ISSN 0304-9892 (Print) ISSN 2455-7463 (Online)
www.vijnanaparishadofindia.org/jnanabha
***Jñānābha*, Vol. 52(2) (2022), 58-67**
(Dedicated to Professor D. S. Hooda on His 80th Birth Anniversary Celebrations)

OPTIMIZATION OF FUZZY ASSIGNMENT PROBLEM USING *R*

Anju Khandelwal [1], Suneet Saxena [2] and Avanish Kumar [3]
[1,2] Department of Mathematics, SRMS College of Engineering and Technology, Bareilly, India
[3] Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi, India
Email: dranju20khandelwal@gmail.com , sunarn2012@gmail.com , dravanishkumar@yahoo.com
(Received : August 04*,* 2022*; Revised : August* 14*,* 2022*, Accepted : September,* 23, 2022*)*

DOI: https://doi.org/10.58250/jnanabha.2022.52206

Abstract

Distributed Computing refers to the solution of a problem by distributed systems of autonomous and heterogeneous computers that are used in communication, networking, and workstation. Distributed computing is a computing technology that allows multiple computers to solve the same problem. Here problem-solving is achieved by communicating and executing the tasks in a distributed environment. Distributed Computing helps in performing computational tasks much faster than single computers. The objective of the problem present in this paper is to optimize the Processor Execution Cost and find the optimum combination which gives the reduced processor execution cost in the distributed computing environment and further provides the reduced program service cost as well as reliability. The algorithm for the present problem is executed in the open-source *R* programming language for optimization.

2020 Mathematical Sciences Classification: MSC 90-04, 90-05, 90C-08, 90C-70

Keywords and Phrases: Distributed Computing System (*DCS*), Processor/Task Execution Cost (*PEC*/*TEC*), Processor Communication Cost (*PCC*), Process Requirement, Open-Source Language.

1. Introduction

Any distributed computer system consists of many software components that are on multiple computers and run as a unit system. The computers in a distributed system are physically and remotely connected together with a communicating network. It can be geographically distant and connected to a wide area network where nodes can easily share data with other nodes. More and more nodes can be added easily in a distributed system i.e., it can be scaled up as needed. The failure or disruption of one or more nodes in any distributed computer system does not have an unauthorized effect on that entire system. In this situation other nodes can continue to communicate with remaining other among themselves.

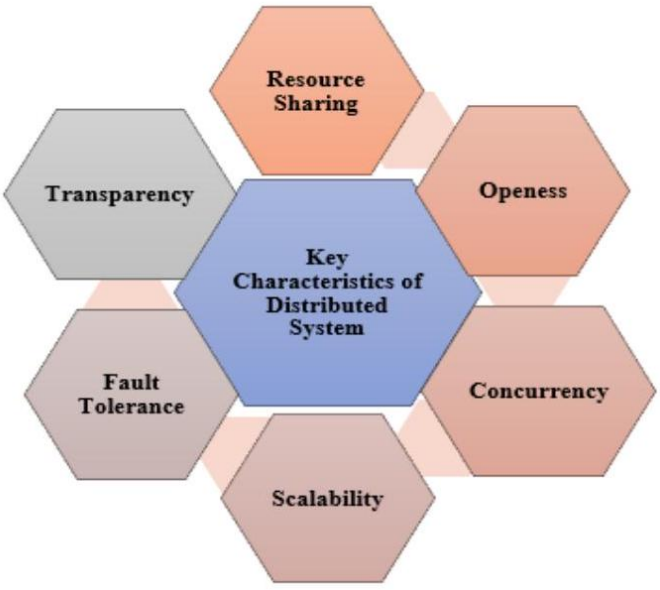

***Figure* 1.1:** Characteristic of Distributed System

This characteristic of distributed computing predominates among all other features of distributed computing Fig. 1.1.

Distributed computing allows different users and computers to share information among themselves. In addition, distributed computing can pass processing power, memory, or storage benefits from one machine to another. A Distributed Computing System is a networked collection of independent machines that can collaborate remotely to achieve an objective. So distributed computing is used in cloud technology that enables this distributed system to operate, collaborate and communicate. Distributed computing results in the development of highly faulttolerant systems that are reliable and performance-driven. Distributed systems allow real-time applications to execute faster and to serve end-user requests quickly.

'*R*' programming is a type of programming language mainly used for statistical computing. '*R*' programming is exclusively based on data science, but '*R*' is also a multi-paradigm language, which means it has several paradigms such as: object-oriented, procedural, reflective, etc.

***Figure* 1.2:** R Environment Features

'*R*' is an open-source programming language developed in 1993 that is an alternative implementation of the language native ' *S* '. It gets its name from its creators Ross Ihaka and Robert Gentleman. The '*R*' language provides a wide variety of statistics related libraries which give an environment conducive to statistical computing and design. In addition, it is also used by many quantitative analysts as a programming tool and has many features Fig. 1.2.

2. Literature Review

Over the years the distributed computing system has become an integral part of executing scientific applications that are usually represented by a workflow model. It is a simple and costeffective way for scientists to demonstrate workflows anytime and anywhere. Distributed computing environment plays an important role in providing resources as a utility on the Internet. Distributed processing environment has emerged as an approach in terms of new governance business and renaissance or revitalization in data innovation. Distributed computing divides a single task among multiple computers while cloud computing provides hardware, software and other infrastructure resources over the Internet. Distributed computing is done when connected via a network to achieve faster than using a personal computer. Technically, if an application syncs information across multiple devices it represents cloud computing as it is using distributed computing.

In 2019, Hao et al.[1] proposed resource scheduling algorithms on the basis of work endurance value. Here, in order to reduce the energy consumption of cloud computing systems, the concepts of performance conditions, work endurance value and relaxation time in cloud systems were studied. Here the task waiting time was also reduced by accommodating massive random task allocation resources in cloud computing systems.

Over the years, the demand for an eco-friendly environment has led to a rapid rise in the green city revolution around the world. The demand for eco-friendly has necessitated the shift of major energy consumers from conventional

electricity grids to renewable energy sources (RES). This is the reason why cloud data centers (DCs) have emerged as important consumers of energy. Kaur, Aujla and Kumar [2, (2022)] provides a comprehensive workload classification in job scheduling and virtual machine placement architecture for Cloud DC powered by RES and Power Grid is designed. Here, well-established benefits such as reduced operating costs and carbon emissions are expanded by using an advanced heuristic approach.

Fuzzy computing is of great importance in any real-life application. Task assignment is an important part of distributed computing systems. Task assignment requires the processor to allocate tasks for execution to use the appropriate means of computation available. Task allocation requires that it conform to the performance characteristics of that task. Khandelwal [3, (2019)] and Khandelwal-Kumar [4, (2019), 5, (2020)] investigated the task allocation problem that is based on fuzzy computing which is more realistic and general in nature. It is based on distributed computing with task allocation timing and fuzzy communication timing.

Liu et al. [6, (2021)] proposed an online multi-workflow scheduling framework called NOSF. Deadline-constrained workflows with random arrivals and uncertain task execution times can be scheduled with this online multi-workflow scheduling. The computation offloading has become increasingly popular in recent years to reduce energy consumption and enhance smartphone performance. Lu et al.[7, (2022)] proposed and implemented a lightweight offloading framework to deal with heterogeneous architectures between smartphones and servers. This lightweight offloading framework supports offloading of computation-intensive tasks and deploys servers efficiently. The experimental results presented here are valid for multi-task offloading strategy and intensive offloading requests.

Load balancing is like a challenge in cloud network to improve proper resource utilization and makespan. Various static and dynamic scheduling methods are used to troubleshoot load balancing. Mishra Sharma, Rath and Parida [8, (2022)], used two multi-datacenters for load balancing and appropriate task scheduling. In addition, the paper proposed a two-stage load adjustment technique in two multi-datacenters through which users can allocate jobs to different virtual machines in the data centers.

The digital transformation of all traditional systems has led to a massive outflow of data. Most of the data generated by digital devices is challenging to handle and there are problems of latency while providing services to the end users. This problem can be solved by osmotic computing method given by Neha, Panda and Sahu [9, (2021)]. Osmotic computing exploits and integrates cloud, edge and Internet of things platforms as well. Osmotic computing is a growing research area that has laid the foundation for a new state-of-the-art computing paradigm by objectifying scenarios to handle heavy data and computations as per user requirements and resource availability.

Distributed cloud computing is increasingly relevant to both academia and numerous industries as an emerging subfield of computer science. It is a solution to high barriers of entry in configuring and maintaining computing hardware and inflexible platform constraints. Task scheduling is one of the vital aspects of distributed cloud computing which deals with the strategies for assigning tasks to computing resources. There are numerous commonly used task scheduling algorithms that cut down the completion time and increase the throughput of the system. Shi, Suo, Kemp and Hodge [10, (2020)] suggested an algorithm called BMin which augments the performance of the algorithm Min-min, decreased completion time, increased throughput, and improves load balancing of resources- outperforming the classical algorithm. Methods based on migrating crawling agents (migrants) can be used to select and filter web documents on the web server with the search engine side. This significantly reduces the network load caused by web crawlers, but as a migrant web move around, security issues emerge. These are an impediment to the development and maintenance of mobile agent technology. Singhal, Dixit, Agarwal and Sharma [11, (2018)] presented a remote platform-oriented reliability-based approach that is helpful for maintaining the security and integrity of migrants as well as data and remote platforms. Also, Truong and Dustdar [12, (2010)] presented cost models associated with different application performance models are presented. Here the method of determining the cost of various scenarios from the performance model is presented. It presents the estimation, monitoring and analysis of costs associated with scientific applications using realworld applications.

The widespread use of data mining and analysis to support decision making has become possible in today's era of big data. The complex process of collecting massive amounts of data can be modeled as a workflow involving storage, transmission and analysis. Cloud environments provide sufficient computing and storage resources for big data management and analysis, but cost-effective resource provisioning for workflows in the cloud is still a significant challenge. Time limit is the most important thing for any workflow execution. Wu, Ding, Jia, and Li[13, (2020)] addressed the challenge of cost-effective resource provisioning while meeting the real-time requirements of workflow execution by introducing a programmingbased resource provisioning strategy.

Large-scale cluster operation of drones has gradually become a reality with the continuous development of computer and network technology due to Yadav, Kumar and Gupta [14, (2006)]. In UAV cluster combat, one of

the most challenging challenges is the proper allocation of *UAV* cluster combat tasks and intelligent adaptation control of the *UAV* cluster. Therefore, solving the task allocation problem and finding the optimal solution is an *NP*-hard problem model. Zhang and Chen [15, (2021)] used a *CSA* based approach to optimize the four objectives simultaneously in *multi-UAV* task allocation to maximize the number of tasks allocated. Solving this problem maximizes the benefits of performing tasks, minimizes resource costs and minimizes time costs.

3. Objective

The objective of this research problem is to execute the large-scale program to assign its various tasks to the most suitable processors in the distributed processing environment. The assignment of tasks shall be based on execution cost that is to be formulate in such a way that the overall cost along with reliability is to be optimized under the set of given constraints. The cost and reliability functions to measure processor execution cost, processor communication cost, processor execution reliability and processor communication reliability are then had to be formulated. The load on each processor has been also taken care off to balance the load of each and every processor. It is presumed that number of tasks shall always be greater than number of available processors of the Distributed Computing System. The algorithm is implemented on an open-source '*R*' language to get an index that will define the optimal system cost. Here, the index will be obtained by using processor execution cost and reliability. The entire process will definitely provide the analyses of the distributed system and provide the overall optimal system cost with reliability.

4. Technique

Let the given system consists of a set of n processors $P = \{p_1, p_2, p_3, p_4, \ldots, p_n\}$, interconnected by communication links and a set of m tasks $T = \{t_1, t_2, t_3, t_4, \ldots, t_m\}$, . The Processor Execution Cost of individual tasks corresponding to each processor are given in the form of matrices $ECM(,)$ of order $m \times n$. The Processor Communication Cost is taken in the square symmetric matrices $CCM(,)$ of order n respectively. The assignment of tasks to processors may be done in different ways. The overall processor execution cost [Ecost] is expressed as the sum of execution costs along with communication cost of all the tasks as follows:

$$\text{Ecost} = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} EC_{ij} x_{ij} \right\} + \sum_{j=1}^{n} \left\{ \sum_{j=1}^{n} CC_{ij} y_{ij} \right\} \right],$$

The Processor Execution Reliability of individual tasks corresponding to each processor are given in the form of matrices $ERM(,)$ of order $m \times n$. The Processor Communication Reliability is taken in the square symmetric matrices $CRM(,)$ of order n respectively. The overall processor execution reliability [Ereliability] is expressed as the product of execution reliability along with communication reliability of all the tasks as follows:

$$\text{Ereliability} = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} ER_{ij} x_{ij} \right\} * \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} CR_{ij} x_{ij} \right\} \right],$$

where,

$$x_{ij} = \begin{cases} c1, \text{ if } i^{\text{th}} \text{ task is assigned to } j^{\text{th}} \text{ processor} \\ 0, \quad \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, \text{ if the task assigned to processor } i \text{ communicate} \\ \text{with the task assigned to processor } j \\ 0, \quad \text{otherwise} \end{cases}$$

$$\text{Index} = \frac{\text{Ereliability}}{\text{Ecost}}.$$

5. Algorithm

To give an algorithmic representation to the technique mentioned in the section 4, let us consider a system in which a set of m tasks $T = \{t_1, t_2, t_3, t_4, \ldots, t_m\}$ is to be executed on a set of n available processors $P = \{p_1, p_2, p_3, p_4, \ldots, p_n\}$, here. $m >> n$.

Step1. Input: m, n, *ECM* (,), *ERM* (,), *CCM* (,), *CRM* (,)

Step2. Select the task to form the combinations with other task(s), (say (t_i , t_k)). Store it in *TCOMB* (,).

Step3. Add k^{th} row of *ECM* (,) to its i^{th} row. If all the values are become infinite, get next value of *TCOMB* (,) then repeat else go to step 4.

Step4. Modify CCM (,), *CRM* (,), *ECM* (,) and *ERM* (,).

Step5. Replace the corresponding values of k^{th} row and k^{th} column by zero in *CCM* (,) and then add k^{th} row to i^{th} row and k^{th} column to i^{th} column, after that delete the k^{th} row and k^{th} column from *CCM* (,).

Step6. Replace the corresponding values of k^{th} row and k^{th} column by one in *CRM* (,) then multiply k^{th} row to i^{th} row and k^{th} column to i^{th} column, after that delete the k^{th} row and k^{th} column from *CRM* (,).

Step7. Modify the *ECM* (,) by adding k^{th} row to i^{th} row and thereby deleting k^{th} row.

Step8. Modify the *ERM* (,) by multiplying k^{th} row to i^{th} row and thereby deleting k^{th} row.

Step9. Store modified, *ECM* (,), *ERM* (,), *CCM* (,) and *CRM* (,) to *NECM* (,), *NERM* (,), *NCCM* (,) and *NCRM* (,) respectively.

Step10. Apply Assignment Algorithm [Hungarian method for task allocation].

Step11. Evaluate Execution Cost [Ecost], Execution Reliability [Ereliability] and Index by using equation (4.1), (4.2) and (4.3).

Step12. Stop

***Flowchart* 5.1: Algorithm of Proposed Technique**

6. Implementation

To analyses the performance of the Distributed Computing System, the suggested algorithm is then executed in the open-source R programming language to find the set of combinations, and then evaluate the index values for each combination, the index values shall be based upon the processor execution costs and execution reliabilities for the corresponding combinations. In order to implement the proposed algorithm following two examples have been illustrated.

Example 6.1. Here, we consider a distributed environment consisting of a set $T = t_1, t_2, t_3, t_4$ of 4 tasks of a large-scale program and a set $P = p_1, p_2, p_3$ of 3 processors in a distributed system.

$$ECM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_2 \\ t_1 & 8 & 12 & 7 \\ t_2 & 9 & 8 & 11 \\ t_3 & 12 & 9 & 6 \\ t_4 & 10 & 11 & 12 \end{array} \qquad ERM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_2 \\ t_1 & 0.997 & 0.996 & 0.994 \\ t_2 & 0.993 & 0.998 & 0.992 \\ t_3 & 0.998 & 0.991 & 0.994 \\ t_4 & 0.997 & 0.993 & 0.998 \end{array}$$

$$CCM(,) = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0 & 3 & 6 & 9 \\ t_2 & 3 & 0 & 4 & 5 \\ t_3 & 6 & 4 & 0 & 7 \\ t_4 & 9 & 5 & 7 & 0 \end{array} \qquad CCM(,) = \begin{array}{c|cccc} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0.000 & 0.994 & 0.996 & 0.995 \\ t_2 & 0.994 & 0.000 & 0.992 & 0.993 \\ t_3 & 0.996 & 0.992 & 0.000 & 0.992 \\ t_4 & 0.995 & 0.993 & 0.992 & 0.000 \end{array}$$

The implementation of the example through *R* is represented in Fig. 6.1 and its output is shown in Fig. 6.2 as obtained through *R* Console. The final as well optimal assignment of tasks is mentioned in Fig. 6.3.

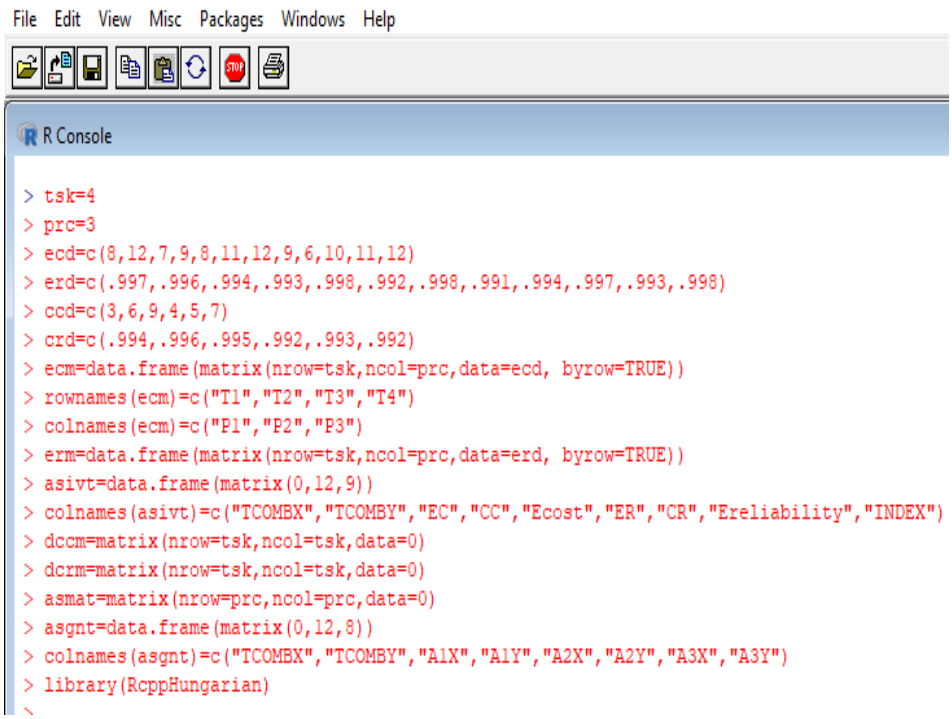

***Figure* 6.1:** Program on *R* Console

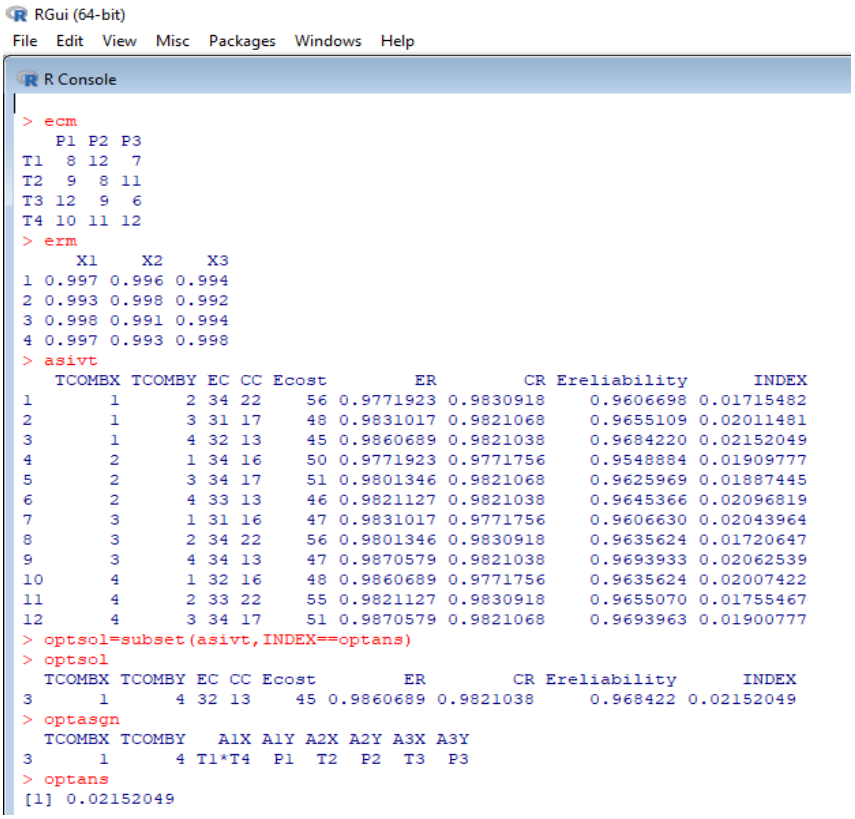

***Figure* 6.2:** Output on *R* Console

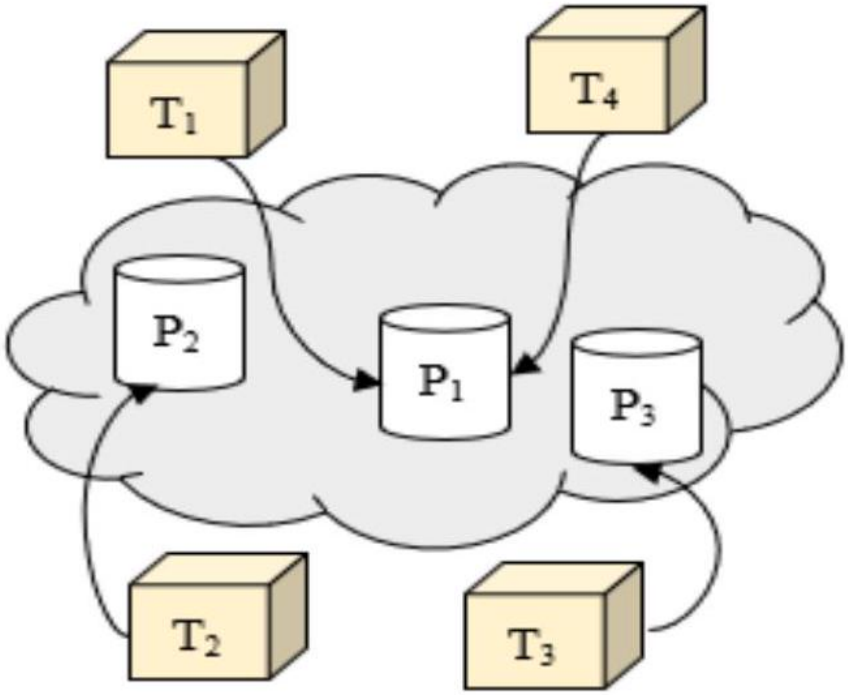

***Figure* 6.3:** Task Execution on Processor

Example 6.2. Consider another similar example in which the distributed environment consisting of a set $T = \{t_1, t_2, t_3, t_4, t_5\}$ of 5 tasks of any large computer executable program and a set $P = \{p_1, p_2, p_3\}$ of 3 processors of any distributed system. The details are shown in Fig. 6.4, Fig. 6.5 and Fig. 6.6 respectively.

```
RGui (64-bit)
File  Edit  View  Misc  Packages  Windows  Help
R Console
> ecm
    P1  P2  P3
T1   8  12   7
T2   9   8  11
T3  12   9   6
T4  10  11  12
T5   7   6   2
> erm
     X1    X2    X3
1 0.997 0.996 0.994
2 0.993 0.998 0.992
3 0.998 0.991 0.994
4 0.997 0.993 0.998
5 0.992 0.995 0.996
> tcrm
     X1    X2    X3    X4    X5
1 0.000 0.994 0.996 0.995 0.993
2 0.994 0.000 0.992 0.993 0.995
3 0.996 0.992 0.000 0.992 0.996
4 0.995 0.993 0.992 0.000 0.992
5 0.993 0.995 0.996 0.992 0.000
> tccm
  X1 X2 X3 X4 X5
1  0  3  6  9  8
2  3  0  4  5  6
3  6  4  0  7  9
4  9  5  7  0  5
5  8  6  9  5  0
>
```

***Figure* 6.4:** Process on *R* Console

RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
> optans
[1] 0.020522
>
>
> optsol
   TCB1X TCB1Y TCB2X TCB2Y EC CC Ecost       ER       CR Ereliability    INDEX
16     1     4     3     5 34 13    47 0.982125 0.982104     0.964549 0.020522
68     3     5     1     4 34 13    47 0.982125 0.982104     0.964549 0.020522
>
>
> optasgn
   TCB1X TCB1Y TCB2X TCB2Y   A1X A1Y A2X A2Y   A3X A3Y
68     3     5     1     4 T1*T4  P1  T2  P2 T3*T5  P3
>
>
```

***Figure* 6.5:** Output on *R* Console

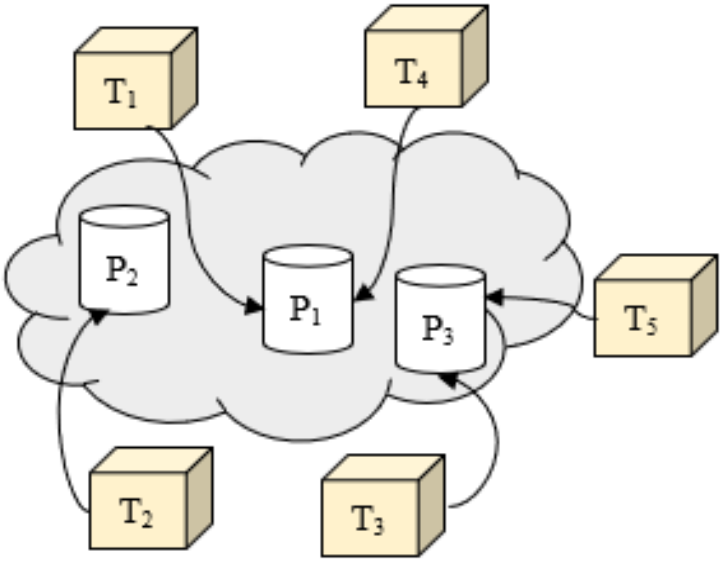

***Figure* 6.6:** Optimal Assignment of Task on Processor

7. Conclusion

In distributed system, computation workload is spread across several connected processors. It creates an execution environment where application components look at specific geographical locations that are chosen based on application needs. Also, in distributed cloud computing takes the cloud computing model and distributes it to different geographic locations in a connected manner. Here we have chosen the problem in which number of tasks are more than the number of processors in the distributed environment. On implemented the proposed algorithm the set of index values are obtained which are based on cost and reliability. The minimum value of index indicates the overall optimal system cost. The developed algorithm in *R* programming gives the optimum result for the examples 6.1 and 6.2. These evaluated results have been mentioned in the following Table 7.1 and Table 7.2:

***Table* 7.1:** Optimum Solution of Example 6.1

Task	Processor	*EC*	*CC*	Ecost	***ER***	***CR***	Ereliability	Index
$t_1 * t_4$	p_1	2*32	2*13	45	0.9860689	0.9821038	0.968422	0.02152049
t_2	p_2							
t_3	p_3							

***Table* 7.2:** Optimum Solution of Example 6.2

Task	Processor	***EC***	***CC***	**Ecost**	***ER***	***CR***	Ereliability	Index
$t_1 * t_4$	p_1	2*34	13	47	0.982125	0.982104	0.964549	0.020522
t_2	p_2							
$t_3 * t_5$	p_3							

The pictorial representations of the input data of the Example 6.1 and example 6.2 are shown in Fig. 6.1, Fig. 6.4 and their output is shown through Fig. 6.2 and Fig. 6.5 respectively. The optimal assignment is shown through Fig. 6.3 and Fig. 6.6 respectively. The present algorithm is the implementation of the approach discussed by the Yadav, Kumar and Gupta [14, (2006)] in R open source. Here complexity is found to be very less in comparison to the approach discussed by Yadav et al. [14]. Although the studied algorithm implemented on the distributed environment, but it is very easily extended to the cloud environment. Future researchers are advised to explore more parameters such as, storage, speed, inter processor distance, computing, and other network resources.

References

[1] L. Hao, B. Li, K. Li and Y. Jin, Research for Energy Optimized Resource Scheduling Algorithm in Cloud Computing Base on Task Endurance Value, *International Conference on Artificial Intelligence and Computer Applications,* (2019), 279-282, China. http://doi/10.1109/ICAICA.2019.8873435http://doi/10.1109/ICAICA.2019.8873435.

[2] K. Kaur, G. S. Aujla and N. Kumar, A Multi-Objective Optimization Scheme for Job Scheduling in Sustainable Cloud Data Centers, *IEEE Transactions on Cloud Computing,* **10**(1) (2022), 172-186. {http://doi/10.1109/TCC.2019.2950002}{http://doi/10.1109/TCC.2019.2950002}.

[3] A. Khandelwal, Fuzzy based Amalgamated Technique for Optimal Service Time in Distributed Computing System, *International Journal of Recent Technology and Engineering,* **8**(3) (2019), 6763-6768. Doi:10.35940/ijrte.C4783.098319.

[4] A. Khandelwal and A. Kumar, Framework and Evolution of Task Orientation using Fuzzy and GA, *International Journal of Recent Technology and Engineer*ing, **8**(2) (2019), 5475-5479. Doi:10.35940/ijrte.B6988.078219.

[5] A. Khandelwal and A. Kumar, Evaluation of Service Time in DCS using Fuzzy and Clustering Technique, *International Conference on Computational Performance Evaluation (ComPE),* (2020), 028-032, Shilong. http://doi/10.1109/ComPE49325.2020.9200122http://doi/10.1109/ComPE49325.2020.9200122.

[6] J. Liu, J. Ren, W. Dai, D. Zhang, P. Zhou, Y. Zhang, G. Min and N. Najjari, Online Multi-Workflow Scheduling under Uncertain Task Execution Time in IaaS Clouds, *IEEE Transactions on Cloud Computing,* **9**(3) (2021), 1180-1194. http://doi.10.1109/TCC.2019.2906300http://doi.10.1109/TCC.2019.2906300.

[7] J. Lu, Q. Li, B. Guo, J. Li, Y. Shen, G. Li and H. Su, A Multi-Task Oriented Framework for Mobile Computation Offloading, *IEEE Transactions on Cloud Computing,* **10**(1) (2022), 187-201. https://doi.10.1109/TCC.2019.2952346https://doi.10.1109/TCC.2019.2952346.

[8] S. C. Mishra Sharma, A. K. Rath and B. R. Parida, Efficient load balancing techniques for multi-datacenter cloud milieu, *International Journal of Information Technology,* **14**(2) (2022), 979-989. https://doi.org/10.1007/s41870-020-00529-2https://doi.org/10.1007/s41870-020-00529-2.

[9] B. Neha, S. K. Panda and P. K. Sahu, An efficient task mapping algorithm for osmotic computing-based ecosystem, *International Journal of Information Technology,* **13**(4) (2021), 1303-1308. https://doi.org/10.1007/s41870-021-00715-whttps://doi.org/10.1007/s41870-021-00715-w.

[10] Y. Shi, K. Suo, S. Kemp and J. Hodge, A Task Scheduling Approach for Cloud Resource Management, 2020 *Fourth World Conference on Smart Trends in Systems, Security and Sustainability* (WorldS4), London (2020), 131-136. http://doi/10.1109/WorldS450073.2020.9210422http://doi/10.1109/WorldS450073.2020.9210422.

[11] N. Singhal, A. Dixit, R. P. Agarwal and A. K. Sharma, A reliability based approach for securing migrating crawlers, *International Jounal of Information Technology,* **10**(1) (2018), 91-98. https://doi.org/10.1007/s41870-017-0065-0https://doi.org/10.1007/s41870-017-0065-0.

[12] H. L. Truong and S. Dustdar, Composable cost estimation and monitoring for computational applications in cloud computing environments, *Procedia Computer Science,* (2010), 2175-2184. https://doi.org/10.1016/j.procs.2010.04.243https://doi.org/10.1016/j.procs.2010.04.243.

[13] L. Wu, R. Ding, Z. Jia and X. Li, Cost-Effective Resource Provisioning for Real-Time Workflow in Cloud, *Journal of Complexity,* **1** (2020), 1-15. https://doi.org/10.1155/2020/1467274https://doi.org/10.1155/2020/1467274.

[14] P. K. Yadav, A. Kumar and A. R. Gupta, An Exhaustive Approach of Performance Analysis to the Distributed Systems based on Cost Assignments, S*outh East Asian Journal of Mathematics and Mathematical Sciences,* **5**(1) (2006), 29-44.

[15] X. Zhang and X. Chen, UAV Task Allocation Based on Clone Selection Algorithm, *Wireless Communications and Mobile Computing,* **1** (2021), 1-9. https://doi.org/10.1155/2021/5518927https://doi.org/10.1155/2021/5518927