JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

Task Scheduling Using Proposed Enhanced Particle Swarm Optimization Approach in Manufacturing Grid

Avijit Bhowmick[1],

[1]Dr. B. C. Roy Engineering College, Durgapur, West Bengal, India.

Arup Kr Nandi[2],

[2]Principal Scientist, CSIR-CMERI, Durgapur, West Bengal, India.

Goutam Sutradhar[3]

[3]Director, National Institute of Technology, Manipur, India.

Corresponding author: Avijit Bhowmick, email- mr.avijit.bhowmick@gmail.com

Abstract— One of the most significant challenges to solve in order to fully realize the potential of a Manufacturing Grid-based infrastructure is the issue of work planning and scheduling. Even though it is very rare to acquire the perfect result & output in such distributed environment like Manufacturing Grid applying traditional algorithms, evidence from several experiments suggests that it is possible to get a sub-optimal solution by making use of heuristic algorithms. Utilizing either the PSO algorithm or the genetic algorithm approach are all feasible options for producing efficient schedules for completing tasks. All of these algorithms are designed to accomplish the same goal, which is to provide a schedule that will enable activities to be completed in the shortest amount of time feasible. We have suggested an enhanced PSO algorithm in this paper which is performing better than GA and PSO. We compare and contrast two different heuristic approaches GA and PSO with our enhanced PSO algorithm to Grid-based task scheduling to exhibit their similarities, differences and performance.

Keywords— Task scheduling, Genetic algorithms, Particle swarm optimization, and cloud computing.

1. Introduction

Modern manufacturing has experienced enormous change as a result of the fierce competition, economic and resource globalization, and the rapid advancement of superior manufacturing, information, computer, and management technology. The 1960s saw an increase in production scale, the 1970s saw a decrease in production costs, the 1980s saw an increase in product quality, and the 1990s saw an increase in market responsiveness. The present decade sees manufacturing placing more of an emphasis on knowledge and service. The introduction of computer and information technology, as well as the swift growth and application of Internet technologies, have all accelerated the advancement of manufacturing. Now, the ideas of computational grid and cloud computing have been combined to form the idea of Manufacturing Grid. Computing on grids, computing on dispersed networks, and computing in parallel all came before the advent of cloud computing. Manufacturing Grid is derived from the concept of computational grid. [1] It is a novel kind of computing for organizations that includes sharing a pool of resources such as computing resources and storage space. a number of dynamically available, networked computers that are shown as single or more integrated computing resources make up a distributed computing system, a sort of parallel and distributed system [2]. These characteristics are achieved through the use of service-level agreements that are negotiated between the service provider and the

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

consumers of the system. The cloud generally provides users with access to these three distinct types of services. Those who use the cloud have access to a variety of features, including a storage system and processing capabilities, thanks to the first kind of cloud service. The second kind of platform, and it enables users to construct applications on a server that is shared with other users. It gives customers the ability to access and utilize software without first requiring them to download and install the programme on their local workstations [3]. One of the biggest and most difficult problems in distributed environment is task scheduling, thus many academics have looked at the best way to schedule work on resources that are currently in use. Nonetheless, in most cases, arranging the timing of jobs is an NP-complete problem [4]. In recent years, heuristic optimization methods have gained popularity as a means of solving NP-complete problems. After a lengthy computation process, a solution to the issue that is not the best one imaginable is found using an evolutionary algorithm. To get the solution more quickly, a number of other heuristic-based methods are used. [5]. The genetic algorithm and simulated annealing both use natural events as models for their respective processes. A genetic algorithm may be thought of as a simulation of the process of natural selection. The fitness function assigns a ranking to each generation's solutions; The crossing over as well as mutation operators are then used to make new solutions, together with the progeny of the individuals thought to have the greatest fitness. There is discussion on task scheduling by employing a genetic algorithm [7], and there is also discussion on numerous variations on the original genetic algorithm [8]. The PSO algorithm is notable for being one of the more modern heuristic ones. Kennedy and Eberhart created this evolutionary algorithm, which is one of several. [9] and imitates how a group of animals acts in a social setting as they cooperate to accomplish a goal, such a flock of birds or a school of fish. Shih Tang Lo and colleagues offer a PSO approach in [10] that takes into consideration the challenge of scheduling resources across several machines.

Computer-aided process planning (CAPP), Computer-aided design (CAD), and computer-aided manufacturing (CAM) software, as well as various machine tools like rapid prototype manufacturing (RPM) also computerized numeric control (CNC), etc., are examples of manufacturing resources that are quite different from computing resources or data resources. So, task scheduling is most important issue in this regard when jobs are assigned effectively in those above-mentioned resources of Manufacturing Grid environment.

So, a manufacturing grid system built on an effective algorithm is necessary to effectively tackle the work scheduling problem in the Manufacturing Grid. PSO combines the flexibility and acceptable calculations of heuristic algorithms with the specific criteria of its kind, such as ease of implementation and dependability of performance. This results in an algorithm that meets both sets of requirements. The PSO method is extensively used in today's world, and its applications are as varied as structural optimization, the analysis and design of control systems, and the training of neural networks.

Within the scope of this study, we investigate how well the PSO technique, the genetic algorithm, and a PSO algorithm with certain modifications perform when applied to the problem of work scheduling within the context of distributed Manufacturing Grid environment.

II. Modeling of Typical Scheduling Problem

First, it is typically assumed that each task that a user submits is distinct from other tasks. In this investigation, we use the assumption that we have n jobs that need to be finished on m resources, and that we are aware of the amount of time required to complete each work

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

on each machine in advance. The overall execution time should be reduced down while at the same time resource utilization should be increased. The first-come, first-served principle is used to decide who gets which job when. In situations when there are more jobs available than there are machines, scheduling algorithms are used to distribute the load. This research makes the assumption that there are more tasks than there are resources available, which indicates that work cannot be transferred from one resource to another. The set of tasks, which consists of n unique tasks, is denoted by the notation $T_i i = \{1,2,3,4....,n\}$ and whereas the set of resources is referred to by the notation R_j $j = \{1,2,3,4...,m\}$.The objective is to create a matrix that reduces the overall resource use and cost of job execution and completion time where the x_{ij}-element of that matrix becomes 1 if task i is gets completed successfully by resource j and or else it becomes 0.
The two most significant conditions are considered in this section:

$$\sum x_{i,j} = 1 \ \forall j \in T \quad \text{.....(1)}$$
$$X_{i,j} \in \{0,1\} \ \forall i \in R, \ \forall j \in T \quad \text{.....(2)}$$

Makespan refers to the amount of time required to finish a project utilizing a certain processing resource and is measured in seconds. The first limitation ensures that just one processing resource is utilized for each task by mandating that this be the case.

A. Genetic algorithm for Task scheduling

As an evolutionary strategy for optimization, the genetic algorithm is one of the most popular one. To get the best outcomes, it makes comparisons between the principles of genetics and natural selection. This technique makes use of the natural operators offered by means of evolutionary algorithm in turn to get better the distinguishing characteristics of likely solutions. A genetic algorithm evolves in a manner that is analogous to the development of biological organisms. This approach is among the most promising models when it comes to replicating natural evolution and ranks among those methods. The components that make up a genetic algorithm may each be thought of as potential answers to the problem at hand. These responses are then improved by genetic operators over the course of multiple generations to represent either an optimal or a suboptimal solution of the trouble. According to the rules that are now in place for evolution, the only solutions with a possibility of making it to the next generation and producing offspring are those that are the most fit amongst the created population. An essential part of a genetic algorithm is the definition of the objective function, the specification of the genetic form of the issue that has to be solved, and the establishment of the genetic operators that will be applied in the solution process. In a genetic algorithm, the computer stands in for nature, individuals stand in for potential answers to the problem, The population symbolizes diversity of prospective replies, the fitness measures the effectiveness of the responses, and genes represent the actual possible responses.

The following is an outline of the numerous stages that make up a genetic algorithm:

Initialization: As a point of departure, a population chosen at random is generated. According to this point of view, each individual has worth as a potential response to the problem at hand.

Fitness function: When an initial population has been produced, a fitness function is applied to it in order to rank the individuals according to their level of quality. Every problem has its own one-of-a-kind fitness function, which is defined by the overall goal of the problem of optimization. The fitness function that is being employed in this scenario is the makespan metric.

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

Elitism and the process of picking the most qualified candidates: Elitism is the practice of passing on unchanged the best members of a population to succeeding generations, whereas selection is the practice of favoring the offspring of those who have higher fitness levels. Elitism is contrasted with selection, which is the practice of favoring the offspring of those who have higher fitness levels. In this particular instance, the selection will be made using a roulette wheel with an obvious bias.

Crossing over: The mating of carefully chosen individuals produces offspring in this area. Two chromosomes, which is the most frequent situation for crossover, one inherited from each parent, exchange sections of their respective DNA. In most cases, a kid will take on certain characteristics that are inherited from both of its parents. This problem demonstrates one method of using a crossover at a single point, which may be seen as an example.

Mutation: The genetic algorithm's supplementary method for investigating the cost surface is known as mutation. Through the process of randomly modifying one or more genes on a pair of chromosomes, mutation contributes to the preservation of genetic variability and prevents the development of a population structure that is restrictive.

B. PSO algorithm used for Task scheduling

B.1 Particle Swarm Optimization (PSO algorithm)

In the article [9], Kennedy and Eberhart present the PSO algorithm, which is one of the most recent iterations of heuristic optimization techniques. The PSO technique is one option that may be used in order to resolve optimization-related challenges. This algorithm creates a model of how a group of organisms, such as a school of fish or a flock of birds, may act in order to achieve their objective. This approach begins with establishing a population of particles as its base element. In evolutionary algorithms, each individual particle represents a different kind of probable solution that may be found. To begin, particles are generated at random; the placement of each particle represents an alternative approach to resolving the problem. Each particle has a defined location vector that is always being updated (X^i), and there is also a velocity vector that is similarly defined and also always being updated resulting to motion in the search space. Both of these vectors are being continually updated.

The following formula, which is based on the method described in Kennedy and Eberhart's [9] publication, may be used to determine any changes that have been made to the position vector:

$$X_{k+1}{}^{i} = X_k^{\,i} + V_{k+1}{}^{i} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(3)$$

In addition to this, the velocity vector is expressed as follows:

$$v_k{}^{i+1} = w_k v_k^{\,i} + c_1 r_1 (pbset_k{}^{i} - x_k{}^{i}) + c_2 r_2 (gbest - x_i{}^{k}) \qquad \ldots\ldots\ldots\ldots\ldots\ldots(4)$$

In the equation (4), we have a pair of constant integers, which are symbolized by the symbols c_1 and c_2, and a pair of regularly distributed numbers, which are denoted by the symbols r_1 and r_2. Both sets of numbers are located in the interval [0, 1]. [$-v_{max}$, $+v_{max}$] denotes the permissible change in velocity vectors within this range. In each new iteration of this approach, an application of a fitness function was put to each particle's position vector in order to determine the goodness of each individual particle. In PSO, the parameters pbest and gbest, which are described in (4), play very important roles. The best possible position (gbest) between any two particles, as well as the best possible placement (Pbest) for the ith particle since the algorithm started executing (Pbest). A particle can seek for both its best local location (pbest) and best global position using a velocity vector that is continually updated (gbest).

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

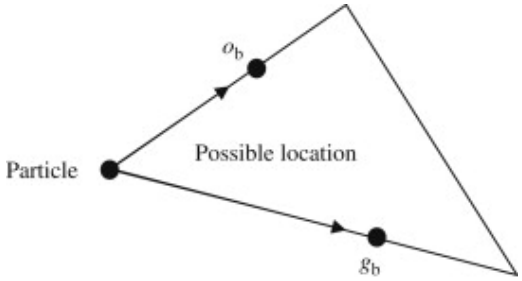
The necessary distance for a particle is represented by its velocity at the k^{th} iteration, with v_{id}^{k} standing for the particle's speed in the iteration's i-th particle. The relative acceleration coefficients are c1 and c2. The definition of the of inertia weight refers to the value of *w*. More comprehensive *w* searches may now be conducted in the enlarged region, whereas fewer such searches can be conducted in the initial zone. Making strategic decisions about the inertia weight and acceleration coefficient is required in order to achieve a balance that is satisfactory between local and global searches. After a particle's current location, its previous velocity, and its best possible position relative to its neighbors are used to calculate the particle's updated velocity, Based on the revised velocity vector, the particle travels in the direction of its new location. which was calculated using the particle's prior velocity. During the iterative procedure, each particle is given a grade by using a proportional evaluation function that has been set Iteratively repeat the process till the algorithm's ending condition has been met.

B.2 Task scheduling is done using the PSO algorithm

Establishing a trustworthy association between PSO particles and the responses that they relate to is the first stage in this process. In order to do this, we need to find out how to get the most out of the resources we have by increasing production as much as possible. PSO is capable of dealing with a wide variety of different optimization problems. If there are n jobs to be done and the objective is to divide them up across m different processing units, then the particles should be defined as *n x m* matrices. Position where m is entire available number of resource nodes with n is total no. of available task that need to be scheduled. Each particle's position vector is made up of two main parts which are as follows:

-The position matrix does not include any values other than zero or one.

-To put this way, each location vector matrix has only one value of 1 and no other values at all.



	T1	T2	T3	T4	T5	T6	T7	T8	T9
M1	1	0	1	0	0	0	0	1	0
M2	0	0	0	1	1	0	0	0	1
M3	0	1	0	0	0	1	1	0	0

Fig.1 Typical particle for 3 processing units and 9 independent tasks

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

Each column in this matrix represents a different machine that performs a certain operation. The second stipulation is that only one machine be used for the completion of each activity. For instance, In the position vector above, task 1 is completed on the first machine., task 2 is carried out on 2nd machine, task 3 is carried out on 3rd machine, and like this. The measures of a position matrix and a velocity matrix have the same dimensions. i.e., m x n . The following are the element ranges: $[-v_{max}, +v_{max}]$. Consequently, for each element in the population velocity matrix (vk is the velocity matrix of the kth particle),)

$$v_k(i,j) \in [-v_{max}, +v_{max}] \quad \text{.....(5)}$$
$$i \in \{1, 2, \ldots\ldots, m\}$$
$$(\forall i,j)$$
$$j \in \{1, 2, \ldots\ldots, n\}$$

Both pbest along with gbest are examples of *m x n* matrices, and much like the velocity matrix, the only possible entries are zeros and ones.

The matrix Pbest is used to represent the optimal beginning position for a single particle, while the variable gbest is used to store the optimal starting position for all particles taken together. Throughout the whole of the process, pbest and gbest are Enhanced by using the relevant formulas that were shown earlier. As the procedure moves forward, the fitness function is used to conduct an evaluation of the particles' overall quality. If the particle's fitness is greater than $pbest_k$, then x_k will be used in place of x_k pbest in the calculation. The pbest of all particles is considered while updating gbest. The current present gbest will be changed with the more recent and advantageous choice if there is a pbest that is better than it.

The velocity matrices of the particles are used in order to determine the most recent updates to their position vectors, which are then utilized in order to determine the new positions of the particles.

$$v_k^{(t+1)}(i,j) = w.\, v_k{}^t(i,j) + c_1 r_1 (pbest_k{}^t(i,j) - X_k{}^t(i,j) + c_2 r_2 (gbest_k{}^t(i,j) - X_k{}^t(i,j) \quad \text{.....(6)}$$

By comparing the values of all the rows and columns in each column, the position vector for the particles is updated. The element associated with the row with the greatest value is updated to 1, while the other elements in the column are set to 0. Our objectives serve as the foundation for the formulation of the fitness function that we use. In this investigation, our primary objective is to determine how we may decrease the amount of time required to complete the final work, also known as the Makespan. The length of time it took for a particular machine to do all of the tasks that were assigned to it is one of the factors that is considered while determining the fitness of a particle. Because the number of iterations required by our methodology is the same as that of the stop criterion, we will proceed in this manner until we reach our goal. The PSO algorithm is shown here in flowchart form.

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

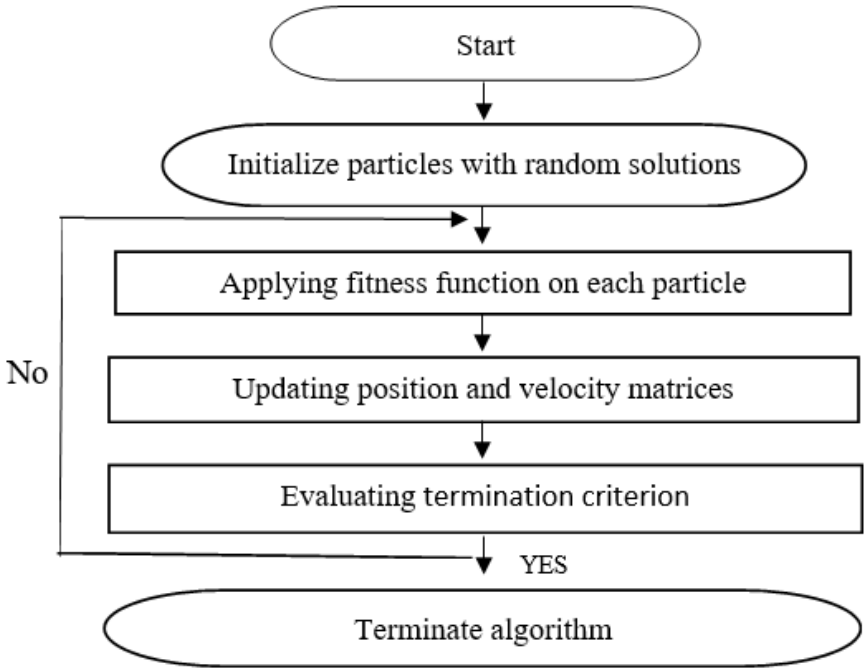
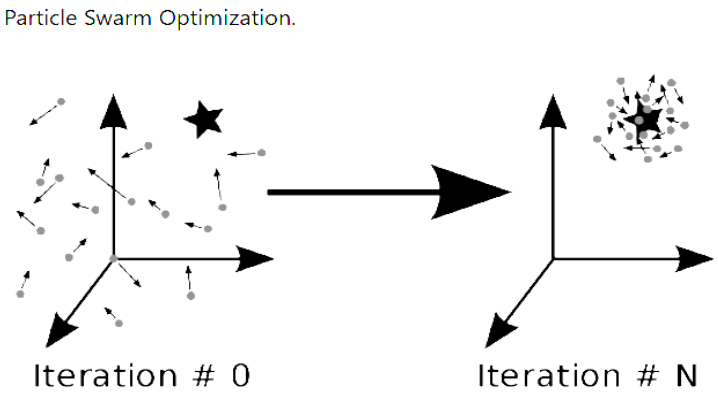


Fig.2 PSO algorithm Flowchart

C. Enhanced PSO Algorithm - Merging FCFS in PSO algorithm

We generate the starting population for a PSO algorithm taking into account the first-come-first-serve (FCFS), which increases the likelihood that the algorithm will converge on the optimal solution. In the first step of this procedure, we organize all of the active jobs and machines such that their level of execution increases. After sorting the tasks and mapping them to the appropriate machines, the next step is to assign the jobs. This is done instead of randomly creating the starting population for a PSO algorithm. As indicated in Figure 3 of the article that accompanies this strategy, the remaining steps of the process adhere to the same basic format as the PSO method that was used originally.



ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

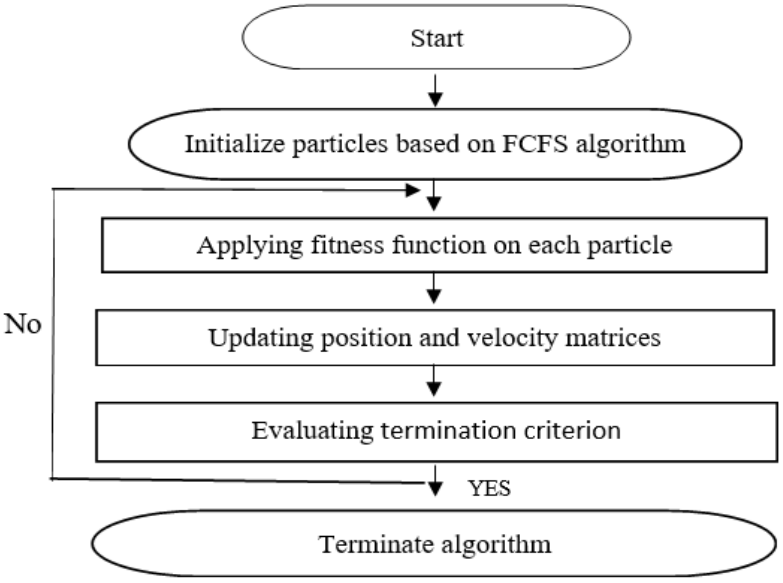
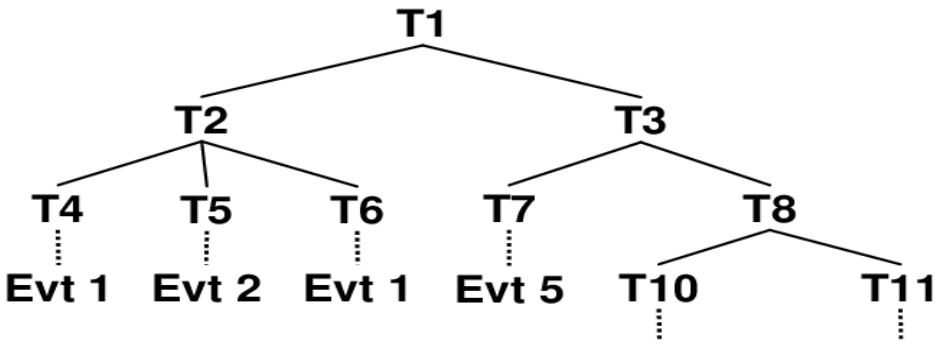


Fig. 3 Enhanced PSO algorithm Flowchart

III. Experiments and Results:

We have experimented using the Genetic algorithm in addition to the conventional PSO algorithm in order to see how well both of these algorithms perform in comparison to the proposed strategy. Whether or not we were successful in solving this task scheduling challenge will depend on Makespan. In order to evaluate the efficacy of our methodology, we put it through its paces using two distinct test scenarios. In the first possible outcome, there was an increase in the number of available positions from 100 to 800. In the second situation, iterations number is changed while number of jobs overall remains constant. Experiments are performed ten times, and the result is determined based on the average value. The makespan metric is used in order to determine how efficient this technique is. The percentage of advancement may be calculated based on the equation (7)

$$\varphi = \left(1 - \frac{\sum \text{Makespan}_{PSO}}{\sum \text{Makespan}_{GA}}\right) X\ 100 \qquad \text{.....(7)}$$



ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

Table I displays the GA parameters, while Table II displays the PSO and Enhanced PSO parameters utilised in Scenario 1:

TABLE I

GA PARAMETRS

Population size	60
Crossover probability	0.8
Mutation probability	0.01
Maximum no. of iterations	130

TABLE II

PSO AND ENHANCED PSO PARAMETERS

Population size	**60**
W	**0.65**
C1	**2**
C2	**2**
Maximum no. of iterations	**130**

TABLE III

FINAL RESULT FOR THREE ALGORITHMS

Scheduling Algorithm	No. of Tasks	No. of Resources	Makespan (s)	Improvement
Genetic Algorithm(GA)	100	4	667	2.25
PSO			667.80	
Enhanced PSO			655.25	
Genetic Algorithm(GA)	200	4	1120	10.65
PSO			1028	
Enhanced PSO			1005	
Genetic Algorithm(GA)	400	4	2030	2.70
PSO			2008	
Enhanced PSO			1976	
Genetic Algorithm(GA)	600	4	3335	3.98
PSO			3245	
Enhanced PSO			3209	
Genetic Algorithm(GA)	800	4	4732	6.69
PSO			4699	
Enhanced PSO			4418	

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

Fig. 4, where the x-axis indicates the no. of jobs and the makespan is represented in y-axis represents, we see how well these three algorithms perform under the given conditions.

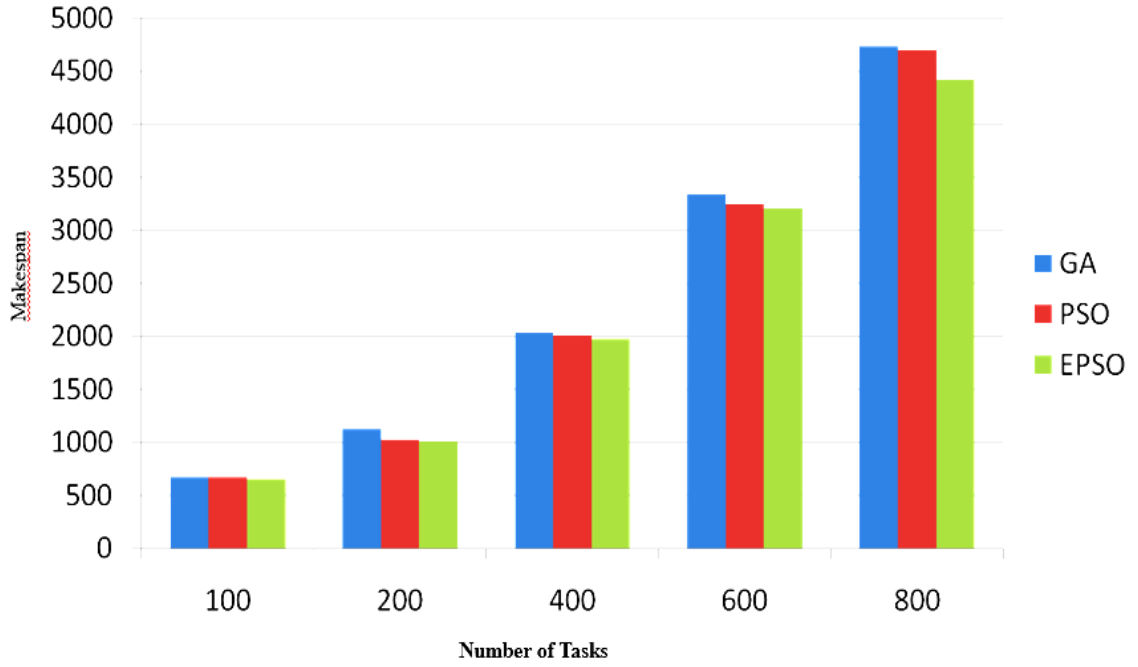


Fig.4 Comparison Performance of GA, PSO, Enhanced PSO Algorithm

Experiments for 100, 200, 400, and 800 jobs were carried out on four virtual machines, as can be seen in Table III and Figure 4, respectively. When measuring the efficacy of these algorithms, the "makespan," or the amount of time it takes for the execution of the very last job on the processing machines to be finished, is the criteria that is taken into consideration.

Based on the findings shown in table III and figure 4, the PSO method outperforms the Genetic algorithm in three sets of tests, although the Genetic algorithm performs better overall, all sets of trials, however, show that the Enhanced PSO method surpasses these two algorithms; more particular, this algorithm's makespan has the least value across all experiments.

The Enhanced PSO method beats these two algorithms, which is the basis for this conclusion. In the second scenario, With the exception of the number of iterations, which is raised from 20 to 360, all of the parameters are identical to those in tables II and III Throughout the course of the experiment, a fixed number of 400 tasks has been used.

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

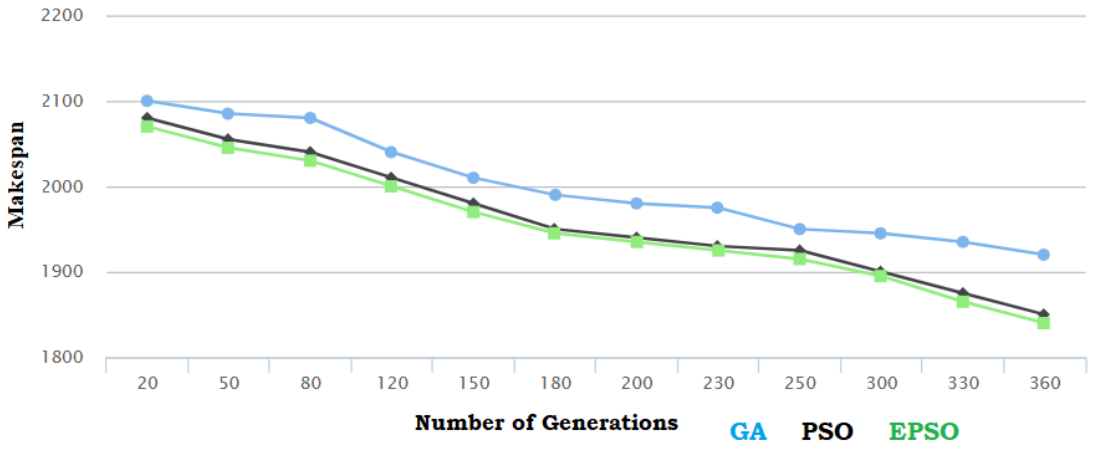


Figure 5 displays the experiment's outcomes.

As can be seen in Figure 5, increasing the generations number causes a drop in makespan. However, since the initial particles at the beginning of the process are adjusted, Solutions with a longer optimal makespan can be delivered by the enhanced PSO algorithm.

IV Conclusion

In this study, we examine how difficult it may be to schedule tasks while utilizing the geographically distributed Manufacturing Grid environment. When it comes to scheduling work in distributed systems, the PSO method and the Genetic algorithm have the highest good successful reputation. We suggest an altered version of the PSO technique, which generates an initial population by combining the FCFS algorithm with the standard PSO algorithm. This is done with the intention of shortening the amount of time the search takes to complete. Both the Genetic algorithm and the PSO algorithm deliver reasonable results, but on average, the PSO method outperforms the Genetic algorithm. According to our research, the PSO method outperforms the Genetic algorithm on average, while the Enhanced PSO algorithm outperforms the other two in terms of lowering makespan. This is due to the Enhanced PSO Algorithm's consideration of additional variables and additionally, the PSO algorithm underwent some changes enhanced by researchers. This approach may be beneficial in the distributive environment of manufacturing Grid to optimize the scheduling of work among the multiple geographically dispersed resources and minimising job completion time.

References:

1. China cloud computing. Peng Liu: cloud computing definition and characteristics, http://www.chinacloud.cn/.2009-2-25.
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud Computing and Emerging IT Platforms”, Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems 25(6), 599–616 (2009).
3. P. Kumar, A. Verma, “Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol2, Issue 5, May 2012.

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

4. Z. Yingfeng, L. Yulin, "Grid Computing Resource Management Scheduler Based on Evolution Algorithm[j]", Computer Engineering Conference, 2003, 29(15):1102175.
5. P. Roy, M. Mejbah, N. Das. "Heuristic Based Task Scheduling in Multiprocessor Systems with Genetic Algorithm by choosing the eligible processor", International Journal of Distributed and Parallel Systems (IJDPS), Vol3, No.4, July 2012.
6. Abraham, R. Buyya, and B. Nath." Nature's heuristics for scheduling jobs on computational Grids", 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India, 2000.
7. H. Yin, H. Wu, J. Zhou, "An Improved Genetic Algorithm with Limited It rat ion for Grid Scheduling", IEEE Sixth International Conference on Grid and Cooperative Computing, GCC 2007, Los Alamitos, CA, pp. 221-227, 2007.
8. R. Verma, S. Dhingra, "Genetic Algorithm for Multiprocessor Task Scheduling", IJCSMS International Journal of Computer Science and Management Studies, Vol.1, Issue 02, pp. 181-185, 2011.
9. J. Kennedy, R.C. Eberhart, "Particle swarm optimization", Proc, IEEE Conf. Neural Netw., vol. IV, IEEE, Piscataway, NJ, 1995, pp.1942-1948.
10. L. Zhang, Y. Chen, B. Yang "Task Scheduling Based on PSO Algorithm in Computational Grid", 2006 Proceedings of the 6th International Conference on Intelligent Systems Design and Applications, vol-2, 16-18 Oct, 2006, Jinan, China.
11. T. Chen, B. Zhang, X. Hao, Y. Dai, "Task scheduling in grid based on particle swarm optimization", The Fifth International Symposium on Parallel and Distributed Computing, ISPDC '06. pp. 238-245, 2006.
12. Wang-Tong Yang, Qin-bao An, Wu-jun jun. Manufacturing Grid and its key technology [J]. Computer Application and Software, 2006, (02).
13. I Foster, C Kesselman, J Nick, S Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration[J]. Global Grid Forum, 2002, (6):22.
14. Fan Yu Shun. Manufacturing grid with the concept of system architecture [J]. Aeronautical Manufacturing Technology, 2005, (10).
15. Hong Deng, Li Chen, Qian-Ni Deng, Cheng-Tao Wang. Virtual enterprise organizational change and manufacturing resource scheduling research based on manufacturing grid. Computer Integrated Manufacturing System, 2006, (08).
16. Cicirello V A, Smith S F. Wasp nests for self2configurable factories[A]. Proceedings of the Fifth International Conference on Autonomous Agents[C] , ACM Press , May-June 2001.
17. Li-Huixian, Chun-Tian Cheng and Liao-Jun Pang. Grid Environment efficient dynamic task scheduling algorithm [J]. Technology Journal of South China University (Natural Sciences), 2006.
18. Deng H, Chen L, Wang C, Deng Q (2006) A grid-based scheduling system of manufacturing resources for a virtual enterprise. Int J Adv Manuf Technol 28(1–2):137–141.

ISSN- 2394-5125 VOL 6, ISSUE 05, 2019

19. Chettry OVK, Gnanasekaran OC (1996) Modelling, simulation and scheduling of flexible assembly systems with coloured Petri nets. Int J Adv Manuf Technol 11(6):430– 438.

20. Chen J, Chen FF (2003) Performance modelling and evaluation of dynamic tool allocation in flexible manufacturing systems using Petri nets: an object-oriented approach. Int J Adv Manuf Technol 21(2):98–109.

21. Wu NQ, Zhou MC (2001) Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems. IEEE Trans Robot Autom 17(5):658–669.