DOI: 10.21917/ijsc.2025.0542

FUZZY SCHEDULING OF SCIENTIFIC WORKFLOWS WITH ENERGY AND SECURITY CONSTRAINTS IN CLOUD

P. Tharani, K. Manimala and A.M. Kalpana
Department of Computer Science and Engineering, Government College of Engineering, Salem, India

Abstract

Cloud infrastructure is a pay-per-use, easily scalable, and accessible model. Based on the requirements of the workflow application, provisioning of resources can be dynamically done as the cloud is elastic in nature. The objective of this work is to execute scientific workflows within user deadlines having minimal expense and the total execution time. Appropriate provisioning and scheduling of all the tasks can effectively decrease the execution time in scientific workflows. This work suggests fuzzy scheduling for improvising the efficacy of energy as well as security. A major role in cloud computing is made by fuzzy logic theory. This work describes an approach for addressing the energy as well as security constraints in the cloud using the fuzzy controller. By referring to the fuzzy inference knowledge base, the duration, energy consumption and trust metrics can be inferred using fuzzy logic, based on the user task types. The system realizes the dynamic scheduling of the resources as per the specific needs. This achieves the purpose of improving the execution ratio and the utilization of resources. It has been demonstrated through the outcomes that the suggested scheduling algorithm can be efficiently deployed on the cloud.

Keywords:

Fuzzy Scheduling, Scientific Workflow, Partial Critical Path (PCP), Energy Factor, Cloud Computing

1. INTRODUCTION

Cloud computing is the instance of a distributed environment, which has developed over time from shared community platforms to utility-based models. The delivery of IT resources over the internet [1] has been possible because of this technology; here, based on the consumption, the users are charged and hence this is a pay-as-you-go model. There are different types of cloud providers based on various types of product offerings. They divide into three categories of as-a-service terms: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

A procedure that comprises a series of steps that are used for simplifying the execution complexity and the application management is referred to as workflow. There is a common model that is comprised of workflow technology that is used to describe a range of scientific applications in distributed systems. In many fields of science such as computer science, chemistry, and physics, there is an important role of workflow technology. The requirement for constructing upon the legacy codes that are very expensive to re-write gives rise to the interest in workflow. The Directed Acyclic Graph (DAG) compute tasks are represented by the nodes; the precedence and the flow constraints between the tasks are represented by the edges. For accessing, managing and processing large amounts of data from a higher level, scientific workflows make use of distributed resources. A distributed collection of storage and computation facilities is required to process and manage such huge amounts of data. These resources are not only shared amongst many users but are also frequently limited [2].

Scientific workflows are computational as well as data-intensive; this is due to the complexity of scientific processes. These workflows are to be executed in distributed as well as high-end computing environments like the cloud computing environments that have emerged of late. Many features of workflows that are different from other computing environments are provided by the cloud computing environment. (1) Computational cloud resources have been exposed as services providing standardized interfaces for accessing services over the network. (2). The quantity and type of computing resources allocated to a workflow are determined by service demands. (3) Due to the ability to dynamically alter the number of resources assigned to a process at runtime, workflow computing resources can be elastically scaled on demand. (4) Resources be allocated when required and not all of them need to assigned at the beginning of the workflow [3].

One of the most important activities that is executed in the cloud computing environment is scheduling. Scheduling is a task that maximizes the profit for increasing the efficiency of the workload in cloud computing. Using the resources appropriately considering the load distributed among the resources to minimize the execution time is the basic objective of planning calculations in the cloud environment. There are three classifications of the process of scheduling: Resource Discovering and filtering- Here, status information of the available resources is gathered following the data center broker's presentation of the data regarding the resources' current accessibility. Resource Selection: Based on the task and the resource parameter, in this stage, resource selection is done. This is the stage of selection. Task Submission: Here, the tasks are allocated to the resources [4].

Cloud computing has evolved from the principles of grid computing, service-oriented computing, and virtualization. Thus, whatever scheduling algorithms have been developed for these system types, the same ones can also be used for clouds. The main traits of the scheduling algorithms are their distinguishable characteristics: Target System: These are the systems developed for the scheduling algorithms; these can be a cloud computing system, grid system or heterogeneous system. Optimization criterion: The metrics that are considered for decision making by the schedulers are makespan and the cost and these are specified by the cloud user. The computer system consists of multiple cores based on scheduling algorithms in resource selection [5].

On-demand resources: Depending on demand, resources might be planned or leased for an extended period. During the execution of the workflow, the scheduling algorithm treats the on-demand leasing of resources as a "single expense". Reserved Resources: The use of these resources is considered by the algorithm for a long period. Levels in a Service Level Agreement (SLA): The SLAs can be hierarchically organized by the

scheduling algorithms. Clients and providers are allowed to directly interact and negotiate the prices and the capacities of the resources by the SLAs with a single level. Heuristics information: It guides the direction of search and is some problem-based value. Heuristics are designed based on the workflow problem. Multicore, bandwidth, makespan and cost are the main heuristics.

These are some of the common aims for the scheduling schemes of the workflows: Budget: It defines the expenses incurred by the consumers for using the cloud resources. Deadline: The limit of time for executing the workflow and also its support is an important Quality of Service (QoS) requirement. Reliability: Scheduling techniques such as active replications and backup/restart schemes that correspond to resource and time redundancy may be used to increase the task's likelihood of completion. Availability: tasks are not only executed faster but the executions are also terminated quickly due to the proper scheduling of workflows. The availability of cloud resources is improvised by this [6]. Minimizing the makespan: The time at which the final workflow task execution is completed is referred to as makespan. Supporting SLA: A document that contains various considerations from the service providers and the consumers is referred to as the SLA that comprises the discussions regarding the delivery performance assurance of the QoS. Security: Some of the features of the cloud may be misused by the attackers and the components are also misused for launching cloud-specific attacks. Load Balancing: To avoid the overloading of any of the cloud resources, a scheduler has to optimize the usage of resources.

The user should have some knowledge of the energy that is needed for executing their application in the model of energy-constrained provisioning. This should be based on the provider's information and a workflow ensemble for execution that specifies the energy constraints has been submitted. For the total consumption of energy to stay within an energy budget limit, provisioning is made. Particularly, for executing as many workflows as possible for the given energy constraints, a scheduling plan for the ensemble is developed. When the overall energy usage is less than the energy, the group's workflow is approved and added. Different amounts of energy to be spent for the execution of workflows are required by different execution schemes. Static and dynamic energy are the two schemes for the consumption of energy. The former refers to the energy that is consumed by the system resources when idle and the latter refers to the energy consumed when the applications are running and this can be adapted by the workflow depending on the resources that are required by the tasks [7].

Because of the unreliability of the distributed systems and the intrinsic uncertainty, security is an important issue in cloud computing. For instance, in Google, the event of file leakage is not allowed; also, interrupting Amazon's simple storage service S3 leads to the paralysis of the service. It is a subjective statement of the trustor regarding the security, honesty, reality, and dependability of a service held by the trustee and affects security in a large, dispersed internet environment. The user's decision about the authenticity, integrity, reliability and stability of services delivered by Cloud Service Providers (CSPs) is replicated by trust. Identifying if the cloud service can live up to its identity is referred to as authenticity. Verifying whether the cloud services behave as claimed is referred to as integrity. The ability of the cloud services to ensure data and application security is referred to as reliability. The stability of cloud services is referred to as stability and the likelihood of the effective execution of user's tasks is referred to as stability. Fraudulent services exist in real cloud environments as the resources and the service types are generally uncertain and they change dynamically. This makes it adverse for the users to claim appropriate services. Hence, it is of paramount importance to evaluate the trust of the cloud services and to select high-quality cloud services for the users [8].

This work proposes PCP, robustness and fuzzy logic for cloud-based scheduling. The remainder of the work is structured as follows: In the second part, the relevant literary works are examined. The various techniques used in the work are discussed in the third section. The fourth portion discusses the empirical findings, while the fifth section provides the work's conclusion.

2. RELATED WORKS

For maximizing resource utilization, a delay-constrained optimization problem has been formulated by Zhu et al. [9]. Also, to minimize the cloud overhead within the execution time bound that is user-specified, it has suggested a two-step workflow scheduling algorithm. It has been shown via the extensive simulation outcomes that lower computing overhead are achieved using this approach consistently; also within the execution time limits, higher resource utilization has been achieved. The total execution time has been decreased using the approach; this is done by strategically selecting the appropriate mapping nodes for the modules that are prioritized.

The issue of workflow scheduling in cloud computing and utilities has been taken into consideration by Cai et al. [10]. This takes care of the assignment of tasks so that while the precedence constraints are met along with meeting the workflow deadlines, the total rental cost of the resources is also minimized. For solving small problem instances, A Mixed Integer programming (MILP) model is developed. Since this is a problem of non-deterministic polynomial (NP) hardness, a Critical Path-based Iterative (CPI) heuristic is created. This heuristic can find accurate solutions for huge problem instances wherein there is an iterative construction of several complete critical paths; dynamic programming is used to do this, with the longest and least expensive services for unscheduled activities and service assignments for scheduled ones. After relaxing every critical path to a Multi-stage Decision Process (MDP) problem, a dynamic programming-based Pareto method is suggested for optimization. It has been shown via empirical outcomes that the existing state-of-the-art algorithms are outperformed by the suggested CPI heuristic on many of the problem instances.

Completion Time Driven Hyper-Heuristic (CTDHH) is a proposed method for cost optimization of Scientific Workflow Scheduling (SWFS) in a cloud setting by Alkhanak and Lee [11]. Four of the popular population-based meta-heuristic algorithms have been employed by the CTDHH approach and these act as Level Heuristic (LLH) algorithms. Furthermore, after each run, the CTDHH approach dynamically chooses an appropriate algorithm from the LLH algorithm pool by including the optimal workflow completion time and functioning as a high-level selector; this is how it enhances the native random selection technique. The suggested CTDHH approach has been evaluated

using practical cloud-based experimentation; there are five baseline approaches with which it has been compared- of these, four are population-based approaches and one is an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA). For evaluating the computational and data intensities, many different scenarios have been taken into consideration. The suggested approach has generated good results across experimental scenarios and hence proved its efficacy.

The cloud centers are comprised of Virtual Machines (VMs) as well as Physical Machines (PMs). Huge amounts of energy are consumed by the data centers as a result of improper utilization of resources as well as the non-existence of effective scheduling algorithms for performing task resource mapping. Carbon emissions, high maintenance costs and heavy consumption of energy are some of the effects of this issue. For addressing these issues and the challenges that are associated, Mohanapriya et al. [12] proposed a Power Efficient Scheduling and VM Consolidation (PESVMC) algorithm. The various works in literature that have been carried out pay attention to the techniques of energy management to hardware level support for decreasing the consumption of energy. The flexibility of the virtualization technology has been taken into consideration and the suggested algorithm has emphasized the software level; there are two phases: the VM scheduling phase, and the VM consolidation phase. WorkflowSim is used for performing the experimental evaluation and not only is better resource utilization achieved, but also the suggested algorithm achieves significant consumption of energy.

A Particle Swarm Optimization (PSO) based heuristic has been presented by Pandey et al. [13]. It can take into consideration the costs of data transmission as well as computation and can thus schedule applications to cloud resources. Experiments have been conducted using workflow models and by changing the costs of communication as well as computation. When employing the PSO and the current "Best Resource Selection" (BRS) method, the cost of savings has been calculated. It has been shown via the outcomes that PSO can attain upto thrice the savings in cost compared to BRS and that there is an effective workload distribution among the resources.

A heuristic scheduling approach based on Cat Swarm Optimization (CSO) has been proposed for allocating an application's tasks to the available resources suggested by Bilgaiyan et al. [14]. The execution cost of the task on different resources as well as the data transmission cost between two dependent resources are considered in the CSO heuristic algorithm. By using a hypothetical workflow, the authors have experimented with the suggested CSO algorithm; they have compared the outcomes of the workflow scheduling with the existing PSO algorithm. It has been shown by the experimental outcomes that (1) The CSO provides an ideal Task-to-Resource (TOR) scheduling plan that reduces the overall cost. (2) The task improvises in terms of the number of iterations using the CSO over the existing PSO. There is an optimal workload distribution amongst various resources using CSO.

As there are more and more applications that involve data of huge sizes, the current computing systems require greater data handling and processing capabilities. This is why cloud services are expensive. There is a requirement for effective scheduling so that the tasks are allocated resources to optimize the overall cost. A Bat Algorithm (BA) application for scheduling workflow or data-intensive applications in a cloud computing environment was presented by Sagnika et al. [15]. After executing the algorithm, the outcomes have been compared with two popular algorithms-PSO and CSO. The suggested BA algorithm has proven to present a fair distribution of load alongside the optimal cost of processing with better convergence.

Verma and Kaushal [16] accessed a non-dominance sort-based Hybrid Particle Swarm Optimization (HPSO) algorithm. These suggested approaches link the multi-objective PSO with the previously suggested Budget and deadline-constrained Heterogeneous Earliest Finish Time (BDHEFT) algorithm. Makespan and cost are the two conflicting heuristics that the HPSO tries to optimize using the deadline and the budget constraints. The energy that is consumed when the workflow schedule is created is also minimized along with these two conflicting objectives. A set of Pareto optimal solutions has been generated by the suggested algorithm and this is where the best solution can be chosen from.

To strictly restrict other Genetic Algorithms (GA), an adaptive penalty function was proposed by Liu et al. [17]. Also, for adjusting the probability of crossover and mutation, the co-evolution approach was used and this was able to enhance the speed of convergence and also pre-empt prematurity. The algorithm has also been compared with standards like random, PSO, Heterogeneous Earliest Finish Time (HEFT), and GA in a WorkflowSim simulator on four representative scientific workflows. The results demonstrate that the proposed method outperformed the other well-known ones in terms of meeting time requirements and lowering the overall cost of execution.

Guo et al. [18] presented a Workflow Task Scheduling algorithm based on the Resources' Fuzzy Clustering (FCBWTS). The primary goal of scheduling is to minimize the makespan of the precedence-constrained applications represented by the directed acyclic graph. The resource features of cloud computing are taken into account in FCBWTS, a collection of attributes. These traits which have been used for delineating the artificial performance of the resource processing units have been described in this work. The fuzzy clustering approach pre-treats the processing unit network with these characteristics, and the ready task's execution duration affects the critical path. This will help realize the reasonable partition of the processor network. This expense of deciding which of the processors can execute the current task is decreased.

A new adaptable cloud workflow scheduling model has been suggested by Li et al. [19]. There are two stages into which the workflow scheduling of the new model has been divided so that the user requirements are better analyzed and customizable services are provided macro multi-workflow scheduling as the unit of cloud user and the micro single workflow scheduling. The trust approach is integrated into workflow scheduling. Workflows are classified as time-sensitive and cost-sensitive in a single workflow scheduling level. As per QoS demand parameters of different workflows using the fuzzy clustering method, these are balanced. It has been shown via simulation outcomes that there are some benefits of the new scheme in decreasing the completion time of the workflow; high user satisfaction and higher success rates are also achieved using this scheme.

3. METHODOLOGY

Effective resource utilization while managing the load between the resources to decrease the execution time and increase the efficacy of workload is the main objective of cloud computing. This section discusses the PCP, robustness and proposed fuzzy scheduling.

3.1 PARTIAL CRITICAL PATH

In workflow scheduling, critical path heuristics have been used widely. The critical route of a workflow is the longest execution path between the workload's entry and exit tasks. Mostly, those tasks that belong to the critical paths are first scheduled by these heuristics that allocate resources to them so that they are processed first. This helps in minimizing the execution time of the entire workflow. Based on a similar heuristic, the proposed algorithm would first schedule the critical nodes while minimizing the cost of carrying out the critical path while satisfying user deadlines and maintaining execution time. Every critical node after being scheduled has a start time or deadline for its direct predecessors in the workflow known as parent nodes. Thus, by considering every critical node as an exit node with its start time as the deadline, it can carry out the same procedure; this leads to the creation of PCP that ends the critical node leading to a node already scheduled. This task recursively goes on in the PCP algorithm until all of the tasks are successfully scheduled [20].

The Critical Parent (CP) t_j is the parent t_p, whose sum of start time, data transfer time and execution time is maximum among other parent nodes. The PCP of the node t_j is a collection of tasks for which there is a high degree of dependence. By detecting the unallocated parents, PCP is found. Unallocated parents are nodes not allocated to any PCP. Furthermore, by finding the unallocated critical parent of the node, PCP is created and this is recursively done until there are no more unallocated parents. This procedure optimizes time and cost by identifying the PCP-PCPs that can be scheduled on a single resource. Workflows are divided by this algorithm into smaller groups of tasks that help to schedule. The PCPs of the workflows are mutually exclusive meaning, only one PCP is allocated to one task [21].

The VM having the most robustness type is selected for every PCP. The quantity of clack that will be added to the PCP execution time is the robustness type. The extent of execution time variation tolerated by a PCP is dictated by it. There are four robustness types in PCP-1) No robustness: this will refrain from adding any slack time to the PCP's execution time. 2) Slack: This adds a predetermined amount of time to the PCP execution time and allows for execution time variances up to a particular point. 3) One node failure: This adds the largest execution time of the PCP nodes to the execution time of the PCP and it gives enough slack time for handling the failure of the task having the largest execution time in PCP. 4) Two Node Failure: in this case, the PCP execution time is increased by the execution times of the two largest nodes; Only when there are three nodes comprised in a PCP this is done. Up to two disk failures can be tolerated by PCP with this robustness type. Upto two node failures are tolerated by four robustness types; it is also possible to develop robustness types with a higher amount of node failures.

3.1.1 PCP Algorithm:

Allocate Resources (PCP)

//Allocate a suitable robust resource to the PCP

Input : PCP

Output : Robust Resource for PCP

//Create Solution Set SS;

for Every Instance type do

for Every Robustness type do

Create Solution set with PCP_t and PCP_c

FS = null;

Calculate PCP_b according to equation 6;

//Create a Feasible Solution Set FS;

for Every solution in SS do

time = PCP_t + TopLevel + BottomLevel;

if time <=D and PCP_c <= PCP_b then

Add to FS

//finds the next solution according to the chosen policy

RobustResource = findBestSolution(FS, Policy);

Assign every task in PCP to the RobustResource

The above resource allocation algorithm explains the process of VM selection which attains a robust result. The complete solution set $SS = \{s_1,s_2,\ldots,s_{m*l}\}$ is produced in which m and l are the number of VM types and robustness types respectively. This solution set SS includes all robustness types for every VM type defined. Each solution, $s_i=\{vt_i,RT_i,PCPc_i,PCPt_i\}$ includes a robustness type (RT_i) PCP cost ($PCPc_i$) and PCP execution time ($PCPt_i$) for VM type vt_i. As m and l are generally not large, the time and space required are reasonable.

The solution set *SS* is lowered depending on the deadline and budget restrictions into a smaller set of possible solutions. The deadline constraint D is measured by merging the selected PCP execution time instance and robustness type with the top and bottom levels as shown in Eq.(1).

$$TopLevel+PCP_t+BottomLevel\leq D \quad (1)$$

Here, the sum of execution times of nodes on the longest path from the entry to the first node and from the end node of PCP to the exit node is the top level and bottom level of PCP. Budget Constraint is evaluated by the following Eq.(2):

$$PCP_c\leq PCP_b \quad (2)$$

where PCP_c represents the total cost of the PCP. PCP Budget, PCP_b, means the amount spent on the PCP. This can be decomposed from the overall budget based on Eq.(3):

$$PCP_b=(PCP_t|TT)*B \quad (3)$$

where TT represents the workflow's total time is measured by adding the execution times of the tasks on the reference VM type, $vt_{ref.}$ VM with the minimum MIPS value is taken as the reference type, $vt_{ref.}$ PCP_t which means that the total execution time of the PCP is on $vt_{ref.}$ When PCP_b is less than LPr, which is the price is essential to perform on the cheapest resource, then PCP_b is assigned the value LPr.

Using these two constraints as given in the allocate resource algorithm, a feasible solution set FS is generated. The appropriate VM type vt_i for a PCP is chosen using the findBestSolution,

method described in Algorithm 2, based on the resource selection policy from the feasible solution set FS.

3.2 ROBUSTNESS SCHEDULING

A schedule that is not affected by the workflow processing time disturbances is referred to as a robust schedule. The measure of the degree of "insensitiveness" is given by the robustness of the schedule. This is also defined as a linear mixture of assumed makespan and delay as per one of the first attempts in formalizing the definition of schedule robustness. Nonetheless, it limits the applicability as the definition combines the idea of robustness with optimization criteria of makespan minimization. Although the empirical formula for measuring robustness has been designed by the authors as an objective function to be optimized, there is no way to evaluate the schedule's robustness. It assumed that the schedule's robustness must show the stability of the actual makespan concerning the expected one. Both the expected makespan and the robustness must be considered by the overall performance of a schedule. In this regard, two definitions are proposed [22]:

1. **Definition 1**: Let $M_0(s)$ denote the expected makespan of schedule s obtained with expected workflow execution time and $M_i(s)$ the real makespan with i^{th} realization of expected workflow execution times. The relative schedule tardiness is:

$$\delta_i(s) = \frac{\max(0, M_i(s) - M_0(s))}{M_0(s)} \tag{4}$$

The first definition of the robustness of schedules is Eq.(5):

$$R_1(s) = \frac{1}{E(\delta_i(s))} \tag{5}$$

where $E(\cdot)$ represents the expectation operator.

2. **Definition 2**: $M_0(s)$ and $M_i(s)$ are defined as above. N realizations of the expected workflow execution times are performed. Let $M=\{M_i(s)|M_i(s)>M_0(s)\}$. The schedule miss rate is $\alpha(s)=(|M|)/N$. Then, the second definition of the robustness of schedules is Eq.(6):

$$R_2(s) = \frac{1}{\alpha(s)} \tag{6}$$

3.3 PROPOSED FUZZY SCHEDULING

The performance of the majority of useful scheduling methods in heterogeneous systems is compared to their computational complexity. There should be appropriate and timed execution of tasks in real time. Identifying minimal scheduling time in real-time constrained multiprocessor systems is an NP-hard problem. There is no deterministic response time in real-time scheduling algorithms. For the analysis of the robustness of a system, determining the timing behavior is crucial. The complexities of the scheduling problems are increased due to the inherent uncertainties in dynamic real-time systems. For arranging real-time periodic and non-periodic tasks in the systems, a fuzzy scheduling approach has been employed to alleviate these issues. When non-critical overload condition occurs, static and dynamic optimal scheduling algorithms fail. To improve the performance of heterogeneous systems, knowledge-based algorithms can be devised [23].

Fuzzy logic is employed by the suggested fuzzy controller. This was introduced in 1965 by Zadeh. There is no strict allocation of elements to sets like binary in fuzzy logic. In its place, each element has a degree of membership to a set which is denoted between 0 and 1. A fuzzy system needs to be constructed for applying the fuzzy logic to specific issues like scheduling between cloudlets or VMs. There are three steps for the construction of a fuzzy system- Fuzzification: here the fuzzy sets are assigned with the degree of membership of input value. The degree of membership is given by μ: X→ [0, 1], where X is the set of input values. As a result, every input value is converted to a value between 0 and 1. Inference engine: According to the rule sets, this system maps input regions to output regions using rules. De-Fuzzification: a numerical output value is generated from the output set [24].

Fuzzy set theory is responsible for representing uncertainties is, it admits circumstances either moderately true or false. Fuzzy logic is effective for treating random uncertainty where it is not possible to predict a sequence of events. A collection of fuzzy rules in the fuzzy control system illustrates a control decision mechanism to modify the consequences of certain systemic causes. The objective is to replace a skilled human operator with a fuzzy rule-based system. An online decision for adapting the system behavior to ensure optimality in some cases is determined by the current state of a network-based inference engine comprising a fuzzy rule base [25].

There are a series of steps for designing the fuzzy control step. The first is defining the input and the control variables. It is necessary to quantify each variable. For every variable quantified, it is assigned to a membership function. This necessitates creating a fuzzy rule basis that establishes which control action ought to occur under various input circumstances. The rules are written in the if-then form. For evaluating the individual if-then rules in the rule base, an implication formula is used. For aggregating the rule outcomes so that a fuzzy output set is obtained, a composition rule is employed. Mamdani minimum inference method has been used in the suggested fuzzy system as a fuzzy inference technique.

The input of the fuzzy is security and energy. The security considers the trust computation model. Direct trust [26]:

Let $\phi_{i,j}(t)$ represent the total number of historical trust feedbacks of Cloud Service Users (CSU) CSU_i for CSP_j through window t, t=1,2,…W, where $\sum_{t=1}^{W}\phi_{i,j}(t) = \phi_{i,j}$. Let a trust feedback rating be represented as $f_{i,j}$, $0 \le f_{i,j} \le 1$ in which 0 and 1 means untrustworthy and trustworthy respectively. Let $f_{i,j}$ specify the average trust feedback ratings of CSU_i for CSP_j during window t.

$$sm_{i,j}^{t}\{T\} = \begin{cases} \dfrac{f_{i,j}^{(-t)} - 0.5}{0.5}, & \text{if } f_{i,j}^{(-t)} \in [0.5,1] \\ 0, & \text{if } f_{i,j}^{(-t)} \in [0,0.5) \end{cases}$$

$$sm_{i,j}^{t}\{-T\} = \begin{cases} \dfrac{0.5 - f_{i,j}^{(-t)}}{0.5}, & \text{if } f_{i,j}^{(-t)} \in [0,0.5] \\ 0, & \text{if } f_{i,j}^{(-t)} \in (0.5,1] \end{cases}$$

$$sm_{i,j}^{t}\{U\} = 1 - sm_{i,j}^{t}\{T\} - sm_{i,j}^{t}\{-T\} \tag{7}$$

Assume that if $f_{i,j}^{(-t)}$ is 0.5 then the assessment has the highest ambiguity equal to 1; if $f_{i,j}^{(-t)} < 0$ or $f_{i,j}^{(-t)} > 0.5$, then the estimation consists of trust and uncertainty. It alters the $f_{i,j}^{(-t)}$ values to the possibilities that the subjective trust evidence of CSU_i for CSP_j in time window t is trustworthy $m_{i,j}^t\{T$, untrustworthy $m_{i,j}^t\{-T\}$, and indeterminate $m_{i,j}^t\{U\}$, as in Eq.(7):

The local subjective trust value of CSU_i for CSP_j in time window t is represented as in Eq.(8):

$$LST_{i,j}^T = (lsm_{i,j}^t\{T\}, lsm_{i,j}^t\{-T\}, lsm_{i,j}^t\{U\}) \tag{8}$$

where, $lsm_{i,j}^t\{T\}, lsm_{i,j}^t\{-T\}, lsm_{i,j}^t\{U\}$ represents the possibility of the local subjective trustworthiness, untrustworthiness, and local subjective uncertainty, respectively, are measured as in Eq.(9):

$$LST_{i,j}^t = \begin{cases} lsm_{i,j}^t\{T\} = \mu \times sm_{i,j}^{t-1}(T) + (1-\mu) \times \dfrac{\overline{f}_{i,j}^t - 0.5}{0.5} \\ lsm_{i,j}^t\{-T\} = \mu \times sm_{i,j}^{t-1}(-T) + (1-\mu) \times \dfrac{0.5 - \overline{f}_{i,j}^t}{0.5} \\ lsm_{i,j}^t\{U\} = 1 - sm_{i,j}^{t-1}(T) - sm_{i,j}^{t-1}(-T) \end{cases} \tag{9}$$

where $0 \leq \mu \leq 1$ is a weight factor.

For t=0, it is set $LST_{i,j}^0 = (0.5, 0, 1)$.

Indirect trust is the belief that one entity holds on another entity in a particular context depending on the references obtained from its peer entities' experience with that entity. The indirect trust of platform A about platform B is measured by merging satisfaction and certification opinion of the recommenders on the platform belongings. $A^{\text{ind}}O_{B,(c_i,p_j)}$ or $A^{\text{ind}}O_{B,(p_k)}$ or $A^{\text{ind}}O_{B,(c,p_l)}$ is the complete recommended opinions concerning various types of properties of platform A on B. The complete opinion is measured from the individual opinions of A's recommenders using a consensus $\oplus$ operator. The recommender opinions based on the service platform B are discounted (using a discounting ($\oplus$) operator) based on A's opinion (positive or negative experience) on the recommender in Eq.(10)-Eq.(12).

$$^{A-ind}O_{B,(c_i,p_j)} = (^{A}O_{R_1} \otimes {}^{R_1}O_{B,(c_i,p_j)}) \oplus \ldots \ldots \oplus (^{A}O_{R_m} \otimes {}^{R_m}O_{B,(c_i,p_j)}) \tag{10}$$

$$^{A-ind}O_{B,(p_k)} = (^{A}O_{R_1} \otimes {}^{R_1}O_{B,(p_k)}) \oplus \ldots \ldots \oplus (^{A}O_{R_m} \otimes {}^{R_q}O_{B,(p_k)}) \tag{11}$$

$$^{A-ind}O_{B,(c,p_l)} = (^{A}O_{R_1} \otimes {}^{R_1}O_{B,(c,p_l)}) \oplus \ldots \ldots \oplus (^{A}O_{R_m} \otimes {}^{R_s}O_{B,(c,p_l)}) \tag{12}$$

3.4 ENERGY MODEL

There are two consumptions- static energy consumption and dynamic energy consumption in the power consumption for application execution denoted respectively by E_{static} and $E_{dynamic}$ The static consumption of energy has been ignored here as the dynamic consumption of energy is more significant [27]. The dynamic power dissipation P_k^j of VM type vm_k in the voltage level j is described in Eq.(13).

$$P_k^j = \lambda_k . (v_k^j)^2 . f_k^j \tag{13}$$

where the constant parameter λ_k is associated with the dynamic power based on VM type and capacity v_k^j. This means that the supply voltage at level j on the VM of type k and parameter f_k^j is the frequency with matching v_k^j. The parameter f_k^j and v_k^j are in amount of the computing capacity p_k^j and hence they are in the range $f_k^{\min}$, $f_k^{\max}$ and $v_k^{\min}$, $v_k^{\max}$ respectively.

The energy consumption in runtime $t_{runtime}$ is measured as the Eq.(14).

$$E_k^j = P_k^j . t_{runtime} = \lambda_k . (v_k^j)^2 . f_k^j . t_{runtime} \tag{14}$$

The product of power and time is referred to as energy. Here, execution time and frequency are inversely proportional. Hence, the power needed is proportional to the square of the voltage. Thus, energy consumed can be effectively decreased by decreasing the voltage. For instance, a seventy percent decrease in voltage leads to a fifty percent decrease in power required. Due to their inability to scale to zero, the supply voltage and frequency stay in their lowest voltage condition for the greatest energy savings. Hence, the energy consumption of idle time is defined by Eq.(15):

$$E_{idle} = \lambda_k . (v_k^{\min})^2 . f_k^{\min} . t_{idletime} \tag{15}$$

The total energy consumption of a VM instance is represented by the Eq.(16).

$$E_{total} = E_k^j + E_{idle} \tag{16}$$

4. RESULTS AND DISCUSSION

In this section, the PCP, robustness cost time and proposed fuzzy scheduling methods are used. Experiments are carried out using a 0.1 to 1 energy factor. The mean of tolerance time and mean of makespan as shown in tables 1 and 2 and figures 1 and 2.

Table.1. Mean of tolerance time

Energy Factor	PCP	Robustness cost time	Proposed Fuzzy Scheduling
0.1	24	25	26
0.2	120	125	129
0.3	160	165	170
0.4	185	191	198
0.5	232	241	250
0.6	245	254	263
0.7	290	301	311
0.8	365	377	390
0.9	445	459	475
1	634	658	683

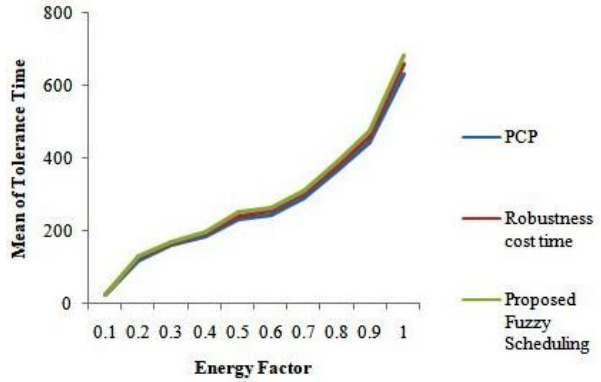

Fig.1. Mean of tolerance time for proposed fuzzy scheduling

From Fig.1, it can be observed that the proposed fuzzy scheduling has a higher mean tolerance time of 6.97% for PCP and 3.47% for robustness cost time.

Table.2. Mean of makespan

Energy Factor	PCP	Robustness cost time	Proposed Fuzzy Scheduling
0.1	332	320	296
0.2	348	336	321
0.3	362	337	311
0.4	374	345	325
0.5	376	355	344
0.6	382	363	346
0.7	398	383	359
0.8	442	420	390
0.9	448	433	399
1	490	468	452

From Fig.2, it can be observed that the proposed fuzzy scheduling has a higher mean of makespan by 10.91% for PCP and by 5.94% for robustness cost time.

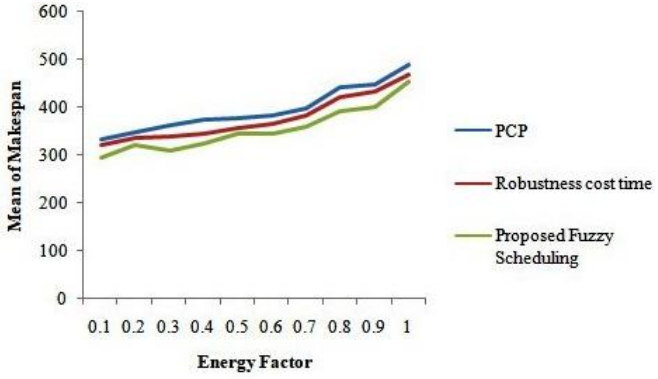

Fig.2. Mean of makespan for proposed fuzzy scheduling

5. CONCLUSION

A set of characteristics that are ideal for cloud infrastructure execution are provided by scientific workflows; they can scale the resources based on the requirements of the application. In terms of scheduling overhead time, the PCP schedulers achieve effective outcomes. Robust scheduling algorithms should comprise resource allocation policies for scheduling workflow tasks on heterogeneous cloud resources while minimizing the makespan and the cost. A new approach for scheduling algorithms on the cloud known as the fuzzy scheduling technique has been proposed in this work. This technique allocates resources to the VM ID in such a way that the least completion time is taken. Results show that the proposed fuzzy scheduling has a higher mean of makespan by 10.91% for PCP and by 5.94% for robustness cost time.

REFERENCES

[1] M.A. Rodriguez and R. Buyya, "Deadline-based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds", *IEEE Transactions on Cloud Computing*, Vol. 2, No. 2, pp. 222-235, 2018.

[2] F. Fakhfakh, H.H. Kacem and A.H. Kacem, "Workflow Scheduling in Cloud Computing: A Survey", *Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, pp. 372-378, 2018.

[3] C. Lin and S. Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing", *Proceedings of International Conference on Cloud Computing*, pp. 746-757, 2019.

[4] P. Thakur and M. Mahajan, "Different Scheduling Algorithm in Cloud Computing: A Survey", *International Journal of Modern Education and Computer Science*, Vol. 5, No. 2, pp. 45-58, 2017.

[5] J. Elayaraja and S. Dhanasekar, "A Survey on Workflow Scheduling in Cloud using Ant Colony Optimization", *International Journal of Computer Science and Mobile Computing*, Vol. 3, No. 4, pp. 39-44, 2018.

[6] M. Masdari, S. ValiKardan, Z. Shahi and S.I. Azar, "Towards Workflow Scheduling in Cloud Computing: A Comprehensive Analysis", *Journal of Network and Computer Applications*, Vol. 66, No. 4, pp. 64-82, 2016.

[7] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski and R. Sakellariou, "Energy-Constrained Provisioning for Scientific Workflow Ensembles", *Proceedings of International Conference on Cloud and Green Computing*, pp. 34-41, 2018.

[8] X. Li, W. Hu, T. Ding and R. Ruiz, "Trust Constrained Workflow Scheduling in Cloud Computing", *Proceedings of International Conference on Systems, Man and Cybernetics*, pp. 164-169, 2017.

[9] M. Zhu, Q. Wu and Y. Zhao, "A Cost-Effective Scheduling Algorithm for Scientific Workflows in Clouds", *Proceedings of International Conference on Performance Computing and Communications Conference*, pp. 256-265, 2018.

[10] Z. Cai, X. Li and J.N. Gupta, "Critical Path-based Iterative Heuristic for Workflow Scheduling in Utility and Cloud Computing", *Proceedings of International Conference on Service-Oriented Computing*, pp. 207-221, 2020.

[11] E.N. Alkhanak and S.P. Lee, "A Hyper-Heuristic Cost Optimisation Approach for Scientific Workflow Scheduling in Cloud Computing", *Future Generation Computer Systems*, Vol. 86, No. 8, pp. 66-78, 2018.

[12] N. Mohanapriya, G. Kousalya, P. Balakrishnan and C. Pethuru Raj, "Energy Efficient Workflow Scheduling with Virtual Machine Consolidation for Green Cloud Computing", *Journal of Intelligent and Fuzzy Systems*, Vol. 34, No. 3, 1561-1572, 2018.

[13] S. Pandey, L. Wu, S.M. Guru and R. Buyya, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", *Proceedings of International Conference on Advanced Information Networking and Applications*, pp. 400-407, 2019.

[14] S. Bilgaiyan, S. Sagnika and M. Das, "Workflow Scheduling in Cloud Computing Environment using Cat Swarm Optimization", *Proceedings of International Conference on Advance Computing*, pp. 680-685, 2017.

[15] S. Sagnika, S. Bilgaiyan and B.S.P. Mishra, "Workflow Scheduling in Cloud Computing Environment using Bat Algorithm", *Proceedings of International Conference on Smart System, Innovations and Computing*, pp. 149-163, 2020.

[16] A. Verma and S. Kaushal, "A Hybrid Multi-Objective Particle Swarm Optimization for Scientific Workflow Scheduling", *Journal of Parallel Computing*, Vol. 62, No. 6, pp. 1-19, 2017.

[17] L. Liu, M. Zhang, R. Buyya and Q. Fan, "Deadline-Constrained Coevolutionary Genetic Algorithm for Scientific Workflow Scheduling in Cloud Computing", *Concurrency and Computation Practice and Experience*, Vol. 8, No.4, pp. 1-12, 2018.

[18] F. Guo, L. Yu, S. Tian and J. Yu, "A Workflow Task Scheduling Algorithm based on the Resources' Fuzzy Clustering in Cloud Computing Environment", *International Journal of Communication Systems*, Vol. 28, No. 6, pp. 1053-1067, 2023.

[19] W. Li, J. Wu, Q. Zhang, K. Hu and J. Li, "Trust-Driven and QoS Demand Clustering Analysis based Cloud Workflow Scheduling Strategies", *Journal of Cluster Computing*, Vol. 17, No. 7, pp. 1013-1030, 2023.

[20] S. Abrishami, M. Naghibzadeh and D.H. Epema, "Cost-Driven Scheduling of Grid Workflows using Partial Critical Paths", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 5, pp. 1400-1414, 2022.

[21] D. Poola, S.K. Garg, R. Buyya, Y. Yang and K. Ramamohanarao, "Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds", *Proceedings of International Conference on Advanced Information Networking and Applications*, pp. 858-865, 2017.

[22] Z. Shi, E. Jeannot and J.J. Dongarra, "Robust Task Scheduling in Non-Deterministic Heterogeneous Computing Systems", *Proceedings of International Conference on Cluster Computing*, pp. 1-10, 2018.

[23] M.R.A. Kulkarni, S.H. Patil and N. Balaji, "Fuzzy Real Time Scheduling on Distributed Systems to Meet the Deadline of Applications", *International Journal of New Technology and Research*, Vol. 2, No. 4, pp. 56-58, 2016.

[24] S.S. Devi and G. Muthulakshmi, "An Enhanced Optimization Technique for Scheduling in Cloud based Applications", *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, Vol. 3, No. 3, pp. 918-925, 2018.

[25] A. Mehranzadeh and S.M. Hashemi, "A Novel-Scheduling Algorithm for Cloud Computing based on Fuzzy Logic", *International Journal of Applied Information Systems*, Vol. 5, No. 7, pp. 58-68, 2018.

[26] W. Fan and H. Perros, "A Novel Trust Management Framework for Multi-Cloud Environments based on Trust Service Providers", *Knowledge-based Systems*, Vol. 70, No.10, pp. 392-406, 2014.

[27] Z. Li, J. Ge, H. Hu, W. Song, H. Hu and B. Luo, "Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds", *IEEE Transactions on Services Computing*, Vol. 10, No. 4, pp. 113-120, 2021.