}. \tag{5}$$

Then, fuzzy relation $F$ among $x$ and $y$ is established as a fuzzy subset of $x \times y$, in which $I_{xy}$ is the membership function in the interval $[0, 1]$. The fuzzy similarity between the processing units is referred to as $I_{xy}$ in the formula:

$$I_{xy} = \begin{cases} 1 & x = y \\ [0, 1] & x \neq y \end{cases}. \tag{6}$$

The fuzzy similarity is based upon the fuzzy relations, such as equivalence relation, fuzzy relation and similarity relation, which are necessary for reflexivity and symmetry.

**Manhattan distance based gang formation**: The process of grouping the elements or objects into clusters for a problem with a certain objective is called gang formation. It can expose formerly hidden relations in a multifaceted data set. Finding the similarity between two objects is the main issue in gang formation, so that the gang can be formed with a high similarity between the objects. Manhattan distance computes the absolute differences between the coordinates of a pair of objects. This means that the distance between two items is the total sum of the differences of their parallel elements. Figure 2 presents the process flow of the proposed FMDSGR.

The similarity between the data items of a gang is measured using the Manhattan distance $M$. The formula for this distance between a standardized data item and a center point is:

$$M_t = \sum_{x=1}^{M}\sum_{y=1}^{N} \left\| I'_{xy} - c_t \right\|. \tag{7}$$

In Eq. (7) $c_t = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} u''_{xy}}{MN}$ represents the center point of the $t$ characteristic value calculated after the extremely standardized output.

The maximum number of center points depends upon the minimum value of the objective function. Then, the gangs are formed using those center points represented as a gang. Members other than the center point could not be assigned to any gang. Therefore, they belong to two or more gangs.

After forming the gang, we have to decide, which gang executes the task. Hence, we have to add the separate characteristics of each gang and sort it in the descending order. We can decide, which gang processes the task based on the arrangement.

The objective of this algorithm is to reduce the completion time of every task in the workflow, whereas the start time and the finish time are the two important factors affecting the completion time. If the task is allocated to the different processing units, the execution implementation is also different. We schedule each task based on its suitable processing unit, which minimizes the finishing time of the task.

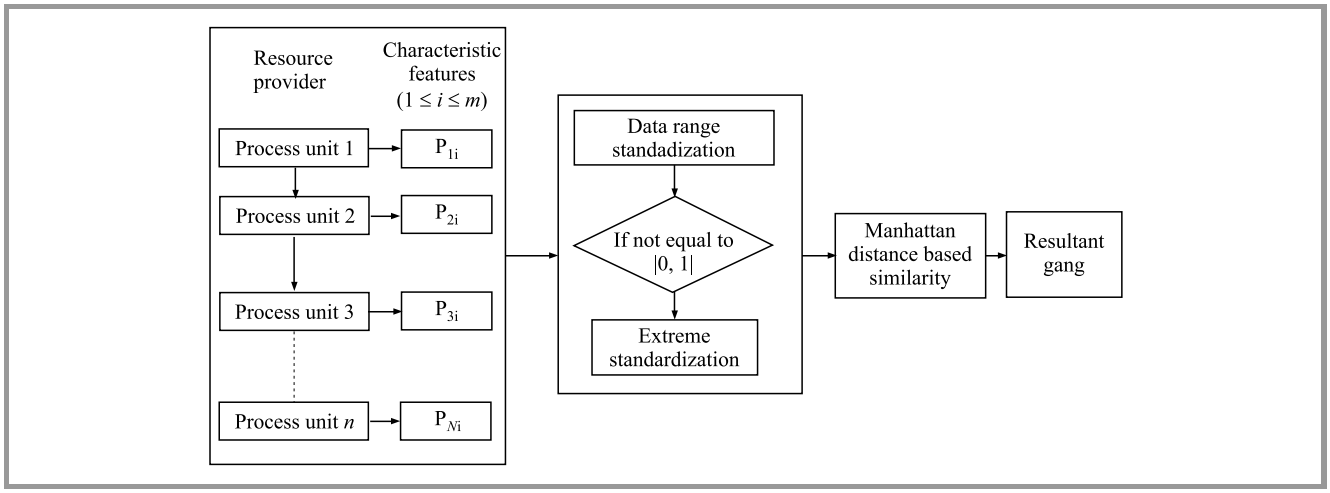


***Fig. 2.*** Process flow of proposed FMDSGR.

**Algorithm 1**: FMDSGR flowchart

**Input**: $N$ number of processors with characteristics
**Output**: gang
*The algorithm comprises the following steps*:

1. Random selection of the gang center
2. Initialization of resources $U = [u_{ij}]$ matrix, $U^{(0)}$
3. Data standardization
$$u'_{xy} = \frac{u_{xy} - u_{xy}}{\delta_{xy}}$$
4. Extreme standardization
$$u''_{xy} = \frac{u'_{xy} - u'_{y\min}}{u'_{y\max} - u'_{y\min}}$$
5. Fuzzy similarity matrix
$$F^e = [I'_{xy}]_{(n+1)\times(n+1)} = \begin{bmatrix} I'_{11} & I'_{12} & \cdots & I'_{1(n+1)} \\ I'_{21} & I'_{22} & \cdots & I'_{(n+1)(n+1)} \\ \vdots & \vdots & \vdots & \vdots \\ I'_{(n+1)1} & I'_{(n+1)2} & \cdots & I'_{2(n+1)} \end{bmatrix}_{(n+1)\times(n+1)}$$
6. At $k$-step: calculate the centers vectors $C^{(k)} = [c_t]$ with $U^{(k)}$
$$c_t = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} u''_{xy}}{MN}$$
7. Update $U^{(k)}$, $U^{(k+1)}$
$$M_t = \sum_{x=1}^{M}\sum_{y=1}^{N} \|I'_{xy} - c_t\|$$

If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$ or the minimum objective is achieved, then Stop. Otherwise return to step 2.

### *3.2. Task Scheduling Using GPS Algorithm*

Gang based priority scheduling is used to reduce the scheduling length (makespan) of the task scheduling workflow. The scheduling length gets reduced every time the task is performed and a ready task list (RTL) is established for priority allocation. If there is any task in the RTL that denotes the scheduled tasks of the parent node, the priority of the task is based up on the current-to-exit length (CEL) value. It represents the largest distance from the present node to the existing node:

$$CEL(t_x) = \frac{W(t_x)}{M_p} + \max_{t_y \in succ(t_x)} \left\{ \frac{T_{xy}}{M_c} + CEL(t_y) \right\}, \qquad (8)$$

where $M_p$ is the median of the computing ability of the processing units, $M_c$ is the median of the transmission ability between processing units, $T_{xy}$ is the inter-task communication between $t_x$, and $t_y \in succ(t_y)$ is the closest successor or child of the task $t_x$. The larger the CEL, the greater its priority. If greater the attribute priority value of a task means priority is also greater. After that, group the workflow task into gangs or jobs based on their priority.

Let us consider all the tasks within the same job being allocated at the same time. The job consists of a set of tasks called gangs and each task in the gang starts at the same time. Assume that every gang $G$ consists of $x$ tasks and $1 \le x \le \frac{P}{s}$, in which $s = 2$ and $P$ denotes the processor. Gang thus formed by cluster the task of parallel jobs and allocate the tasks simultaneously to different processors. The degree of parallelism of a gang is the number of the task in a job. The size of the gang is equal to the number of tasks in a gang $G$. If $p$ is the number of processors required by gang $G$, then $1 \le x \le p \le \frac{P}{s}$. Before scheduling, we should know that a small gang requires a limited number of the processing units, and a large gang requires a large processing unit.

#### 3.2.1. Gang Size Dissemination

The number of tasks of the gang (job) is evenly disseminated with in the range $\lfloor 1, \ldots, \frac{P}{s} \rfloor$. The mean of the gang size is represented as:

$$\gamma = \left\lceil \frac{1 + \frac{P}{s}}{2} \right\rceil. \qquad (9)$$

The number of jobs that can be processed in parallel depends on the gang size and the scheduling policy. The

scheduling algorithm selects the task in the gang which is on the top of RTL. The tasks will be released after executing each task from the first gang, so the RTL is updated after each task allocation process. The task from the gang with the highest priority completes the task with a minimum completion time during scheduling. The processor with the nearest center unit is selected when two processors have the same completion time.

#### 3.2.2. Computing the Primitive Begin Time (PBT)

While allocating tasks to the suitable processor, in order to achieve the smallest PBT value, calculate all the processing units' PBT assigned by $t_i$ to decide about the processing unit $p_1$.

$$PBT(t_x, p_y) = \max\left\{RFT(t_x, P_y), \max_{t_y \in PRED(t_x)}(AT(t_x, t_y, p_y)\right\}, \tag{10}$$

where $pred(t_x)$ is the nearest predecessor of $t_x$, $RFT(t_x)$ is the real finish time of the task $t_x$.

$$RFT(t_x, p_y) = RST(t_x, p_y) + \frac{w(t_x)}{w(p_y)}. \tag{11}$$

When $t_y \in PRED(t_x)$ and tasks $t_x$ and $t_y$ on $p_x$ and $p_y$. $w(t_x)$ is the computation time of the task $t_x$, the processing unit $p_y$ computation time is represented as $w(p_y)$. The obtained time on $t_x$ and $t_y$ on $p_x$ and as $p_y$ is:

$$AT(t_x, p_y) = RST(t_x, p_y) + \frac{C(t_x, t_y)}{C(p_x, p_y)}. \tag{12}$$

In Eq. (12) $C(t_x, t_y)$ is the inter-task communication time between $t_x$ and $t_y$, $C(p_x, p_y)$ is the communication time of the processing unit between $p_x$ and $p_y$.

#### 3.2.3. Computing the Primitive Completion Time

In the same way, the processing unit $p_2$ is selected and provides the minimum PCT value:

$$PCT(t_x, p_y) = PBT(t_x, p_y) + w_{xy}, \tag{13}$$

where $w_{xy}$ is the computing cost of the processing unit.

If the selected processing units are the same, then each task from the high priority gang is allocated to the first processor. Then, simultaneously update the LRT. If the selected processing unit is different, then select the nest gang of the task in the RTL with a low priority. Continue the step until the processors $p_1$ and $p_2$ become equal, and simultaneously update RTL.

The task with the highest priority is scheduled on the processing unit that can complete the task with the minimum completion time. When there is no vacancy in the optional set, we choose the processing unit of the cluster with the highest comprehensive performance in the backup set and calculate the completion time of the current task on this processing unit. If the completion time of this processing unit in the backup set is lower than the smallest completion time of the processing unit in an optional set, we put the current task schedule on the one in the backup set. If there are two processing units that have the same completion time, we can choose the processing unit whose network location is nearer to the center unit. Because of the consideration of communication cost, whenever possible, we place the tasks on the critical workflow path of the same processing unit.

**Algorithm 2**: Gang based priority scheduling

**Input**: Current LRT[LRT= $G_1, G_2, \ldots, G_n$];
$G_1$ – largest gang, $G_n$ – smallest gang;
$G = p_1, \ldots, p_n$
Task: Still exist tasks that haven't been allocated,
$T = t_1, t_2, \ldots, t_n$
LRT [head]: Head of the LRT
LRC [tail]: Tail of the LRT

**Output**: Refresh LRT to improve the resources scheduling performance

```
1: Cluster the processing unit with fuzzy theory
2: Compute each node's CEL
3: Establish the LRT
4: Choose the first LRT
5: While (Gang) do {
6: If (LRC→LRT [head]) Then {
        Compute the EST and choose the minimum
        value processing unit p1.
        Compute the EFT and choose the minimum
        value processing unit p2.
7: If (p1 = p2) Then {
                    Allocate each task of ti to p1
                    Remove task of ti from LRT,
                    update the LRT
              Else
                    To the next task in LRT
                  }
                    Else
                    //LRC→LRT[tail]
                    Go back to the first task
                    in LRT,
                    Compute the difference, and
                    Choose the processing unit.
8: Allocate ti to pi and delete ti from LRT
                  }          }
```

## 4. Results and Discussion

The experimental results were achieved by using the CloudSim toolkit as a simulation platform to simulate heterogeneous cloud environments. CloudSim [26] was also proposed by R. Buyya, extending the range of features of GridSim end enabling, modeling and simulating cloud environments, data centers, virtual machines, cloudlets, etc.

Five characteristic features have been employed in this paper to improve scheduling performance in the resource network:

- processing performance $t_1$: the average computing ability among processing units in the resource system, it represents the average time per task executed by a processing unit. If the resultant value is small, this means that the processing unit costs are also lower during computation;
- average communication ability $t_2$: the average communication ability value of the processor-connected links that can be calculated from the average weight of edges connected to the processing elements;
- maximum transmission capacity $t_3$: the maximum number of edges connected to a processing unit with the transmission capacity;
- network position $t_4$: position of the processing element in the network, defined as the network position. If the processing unit is far away from the center of the network, then the value is greater;
- number of links $t_5$: the number of links that are connected to a processing element.

Table 1 shows the characteristics of the original data $U$ for the processing units. The cloud resources should be formed as gangs to use the resources effectively and to divide the processing units.

Table 1
Original dataset

| Processing unit | Characteristic features | | | | |
|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ |
| $P_0$ | 20 | 0.05 | 0.05 | 6 | 1 |
| $P_1$ | 30 | 0.08 | 0.05 | 9 | 2 |
| $P_2$ | 35 | 0.1 | 0.1 | 10 | 2 |
| $P_3$ | 50 | 0.13 | 0.1 | 14 | 4 |
| $P_4$ | 45 | 0.13 | 0.1 | 13 | 4 |
| $P_5$ | 50 | 0.13 | 0.1 | 14 | 3 |
| $P_6$ | 30 | 0.1 | 0.1 | 9 | 1 |
| $P_7$ | 28 | 0.1 | 0.1 | 8 | 1 |
| $P_8$ | 48 | 0.11 | 0.03 | 13 | 5 |
| $P_9$ | 33 | 0.1 | 0.1 | 9 | 1 |
| $P_{10}$ | 30 | 0.08 | 0.08 | 8 | 1 |
| $P_{11}$ | 35 | 0.1 | 0.1 | 10 | 1 |
| $P_{12}$ | 40 | 0.12 | 0.1 | 11 | 1 |

In the proposed work, first the original data $U$ can be standardized. Then, the similarity is calculated based on the Manhattan distance, and we obtain the overall gang formation results as gang 1: $\{P_0, P_2, P_6, P_7, P_7, P_9, P_{10}, P_{11}, P_{12}\}$ and gang 2: $\{P_1, P_3, P_4, P_5, P_8\}$. The first gang result forms gang 1, and the next gang result forms gang 2.

The proposed work ensures better performance when the data sets are different or detached from one another. Manhattan distance-based similarity of the presented method is compared with the arithmetic mean method in Table 2. The successful result has not been achieved due to the random selection of gang center in the arithmetic mean method, and this method does not work well for categorical data, i.e. it is applicable only when the mean is defined.

Table 2
Comparison of gangs in Manhattan distance and arithmetic mean methods

| Manhattan distance (MFC) | | Arithmetic mean | |
|---|---|---|---|
| Gang 1 | Gang 2 | Gang 1 | Gang 2 |
| $P_0$ | $P_1$ | $P_0$ | $P_3$ |
| $P_2$ | $P_3$ | $P_1$ | $P_4$ |
| $P_6$ | $P_4$ | $P_2$ | $P_5$ |
| $P_7$ | $P_5$ | $P_6$ | $P_6$ |
| $P_9$ | $P_8$ | $P_7$ | – |
| $P_{10}$ | – | $P_9$ | – |
| $P_{11}$ | – | $P_{10}$ | – |
| $P_{12}$ | – | $P_{11}$ | – |
| – | – | $P_{12}$ | – |

In order to decide which gang to choose to execute the task at hand, we have to add any of the characteristics of the processor in the gang and sort the results in the descending order. In this paper, we have used the average communication ability $t_2$, and the maximum transmission capacity $t_3$ as our characteristics, and added all $t_2$ and $t_3$ features of the gang 1 and gang 2 separately.
The total average communication ability equals 0.74 for gang 1 and 0.59 for gang 2. The total maximum transmission capacity amounts to 0.75 for gang 1 and 0.38 for gang 2.
After calculating the characteristics, gangs with high average communication and transmission capacity values have been sorted in the descending order as [0.75, 0.59] and [0.75, 0.38]. From this, we can decide that gang 1 is to to execute the task. Thus, the Manhattan distance based fuzzy clustering (MFC) is an effective technique for gang formation enabling a considerable reduction in the cost of deciding which processor is to execute the task and the selection of the gang depends on the performance characteristics.
The various performance metrics to compute the results are:

**Makespan**: makespan (M) is referred to as the scheduling length. It is measured by calculating the ending time of the final task using the proposed algorithm:

$$M = \max\left\{PCT(t)\right\}, \tag{14}$$

in which $PCT(t)$ is the primitive completion time of the task.

**Scheduling length ratio**: the scheduling length ratio (SLR) in the lower bound is the time taken to execute the task on a serious path. SLR is defined to normalize the schedule length:

$$SLR = \frac{M}{\sum\limits_{t_i \in CP_{\min}}^{n} \min\limits_{p_j \in P}\{w_{i,j}\}}. \tag{15}$$

SLR is defined as the ratio of makespan to the total sum of minimum computation cost $CP_{\min}$ of all tasks. The resultant graph value of SLR is not less than one. When the SLR output is low, then the scheduling algorithm provides better performance.

**Speed up**: the ratio of parallel execution time and the sequential execution time is the speedup (S). The scheduling length on a limited number of processors is the parallel execution time and the sum of entire computation time of every task is the sequential execution time:

$$S = \min_{p_j \in P} \left\{ \frac{\sum\limits_{i=1}^{n} w_{i,j}}{M} \right\}, \tag{16}$$

where the total sum of the computational time of tasks is defined as $\sum\limits_{i=1}^{n} w_{i,j}$, $i = 1, 2, 3, \ldots, n$.

**Efficiency**: the efficiency is the measure of the utilization of the processor of a parallel program:

$$\eta = \frac{S}{NP}, \tag{17}$$

where $S$ stands for speed up and NP is the number of processors.

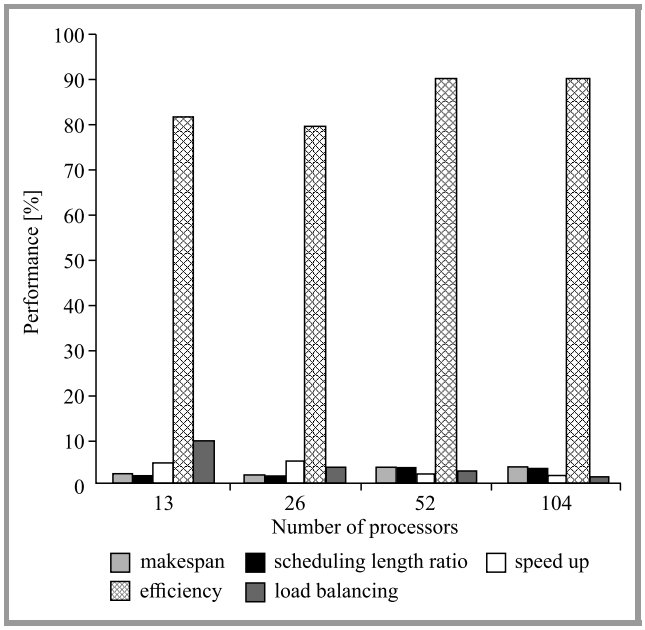


*Fig. 3.* Performance comparison of the proposed GPS method for different number of processors.

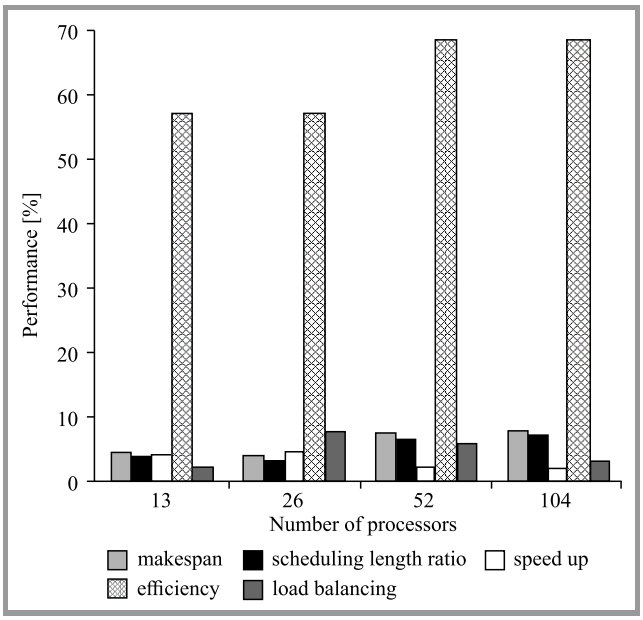


*Fig. 4.* Comparison of the existing LJFS scheduling.

**Load Balancing**: the ratio of the overall makespan of processors and the average execution time is the measure of load balancing (LB):

$$LB = \frac{M}{A}, \tag{18}$$

where $M$ is scheduling length. The average $A$ is taken from the ratio of a sum of the processing time of each processor and the number of processors used.

The proposed fuzzy-based gang priority scheduling method is compared with the largest come first serve (LJFS), Yarn, and Mesos scheduling method.

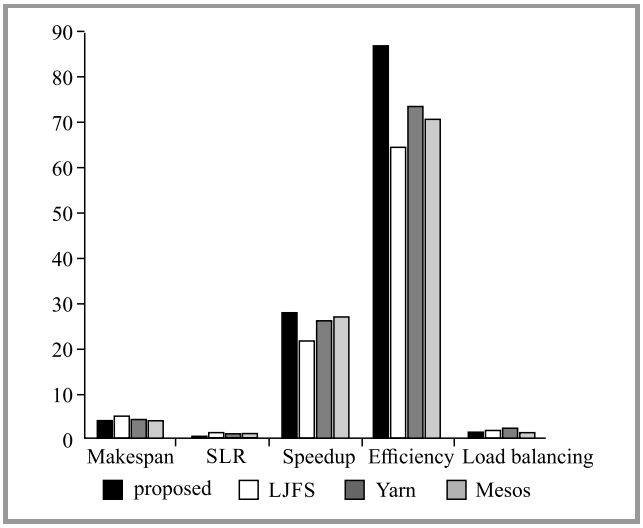


*Fig. 5.* Comparison of the proposed GPS method with the LJFS scheduling.

The comparison of the various performance metrics shows that the proposed method provides better performance when

compared with the LJFS method, as presented in Figs. 3 and 4.

The overall comparison of the proposed method with the LJFS, Yarn, and Mesos scheduling is given in Fig. 5. As a result, the proposed method minimizes the waiting time between tasks and reduces the idle time interval among the processing elements.

## 5. Conclusion

The work proposed in this paper has led to developing efficient pre-processing (gang forming) and task scheduling of cloud resources. Similar resources are ganged based on the FMDSGR approach to reduce the scheduled time while selecting the processor. The proposed pre-processing algorithm is fast, robust and easier to understand. When data sets are different or separated from each other, improved results are achieved. Thus, the resultant method enables to form gangs more effectively when compared with the existing arithmetic mean method. The resources can be utilized effectively with the resultant gang. The scheduling length of the workflow is prioritized using the GPS algorithm. Finally, the QoS performance metrics have been evaluated in terms of makespan, SLR, speedup, efficiency and load balancing. The proposed method provides improved performance when compared with the existing LJFS, Yarn, and Mesos method.

## References


[1] M. Armbrust *et al.*, "A view of cloud computing", *Commun. of the ACM*, vol. 53, no. 4, pp. 50–58, 2010 (doi: 10.1145/1721654.1721672).

[2] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis", *J. of Network and Comp. Appl.*, vol. 66, pp. 64–82, 2016 (doi: 10.1016/j.jnca.2016.01.018).

[3] J. Ma, W. Li, T. Fu, L. Yan, and G. Hu, "A novel dynamic task scheduling algorithm based on improved genetic algorithm in cloud computing", in *Wireless Communications, Networking and Applications*, Q.-A. Zeng, Ed. Springer, 2016, pp. 829–835.

[4] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", *J. of Internet Serv. and Appl.*", vol. 1, no. 1, pp. 7–18, 2010.

[5] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. Da Fonseca, "Scheduling in hybrid clouds", *IEEE Commun. Mag.*, vol. 50, no. 9, pp. 42–47, 2012.

[6] B. Saovapakhiran, M. Devetsikiotis, G. Michailidis, and Y. Viniotis, "Average delay SLAs in cloud computing", in *Proc. IEEE Int. Conf. Commun. ICC 2012*, Ottawa, ON, Canada, 2012, pp. 1302–1308 (doi: 10.1109/ICC.2012.6364548).

[7] S. Saha, S. Pal, and P. K. Pattnaik, "A novel scheduling algorithm for cloud computing environment", in *Computational Intelligence in Data Mining – Volume 1*, H. S. Behera and D. P. Mohapatra, Eds. Springer, 2015, pp. 387–398.

[8] K. Chandran, V. Shanmugasudaram, and K. Subramani, "Designing a fuzzy-logic based trust and reputation model for secure resource allocation in cloud computing", *Int. Arab J. of Inform. Technol.* (*IAJIT*), vol. 13, no. 1, pp. 30–37, 2016.

[9] T. Luis, C. A. Caminero, B. Caminero, and C. Carrion, "A strategy to improve resource utilization in Grids based on network-aware meta-scheduling in advance", in *Proc. 12th IEEE/ACM Int. Conf. on Grid Comput. GRID 2011*, Lyon, France, 2011, pp. 50–57, 2011.

[10] X. Li *et al.*, "Cloud tasks scheduling meeting with QoS", in *Proceedings of the 2015 International Conference on Electrical and Information Technologies for Rail Transportation*, Y. Qin, L. Jia, J. Feng, M. An, and L. Diao, Eds. Springer, 2016, pp. 289–297.

[11] N. Moganarangan, R. G. Babukarthik, S. Bhuvaneswari, M. S. Saleem Basha, and P. Dhavachelvan, "A novel algorithm for reducing energy-consumption in cloud computing environment: Web service computing approach", *J. of King Saud Univ. – Comp. and Inform. Sci.*, vol. 28, no. 1, pp. 55–67, 2016.

[12] J. Dümmler, R. Kunis, and G. Rünger, "SEParAT: scheduling support environment for parallel application task graphs", *Cluster Comput.*, vol. 15, no. 3, pp. 223–238, 2012.

[13] J. J. Durillo, H. M. Fard, and R. Prodan, "MOHEFT: A multi-objective list-based method for workflow scheduling", in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. and Sci. CloudCom 2012*, Taipei, Taiwan, China, 2012, pp. 185–192 (doi: 10.1109/CloudCom.2012.6427573).

[14] K. R. Remesh Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud", in *Innovations in Bio-Inspired Computing and Applications*, V. Snášel *et al.*, Eds. Springer, 2016, pp. 67–78.

[15] G. Peng, H. Wang, J. Dong, and H. Zhang, "Knowledge-based resource allocation for collaborative simulation development in a multi-tenant cloud computing environment", *IEEE Trans. on Services Comput.*, 2016 (doi: 10.1109/TSC.2016.2518161).

[16] F. Koch, D. M. Assunção, C. Cardonha, and A. S. M. Netto, "Optimising resource costs of cloud computing for education", *Future Gener. Comp. Systems*, vol. 55, pp. 473–479, 2016 (doi: 10.1016/j.future.2015.03.013).

[17] D. Saxena, R. K. Chauhan, and R. Kait, "Dynamic fair priority optimization task scheduling algorithm in cloud computing: concepts and implementations", *Int. J. of Comp. Netw. and Inform. Secur.*, vol. 8, no. 2, pp. 41–48, 2016.

[18] J. Choi, T. Adufu, Y. Kim, S. Kim, and S. Hwang, "A job dispatch optimization method on cluster and cloud for large-scale high-throughput computing service", in *Proc. Int. Conf. on Cloud and Autonom. Comput. ICCAC 2015*, Cambridge, MA, USA, pp. 283–290.

[19] I. A. Moschakis and H. D. Karatza, "Evaluation of gang scheduling performance and cost in a cloud computing system", *The J. of Supercomput.*, vol. 59, no. 2, pp. 975–992, 2012.

[20] I. A. Moschakis and H. D. Karatza, "Performance and cost evaluation of Gang Scheduling in a Cloud Computing system with job migrations and starvation handling", in *Proc. 16th IEEE Symp. on Comp. and Commun. ISCC 2011*, Kerkyra, Corfu, Greece, 2011, pp. 418–423. IEEE, 2011.

[21] Y. Cao, Z. Wu, T. Liu, Z. Gao, and J. Yang, "Multivariate process capability evaluation of cloud manufacturing resource based on intuitionistic fuzzy set", *The Int. J. of Adv. Manufactur. Technol.*, vol. 84, no. 1–4, pp. 227–37, 2016.

[22] S. Zhang, Z. Qian, Z. Luo, J. Wu, and S. Lu, "Burstiness-aware resource reservation for server consolidation in computing clouds", *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 27, no.4, pp. 964–77, 2016.

[23] J. Shi, J. Luo, F. Dong, J. Zhang, and J. Zhang, "Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints", *Cluster Comput.*, vol. 19, no. 1, pp. 167–182, 2016 (doi: 10.1007/s10586-015-0530-0).

[24] A. T. Saraswathi, Y. R. Kalaashri, and S. Padmavathi, "Dynamic resource allocation scheme in cloud computing", *Procedia Comp. Science*, vol. 47, pp. 30–36, 2015 (doi: 10.1016/j.procs.2015.03.180).

[25] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: issues and challenges", *J. of Grid Comput.*, vol. 14, no. 2, pp. 217–264, 2016.

[26] A. Sfrent and F. Pop, "Asymptotic scheduling for many task computing in big data platforms", *Inform. Sciences*, vol. 319, pp. 71–91, 2015 (doi: 10.1016/j.ins.2015.03.053).

[27] M. A. Vasile, F. Pop, M. C. Nita, and V. Cristea, "MLBox: Machine learning box for asymptotic scheduling", *Inform. Sciences*, 2017 (doi: 10.1016/j.ins.2017.01.005).

[28] M. A. Vasile, F. Pop, R. I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing", *Future Gener. Comp. Systems*, vol. 51, no. C, pp. 61–71, 2015.

[29] K. Li, "Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment", *Future Gener. Comp. Systems*, 2017 (doi: 10.1016//j.future.2017.01.010).

[30] B. P. Rimal and M. Maier, "Workflow scheduling in multi-tenant cloud computing environments", *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 28, no. 1, pp. 290–304, 2017.

[31] R. Calheiros, R. Ranjan, A. Beloglazov, C. DeRose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Pract. and Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

**S. Sharon Priya** completed her B.Tech. in Information Technology and M.Tech. in Computer Science Engineering from Anna University in 2008. She is presently working as an Assistant Professor at the Department of Computer Science and Engineering at B. S. Abdur Rahman University, Chennai, India. Her areas of interest include computing and image processing. At present, she is pursuing Ph.D. from B.S. Abdur Rahman University. She has published/presented several research papers in international journals and at conferences.
E-mail: sharonpriyaphd@gmail.com
sharonpriya@bsauniv.ac.in
Department of Computer Science and Engineering
B. S. Abdur Rahman University
Chennai, Tamil Nadu, India

**K. M. Mehata** obtained his Ph.D. from the Indian Institute of Technology, Madras, and has been, since 2009, Professor & Dean at the School of Computer, Information and Mathematical Sciences at B. S. Abdur Rahman University since 2009. He guided more than fifteen doctoral candidates and published about 70 research papers concerned with image processing, computer networks, software engineering, web mining and medical informatics. He is currently guiding 10 doctoral students. He is an expert member of AICTE, National Board of Accreditation, DRDO, MHRD and other universities' academic councils study boards.
E-mail: mehatakmphd@gmail.com
School of Computer
Information and Mathematical Sciences
B. S. Abdur Rahman University
Chennai, Tamil Nadu, India

**W. Aisha Banu** has a Ph.D. degree and focuses on information retrieval and natural language processing. She has a rich teaching experience of seventeen years and has published research papers in peer-reviewed journal and has presented them at conferences. She is a life member of ISTE and a member of ACM.
E-mail: aisha@bsauniv.ac.in
Department of Computer Science and Engineering
B. S. Abdur Rahman University
Chennai, Tamil Nadu, India