International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-VXXXX
Volume X Issue X, 1-4, Month 2022

Type of Article *(Original Article / Review Article / Short Communication)* (Size 11)

A Comparison of Optimization Algorithms for Task Allocation in Vehicular Fog Computing

Dhanashree Toradmalle[1], Pramila Shinde[2], Sachin Singh[3], Tarang Ahuja[4], Yash Rane[5]

[1] *Associate Professor, Computer Engineering, K J Somaiya Institute of Technology, Mumbai, Maharashtra, India*
[2] *Associate Professor, Computer Engineering, Shah and Anchor Kutchhi Engineering College, Mumbai, Maharashtra, India*
[3] *Student, Information Technology, Shah and Anchor Kutchhi Engineering College, Mumbai, Maharashtra, India*
[4] *Student, Information Technology, Shah and Anchor Kutchhi Engineering College, Mumbai, Maharashtra, India*
[5] *Student, Information Technology, Shah and Anchor Kutchhi Engineering College, Mumbai, Maharashtra, India*

[1]dhanashree.t@somaiya.edu
[2]pramila.shinde@sakec.ac.in
[3]sachin.singh_19@sakec.ac.in
[4]tarang.ahuja_19@sakec.ac.in
[5]yash.rane_19@sakec.ac.in

Received: Revised: Accepted: Published:

Abstract *- Through vehicular ad hoc networks, fog computing, which is situated between vehicles and the cloud, gives vehicles more processing, networking and storage power. The traditional vehicular networks are added to the fog computing paradigm. This allows us to support more pervasive vehicles, improve communication efficiency, and do away with limitations of conventional vehicular networks. The well-known problems with vehicular fog computing that cause energy waste are latency and delay. To address such issues, in this work, we plan to develop a task allocation system so that we can allocate tasks to the most optimal node such that we can help vehicular fog computing applications get a faster response. Genetic Algorithm and Particle Swarm Optimization Algorithm are the two-optimization algorithm that we have implemented to do a comparison among the outputs of MinMakeSpan, MinTotalCost, Global best array, which specifically tells us about the task allocated to a particular node, and TotalCost, MakeSpan, Optimal Function value of each algorithm in order to determine the optimal algorithm for task allocation.*

Keywords - *Vehicular fog computing; Task Allocation; Genetic Algorithm; Particle Swarm Optimization.*

1. Introduction

Vehicles are given processing, networking, and storage power using vehicular fog computing, which links them to the cloud. Our goal is to create an ideal task allocation system for Vehicular Fog Computing.
Fog Nodes and Cloud Servers are the two main parts of the system, according to the Vehicular Fog Computing or Fog Computing standard architecture.

- Once a request is generated by a user, a task is created and sent to a fog node.
- The task is transferred to the Cloud Server if the Fog Node is unable to complete it, and the Cloud Server subsequently sends the completed task back to the server.
- Here, if the task is sent to the Cloud Server, a lot of computational energy is wasted and the result generation will take longer due to latency.

In our proposed system, when a user generates a request

- A task is created, and based on the computational requirements of the task, an appropriate fog node is selected from the multiple available fog nodes to execute the task.
- If none of the fog node devices are able to do so, the task is forwarded to the Cloud Server and the result is generated.

By using our proposed system, we can help vehicular network applications reduce latency, delay and computational usage.

2. Related Work

Various problems with the centralized Cloud architecture are being caused by IoT applications. For example, IoT is unable to support applications like connected cars. [1], traffic signals [6], etc. These issues can

be overcome using fog computing. Cloud Fog Computing consists of different phases architecture, Algorithm, Resource Pooling, and Security. The most important phase is the Algorithm which consists of Resource allocation and task offloading.

Table 1. Comparison between Cloud computing and Fog computing.

Parameter	Cloud Computing	Fog Computing
Latency	High	Low
Response time of the system	Low	High
Architecture	Centralized	Distributed
Communication with devices	From a distance via Internet, Multiple hops	Directly from the edge via Various protocols and standards, One hop
Data processing	Far from the source of information in long-term time	Close to the source of information in short-term time
Computing capabilities	Higher (Cloud Computing does not provide any reduction in data while sending or transforming data)	Lower (Fog Computing reduces the amount of data sent to Cloud computing.)
Server nodes	Few nodes with scalable storage and computing power	Very large nodes with limited storage and computing power
Security	Less secure, Undefined	More secure, Can be defined
Working environment	Warehouse-size building with air conditioning systems	Outdoor (e.g., Streets, gardens) or indoor (e.g., Restaurants)
Working environment	Within the Internet	At the edge of the local network

According to [4], Emerging automotive applications like cooperative lane changing and real-time situational awareness necessitate enough computing power at the edge to complete time-sensitive and data-intensive tasks. Task distribution across fixed and mobile nodes with restrictions on service delay, quality loss, and fog capacity is accomplished using non-linear programming (NLP).

[3] proposed Algorithms to Optimize Resource and Task Allocation Problems for Cloud-Fog Environment. It consists of both cloud and fog nodes in hybrid mode. Each node has its own processing power, memory, and bandwidth usage cost A number of issues were produced at random using the properties suggested by Nguyen et al. [3].

Table 2. Properties of the nodes.

Parameter	Fog Nodes	Cloud Nodes
CPU rate (MIPS)	[500,1500]	[3000,5000]
CPU usage cost	[0.1,0.4]	[0.7,1.0]
Memory usage cost	[0.01,0.03]	[0.02,0.05]
Bandwidth usage cost	[0.01,0.02]	[0.05,0.1]

Table 3. Properties of the tasks to be assigned.

Property	Value
Number of instructions (10^9 instructions)	[1]
Memory required (MB)	[50,200]
Input file size (MB)	[10,100]
Output file size (MB)	[10,100]

In [3], the problem formulation is done for task allocation which consists of independent tasks. The problem formulation consists of Execution time, Makespan, Cost, and UtilityFunction. [3] have defined the Utility function which helps in finding the best task allocation using the Genetic algorithm and Modified Particle Swarm Optimization algorithm. These algorithms are used by [3] to allocate nodes to the independent tasks using the generated dataset from the above characteristics tables. A comparative analysis of these algorithms is done to select the best one. The main focus of [3] is to compare different algorithms for task allocation and variation of alpha value in the utility function.

According to [2], the utility function gives more accurate results rather than the algorithm. It uses a mixed-integer linear model that the optimization solver can resolve to get a globally optimal answer. The solution can be varied based on the changes made to the trade-off coefficient α.

According to [5], VFC-related task scheduling and resource allocation have recently attracted a lot of attention. The majority of these projects in VFC right now have a primary focus on reaction time optimization or energy reduction. The optimization problem for maximization is solved using a genetic algorithm-based approach.

For various aspects of Fog computing there are survey papers available. For example, for healthcare systems, Shi et al. [11] reviewed key features of fog computing. Furthermore, Yi et al. [12] contributed to the overview of fog computing by discussing different application scenarios of fog computing and several issues that may arise during their implementation.

Central fog services are positioned at the software defined resource management layer in the proposed architecture [13]. It offers a cloud-based middleware for analysing, orchestrating, and monitoring fog cells as well as stopping autonomous behaviours by fog colonies. In addition, Bonomi et al [14] performed an investigation of the integration of IoT and fog computing focused on crucial facets of the latter and how it extends and complements cloud computing.

Yousefpour et al. [15] proposed a policy that uses fog and fog communication in order to minimise service delay by load sharing. Apart from queue lengths, the policy also considers various types of requests with different processing time to calculate offloading.

Mahmud et al[16] gives taxonomy, problems, and characteristics of the fog computing environment. They demonstrated how cloud, fog, edge, mobile cloud, and mobile edge computing differ from one another. They examined networking equipment, fog node configuration, and other fog computing characteristics.

Abdi et al. [17] offered a MPSO coupled with a comparison to two well-known sstrategies, namely PSO and GA, for the effectiveness of task scheduling. They concentrated on speeding up task completion.

Guo et al. [18] proposed a task allocation problem in fog-cloud computing. However, for determining latency and power consumption, a simple model is being used. Specifically, the terminal sends a task to the base station. Base stations receive tasks and send them to edge cloud servers for execution. Results are then returned to the end user.

Huynh Thi Thanh Binh [22] presented a job scheduling approach fog computing based on evolutionary algorithms. This process aims to offer a trade-off between time and expense.

Deng et al. [21] used existing optimization techniques such as convex optimization [22], nonlinear integer [23] and Hungarian method [24] to solve three independent sub-problems

3. Task Allocation Problem

The materials and methods section should contain suffi

3.1. System Architecture

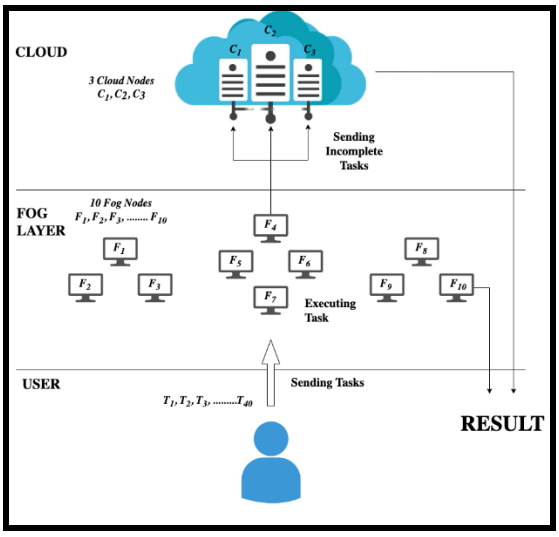

Fig 1. Proposed Vehicular Fog Computing Architecture

The above Fig 1. represents the modified architecture based on our proposed system, the nodes work together regionally and simultaneously connect with the cloud to satisfy the needs of their users.

According to Fig 2. A request is sent by the user (Step 1) to the fog node and the task gets executed (Step 2) based on the various characteristics of the task and the result generated is sent to the user (Step 3). If the fog node is not able to complete the task (Step 4), then the task is sent to the most suitable cloud node which is chosen from the multiple available cloud nodes, and the task gets executed (Step 5) based on the various characteristics of the task and the result is forwarded to the user (Steps 6 and 7).

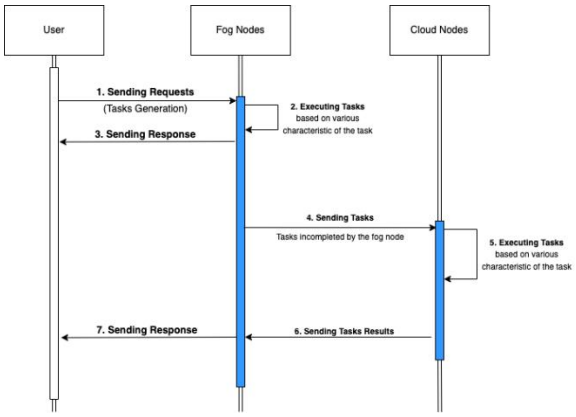

Fig 2. Implemented Task Allocation in Vehicular Fog Computing

3.2. Problem Formulation

The Fog Node Layer breaks down user requests into independent tasks that may be handled using Vehicle based Fog computing infrastructure. These tasks' characteristics include the number of instructions, the amount of memory needed, and the size of the input and output files. Let T_k indicate the kth task, the system receives a collection of n distinct tasks each time, as follows:

$$T = \{T_1, T_2, T_3 \dots, T_n\}$$

The Cloud and Fog nodes that make up the Vehicular Fog computing infrastructure share characteristics including CPU rate, CPU usage cost, memory usage fee, and bandwidth usage fee. Even while cloud nodes are frequently more powerful than fog nodes in this situation, their cost is higher. The set of m processors including c Cloud nodes and f Fog nodes in the system ($N = N_{cloud} \cup N_{fog}$) is expressed as

$$N = \{N_1, N_2, N_3, \dots, N_m\}$$

Execution time:

ExeTime$\left(T_k^i\right)$ time taken for execution of T_k handled in node N_i, which is calculate by:

$$ExeTime\left(T_k^i\right) = \frac{length\left(T_k^i\right)}{CPUrate(N_i)}$$

Makespan:

Makespan, which is measured from the moment a request is received until the last task is completed, or when the final machine is finished, is the total amount of time required for the system to complete all tasks., which is calculated by:

$$Makespan = \max_{1<i<m} [EXT(N_i)]$$

MinMakespan:

Let MinMakespan be the lower bound of Makespan, which is the system's total job completion time in the quickest amount of time. Ideally, MinMakespan will be reached when every node completes every task at the same time and is calculated by:

$$MinMakespan = Ext(N_1) = \cdots = Ext(N_m)$$

thus,

$$MinMakespan = \frac{\sum_{1 \le k \le n} length(T_k)}{\sum_{1 < i < n} CPUrate(N_i)}$$

Total Cost:

Processing costs, memory costs, and bandwidth costs all need to be covered when a task is completed. The cost estimated when node N_i processes the task T_k is expressed as:

$$Cost(T_k^i) = c_p(T_k^i) + c_m(T_k^i) + c_b(T_k^i)$$

Processing cost is defined as:

$$c_p(T_k^i) = c_1 * ExeTime(T_k^i)$$

Memory usage cost is defined as:

$$c_m(T_k^i) = c_2 * Mem(T_k^i)$$

Bandwidth usage cost is defined as:

$$c_b(T_k^i) = c_3 * Bw(T_k^i)$$

Total cost for all tasks to be executed is calculated as follows:

$$TotalCost = \sum_{T_k^i \in NodeTasks} Cost(T_k^i)$$

MinTotalCost:

When each task is given to the least expensive node, MinTotalCost is obtained which is the minimum cost required to complete a set of tasks T. It is simple to determine which node processes task T_k with the lowest cost, known as $MinCost(T_k)$. As a result, MinTotalCost is particular to a particular set of tasks T and is defined as:

$$MinTotalCost = \sum_{T_k \in T} MinCost(T_k) = \sum_{T_k \in T} \min_{1 \le i \le m} \left(Cost(T_k^i)\right)$$

Utility Function:

The utility function that determines the trade-off between Makespan and total cost can be defined as follows:

$$F = \alpha * \frac{MinMakespan}{Makespan} + (1 - \alpha) * \frac{MinTotalCost}{TotalCost}$$

3.3. Problem Statement

In order to handle latency-sensitive real-time applications in the vehicular networks, we need to create a task allocation system that will provide task assignment strategies, where the goal is to reduce processing costs and execution time.
The problem can be stated mathematically as

Input:

$T = \{T_1, T_2, T_3, \dots, T_n\}$: *a set of independent tasks*
$N = \{N_1, N_2, N_3, \dots, N_m\}$: *a set of processing nodes*

Output:

$NodeTasks = \{T_1^a, T_2^b, T_3^c, \dots, T_n^p\}$: *an assignment of all tasks*

Objective:

Maximise the utility function F:

$$F = \alpha * \frac{MinMakespan}{Makespan} + (1 - \alpha) * \frac{MinTotalCost}{TotalCost}$$

4. Implementation

4.1. Genetic Algorithm

- Encoding of Chromosomes

A chromosome-specified individual serves as a stand-in for a task allocation solution in a genetic algorithm. Chromosomes are encoded in an n-dimensional array with n genes. Indicating that the associated task is assigned to the node with the number k, each gene has a value of an integer k in the range [1; m], where m is the number of nodes. Gene sequence numbers and task sequence numbers are the same.

- Population Initialization

The set of all individuals utilised in the GA to identify the best course of action is known as the initial population. Assume that there are N people in the population. The N individuals are initialised randomly in order to cover a large portion of the search space and to guarantee the variety of the population in the first generation. The next generation is formed by selecting people from the starting population and performing various operations on them.

- Fitness Function

An individual's superiority in the population is determined by their fitness function. The individual with a high fitness value is an excellent solution. The utility function F makes

an estimate of each person's fitness value. The ability of a person to survive or perish through each generation depends on their fitness worth.

- Genetic Operators

Crossover operator

In order to produce offspring that inherit beneficial genes from parents, chromosomal encoding as an array of numbers is used. Figure 3 illustrates this process. A new individual is produced by randomly selecting two crossover points, where the first parent trades the middle gene segment with the second parent while leaving the other genes unaltered.

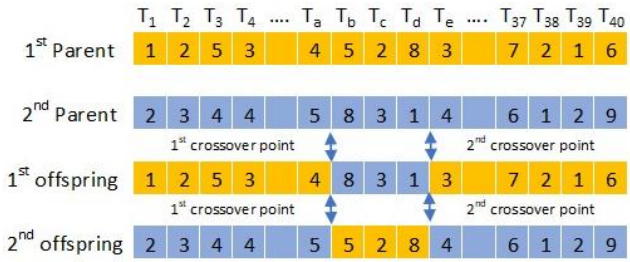

Fig 3. Genetic Crossover Operator

Selection strategy

Since the genes of the weaker individuals are not eliminated as quickly and continue to contribute to the exploration of new regions on the search area, this greedy selection preserves the population's diversity over generations. At the same time, the good gene segments are not repeated excessively among members of one population.

Mutation operator

After the processes of crossover and natural selection, N new members of the population are created. Every person takes part in one-point mutation at a rate of. The mutant gene's position is randomly chosen and changed to a different value; this assigns the task to be processed in a different node.

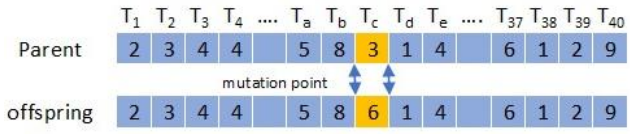

Fig 4. Mutation Operator

4.2. Modified Particle Swarm Optimization Algorithm

- Particle: Position and Velocity Matrix

Based on its velocity, each particle moves to a new position. The position matrix and velocity matrix share the exact same dimensions, i.e., $m \times n$. Its elements are real numbers in the range $[-V_{\max}, V_{max}]$. Basically, the kth particle has velocity V_k satisfying the condition:

$$V_k(i,j) \in [-V_{\max}, V_{max}](\forall\, i,j),$$
$$i \in \{1,2,3,\dots,m\},$$
$$j \in \{1,2,3,\dots,n\}$$

	T_1	T_2	T_3	T_4	T_5	T_6	...	T_{37}	T_{38}	T_{39}	T_{40}
N_1	0	1	0	1	1	0		1	0	1	0
N_2	0	0	0	0	0	1		0	0	0	0
N_3	1	0	1	1	0	0		0	1	0	1
...											
N_{12}	1	0	1	1	0	1		0	1	0	1
N_{13}	0	0	1	0	0	1		0	1	0	1

Fig 5. Velocity Matrix

- Population Initialization

Assuming that a population of N particles exists, each particle's position in the initial population is initially produced at random, ensuring that particles are dispersed throughout the search space. Additionally, enabling dynamic exploration, the velocity matrices of the particles are initialised stochastically.

- Fitness Function

A fitness function is used to evaluate particles; the fitness value reveals the quality of the solution the particle expresses as well as how it affects the population. The utility function F calculates the value of fitness. The solution is better when the fitness value is higher.

- Personal Best and Global Best

Particles move around in many different directions. The global best (gBest) denotes the perfect location that any particle of the entire population has figured since the inception, while the personal best $pBest_k$ indicates the perfect position that the kth particle has encountered. As position matrices, pBest and gBest are therefore $m \times n$ binary matrices. In the PSO process, the two variables pBest and gBest are crucial because they enable every particle to look around both its own best position and the global best position. Every time step, personal best and global best should be updated. If a higher fitness value is attained, the kth particle's personal best, pBestk, is replaced with X_k when it shifts to the new position, X_k. If any pBest's fitness value exceeds that of the current gBest, then that pBest will take its place.

5. Result

5.1. Genetic Algorithm

After executing our implementation of Genetic Algorithm for 40 tasks with Alpha value as 0.5, an output is generated as seen in Fig 6. which includes the values of MinMakeSpan, MinTotalCost, Global best array which specifies the task allocated to a particular node, TotalCost, MakeSpan and Optimal Function value.

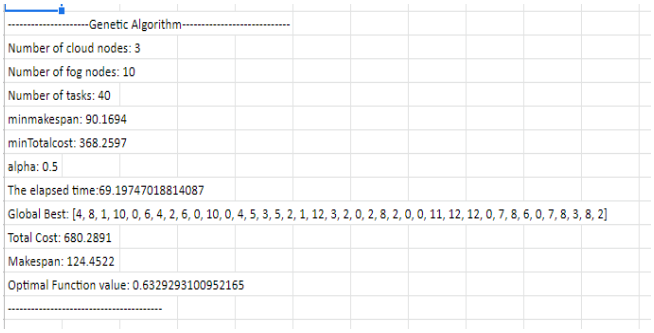

Fig 6. Implementation Output for Genetic Algorithm

We also create a Gantt chart as seen in Fig 7. based on the Global Best array that is generated which pictorially shows the various tasks allocated to the various nodes.

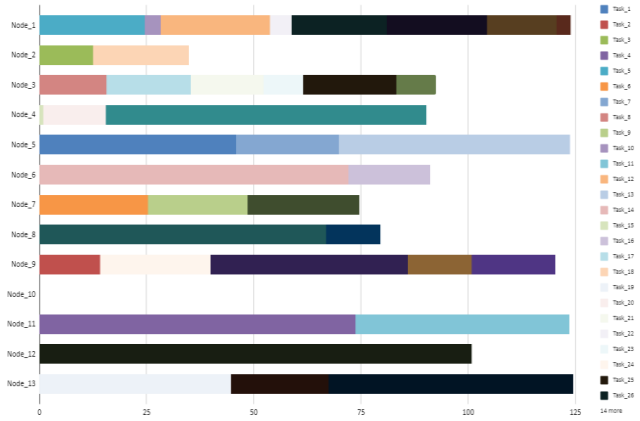

Fig 7. Gantt Chart for Genetic Algorithm

5.2. Modified Particle Swarm Optimization

After executing our implementation of Modified Particle Swarm Optimization Algorithm for 40 tasks with Alpha value as 0.5, an output is generated as seen in Fig 8. which includes the values of MinMakeSpan, MinTotalCost, Global best array which specifies the task allocated to a particular node, TotalCost, MakeSpan and Optimal Function value.

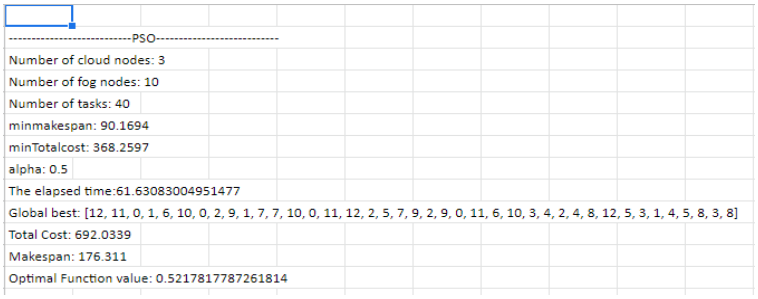

Fig 8. Implementation Output for Modified Particle Swarm Optimization

We also create a Gantt chart as seen in Fig 9. based on the Global Best array that is generated which pictorially shows the various tasks allocated to the various nodes.

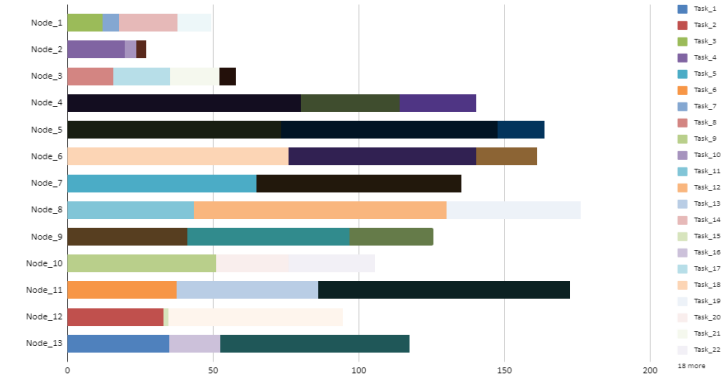

Fig 9. Gantt Chart for Modified Particle Swarm Optimization

6. Analysis

6.1. Comparative Analysis

We perform a comparative analysis on both the algorithms, namely Genetic Algorithm and Modified Particle Swarm Optimization Algorithm, to generate a table which compares the values of MakeSpan, TotalCost and Utility Function for both the optimization algorithms for varied number of tasks ranging from 40 tasks to 240 tasks at an increment of 40 tasks, based on which we can determine the optimal algorithm.

Table 4. Comparative Analysis Between GA and MPSO.

	makeSpan (s)		Total Cost (G$)		Utility Function Value(0-1)	
Number of Task	PSO	Genetic algo	PSO	Genetic algo	PSO	Genetic algo
40	**157.95568**	214.01068	697.635415	**642.54713**	0.5543369416	**0.5831334157**
80	324.83351	**281.64829**	1625.49926	**1530.357945**	0.5772191059	**0.6398722563**
120	516.896235	**472.71435**	2221.471655	**2163.1539**	0.5777456038	**0.6131759715**
160	698.212985	**549.83356**	2975.77659	**2870.275605**	0.4922958606	**0.5770851843**
200	790.878405	**787.959415**	3592.30025	**3314.173465**	0.595287397	**0.6249515778**
240	1026.661775	**1001.826385**	5111.289815	**4969.472575**	0.5289928007	**0.5428196911**

Based on the comparative analysis, we create a graphical representation comparison of the Two Algorithms.

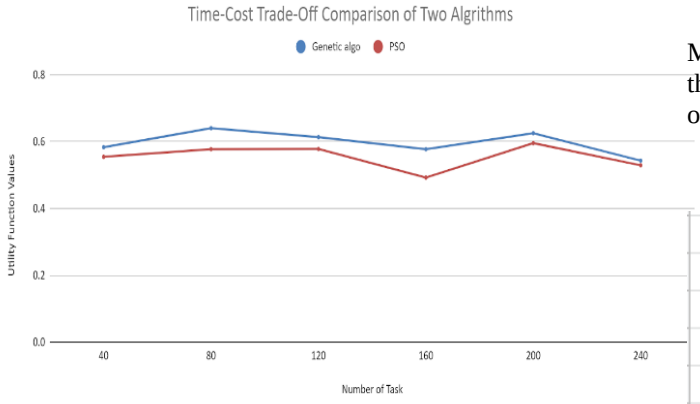

Fig 10. Graphical Representation for Comparative Analysis

6.2. Alpha Comparison for Genetic Algorithm

We generate a table for the obtained values of MakeSpan, TotalCost and Utility function upon changing the values of Alpha in the range of 0 to 1, with an increment of 0.1, to determine the optimal alpha value.

Table 5. Alpha Value Comparison for Genetic Algorithm.

alpha value	makespan	total cost	utility funciton
0	4467.6956	1353.5151	0.9779570985
0.1	3873.6742	1354.9993	0.8866304345
0.2	3683.703	1353.555	0.7979755433
0.3	3166.8276	1362.8332	0.7071660533
0.4	1454.1948	1453.9638	0.6254376103
0.5	385.8773	2329.2822	0.6572307554
0.6	425.9567	2256.0119	0.6402776452
0.7	508.7719	2239.6615	0.5734646057
0.8	347.6923	2434.1832	0.7712636217
0.9	453.3332	2237.7792	0.6307879137
1	456.7388	2170.0262	0.6304156774

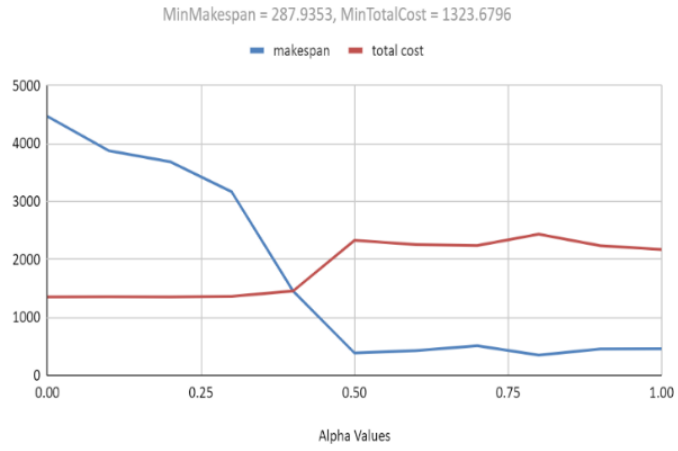

Fig 11. Graphical Representation for Alpha Comparison of GA

6.3. Alpha Comparison for Modified Particle Swarm Optimization

We generate a table for the obtained values of MakeSpan, TotalCost and Utility function upon changing the values of Alpha in the range of 0 to 1, with an increment of 0.1, to determine the optimal alpha value.

Table 6. Alpha Value Comparison for Modified Particle Swarm Optimization

alpha value	makespan	total cost	utility funciton
0	1047.3814	1908.9686	0.6934004572
0.1	821.6375	1926.9581	0.6532784512
0.2	900.5848	1885.1936	0.6256602594
0.3	560.27	2117.9588	0.5916619943
0.4	466.1922	2213.4948	0.6058555269
0.5	566.2767	2223.4506	0.5518989108
0.6	445.4192	2307.4733	0.6173215872
0.7	519.1349	2180.8307	0.5703394982
0.8	471.3216	2371.3811	0.6003662478
0.9	537.6941	2300.4281	0.5394907644
1	474.733	2313.7891	0.6065205073

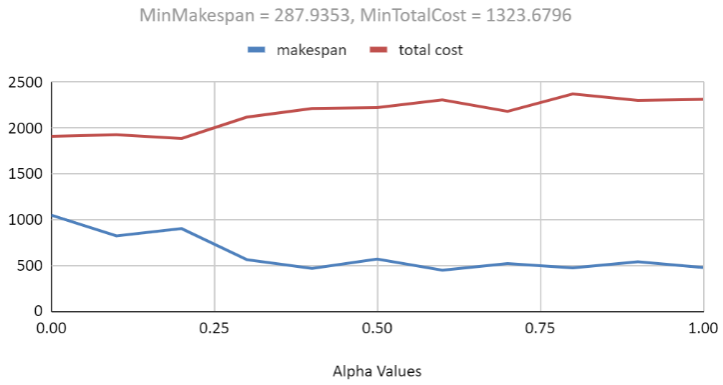

Fig 12. Graphical Representation for Alpha Comparison of MPSO

7. Conclusion

We concentrated on task distribution issues for vehicular fog computing applications. Utilising the Modified Particle Swarm Optimization Algorithm and Genetic Algorithm for optimization we have done a performance evaluation in various scenarios based on the output parameters like the values of MinMakeSpan, MinTotalCost, Global best array which specifies the task allocated to a particular node, TotalCost, MakeSpan and Optimal Function. Genetic Algorithm outperformed Modified Particle Swarm Optimization Algorithm over 6 sets of tasks, starting from 40 tasks and ending at 240 tasks with an increment of 40 tasks each. Genetic Algorithm

obtained a significantly shorter scheduling length. Additionally, the genetic algorithm might adapt to the user's need for cost-effectiveness or high-performance processing. We have achieved our goal of determining the optimal algorithm for task allocation that we have implemented for general situations and that can be extended to work with various applications of Vehicular Fog Computing.

References

[1] Atlam, Hany & Walters, Robert & Wills, Gary. (2018). Fog Computing and the Internet of Things: A Review. Big Data and Cognitive Computing. 2. 10.3390/bdcc2020010.

[2] J.-F. Tsai, C.-H. Huang, and M.-H. Lin, "An Optimal Task Assignment Strategy in Cloud-Fog Computing Environment," *Applied Sciences*, vol. 11, no. 4, p. 1909, Feb. 2021, doi: 10.3390/app11041909.

[3] Nguyen, Binh Minh & Binh, Huynh & Anh, Tran & Do, Son. (2019). Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment. Applied Sciences. 9. 1730. 10.3390/app9091730.

[4] C. Zhu, G. Pastor, Y. Xiao, Y. Li and A. Ylae-Jaeaeski, "Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing," 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2018, pp. 1-9, doi: 10.1109/SAHCN.2018.8397129.

[5] Tang, C., Xia, S., Li, Q. *et al.* Resource pooling in vehicular fog computing. *J Cloud Comp* 10, 19 (2021). https://doi.org/10.1186/s13677-021-00233-x

[6] Atlam, Hany & Walters, Robert & Wills, Gary. (2018). Fog Computing and the Internet of Things: A Review. Big Data and Cognitive Computing. 2. 10.3390/bdcc2020010.

[7] Z. Constantinescu and M. Vladoiu, "Towards Vehicular Fog Computing," 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)

[8] VC. Huang, R. Lu and K. -K. R. Choo, "Vehicular Fog Computing: Architecture, Use Case, and Security and Forensic Challenges," in IEEE Communications Magazine, vol. 55, no. 11, pp. 105-111, Nov. 2017, doi: 10.1109/MCOM.2017.1700322.

[9] H. -J. Hong, "From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017, pp. 331-334, doi: 10.1109/CloudCom.2017.53.

[10] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura and S. Mahmoud, "UAVFog: A UAV-based fog computing for Internet of Things," 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2017, pp. 1-8, doi: 10.1109/UIC-ATC.2017.8397657.

[11] Shi, Y.; Ding, G.; Wang, H.; Roman, H.E.; Lu, S. The fog computing service for healthcare. In Proceedings of the 2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), IEEE, Beijing, China, 28–30 May 2015; pp. 1–5.

[12] Yi, S.; Li, C.; Li, Q. A survey of fog computing: Concepts, applications and issues. In Proceedings of the 2015 Workshop on Mobile Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 37–42.

[13] Suárez-Albela, M.; Fernández-Caramés, T.; Fraga-Lamas, P.; Castedo, L. A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications. Sensors 2017, 17, 1978

[14] Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In Big Data and Internet of Things: A Roadmap for Smart Environments; Springer: Berlin, Germany, 2014; pp. 169–186.

[15] Yousefpour, A.; Ishigaki, G.; Jue, J.P. Fog computing: Towards minimizing delay in the internet of things. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 17–24.

[16] Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In Internet of Everything; Springer: Singapore, 2018; pp. 103–130

[17] Abdi, S.; Motamedi, S.A.; Sharifian, S. Task scheduling using modified PSO algorithm in cloud computing environment. In Proceedings of the International Conference on Machine Learning, Electrical and Mechanical Engineering, Dubai, UAE, 8–9 January 2014; pp. 8–9.

[18] Guo, X.; Singh, R.; Zhao, T.; Niu, Z. An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–7.

[19] Ningning, S.; Chao, G.; Xingshuo, A.; Qiang, Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. China Commun. 2016, 13, 156–164.

[20] Bitam, S.; Zeadally, S.; Mellouk, A. Fog computing job scheduling optimization based on bees swarm. Enterp. Inf. Syst. 2017, 12, 373–397.

[21] Deng, R.; Lu, R.; Lai, C.; Luan, T.H.; Liang, H. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J. 2016, 3, 1171–1181

[22] He, J.; Cheng, P.; Shi, L.; Chen, J.; Sun, Y. Time synchronization in WSNs: A maximum-value-based consensus approach. IEEE Trans. Autom. Control 2014, 59, 660–675.

[23] Li, D.; Sun, X. Nonlinear Integer Programming; Springer Science & Business Media: Berlin, Germany, 2006; Volume 84.

[24] Kuhn, H.W. The Hungarian method for the assignment problem. Naval Res. Logist. (NRL) 2005, 52, 7–21.

[25] Binh, H.T.T.; Anh, T.T.; Son, D.B.; Duc, P.A.; Nguyen, B.M. An Evolutionary Algorithm for Solving Task, Scheduling Problem in Cloud–Fog Computing Environment. In Proceedings of the SOICT 9th Symposium on Information and Communication Technology, Da Nang City, Vietnam, 6–7 December 2018; pp. 397–404.