Analysis of Performance Metrics for Task Scheduling with Task Duplication using Hybrid Meta-heuristic Techniques Under Cloud Environment

Rachhpal Singh
P.G. Deptt. Of Computer Sc. And Applications
Khalsa College, Amritsar

Jaspreet Bedi
P.G. Deptt. Of Computer Sc. And Applications
BBK DAV College for Women, Amritsar

Anurag Gupta
P.G. Deptt. Of Computer Sc. And Applications
BBK DAV College for Women, Amritsar

ABSTRACT
Task scheduling in the cloud is taken as NP-Complete problem and meta-heuristic scheduling methods are frequently used to deal with these problems. Here, hybrid meta-heuristics with a task duplication method was developed to optimize task scheduling problem. The suggested method makes use of Particle Swarm Optimization and Genetic Algorithm characteristics. A cloud-based methodology was created by taking into account a well-known Fast Fourier Transformation approach utilizing Directed Acyclic Graph in order to mimic the proposed technique. The hybrid meta-heuristics can schedule the workloads on available top-tier servers in an optimistic manner. Additionally, task duplication may result in less communication between processors. Various tests demonstrated for better performance of proposed method than other scheduling techniques.

Keywords
Genetic Algorithm, Task Scheduling, Task Duplication, Particle Swarm Optimization, Meta-heuristics Technique, Directed Acyclic Graph, Fast Fourier Transformation.

.

1. INTRODUCTION
Cloud computing is known as a provider of dynamic services using very large scalable and virtualized resources over the Internet. [1]. Task scheduling is essential in Cloud Computing (CC) and resource scheduling processes be used for this [2]. The tasks/jobs submitted to this cloud environment needs to be executed on time using the resources available so as to achieve proper resource utilization, efficiency and lesser makespan which in turn requires efficient task scheduling algorithm for proper task allocation. [3] while minimizing the overall cost of allocation is crucial in a CC environment [4]. As per the nature of the workflow scheduling for cloud computing environment researchers are dedicated to find out the optimal or near optimal solution based on different heuristics [5]. Task or Job duplication are some satisfied precedence restrictions in raising a system and loading of jobs to get a timetable from a solution to linear approach [6]. The job duplication based scheduling strategy can reduce interprocessor communication. Task duplication's main objectives [7] are to break the deadlock, reduce communication costs, and enhance the program's process-to-communication ratio [8]. The use of Genetic Algorithm (GA) based scheduling to plan out tasks in a cloud setting has been proven to be promising [9]. However, it has a slow rate of convergence and occasionally gets stuck in local optima [10]. Another finding indicates that PSO (Particle Swarm Optimization) based scheduling approach is frequently employed scheduling method in cloud environments [11]. However, PSO occasionally has a problem with the wrong particle being originally selected. To avoid any local minima trapping and also premature convergence through best possible utilization of GA's mutation operator over the entire population [12], a hybrid technique of GA and PSO can be applied.

Section II is literature review. Section III goes on to examine the suggested methodology in more detail utilizing various techniques. In Section IV, Experimental details, a simulation environment, a test bed, and the necessary settings are all illustrated. Proposed solution is contrasted with the current scheduling methodology. In Section V The proposed technique is proved as more effective than earlier techniques in the form of conclusion.

II. RELATED WORK

Task scheduling in Cloud Computing is one of the most difficult problems. Issue relates to the class of non-deterministic polynomial-time (NP)-hard issues. Several heuristics as well as Meta-heuristic methods are considered for finding optimum solution. The target is the best usability of computing resources and speedup turnaround time. As per literature, a variety of meta-heuristic algorithms have been utilized viz., Genetic algorithms, Variable neighborhood search (VNS), Particle Swarm Optimization, Artificial Bee Colony (ABC), Ant colony optimization (ACO), and their hybrid variations. Following are the major milestones in this regard.

GA enabled PSO known as PSOGA is a mixture of two techniques was introduced by Agarwal et al. PSOGA makes use of the intensification and diversification properties of the PSO. In majority of the cases, the proposed algorithm presents less makespan time improvement in system performance, demonstrating the proposed PSOGA algorithm's superiority over other techniques that are taken into consideration [13].

Task Assignment issue (TAP) is NP-hard issue and Visalakshi, P., and S. N. Sivanandam published a Hybrid Particle Swarm Optimization (HPSO) technique for tackling it. When used to solve the job assignment problem, the HPSO produces a better outcome than the Normal PSO. The outcomes of PSO and HPSO are also contrasted with those of the Genetic Algorithm (GA), a well-liked heuristic optimization technique. The outcomes suggest that the PSO performs superior to the GA [14].

In order to reduce the overall execution time, Kumar et al. suggested a hybrid task scheduling approach that combined Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) techniques. Particle swarm optimization (PSO) assisted the hybrid Genetic method-Particle Swarm Optimization (GA-PSO) method in achieving superior results in comparison to those of a normal genetic algorithm, Min-Min, and Max-Min algorithms [15].

For accurate interval computation, Li et al. proposed probability oriented preference-ratio approach having ranking interval. The goal of the hybrid genetic algorithm (GA) and particle swarm optimization (PSO) algorithm is to arrive at a good solution. Six trials that were adapted from well-known IPPS benchmark issues were utilized to gauge how well the suggested approach performed. The experimental findings show that the suggested algorithm has improved significantly and is useful for the uncertain IPPS problem [16].

In order to enhance work scheduling behavior, Velliangiri et al. presented Hybrid Electro Search with a Genetic Algorithm (HESGA), which takes into account factors including makespan, load balancing, resource utilization, and multi-cloud cost. The best global optimal solutions are produced by the Electro search algorithm, whilst the best local optimal solutions are produced by the genetic algorithm. The suggested technique outperforms current scheduling methods as the Hybrid PSO and GA known as HPSOGA are better than ACO and GA etc. [17].

After improving evolutionary algorithms and particle swarm optimization techniques, Jana et al. introduced a unique scheduling algorithm that can reduce waiting times for specific clients in a cloud environment and improve response times from cloud providers [18].

In order to optimize the task scheduling of AGVs, Mousavi et al. created a mathematical model and integrated it with evolutionary algorithms (genetic algorithm (GA), particle swarm optimization (PSO), and hybrid GA-PSO). The objectives were to reduce the makespan and number of AGVs while taking into account the AGVs' battery charge. The hybrid GA-PSO outperformed the other two algorithms and delivered the best outcome [19].

By incorporating genetic operators (crossover/mutation) for updating particles and combining PSO with operators of GA known as GPSO, Niu et al. reinterpreted and modified PSO. This is successfully used to address the posed issue. The GPSO is evaluated using some benchmarks for evaluating fuzzy processing-time. In comparison to GA, proposed method's viability and effectiveness are evaluated [20].

A hybrid particle swarm optimization and hill climbing technique was suggested by Dordaie to shorten the task scheduling timeline. Experimental findings based upon some scientific and random DAG demonstrated that the suggested approach outperforms the currently used heuristic and particle swarm optimization algorithms in terms of makespan [21].

Particle swarm optimization (PSO) and the genetic algorithm (GA) are two heuristics algorithms that Nzanywayingoma et al. devised. The performance of the experimental results compared to benchmark functions revealed that the particle swarm optimization (PSO) outperforms the genetic algorithm (GA), but that they both exhibit comparable characteristics due to their population-based search techniques. The outcomes also demonstrated that the suggested hybrid models outperformed the conventional PSO, substantially shortened execution times, and required fewer computing resources for processing [22].

Fu et al. suggested the PSO_PGA, which is based upon phagocytosis. The new technique is compared to various other existing algorithms through simulation experiments, and the findings demonstrate that the suggested algorithm has higher convergence accuracy and greatly reduces the overall completion time of cloud workloads. It demonstrates the algorithm's efficiency in scheduling cloud-based tasks [23].

Arzoo and Anil Kumar introduced Ant particle swarm genetic algorithm (APSGA), a mix of PSO-ACO-GA applied for scheduling tasks on CC VM (virtual machines). ACO distributes the job on particular virtual machines and produces enhanced results for characteristics like CPU utilization, makespan, and execution time. The CPU utilization and execution time have been maximized [24].

Ghosh devised a hybrid technique that skillfully integrates GA with Particle Swarm Optimization (PSO) while scheduling Grid jobs. The hybrid GA-PSO attempts to shorten the flow time and schedule makespan. Outputs of comparison table showed about proposed method performance that was best than other algorithms [25].

Musa et al. suggested an improved combination of the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) (GA-PSO) to get beyond the limitations of randomization and speed up convergence. Applying the enhanced GA-PSO with SPV is suggested for scheduling workloads to cloud computing resources effectively. The outcome showed that, in terms of makespan and resource utilization, the proposed GA-PSO algorithm outperformed the traditional hybrid GA-PSO algorithm [26].

Agarwal et al. considered task-scheduling using PSO to avoid premature convergence and to speed up the convergence of standard PSO. They compared this to some task scheduling schemes considering max-min methods, PSO, modified-PSO, GA, for finding minimum execution and completion time [27].

Singh et al. suggested a hybrid GA-modified PSO approach to effectively distribute jobs to the resources. The Hybrid GA (Genetic Algorithm)-modified PSO technique seeks to be less makespan, less costly, and minimize the energy consumption across heterogeneous resources in cloud-fog computing scenarios by balancing the burden of dependent operations. Outputs of experiment showed about hybrid GA-modified PSO for reduction execution time comparative to other algorithms [28].

Optimising several objectives, Liu et al. introduced a SLPSO model and a local search methodology defined by evaluating the fitness function for better swarm results [29]. While for reducing original PSO's execution time for task scheduling in CC environment Pirozmand et al. proposed a multi-adaptive based learning approach called IPSO using some criterias like load balancing, stability, makespan and efficiency [30].

III. PROPOSED METHODOLOGY

For task scheduling system in heterogeneous computing environment, an inter-connected complete network with high end speed processors is considered. On all interconnected channels, the entire system is operated at the same bandwidth and processing speed. A DAG G with n-1 edges, also known as sub-task dependencies and n vertices, also known as jobs, is taken into consideration as shown in figure 1.

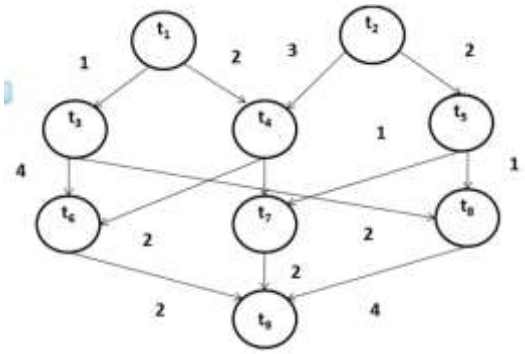

Figure 1: DAG

Determining of communication cost between sub-tasks is necessary if they are not connected to the same server in diverse environment. The entire system is assumed to operate at same bandwidth as well as same processing speed in this connected system. Here, DAG known as G having n edges, or sub-task dependencies, and n vertices, or jobs was analyzed as shown in figure 2.

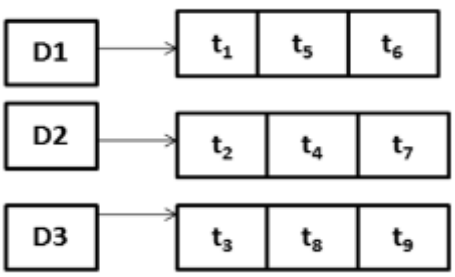

Figure 2: Task dependencies

The communication cost between two subtasks is needed to be calculated if they are not connected to the same server . Communication cost in the form cost matrix for nine jobs processing on three servers is assumed as in figure 3.

Cost matrix for 9 jobs assignment on 3 servers

Jobs	$Ð_0$	$Ð_1$	$Ð_2$	$\overline{E}$(blevel)	Order
t_1	31	32	33	32	1
t_2	32	28	33	31	2
t_3	34	30	38	34	5
t_4	47	50	39	45	4
t_5	35	32	38	35	3
t_6	29	34	38	33	8
t_7	39	32	34	34	6
t_8	34	30	35	33	9
t_9	35	32	35	34	7

Figure 3: Cost Matrix

Functioning of GA:

Solutions are referred to as chromosomes or people in GA. The selection, crossover, and mutation operators are used by the GA for identifying the better-optimized values and to converge. An initial population that is a collection of random solutions is created. Once objective values are evaluated for every random solution, selection operator goes into operation. Make-span is employed as fitness function. Mutation and crossover are GA's operators for refining chosen random solutions with the smallest makespan. Up until the halting requirement is not met, all of these procedures are performed repeatedly. Number of objective function computations (OF_e) are done for finding stopping step in this algorithm. A solution was automatically returned by GA as OF_e == Max_e met. A complete flow of GA is as shown in figure 4.

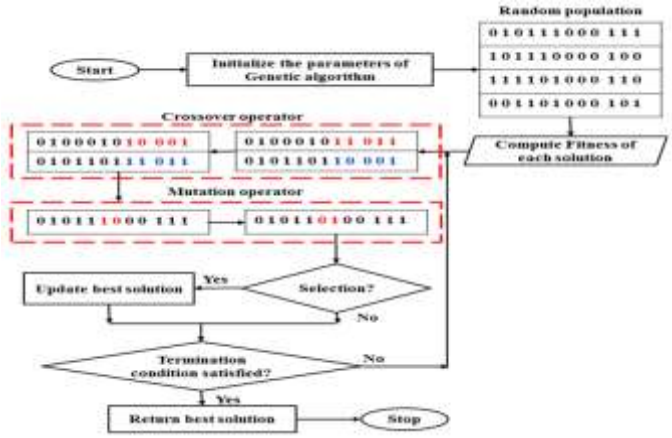

Figure 4: GA flow chart

Population Creation in GA :

The chromosome set is first created through encoding, and each chromosome in the population set has a solution. With the aid of the permutation process, integer values 0, 1, 2, 3, and n 1 are randomly chosen in order to create chromosomes. The initial population size is Psize = 4. For better outputs considering starting population values as indicated in DAG. It presents better priority queues and most usable heuristic rank way-outs as Top-level, bottom-level and Top-bottom ranks and these are represented in Eqs. (1), (2), and (3) as:

$$T(w_a) = \overline{CC(w_a)} + \max_{w_s \in succ(w_a)} (QC(w_a, w_s) + T(w_s)) \ldots\ldots\ldots\ldots (1)$$

Where $\overline{CC(w_a)}$ is average computational cost of sub-job w_a , $QC(w_a, w_s)$ is quantity of communication between the sub-job w_a and w_s and T(w_s) is upward rank of sub-job w_a 's successor.

$$B(w_a) = \max_{w_s \in pred(w_a)} (T(w_s) + \overline{CC(w_s)} + QC(w_a, w_s)) \dots\dots\dots\dots (2)$$

Where $B(w_a)$ is downward rank of the sub-job w_a's precedence.

$$TB(w_s) = \begin{cases} 0, & if\ w_a = w_{entry}; \\ \max_{w_s \in pred(w_a)} \left((TB(w_s)) + 1 \right), & Otherwise \end{cases} \dots\dots\dots\dots (3)$$

Where w_s is the sub-job w_a's precedence.

Assignment of sub-job to high-end processors:

Every person should have a primary priority mechanism with a permutation process in the case of an origin population so, for this process, sub-jobs should adhere to the rules of precedence. The fastest server will be given a sub-job if and only if it hasn't previously been scheduled. In the suggested approach, the HEFT technique is used to identify the sub-jobs with the highest level of individual priority. Additionally, it assigns certain sub-jobs to the server(s) in order to reduce the overall makespan. As determined by Equations (4) and (5), Earliest-start-time (E_{st}) of some sub-jobs w_a on computing machines P_i represented as $E_{st}(w_a, P_i)$.

$$E_{st}(w_{entry}, P_i) = 0 \dots\dots\dots\dots\dots\dots\dots (4)$$

$$E_{st}(w_a, P_i) = \max_{w_a \in pred(w_s)} (A_{st}(w_s) + QC(w_a, w_s)) \dots\dots\dots\dots (5)$$

The Actual-start-time of sub-job w_a on processor P_i is symbolized as $A_{st}(w_a, P_i)$. It is obtained by Eq. (6) as:

$$A_{st} = \max(E_{st}(w_a, P_i), available(P_i)) \dots\dots\dots\dots\dots\dots (6)$$

Where $available(P_i)$ is computing time for machines P_i and always in ready position for execution purposed. Note that Earliest-finish-time of sub-job w_a on computing machines P_i are represented by $E_{ft}(w_a, P_i)$, which is obtained by Eq. (7) as:

$$E_{ft}(w_a, P_i) = CCost(w_a, P_i) + A_{st}(w_a, P_i) \dots\dots\dots\dots (7)$$

Where $CCost(w_a, P_i)$ is computational-cost of sub-job w_a on processing machines P_i. Also Actual-finish-time of sub-job w_a on P_i is $A_{ft}(w_a, P_i)$ is obtained by Eq. (8) as:

$$A_{ft}(w_a, P_i) = \min_{1 \le i \le P} E_{ft}(w_a, P_i) \dots\dots\dots\dots\dots (8)$$

A noteworthy accomplishment in the proposed system is the assignment of sub-jobs and the allocation of sub-jobs to high-end processors. For experimental purposes, the work duplication rate in this case is set to 0.5. That means there is a 50% chance that a particular job will be replicated on any high-end processors that are available.

Computing Fitness Function in GA:

Finding and computing the optimum answer from the available chromosomes is the importance of the fitness function. Most researchers working now have used makespan as a fitness function [31]. The term "makespan" or "M_{span}" refers to the longest possible schedule. Eq. (9) gives the definition of makespan as:

$$M_{span} = A_{ft}(w_{exit}) \dots\dots\dots\dots\dots\dots (9)$$

In GA, Eq. (10) calculates a chromosome's Fitness-value (FV) as follows:

$$FV = M_{span} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (10)$$

Crossover operation in GA:

Crossover operator is necessary to achieve variety and improved evolution in the population set. Current research indicates that the crossover operator's function in the suggested evolutionary approach is to alter the population. Diversity on both the parent and child sides will arise with help of the crossover point or crossover rate. Consequently, it produces four siblings from two sets of parents with variety in solution sets, allowing for best-optimized outcomes.

Functioning of Mutation in GA:

Applying varieties in chromosome populations with a certain mutation probability rate, the mutation operator is utilised in the GA. Here, a specific gene is arbitrarily switched out for another gene, and the makespan is calculated to optimise the result. The process is continued until optimal outcomes are achieved.

Termination Requirement:

Since achieving Makespan as "0" is not possible, each section of the algorithm is stopped after 100 function evaluations. The term "function evaluations" refers to the total number of times the so-called "Fitness function" was calculated.

Functioning of PSO:

The GA's upbeat timetable serves as the PSO's initial batch of particles, helping it to overcome the problem that "poorly selected particles tend to poor results." The optimum schedule is determined by taking velocity and position of every particle and by applying Smallest- Position-Value procedure for obtaining all jobs for every particle having all possible combinations. Every permutation's makespan evaluation depends on how the jobs are represented. The makespan evaluation can be used to calculate personal best (pBest). In each cycle, the particle's optimum position is determined by its minimum makespan. The pBest can be enhanced by comparing the new personal with the old one. If the new one is better than the old one and has a lower value than the old one, the two are updated. The set of computed pBest can be used to assess gBest. The gBest is the pBest with the lowest value among all pBest. If the iteration counter goes over the maximum value, the procedure can be stopped (it also compares the maximum use of CPU time). A complete operation of PSO is as shown in figure 5.

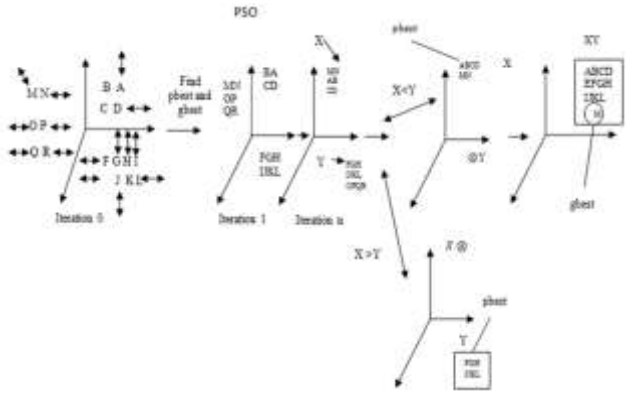

Figure 5: PSO Processing

A complete flow chart showing the relationship of GA, PSO and Task Duplication (TD) is as shown in figure 6.

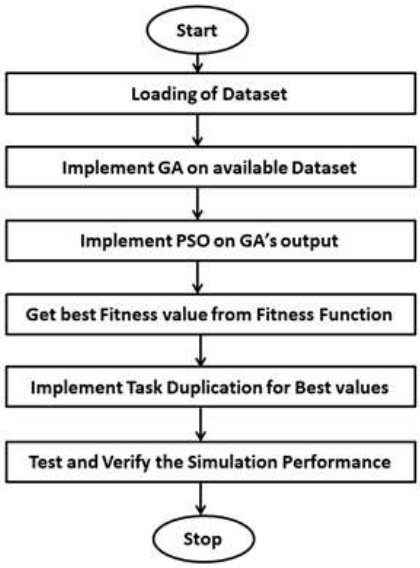

Figure 6: A complete flow showing relationship of GA, PSO and TD

IV. EXPERIMENT AND RESULTS

System performance is enhanced by the proposed GPTD (Genetic and Particle Swarm Optimization with Task Duplication) technique using test-bed criteria. The values of the graph are used for representation of characteristics of contemporary parallel approaches that employ by Fast Fourier Transformation (FFT) graph. Numerous tasks having matrix size m with some new parameters used in the development of the task graph. FFT graph is used for task graphs with appropriate matrix sizes of 2, 3, and 4. FFT has some nodes with client machines and VMs are identical to m2, (m2 + m)/2, and mlog2m + 2m + 1 correspondingly due to the same network layout with changes in matrix sizes. FFT to DAG task graph is as shown in figure 7.

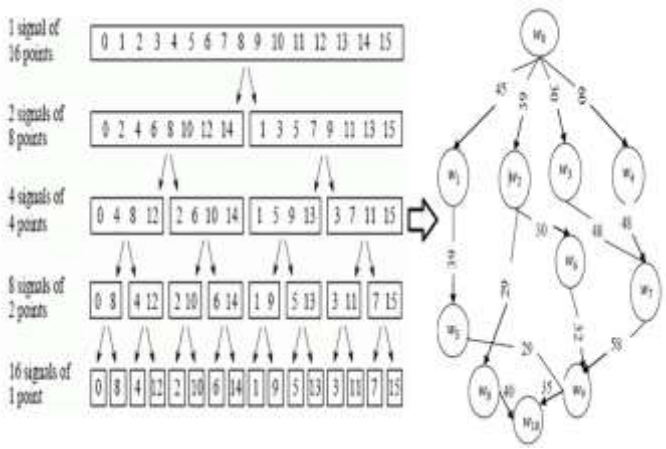

Figure 7: FFT to DAG

Two well-known techniques, including GA and PSO, are used to compare the GPTD methodology. In the MATLAB (2013a) version, GPTD, GA and PSO are implemented. Intel Core i3, 2.9 GHz with 4GB RAM, some simulation processes are run. This study uses a population size value of 300 with 1000 generations are considered and also a binary tournament taken as selection and crossover GA's two major operators. Final sampling rate utilized is 0.05, mr (mutation rate) is 0.01, cr (crossover rate) is 0.5, with some constants as c1=c2=0.2 values.

Simulation process is performed 18 times. For every technique, the minimum, mean, and maximum values of each quality measure are recorded. The purpose of obtaining these minimum, maximum, and mean values is to evaluate the variations that come from the meta-heuristic approaches' random behavior. The algorithms are computed using many criteria, including makespan, speedup, efficiency, and utilization. Speed-up is a metric used to demonstrate superior performance when contrasting the suggested strategies with current methods.

Table 1 compares the makespans (msp) of the three procedures— GPTD, GA and PSO and clearly demonstrates proposed approach's makespan and found GPTD has significantly shorter msp as to other techniques. Table 2 compares GPTD, GA and PSO while taking the metric Speedup (sp) into account, and the results reveal that the GPTD approach is more significant than the other strategies that are currently in use. By using the efficiency metric (ef), Table 3 showed comparison report by considering some parameters of GPTD, which is better than the existing techniques GA and PSO. Taking utilization analysis (ut), Table 4 compares the GPTD, GA and PSO and is more important for the improvement of scheduling criteria by taking into account the utilization parameter.

Table 1: Analysis of Makespan (msp)

PM	*FFT*	*PSO*	*GA*	*GPTD*
2	4	11008 ± 80	11208 ± 125	10040 ± 66
	8	15900 ± 140	17199 ± 190	15599 ± 125
	16	22023 ± 190	23133 ± 280	21400 ± 166
4	4	8280 ± 195	9250 ± 237	4695 ± 177
	8	9585 ± 250	12011 ± 300	9355 ± 200
	16	11390 ± 270	14202 ± 290	10900 ± 220
8	4	4888 ± 170	9280 ± 270	7430 ± 140
	8	7944 ± 190	10222 ± 310	7560 ± 160
	16	7350 ± 180	8790 ± 230	7140 ± 160

Table 2: Analysis of Speedup (sp)

PM	*FFT*	*PSO*	*GA*	*GPTD*
2	4	1.90 ± 0.12	1.70 ± 0.14	1.93 ± 0.11
	8	2.10 ± 0.10	2.00 ± 0.11	2.13 ± 0.12
	16	2.14 ± 0.09	2.00 ± 0.10	2.22 ± 0.07
4	4	2.05 ± 0.12	2.00 ± 0.11	2.18 ± 0.08
	8	2.18 ± 0.14	2.17 ± 0.10	2.33 ± 0.10
	16	2.11 ± 0.10	2.22 ± 0.14	2.30 ± 0.07
8	4	2.18 ± 0.12	2.00 ± 0.19	2.22 ± 0.02
	8	2.11 ± 0.10	2.08 ± 0.16	2.33 ± 0.10
	16	2.19 ± 0.19	2.14 ± 0.18	2.28 ± 0.08

Table 3: Analysis of Efficiency (ef)

PM	*FFT*	*PSO*	*GA*	*GPTD*
2	4	0.86 ± 0.054	0.83 ± 0.058	0.92 ± 0.051

	8	0.88 ± 0.050	0.82 ± 0.060	0.92 ± 0.030
	16	0.88 ± 0.040	0.80 ± 0.070	0.90 ± 0.030
4	4	0.56 ± 0.190	0.48 ± 0.254	0.80 ± 0.094
	8	0.77 ± 0.129	0.68 ± 0.150	0.87 ± 0.070
	16	0.84 ± 0.077	0.68 ± 0.140	0.88 ± 0.041
8	4	0.31 ± 0.280	0.24 ± 0.210	0.44 ± 0.135
	8	0.55 ± 0.150	0.34 ± 0.250	0.68 ± 0.090
	16	0.80 ± 0.090	0.60 ± 0.190	0.83 ± 0.080

Table 4: Analysis of Utilization (ut)

PM	*FFT*	*PSO*	*GA*	*GPTD*
2	4	0.76 ± 0.300	0.66 ± 0.400	0.86 ± 0.200
	8	0.83 ± 0.250	0.73 ± 0.270	0.89 ± 0.150
	16	0.88 ± 0.214	0.80 ± 0.244	0.98 ± 0.044
4	4	0.50 ± 0.200	0.46 ± 0.215	0.56 ± 0.130
	8	0.73 ± 0.150	0.63 ± 0.150	0.80 ± 0.090
	16	0.88 ± 0.074	0.78 ± 0.084	0.98 ± 0.034
8	4	0.20 ± 0.190	0.16 ± 0.320	0.36 ± 0.200
	8	0.43 ± 0.150	0.33 ± 0.230	0.53 ± 0.150
	16	0.68 ± 0.114	0.58 ± 0.164	0.80 ± 0.074

Above comparative tables demonstrated and analyzed some performance metrics mspan, sp, ut and ef. Finally found that suggested methodology performs better than other available methods. Therefore, suggested method is best suited in real-time CC environments, giving cloud consumers excellent availability.

V. CONCLUSION

The majority of current meta-heuristic scheduling mechanisms suffer from local search and slow convergence speed issues according to analysis of scheduling approaches currently used so a hybrid meta-heuristic with a task duplication method is suggested for optimizing task scheduling. The features of GA and PSO are used in the suggested strategy. The well-known FFT problem is taken into account when implementing the cloud-based paradigm utilizing DAG. By leveraging task duplication, the hybrid scheduling method may schedule the tasks effectively while simultaneously reducing inter-processor communication. Numerous tests have revealed that the proposed method performs better than the individual performance of GA and PSO.

REFERENCES

[1]. Pinal Salot , “A survey of various scheduling algorithm in cloud computing environment “,IJRET: International Journal of Research in Engineering and Technology ISSN: 2319-1163

[2]. L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, “A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing,” IEEE Access, vol. 3, pp. 2687–2699, 2015.

[3]. Shubam Mittal, Avita Kattal, "An Optimized Task Scheduling Algorithm in Cloud Computing", Published in 2016 IEEE 6th International Conference on Advajnced Computing, 18 August 2016, DOI: 10.1109/IACC.2016.45

[4]. X. Xu, L. Cao, X. Wang, X. Xu, L. Cao, and X. Wang, "Resource pre-allocation algorithms for low-energy task scheduling of cloud computing," Journal of Systems Engineering and Electronics, vol. 27, pp. 457–469, April 2016.

[5] Gupta, I., Kumar, M. S., & Jana, P. K. (2016, August). Task duplication-based workflow scheduling for heterogeneous cloud environment. In 2016 Ninth International Conference on Contemporary Computing (IC3) (pp. 1-7). IEEE.

[6]. Maiti, Biswaroop, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, and Aravindan Vijayaraghavan. "Scheduling precedence-constrained jobs on related machines with communication delay." In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 834-845. IEEE, 2020.

[7]. Q. Tang, S. F. Wu, J. W. Shi, and J. B. Wei, "Optimization of duplication based schedules on network-on-chip based multi-processor system-on-chips," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 826–837, March 2017.

[8]. C.-I. Park and T.-Y. Choe, "An optimal scheduling algorithm based on task duplication," IEEE Transactions on Computers, vol. 51, pp. 444–448, Apr 2002.

[9]. J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," IEEE Access, vol. 4, pp. 5065– 5082, 2016.

[10]. H. He, G. Xu, S. Pang, and Z. Zhao, "Amts: Adaptive multi-objective task scheduling strategy in cloud computing," China Communications, vol. 13, pp. 162–171, April 2016.

[11]. X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," IEEE Transactions on Automation Science and Engineering, vol. 11, pp. 564–573, April 2014.

[12]. W. E. Costa, M. C. Goldbarg, and E. G. Goldbarg, "New {VNS} heuristic for total flowtime flowshop scheduling problem," Expert Systems with Applications, vol. 39, no. 9, pp. 8149 – 8161, 2012.

[13]. Sharma, M., Singh, G. and Singh, R., 2019. Design of GA and ontology based NLP frameworks for online opinion mining. Recent Patents on Engineering, 13(2), pp.159-165..

[14]. Visalakshi, P., and S. N. Sivanandam. "Dynamic task scheduling with load balancing using hybrid particle swarm optimization." *Int. J. Open Problems Compt. Math* 2, no. 3 (2009): 475-488.

[15]. Monga, Preeti, Manik Sharma, and Sanjeev Kumar Sharma. "Performance analysis of machine learning and soft computing techniques in diagnosis of behavioral disorders." In Electronic Systems and Intelligent Computing: Proceedings of ESIC 2021, pp. 85-99. Singapore: Springer Nature Singapore, 2022.

[16]. Li, Xinyu, Liang Gao, Wenwen Wang, Cuiyu Wang, and Long Wen. "Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time." *Computers & Industrial Engineering* 135 (2019): 1036-1046.

[17]. Velliangiri, S., P. Karthikeyan, VM Arul Xavier, and D. Baswaraj. "Hybrid electro search with genetic algorithm for task scheduling in cloud computing." *Ain Shams Engineering Journal* 12, no. 1 (2021): 631-639.

[18]. Jana, Bappaditya, and Jayanta Poray. "A hybrid task scheduling approach based on genetic algorithm and particle swarm optimization technique in cloud environment." In *Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA*, pp. 607-614. Springer Singapore, 2018.

[19]. Mousavi, Maryam, Hwa Jen Yap, Siti Nurmaya Musa, Farzad Tahriri, and Siti Zawiah Md Dawal. "Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization." *PloS one* 12, no. 3 (2017): e0169817.

[20]. Niu, Qun, Bin Jiao, and Xingsheng Gu. "Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time." *Applied Mathematics and Computation* 205, no. 1 (2008): 148-158.

[21]. Dordaie, Negar, and Nima Jafari Navimipour. "A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments." *ICT Express* 4, no. 4 (2018): 199-202.

[22]. Nzanywayingoma, Frederic, and Yang Yang. "Analysis of particle swarm optimization and genetic algorithm based on task scheduling in cloud computing environment." *International Journal of Advanced Computer Science and Applications* 8, no. 1 (2017).

[23]. Fu, Xueliang, Yang Sun, Haifang Wang, and Honghui Li. "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm." *Cluster Computing* (2021): 1-10.

[24]. Arzoo, and Anil Kumar. "Hybrid ant particle swarm genetic algorithm (APSGA) for task scheduling in cloud computing." In *Information and Communication Technology for Competitive Strategies (ICTCS 2021) Intelligent Strategies for ICT*, pp. 9-20. Singapore: Springer Nature Singapore, 2022.

[25]. Ghosh, Tarun Kumar, Sanjoy Das, and Nabin Ghoshal. "Job scheduling in computational grid using a hybrid algorithm based on genetic algorithm and particle swarm optimization." In *Recent Advances in Intelligent Information Systems and Applied Mathematics*, pp. 873-885. Springer International Publishing, 2020.

[26]. Musa, Nehemiah, Abdulsalam Ya'U. Gital, Fatima U. Zambuk, Ali Muhammad Usman, Mubarak Almutairi, and Haruna Chiroma. "An enhanced hybrid genetic algorithm and particle swarm optimization based on small position values for tasks scheduling in cloud." In *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pp. 1-5. IEEE, 2020.

[27]. Agarwal, Mohit, and Gur Mauj Saran Srivastava. "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing." *Journal of Ambient Intelligence and Humanized Computing* 12, no. 10 (2021): 9855-9875.

[28]. Singh, Gyan, and Amit K. Chaturvedi. "Hybrid modified particle swarm optimization with genetic algorithm (GA) based workflow scheduling in cloud-fog environment for multi-objective optimization." *Cluster Computing* (2023): 1-18.

[29]. Liu, Xiao-Fang, Yongchun Fang, Zhi-Hui Zhan, and Jun Zhang. "Strength Learning Particle Swarm Optimization for Multiobjective Multirobot Task Scheduling." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2023).

[30]. Pirozmand, Poria, Hoda Jalalinejad, Ali Asghar Rahmani Hosseinabadi, Seyedsaeid Mirkamali, and Yingqiu Li. "An improved particle swarm optimization algorithm for task scheduling in cloud computing." *Journal of Ambient Intelligence and Humanized Computing* 14, no. 4 (2023): 4313-4327.