Pi International Journal of Mathematical Sciences
© PIJMS | Volume 1 | Issue 3 | December 2025 | ISSN: 3107-9830 (Online)
Available online at: https://pijms.com/
DOI: https://doi.org/10.59828/pijms.v1i3.14

SYSTEM COST AND RELIABILITY OPTIMIZATION THROUGH FUZZY-BASED TASK SCHEDULING IN DISTRIBUTED COMPUTING SYSTEMS

Soniya Saini [1], Dr. Sandeep Kumar Tiwari[2], Dr. Ankur Nehra[3]

[1]Research Scholar, Department of Mathematics, Faculty of Science, Motherhood University, Haridwar, Uttarakhand, 24766, India

Email ID: soniyasaini599@gmail.com

[2]Supervisor, Department of Mathematics, Faculty of Science, Motherhood University, Roorkee, Uttarakhand, 247667, India

Email ID: fos.sandeep@motherhooduniversity.edu.in

[3]Co-supervisor, Department of Mathematics, Dhanauri P.G. College, Dhanauri, Haridwar, Uttarakhand, 247667, India

Email ID: drankurnehra648@gmail.com

Corresponding Author: Soniya Saini, Research Scholar (soniyasaini599@gmail.com)

Received: 27 November 2025 | **Accepted:** 24 December 2025 | **Published:** 30 December 2025

ABSTRACT

Efficient task scheduling in distributed and cloud-fog computing environments plays a pivotal role in optimizing system performance, reliability, and operational costs. This study presents a comprehensive task scheduling model that integrates cost-aware and reliability-driven strategies to enhance resource utilization while minimizing system downtime and overall execution expense. By incorporating dynamic workload distribution, fuzzy-logic-based decision-making, and hybrid metaheuristic approaches, the proposed model addresses uncertainties and heterogeneity in modern computing systems. Extensive simulations and performance analyses demonstrate that the model achieves significant improvements in system reliability and cost reduction compared to conventional scheduling techniques. The findings offer a practical framework for designing adaptive and efficient scheduling mechanisms in cloud, fog, and IoT-enabled environments.

Keywords: *Task Scheduling, System Reliability, Cost Optimization, Cloud Computing, Fog Computing, IoT, Fuzzy Logic, Hybrid Metaheuristics, Resource Allocation.*

INTRODUCTION

A collection of processors linked by a quick network allows multiple programs to run simultaneously in a distributed real-time system. The scheduling of activities onto the processors, which is frequently NP-complete, has a significant impact on the performance of parallel applications on DRTS. There are two primary types of DRTS job scheduling challenges: static and dynamic. In a dynamic model, scheduling decisions are made at runtime, in contrast to a static model where all the specifics, including the number of tasks and processors, task execution costs across processors, and intertask communication costs, are known beforehand. Static scheduling has been taken into consideration in this work because it enables the employment of complex scheduling algorithms to produce high-quality task scheduling. Numerous scholars have looked into job scheduling issues. When there are two processors, Stone (1977) and Bokhari (1979) have shown how the network flow technique can be used to effectively discover an optimal assignment. The optimal assignment is produced by Bokhari's (1981) shortest tree algorithm when the structure is limited to being a tree. The Bokhari result was extended to the case of several parallel structures by Towsley (1986). The best solutions for the general structure problem in a linear array network with an arbitrary number of processors were given by Lee (1992) and Cho and Park (1994). For heterogeneous DRTS, Ucar et al. (2006) created a task assignment model. The dynamic task scheduling problem for a large program structure in DRTS has been studied by Yadav et al. (2008). The longest dynamic critical route technique was introduced by Daoud and Kharma (2008) as a revolutionary high-performance scheduling strategy for heterogeneous DRTS with a limited number of processors. A task scheduling model was created by Yadav et al. (2011) to effectively schedule work onto the most appropriate processor in order to obtain the best possible outcome. In the process of developing multicore cluster systems, Yang (2014) looked into a novel job scheduling problem and took into account additive intra-node communication time. Data localization and skew efficiency on homogeneous and heterogeneous clusters were studied by Karger and Vakili (2015). In an effort to increase the lifespan of wireless sensor networks, Nayak (2016) created a clustering strategy based on fuzzy logic. Hussain et al. (2016) presented a multi-objective optimization-based solution to the resource scheduling difficulty in inference-limited machine-to-machine communication. In order to maximize the probability of achieving all deadlines, Pathan (2017) proposed a method to calculate the number of backups for each work that should run in the active redundancy. Konar et al. (2017) presented a hybrid quantum approach inspired by evolutionary algorithms for efficient real-time work scheduling. If the jobs are appropriately distributed among the right processors, taking into account the communication links and processor failure rates, a DRTS can execute a parallel program with outstanding dependability. The concept is straightforward: the most dependable processors should tackle operations with high execution costs, and the most dependable links should manage tasks with high communication costs. The majority of research is focused on improving the reliability of DRTS. Genetic algorithms were employed by Vidyarthi and Tripathi (2001) to increase system reliability. A heuristic approach was devised by Attiya and Hamam (2006) in order to optimize system reliability. A fuzzy Bayesian traction control system for rail trains equipped with speed sensors in intelligent transportation systems was created by Noori and Jenab (2013). A mathematical technique for work scheduling in distributed programs to maximize system reliability was presented by Kumar et al. (2016). Task scheduling approaches in cloud computing were developed by Arunarani, A. R., et al. (2019) using a literature study. Alizadeh, M. R., et al. are the authors of the fog computing work scheduling algorithms (2020): an extensive investigation. The autonomic task scheduling algorithm for dynamic workloads was created in 2021 by F. Ebadifard and S. M. Babamir using a load-balancing technique for cloud computing environments. Bezdan, T. et al. (2022)

created a hybridized bat algorithm for multi-objective job scheduling in the context of cloud computing. The decentralized, scalable hybrid scheduling-clustering technique was created by Hajvali, M., and colleagues in 2023 for use in unstable fog-cloud scenarios for real-time applications.

LITERATURE REVIEW

Task scheduling and resource allocation have long been recognized as critical challenges in distributed, parallel, and heterogeneous computing environments. Early foundational work by Stone (1977) introduced multiprocessor scheduling using network flow algorithms, establishing a mathematical basis for optimal task allocation. Bokhari (1979, 1981) extended this with dynamic reassignment and shortest-tree algorithms for optimal task mapping across space and time in distributed processor systems. These studies laid the groundwork for reliability- and cost-aware scheduling in multi-processor environments. Further theoretical advancements were made by Towsley (1986) and Lee et al. (1992), who addressed task allocation in multiprocessor and linear array networks under branching and looping constraints. Cho and Park (1994) and Vidyarthy and Tripathi (2001) later introduced dynamic and genetic-algorithm-based task assignment strategies to enhance reliability in heterogeneous systems. These methods emphasized reducing inter-task communication and maximizing processor utilization. With the rise of heterogeneous distributed systems, performance optimization became a key focus. Daoud and Kharma (2008) proposed high-performance static scheduling for heterogeneous systems, while Ucar et al. (2006) explored task assignment techniques with emphasis on load balancing. Reliability-driven approaches using heuristics and metaheuristics, such as simulated annealing and genetic algorithms, were introduced by Attiya and Hamam (2006) and Yadav et al. (2008, 2011), highlighting the importance of minimizing communication overhead and optimizing processor usage. The integration of fuzzy logic for scheduling and clustering became prominent due to its ability to handle uncertainty in dynamic environments. Noori and Janab (2013) presented fuzzy reliability-based models for intelligent transportation systems, while Nayak (2016) proposed a fuzzy clustering algorithm to extend wireless sensor network lifetime. Similarly, Kumar et al. (2016) introduced fuzzy-based dynamic task scheduling algorithms for distributed systems, enhancing adaptability under fluctuating workloads. As cloud computing matured, research focused on scalable and efficient task scheduling. Arunarani et al. (2019) and Alizadeh et al. (2020) provided comprehensive surveys of cloud and fog task scheduling techniques, identifying gaps in scalability, latency, and energy efficiency. Advanced hybrid and metaheuristic approaches were proposed by Ben Alla et al. (2018) and Bezdan et al. (2022), achieving improved multi-objective optimization in cloud environments. Recent studies emphasize fog-cloud and IoT-based real-time systems, reflecting the need for low-latency, mobility-aware scheduling. Ebadifard and Babamir (2021) proposed autonomic task scheduling with load balancing for dynamic workloads, while Hajvali et al. (2023) introduced decentralized hybrid scheduling-clustering methods for volatile fog-cloud environments. The latest contribution by Ali and Sridevi (2024) integrates mobility-and security-aware fuzzy-logic-based scheduling for IoT devices, directly addressing real-time constraints and enhanced system performance. These approaches indicate a trend toward intelligent, adaptive, and secure scheduling frameworks suitable for emerging cyber-physical systems. Additionally, real-time and safety-critical scheduling has been advanced by Pathan (2017) and Konar et al. (2017) using fault-aware and hybrid quantum-inspired genetic algorithms. Multi-objective resource allocation for M2M and MapReduce environments has been explored by Hussain et al. (2016) and Karger and Vakili (2015), reinforcing the importance of efficient load balancing in large-scale distributed systems. Overall, the literature demonstrates a clear evolution from deterministic and static scheduling models to fuzzy, heuristic, and metaheuristic-based intelligent approaches. By 2025, research trends indicate an increasing focus on fog-cloud integration, IoT systems, security, mobility-awareness, energy efficiency, and real-time adaptability, pointing to the need for unified frameworks capable of addressing multiple constraints simultaneously. Despite these advances, challenges remain in balancing performance, reliability, and scalability in dynamic, heterogeneous, and IoT-enabled environments, motivating further research in this field.

TASKS SCHEDULING PROBLEM

The primary focus of this discussion is the challenge of task scheduling onto processors in a linear array network, which aims to maximize system dependability and cost "n" linearly ordered processors; in other words, intermediary processors must participate in communication between any two processors that are not nearby. The distance $\alpha_{a,x} = \sum_{y=a}^{x-1} \alpha_{y,y+1}$ separates two non-adjacent processors, c_a and c_l, if $\alpha_{a,b}$ is the distance between c_a and c_b. This means that for each unit of information exchanged between any two processors, the cost of communication grows linearly with increasing distance. Each CPU has its own memory and processing power, but there is a certain amount of capacity available to the communication network. Every CPU in the system has a certain failure rate. For every processor in DRTS, there are two possible states: operational and fault. Every processor and path must be operational during the task processing phase and the active data connection time between the path's terminal processors in order for a parallel application to execute properly on DRTS. Consequently, the correct execution of the program depends critically on the assignment of tasks to the right processors and the dependability of system components. Set C of "g" processors receives set S of "f" jobs after some assignment "A." The binary matrix $\left[s_{y,a}\right]$ of order $f \times g$ can be created from the assignment "A," where the element $s_{y,a}$ is set to 1 if the task h_y is assigned to the processor c_a and to 0 otherwise.

System Costs

The imprecise execution expense is the quantity of data required for the job h_y to be executed on the processor c_a is $j_{y,a}$. Applying the following formulas will yield the overall fuzzy execution cost for assignment "A":

$$EXEC(A) = \sum_{y=1}^{f} \sum_{a=1}^{g} j_{y,a} * s_{y,a} . \qquad (1)$$

It is assumed that the inter-task communication cost per unit of information sent for unit distance has a constant value of w, independent of the processors involved in the communication. It is possible to assume $w = 1$ without losing generality. Thus, the inter-task communication cost $\tilde{p}_{y,o} * \alpha_{a,x}$ is caused by the two interacting tasks, h_y and h_o, if they are allocated to separate processors c_a and c_x, respectively. Tasks h_y and h_o have no inter-task communication if they are both assigned to processor c_a, as indicated by $\alpha_{a,a} = 0$. For assignment "A," the overall fuzzy inter-task communication cost is computed as follows:

$$COMM(A) = \sum_{\substack{y=1 \\ y \neq o}}^{f} w_{y,o} * s_{y,a} * s_{o,x} * \alpha_{a,x} \qquad (2)$$

The total cost $TCOST(A)$ of the system for an assignment 'A' is calculated as:

$$TCOST(A) = EXEC(A) + COMM(A) \qquad (3)$$

$$= \sum_{y=1}^{f} \sum_{q=1}^{g} j_{o,q} * s_{y,q} + \sum_{\substack{y=1 \\ y \neq o}}^{f} \tilde{s}_{y,o} * s_{y,q} * s_{o,x} * \alpha_{q,x}$$

The task scheduling issue can therefore be resolved by determining which assignment "A" has the lowest total cost.

Processor Reliability

The probability that the processor will remain functional throughout the completion of all tasks assigned to it is represented by the letter Z_a, which stands for the reliability of the q^{th} processor. If the failure rate of the q^{th} processor is γ_q over a time interval $[0,\ t]$, then the processor's reliability is $\exp(-\gamma_a.h)$. The following equation can be used to determine the processor reliability if, for any task assignment, A," $\sum_{y=1}^{f} \tilde{j}_{y,a} * s_{y,a}$ represents the entire cost needed to complete all of the tasks allocated to AAA, the processor:

$$R_k = \exp\left(-\lambda_k * \sum_{i=1}^{m} \tilde{e}_{i,k} * x_{i,k}\right). \qquad (4)$$

Path Reliability

The probability that the path ax will operate to transfer data between the path's terminal processors is known as path reliability, and it is symbolized by the letter Z_{ax}. An arrangement of links that joins a sender and a recipient is called a communication path. A path's dependability is $\exp(-\rho_{ax}.h)$ if its failure rate ax, is expressed as ρ_{ax} over a time interval $[0,\ t]$. If, for any job assignment, A," $\sum_{\substack{y=1 \\ y \neq o}}^{f} \tilde{s}_{yo} * s_{yq} * s_{ox} * \alpha_{ax}$ represents the total cost required for data transfer between the terminal processors of the path ax, then the path dependability is calculated as follows:

$$Z_{ax} = \exp\left(-\beta_{a,x} * \sum_{\substack{y=1 \\ y \neq o}}^{f} w_{y,o} * s_{y,a} * s_{o,x} * \alpha_{a,x}\right). \qquad (5)$$

System Reliability

The known system reliability, denoted by the letter $Z_{\phi\tau\phi}$, is the likelihood that the system will be able to carry out the complete program correctly. Stated differently, system reliability is the product of path reliability and processor reliability and can be computed as follows:

$$Z_{\phi\tau\phi} = \left[\prod_{a=1}^{g} Z_a\right] * \left[\prod_{a=1}^{g-1} \prod_{x>a} Z_{ax}\right]. \qquad (6)$$

PROPOSED MODEL

In this part, a new heuristic model for handling a task scheduling problem has been developed. Some of the key steps in this approach are defuzzification, cluster construction, task cluster scheduling onto processors, and system cost and reliability calculations. Defuzzification is the first step in the model. Numerous defuzzification methods have been reported in the literature; in this instance, Robust's ranking algorithm is used to transform fuzzy costs into crisp costs. Several defuzzification techniques have been documented in the literature; here, fuzzy costs are converted to crisp costs using Robust's ranking algorithm. The second stage of the model is the formation of work clusters. When processes are executing on different processors and interacting with each other, inter-task communication costs occur. To reduce intertask communication expenses, tasks that need to be communicated often are clustered together. Because it would be difficult to set up a load balancing schedule if the tasks cluster that arises from merging jobs grew too large, there is a limit on how many tasks can be in a cluster defined by $g_w = \left\lceil \frac{f}{g} \right\rceil$. While they are being operated, these clusters will remain fixed. In order to establish a one-to-one relationship between task clusters and processors, the current paradigm sets the number of task clusters equal to the number of processors.

Determining each task pair's membership value based on their communication, which occurs inside the unit interval $[0,1]$, is the aim of the membership function.

$$\rho(s_{yo}) = \frac{s_{yo}}{\max\{s_{yo} : i, j = 1, 2, ..., f\}}.$$

Task clusters that are particularly helpful in handling complex situations are created using linguistic characteristics. Based on their membership values, tasks are divided into the following five linguistic variables.

- Very highly communicating task $(CT_{V.H})$ (00.850, 01.000]
- Highly communicating task (CT_H) (00.650, 00.850]
- Average communicating task (CT_{Avg}) (00.450, 00.650]
- Less communicating task (CT_L) (00.250, 00.450]
- Very less communication tasks $(CT_{V.L})$ [00.000, 00.250]

The final stage of the approach involves allocating work clusters to the right processors in order to maximize system expenditures and improve system reliability. System expenditures are the primary element influencing system reliability. The system will be extremely trustworthy if it is economical, and vice versa. The suggested heuristic model's process is outlined in the following algorithm.

Algo: The algorithm used to create task clusters and schedule them among processors

Step 1: Input f, g, $FECM = [j_{y,a}]$, $FITCCM = [\tilde{s}_{y,o}]$, $CLRFM = [\rho_{s,\tau}]$, $PDM = [\alpha_{s,\tau}]$ and γ_o.

Step 2: Defuzzify the fuzzy costs $j_{y,a}$ and $\tilde{s}_{y,o}$ to crisp costs $c_{y,o}$ and $j_{y,a}$, respectively, then store the costs as matrices $\left[j_{y,a}\right]$ and $\left[c_{i,j}\right]$.

Step 3: Calculate g_s.

Step 4: Determine the membership values $\rho\left(w_{y,o}\right)$, where $y,o = 1,2,...,m$ they are present, and record these values in a matrix $\rho\left(w_{y,o}\right)$.

Step 5: Define the category of each task pair into linguistic variables and store them in the form of a matrix TPLV.

Step 6: Treat each task as a distinct cluster.

Step 7: Establish task pair priorities

$VHCT \to HCT \to ACT \to LCT \to VLCT$ to group the tasks.

Step 8: While (the number of task clusters divided by n) do

Choose the two more important clusters (for example, Cluster(r) and Cluster(s)).

if

the sum of the jobs in Cluster(r) and Cluster(s) $\leq g_w$

then

Combine these clusters to form a single cluster (let's say

$Cluster(r) \bigcup Cluster(s) \to Cluster(r)$).

Else

decide which chores to complete next and pair them with the next,

Higher priorities for their fusion.

Step 9: By merging the tasks in accordance with the tasks in clusters, change the matrices $\left[e_{i,k}\right]$ and BBB $\left[c_{i,j}\right]$.

Step 10: Use the Hungarian approach to identify which processors are assigned to which task clusters. Next, arrange the tasks according to their respective processor positions in a linear array TPOSS.

Step 11: Do the calculation $EXEX(), COMM(), TCOST(), R_{sys}$ using TPOSS.

Step 12: Finish

IMPLEMENTATION OF THE MODEL

The process of the suggested model is better understood by looking at the following example. In this example, there are five processors in the set $P = \{p_1, p_2, ..., p_5\}$ and nine tasks in the set $T = \{t_1, t_2, ..., t_9\}$. The fuzzy inter-task communication cost between tasks, the failure rate of communication links, and the fuzzy execution cost of each task on each processor are provided as matrices $\left[\tilde{e}_{i,k}\right]$ of order 9×5, $\left[\tilde{c}_{i,j}\right]$, $\left[\mu_{x,y}\right]$ of order 5×5, and EEE of order EEE, respectively. Fig. 1 displays the five processors' linear array networks. Processors p_1, p_2, p_3, p_4 and p_5 have failure rates of 0.00012, 0.00015, 0.00011, 0.00013, and 0.00018, in that order. The membership values between each pair of communication tasks and their grading are provided in the form of matrices $\left[\mu\left(c_{i,j}\right)\right]$ and TPLV, respectively, after applying steps 2 through 5 of the current algorithms. In Fig. 2(a), every task in the program is first represented by a polygon in a solid line, each of which is considered as a separate cluster. The tasks are fused into clusters following the application of steps 7 through 8. From Fig. 2(b) to Fig. 2(e), the process for cluster formation is displayed. Five $\{t_6,t_7\},\{t_4,t_9\},\{t_2,t_5\},\{t_1,t_8\}$ and $\{t_3\}$ clusters have been constructed for the program in question, as seen in Fig. 2(e). The fuzzy communication costs between each pair of the program's task clusters are displayed in Fig 3. The following is the optimal technique to use step 10 of the current algorithm to schedule the task clusters onto the processors: $TPOSS=\{((t_6,t_7),p_4),((t_4,t_9),p_2),((t_2,t_5),p_3),((t_1,t_8),p_1),(t_3,p_5)\}$. The optimal scheduling of the task clusters onto the processors is also depicted in Fig. 4. Regarding the case, 403.8, 634.8, 968, and 0.9349, 0.9506, and 0.9693) are the system cost and dependability, respectively. The results are compared with Yadav et al. (2011) model results, which are shown in Table 1. It is possible to conclude from the results that the recommended model outperforms the model created by Yadav et al. (2011) in terms of system reliability and cost. For the sake of comparison, the recommended approach is implemented on multiple randomly generated samples. In these situations, a large number of inputs are employed, and their ranges are contained within preset intervals. At $[1-25]$, $[0-15]$, $[1-5]$, $[0.00025-0.00050]$ and $[0.00010-0.00030]$ intervals, respectively, the execution costs, task-to-task communication costs, processor distance, processor failure rate, and communication connections are measured. A comparison of the system's cost and dependability based on the current model and the model created by Yadav et al. (2011) is shown in Table 2. These comparative results are also displayed in Figs. 5 and 6. The results of the examples show how much better the current scheduling method is than the model proposed by Yadav et al. (2011). After applying steps 2 to 5 of the present algorithm, the membership values between each pair of communicating tasks and their grading are given in the form of matrices $\left[\mu\left(c_{i,j}\right)\right]$ and TPLV respectively. Initially, each task of the program is treated as a distinct cluster and is shown by a polygon in a solid line in Fig. 2(a). After applying the steps 7 to 8, the tasks are fused into clusters. The procedure for the formation of the clusters is shown in Fig. 2(b) to Fig. 2(e). Fig. 2(e) shows that five clusters $\{t_6,t_7\},\{t_4,t_9\},\{t_2,t_5\},\{t_1,t_8\}$ and $\{t_3\}$ are formed for the given program. Fig 3 shows the fuzzy communication costs between each pair of task clusters of the program. By using step 10 of the present algorithm, the optimal scheduling of the task clusters onto the processors is as follows

$$TPOSS=\{((t_6,t_7),p_4),((t_4,t_9),p_2),((t_2,t_5),p_3),((t_1,t_8),p_1),(t_3,p_5)\}.$$

The optimal scheduling of the task clusters onto the processors is also depicted in Fig. 4. Regarding the case, 403.8, 634.8, 968, and 0.9349, 0.9506, and 0.9693 are the system cost and dependability, respectively. The results are compared with Yadav et al. (2011) model results, which are shown in Table 1. It is possible to conclude from the results that the recommended model outperforms the model created by Yadav et al. (2011)

in terms of system reliability and cost. For the sake of comparison, the recommended approach is implemented on multiple randomly generated samples. In these situations, a large number of inputs are employed, and their ranges are contained within preset intervals. At $[1-25]$ $[0.00025-0.00050]$ and $[0.00010-0.00030]$ intervals, respectively, the execution costs, task-to-task communication costs, processor distance, processor failure rate, and communication connections are measured. A comparison of the system's cost and dependability based on the current model and the model created by Yadav et al. (2011) is shown in Table 2. These comparative results are also displayed in Figs. 5 and 6. The results of the examples show how much better the current scheduling method is than the model proposed by Yadav et al. (2011).

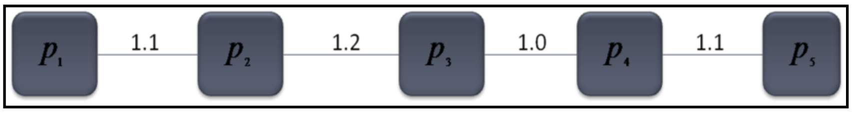


Fig. 1: Linear Array Network of Five Processors

$$FECM=\left[\tilde{e}_{i,k}\right]=\begin{array}{c} \\ t_1\\ t_2\\ t_3\\ t_4\\ t_5\\ t_6\\ t_7\\ t_8\\ t_9\end{array}\begin{array}{c}\begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5\end{array}\\ \begin{bmatrix} (07,09,11) & (08,13,15) & (12,14,15) & (05,08,10) & (11,12,14)\\ (01,03,04) & (02,07,10) & (09,11,12) & (06,08,9) & (12,14,16)\\ (10,12,14) & (06,08,10) & (03,07,10) & (03,05,07) & (04,6,10)\\ (03,07,10) & (04,05,07) & (11,14,17) & (08,10,12) & (07,09,10)\\ (10,12,13) & (10,13,15) & (06,08,09) & (06,07,09) & (04,06,08)\\ (01,3,05) & (04,05,06) & (10,11,12) & (06,08,10) & (07,09,10)\\ (10,14,15) & (10,12,13) & (8,10,11) & (06,09,10) & (08,13,15)\\ (05,07,09) & (06,08,10) & (8,10,12) & (09,11,13) & (12,14,16)\\ (09,11,13) & (06,10,14) & (05,08,11) & (05,07,09) & (10,13,16)\end{bmatrix}\end{array}$$

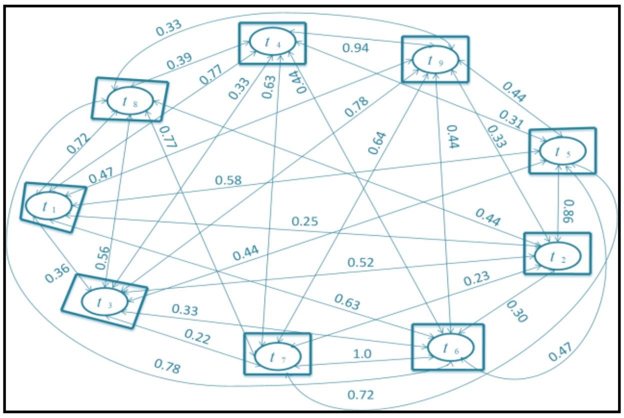
$$FITCCM=\left[\tilde{c}_{i,j}\right]=$$

$$\begin{array}{c} \\ t_1\\ t_2\\ t_3\\ t_4\\ t_5\\ t_6\\ t_7\\ t_8\\ t_9\end{array}\begin{array}{c}\begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9\end{array}\\ \begin{bmatrix} (00,00,00) & (01,02,04) & (02,03,05) & (06,07,08) & (04,05,07) & (04,06,07) & (00,00,00) & (05,06,9) & (03,04,07)\\ (01,02,04) & (00,00,00) & (03,05,06) & (00,00,00) & (06,08,09) & (01,03,04) & (01,02,03) & (03,04,05) & (02,03,04)\\ (02,03,05) & (03,05,06) & (00,00,00) & (01,03,05) & (02,04,06) & (01,03,05) & (00,02,04) & (03,05,07) & (06,07,08)\\ (06,07,08) & (00,00,00) & (01,03,05) & (00,00,00) & (01,03,04) & (01,02,03) & (04,06,07) & (01,03,07) & (06,09,10)\\ (04,05,07) & (06,08,09) & (02,04,06) & (01,03,04) & (00,00,00) & (01,05,06) & (04,07,08) & (00,00,00) & (02,03,04)\\ (04,06,07) & (01,03,4) & (01,03,05) & (01,02,03) & (01,05,06) & (00,00,00) & (08,09,10) & (05,07,09) & (03,04,05)\\ (00,00,00) & (01,02,03) & (00,02,04) & (04,06,07) & (04,07,08) & (08,09,10) & (00,00,00) & (06,07,08) & (04,05,07)\\ (05,06,09) & (03,04,05) & (03,05,07) & (01,03,07) & (00,00,00) & (05,07,09) & (06,07,08) & (00,00,00) & (02,03,04)\\ (03,04,07) & (02,03,04) & (06,07,08) & (06,09,10) & (02,03,04) & (03,04,05) & (04,05,07) & (02,03,04) & (00,00,00)\end{bmatrix}\end{array}$$

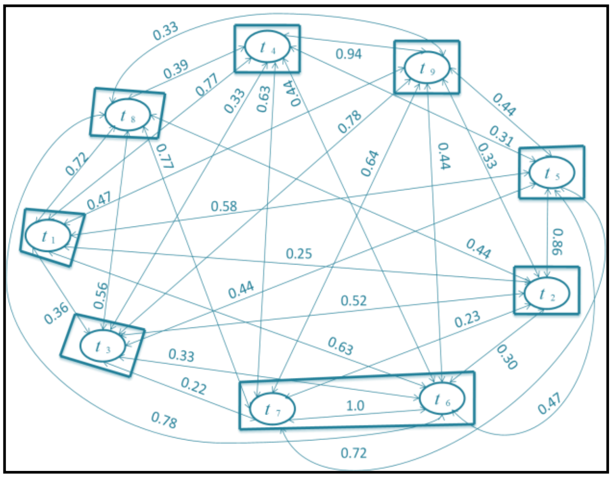
$$CLFRM=\left[\mu_{x,y}\right]=\begin{array}{c} \\ p_1\\ p_2\\ p_3\\ p_4\\ p_5\end{array}\begin{array}{c}\begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5\end{array}\\ \begin{bmatrix} --- & 0.0002800 & 0.0001100 & 0.0001000 & 0.0001200\\ 0.0002800 & --- & 0.0001600 & 0.0001400 & 0.0001300\\ 0.0001100 & 0.0001600 & --- & 0.0001500 & 0.0002200\\ 0.0001000 & 0.0001400 & 0.0001500 & --- & 0.0002500\\ 0.0001200 & 0.0001300 & 0.0002200 & 0.0002500 & ---\end{bmatrix}\end{array}$$

$$\left[\mu\left(c_{i,j}\right)\right]=\begin{array}{c} \\ t_1\\ t_2\\ t_3\\ t_4\\ t_5\\ t_6\\ t_7\\ t_8\\ t_9\end{array}\begin{array}{c}\begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9\end{array}\\ \begin{bmatrix} 00000 & 0.2500 & 0.3600 & 0.7700 & 0.5800 & 0.6300 & 0.0000 & 0.7200 & 0.4700\\ 0.2500 & 00000 & 0.5200 & 0.0000 & 0.8600 & 0.3000 & 0.2300 & 0.4400 & 0.3300\\ 0.3600 & 0.5200 & 00000 & 0.3300 & 0.4400 & 0.3300 & 0.2200 & 0.5600 & 0.7800\\ 0.7700 & 0.0000 & 0.3300 & 00000 & 0.3100 & 0.4400 & 0.6300 & 0.3900 & 0.9400\\ 0.5800 & 0.8600 & 0.4400 & 0.3100 & 00000 & 0.4700 & 0.7200 & 0.0000 & 0.4400\\ 0.6300 & 0.3000 & 0.3300 & 0.4400 & 0.4700 & 00000 & 1.0000 & 0.7800 & 0.4400\\ 0.0000 & 0.2300 & 0.2200 & 0.6300 & 0.7200 & 1.0000 & 00000 & 0.7700 & 0.6400\\ 0.7200 & 0.4400 & 0.5600 & 0.3900 & 0.0000 & 0.7800 & 0.7700 & 00000 & 0.3300\\ 0.4700 & 0.3300 & 0.7800 & 0.9400 & 0.4400 & 0.4400 & 0.6400 & 0.3300 & 00000 \end{bmatrix}\end{array}$$

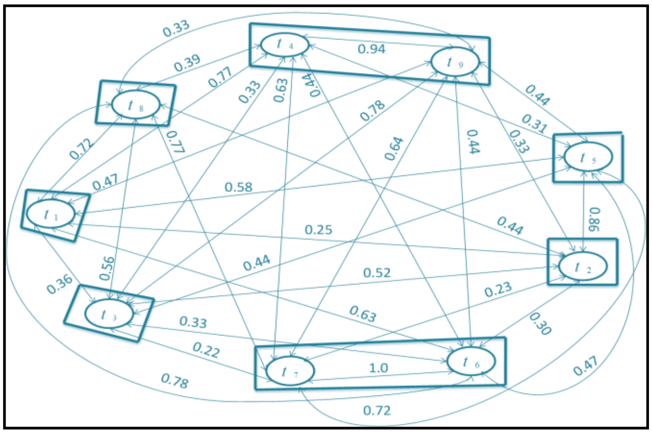
$$TPLV=\begin{array}{c} \\ t_1\\ t_2\\ t_3\\ t_4\\ t_5\\ t_6\\ t_7\\ t_8\\ t_9\end{array}\begin{array}{c}\begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9\end{array}\\ \begin{bmatrix} --- & VLCT & LCT & HCT & ACT & ACT & VLCT & HCT & ACT\\ VLCT & --- & ACT & VLCT & VHCT & LCT & VLCT & LCT & LCT\\ LCT & ACT & --- & LCT & LCT & LCT & VLCT & ACT & HCT\\ HCT & VLCT & LCT & --- & LCT & LCT & ACT & LCT & VHCT\\ ACT & VHCT & LCT & LCT & --- & ACT & HCT & VLCT & LCT\\ ACT & LCT & LCT & LCT & ACT & --- & VHCT & HCT & LCT\\ VLCT & VLCT & VLCT & ACT & HCT & VHCT & --- & HCT & ACT\\ HCT & LCT & ACT & LCT & VLCT & HCT & HCT & --- & LCT\\ ACT & LCT & HCT & VHCT & LCT & LCT & ACT & LCT & --- \end{bmatrix}\end{array}$$



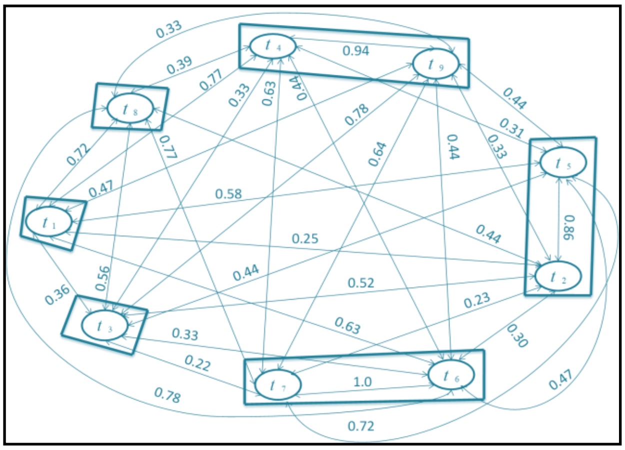
2 (a)



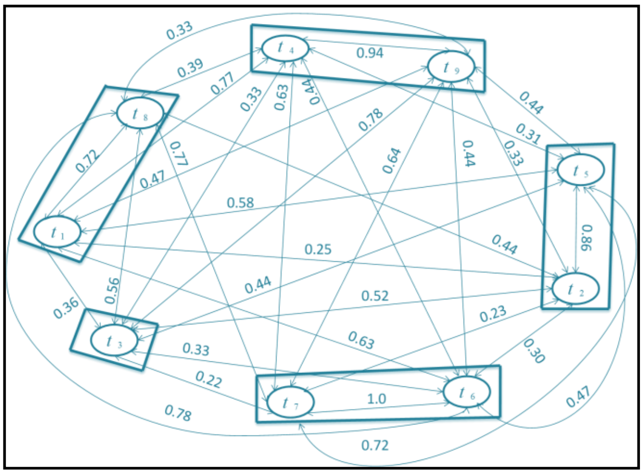
2 (b)



2 (c)



2 (d)



2 (e)

Fig 2: Clustering Steps of the Algorithm

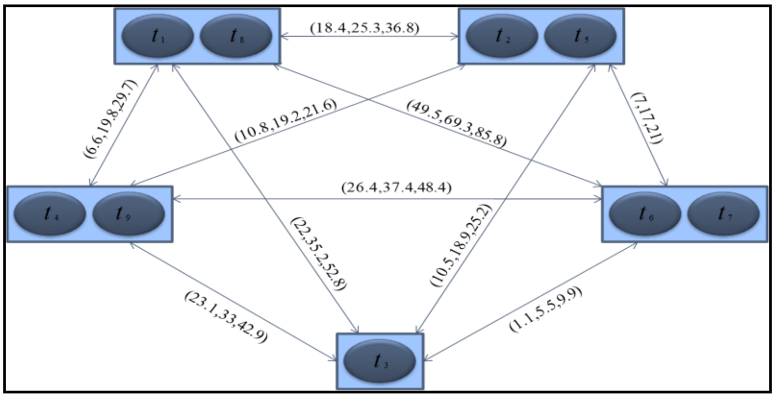


Fig 3: Inter-tasks Communication Task Graph between Task Clusters

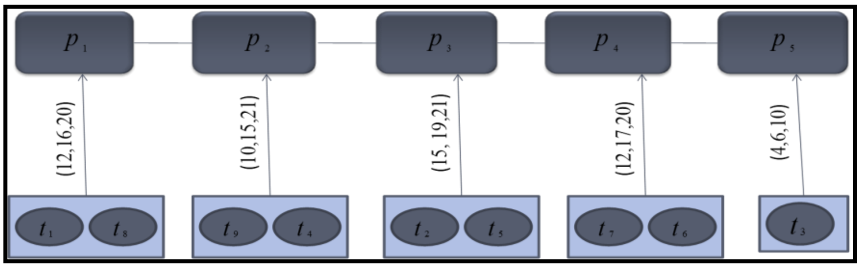


Fig 4: Optimal Tasks Assignment Graph

Table 1: Comparative Results of the Task Program

Processors	Proposed model			Yadav et al. (2011) Model		
	Assigned Tasks	System Cost	System Reliability	Assigned Tasks	System Cost	System reliability
p_1	t_1, t_8	(0403.80,0634.80, 0960.80)	(0.96930,0.95060, 0.93490)	t_6, t_7	(0411,0643.07, 972)	(0.94620,0.91270, 0.88510)
p_2	t_4, t_9			t_1, t_4		
p_3	t_2, t_5			t_8, t_9		
p_4	t_6, t_7			t_2, t_5		
p_5	t_3			t_3		

Table 2: Performance Comparisons of the Proposed Model with Yadav et al. (2011) Model

S. No.	No. of Processors n	No of Tasks m	Proposed model		Yadav et al. (2011) Model	
			System cost	System Reliability	System cost	System Reliability
1.	02	05	(019.60,038.30,052)	(0.99920,0.99840, 0.99290)	(021,041,059)	(0.99660,0.99340, 0.99020)
2.	02	07	(063,091,0116)	(0.99270,0.98950, ,0.98680)	(069,098,0122)	(0.99010,0.98540, 0.98130)
3.	03	05	(098.50,0147,0206)	(0.96710,0.95250, 0.94450)	(0114,0155,0213)	(0.96110,0.94690, 0.93530)
4.	03	08	(0172,0279.80,381.40)	(0.98160,0.97140, 0.96330)	(181.60,287.30,0389)	(0.9862,0.9505, 0.93490)
	04	07	(0336,0485,0634)	(0.98690,0.97650, 0.96340)	(0343.2,4930,0641)	(0.97350,0.96030, 0.94200)
6.	04	09	(0387,0485,0583)	(0.98790,0.97790, 0.96550)	(0397.2,0494,0598)	(0.97970,0.96170, 0.93990)
7.	05	09	(403.80,634.20,0968)	(0.96930,0.95060, 0.93490)	(0411,0643.70,0972)	(0.94620,0.91270, 0.88510)
8.	05	10	(349.60,599.80,0850)	(0.9744,0.95020, 0.92760,)	(0352,0607.5,0854.80)	(0.95430,0.90970, 0.86900)

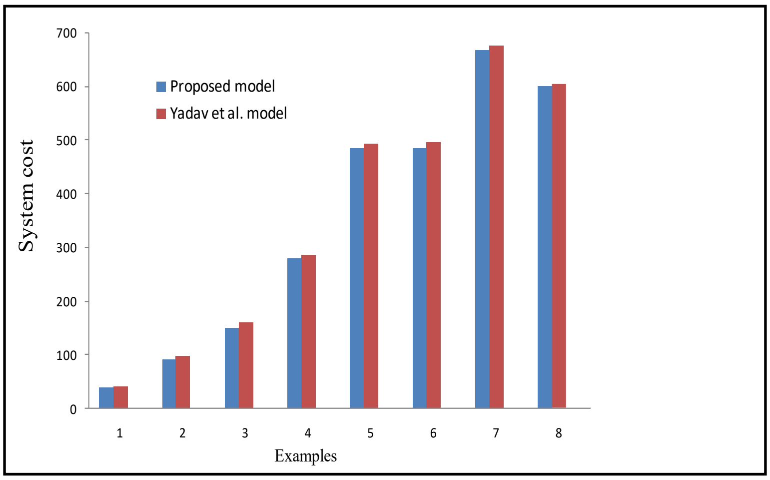


Fig 5: Comparison of System Cost

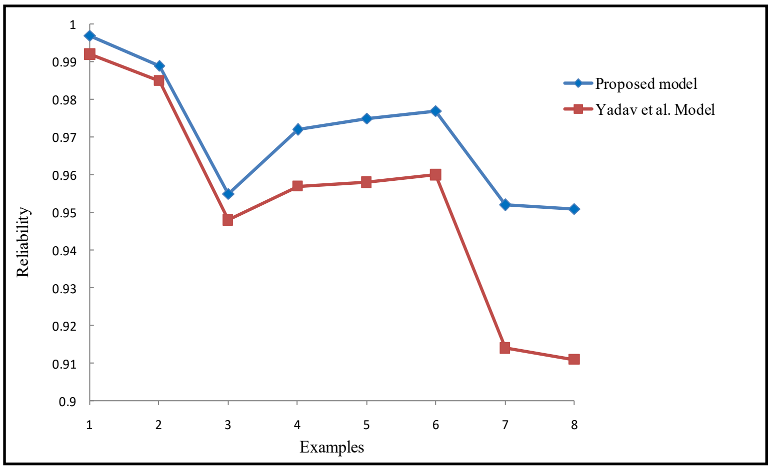


Fig. 6: Comparison of System Reliability

CONCLUSION

This research presented an efficient task scheduling model for distributed real-time systems operating in linear array networks with fuzzy cost considerations, aiming to minimize system cost while maximizing overall reliability. By incorporating heuristic-based scheduling and fuzzy linguistic clustering, the proposed approach effectively addressed processor heterogeneity and communication constraints inherent in distributed and cloud-fog environments. Experimental evaluations on randomly generated task sets demonstrated that the proposed model consistently outperforms the Yadav et al. scheduling model in terms of reliability enhancement and cost optimization. The results confirm the suitability of fuzzy-based decision mechanisms for handling uncertainty in dynamic scheduling scenarios. Looking ahead toward 2025, future work can extend this model to dynamic and large-scale cloud-fog-IoT environments by integrating energy-aware and security-aware constraints, real-time mobility support, and AI-driven adaptive learning techniques. Further validation using real-world workloads and multi-objective optimization frameworks will strengthen the applicability of the model for next-generation intelligent distributed computing systems.

REFERENCES

[1]. Ali, H. S., & Sridevi, R. (2024). Mobility and security-aware real-time task scheduling in fog-cloud computing for IoT devices: a fuzzy-logic approach. The Computer Journal, 67(2), 782-805.

[2]. Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. International Journal of Communication Systems, 33(16), e4583.

[3]. Arunarani, A. R., Manjula, D., & Sugumaran, V. (2019). Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems, 91, 407-415.

[4]. Ben Alla, H., Ben Alla, S., Touhafi, A., & Ezzati, A. (2018). A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for the cloud computing environment. Cluster Computing, 21(4), 1797-1820.

[5]. Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., & Tuba, M. (2022). Multi-objective task scheduling in a cloud computing environment by a hybridized bat algorithm. Journal of Intelligent & Fuzzy Systems, 42(1), 411-423.

[6]. Ebadifard, F., & Babamir, S. M. (2021). Autonomic task scheduling algorithm for dynamic workloads through a load-balancing technique for the cloud-computing environment. Cluster Computing, 24, 1075-1101.

[7]. Hajvali, M., Adabi, S., Rezaee, A., & Hosseinzadeh, M. (2023). Decentralized and scalable hybrid scheduling-clustering method for real-time applications in volatile and dynamic Fog-Cloud Environments. Journal of Cloud Computing, 12(1), 66.

[8]. Bokhari S. H. (1979). Dual Processor Scheduling with Dynamic Reassignment. IEEE Transactions on Software Engineering. 5(4), 341-349.

[9]. Bokhari S. H. (1981). A Shortest Tree Algorithm for Optimal Assignments across Space and Time in a Distributed Processor System. IEEE Transactions on Software Engineering. 7(6), 583-589.

[10]. Stone H. S. (1977). Multiprocessor Scheduling with the Aid of Network Flow Algorithms. IEEE Transactions on Software Engineering. 3(1), 85-93.

[11]. Attiya G. and Hamam Y. (2006). Task Allocation for Maximizing Reliability of Distributed Systems: A Simulated Annealing Approach. Journal of Parallel and Distributed Computing. 66, 1259-1266.

[12]. Cho S. Y. and Park K. H. (1994). Dynamic Task Assignment in Heterogeneous Linear Array Networks for Metacomputing. In Proceedings of the Heterogeneous Computing Workshop, Cancun, Mexico. 66-71.

[13]. Daoud M. I. and Kharma N. (2008). A High Performance for Static Task Scheduling in Heterogeneous Distributed Systems. Journal of Parallel and Distributed Computing. 68, 399-409.

[14]. Hussain F., Anpalagan A., Naeem M., and Khwaja A. S. (2016). Multi-Objective Resource Allocation in Inference-limited M2M Communication Networks. International Journal of Communication Network and Distributed Systems. 16(3), 297-313.

[15]. Karger, M. J. and Vakili, M. (2015). Load Balancing in MapReduce on Homogeneous and Heterogeneous Clusters. International Journal of Communication Network and Distributed Systems. 15(2/3), 149-168.

[16]. Konar D., Bhattacharya S., Sharma K., Sharma S., and Pradhan S. R. (2017). An Improved Hybrid Quantum-Inspired Genetic Algorithm for Scheduling of Real-Time Tasks in a Multiprocessor System. Applied Soft Computing. 53, 296-307.

[17]. Kumar H., Chauhan N. K., Yadav P. K. (2016). Dynamic Tasks Scheduling Algorithm for Distributed Computing Systems under Fuzzy Environment. International Journal of Fuzzy System Applications. 5(4), 77-95.

[18]. Lee C. H., Lee D., and Kim M. (1992). Optimal Task Assignment in Linear Array Networks. IEEE Transactions on Computers. 41(7), 877-880.

[19]. Nayak P. (2016). A Fuzzy Logic-Based Clustering Algorithm for WSN to Extend the Network Lifetime. IEEE Sensors Journal. 16, 137-144.

[20]. Noori K. and Janab K. (2013). Fuzzy Reliability-based Traction Control Model for Intelligent Transportation Systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 43, 229-234.

[21]. Pathan R. M. (2017). Real-Time Scheduling Algorithm for Safety-Critical Systems on Faulty Multicore Environments. Real-Time System. 53, 45-81.

[22]. Towsley D. F. (1986). Allocating Programs Containing Branches and Loops within a Multiple-Processor System. IEEE Transactions on Software Engineering. 12(10), 1018-1024.

[23]. Ucar B., Aykanat C., Kaya K. and Ikinci M. (2006). Task Assignment in Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing. 66, 33-46.

[24]. Vidyarthy D. P. and Tripathi A. K. (2001). Maximizing Reliability of Distributed Computing System with Task Allocation using Simple Genetic Algorithm. Journal of System Architecture 47, 549-554.

[25]. Yadav P. K., Pradhan P., and Singh P. P. (2011). A Fuzzy Clustering Method to Minimize the Inter-task Communication Effect for Optimal Utilization of the Professor's Capacity in Distributed Real-Time Systems. Proceedings of the International Conference on SocProS AISC 130. 151-160.

[26]. Yadav P. K., Singh M. P., Kumar H. (2008). Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Reassignment. Journal of Computer Systems, Networks and Communications. 2008, Article ID-578180, 1-9.

[27]. Yang J. (2014). Complexity Analysis of New Task Allocation Problem using Network Flow Method on Multicore Clusters. Mathematical Programming in Engineering. 2014, Article ID- 723497.

Cite this Article:

*Soniya Saini, Dr. Sandeep Kumar Tiwari, Dr. Ankur Nehra, "System Cost and Reliability Optimization Through Fuzzy-Based Task Scheduling in Distributed Computing Systems", **Pi International Journal of Mathematical Sciences, ISSN:** 3107-9830 (Online), Volume 1, Issue 3, pp. 15-29, December 2025.*

Journal URL: *https://pijms.com/*

DOI: *https://doi.org/10.59828/pijms.v1i3.14*