On Performance Analysis of Hybrid Intelligent Algorithms (Improved PSO with SA and Improved PSO with AIS) with GA, PSO for Multiprocessor Job Scheduling

K.THANUSHKODI
Director
Akshaya College of Engineering
Anna University of Technology, Coimbatore
INDIA
thanush12@gmail.com

K.DEEBA
Associate Professor, Department of Computer Science and Engineering
Kalignar Karunanidhi Institute of Technology
Anna University of Technology, Coimbatore
INDIA
deeba.senthil@gmail.com

Abstract: - Many heuristic-based approaches have been applied to finding schedules that minimize the execution time of computing tasks on parallel processors. Particle Swarm Optimization is currently employed in several optimization and search problems due its ease and ability to find solutions successfully. A variant of PSO, called as Improved PSO has been developed in this paper and is hybridized with the AIS to achieve better solutions. This approach distinguishes itself from many existing approaches in two aspects In the Particle Swarm system, a novel concept for the distance and velocity of a particle is presented to pave the way for the job-scheduling problem. In the Artificial Immune System (AIS), the models of vaccination and receptor editing are designed to improve the immune performance. The proposed hybrid algorithm effectively exploits the capabilities of distributed and parallel computing of swarm intelligence approaches. The hybrid technique has been employed, inorder to improve the performance of improved PSO. This paper shows the application of hybrid improved PSO in Scheduling multiprocessor tasks. A comparative performance study is discussed for the intelligent hybrid algorithms (ImPSO with SA and ImPSO with AIS). It is observed that the proposed hybrid approach using ImPSO with AIS gives better results than intelligent hybrid algorithm using ImPSO with SA in solving multiprocessor job scheduling.

Key-Words: - PSO, Improved PSO, Simulated Annealing, Hybrid Improved PSO, Artificial Immune System (AIS), Job Scheduling, Finishing time, waiting time

1 Introduction

Scheduling, in general, is concerned with allocation of limited resources to certain tasks to optimize few performance criterion, like the completion time, waiting time or cost of production. Job scheduling problem is a popular problem in scheduling area of this kind. The importance of scheduling has increased in recent years due to the extravagant development of new process and technologies. Scheduling, in multiprocessor architecture, can be defined as assigning the tasks of precedence constrained task graph onto a set of processors and determine the sequence of execution of the tasks at each processor. A major factor in the efficient utilization of multiprocessor systems is the proper assignment and scheduling of computational tasks among the processors. This multiprocessor scheduling problem is known to be Non-deterministic Polynomial (NP) complete except in few cases [1].

Several research works has been carried out in the past decades, in the heuristic algorithms

for job scheduling and generally, since scheduling problems are NP- hard i.e., the time required to complete the problem to optimality increases exponentially with increasing problem size, the requirement of developing algorithms to find solution to these problem is of highly important and necessary. Some heuristic methods like branch and bound and prime and search [2], have been proposed earlier to solve this kind of problem. Also, the major set of heuristics for job scheduling onto multiprocessor architectures is based on list scheduling [3]-[9], [16]. However the time complexity increases exponentially for these conventional methods and becomes excessive for large problems. Then, the approximation schemes are often utilized to find a optimal solution. It has been reported in [3], [6] that the critical path list scheduling heuristic is within 5 % of the optimal solution 90% of the time when the communication cost is ignored, while in the worst case any list scheduling is within 50% of the optimal solution. The critical path list scheduling no longer provides 50% performance guarantee in the presence of non-negligible intertask communication delays [3]-[6], [16]. The greedy algorithm is also used for solving problem of this kind. In this paper a new hybrid algorithm based on Improved PSO (ImPSO) and AIS is developed to solve job scheduling in multiprocessor architecture with the objective of minimizing the job finishing time and waiting time.

In the forth coming sections, the proposed algorithms and the scheduling problems are discussed, followed by the study revealing the improvement of improved PSO.

In the next section, the process of job scheduling in multiprocessor architecture is discussed. Section 3 will introduce the application of the existing optimization algorithms and proposed improved optimization algorithm for the scheduling problem. Section 4 discusses the concept of simulated annealing, section 5 discusses AIS & 6, 7 and 8 discusses proposed Hybrid algorithms and followed by discussion and conclusion.

2 Job Scheduling in Multiprocessor Architecture

Job scheduling, considered in this paper, is an optimization problem in operating system in which the ideal jobs are assigned to resources at particular times which minimizes the total length of the schedule. Also, multiprocessing is the use of two or more central processing units within a single computer system. This also refers to the ability of the system to support more than one processor and/ or the ability to allocate tasks between them. In multiprocessor scheduling, each request is a job or process. A job scheduling policy uses the information associated with requests to decide which request should be serviced next. All requests waiting to be serviced are kept in a list of pending requests. Whenever scheduling is to be performed, the scheduler examines the pending requests and selects one for servicing. This request is handled over to server. A request leaves the server when it completes or when it is preempted by the scheduler, in which case it is put back into the list of pending requests. In either situation, scheduler performs scheduling to select the next request to be serviced. The scheduler records the information concerning each job in its data structure and maintains it all through the life of the request in the system. The schematic of job scheduling in a multiprocessor architecture is shown in Fig.1

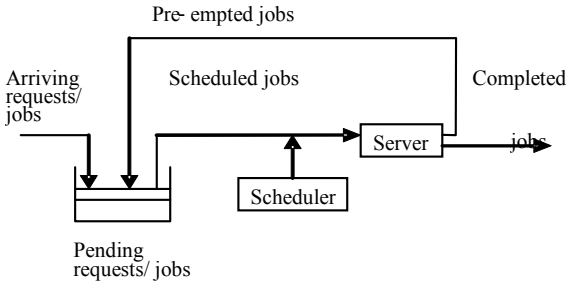


Fig 1. A Schematic of Job scheduling

2.1 Problem Definition

The job scheduling problem of a multiprocessor architecture is a scheduling problem to partition the jobs between different processors by attaining minimum finishing time and minimum waiting time

simultaneously. If N different processors and M different jobs are considered, the search space is given by equation (1),

$$\text{Size of search space} = \frac{(M \times N)!}{(N!)^M} \quad (1)$$

Earlier, Longest Processing Time (LPT), and Shortest Processing Time (SPT) and traditional optimization algorithms was used for solving these type of scheduling problems [10],[18]-[21],[27],[29]. When all the jobs are in ready queue and their respective time slice is determined, LPT selects the longest job and SPT selects the shortest job, thereby having shortest waiting time. Thus SPT is a typical algorithm which minimizes the waiting time. Basically, the total finishing time is defined as the total time taken for the processor to completed its job and the waiting time is defined as the average of time that each job waits in ready queue. The objective function defined for this problem using waiting time and finishing time is given by equation (2),

$$\text{Minimize} \quad \sum_{n=1}^{m_n} \omega_n f_n(x) \quad (2)$$

3 Optimization Techniques

Several heuristic traditional algorithms were used for solving the job scheduling in a multiprocessor architecture, which includes Genetic algorithm (GA), Particle Swarm Optimization (PSO) algorithm. In this paper a new hybrid proposed improved PSO with AIS is suggested for the job scheduling NP-hard problem. The following sections discuss on the application of these techniques to the considered problem.

3.1 Genetic Algorithm for Scheduling

Genetic algorithms are a kind of random search algorithms coming under evolutionary strategies which uses the natural selection and gene mechanism in nature for reference. The key concept of genetic algorithm is based on natural genetic rules and it uses random search space. GA was formulated by J Holland with a key advantage of adopting population search and exchanging the information of individuals in population [10], [11], [13], [15]-[22],[41]-[43]

The algorithm used to solve scheduling problem is as follows:

Step 1: Initialize the population to start the genetic algorithm Process.For initializing population, it is necessary to input number of processors, number of jobs and population size.

Step2: Evaluate the fitness function with the generated populations. For the problem defined, the fitness function is given by,

$$F = \begin{cases} V - TotalFinishingTime & f < V \\ \quad -\beta WaitingTime & \\ 0 & f \geq V \end{cases} \quad (3)$$

Where 'V ' should be set to select an appropriate positive number for ensuring the fitness of all good individuals to be positive in the solution space.

Step3: Perform selection process to select the best individual based on the fitness evaluated to participate in the next generation and eliminate the inferior. The job with the minimal finishing time and waiting time is the best individual corresponding to a particular generation.

Step4: For JSP problem, of this type, two – point crossover is applied to produce a new offspring. Two crossover points are generated uniformly in the mated parents at random, and then the two parents exchange the centre portion between these crossover points to create two new children. Newly produced children after crossover are passed to the mutation process.

Step 5: In this step, mutation operation is performed to further create new offsprings, which is necessary for adding diversity to the solution set. Here mutation is done, using flipping operation. Generally, mutation is adopted to avoid loss of information about the individuals during the process of evolution. In JSP problem, mutation is performed by setting a

random selected job to a random processor.

Step6: Test for the stopping condition. Stopping condition may be obtaining the best fitness value with minimum finishing time and minimum waiting time for the given objective function of a JSP problem or number of generations.
If stopping condition satisfied then goto step 7 else Goto step2

Step 7: Declare the best individual in the complete generations. Stop.

The flowchart depicting the approach of genetic algorithm for JSP is as shown in Fig.2.

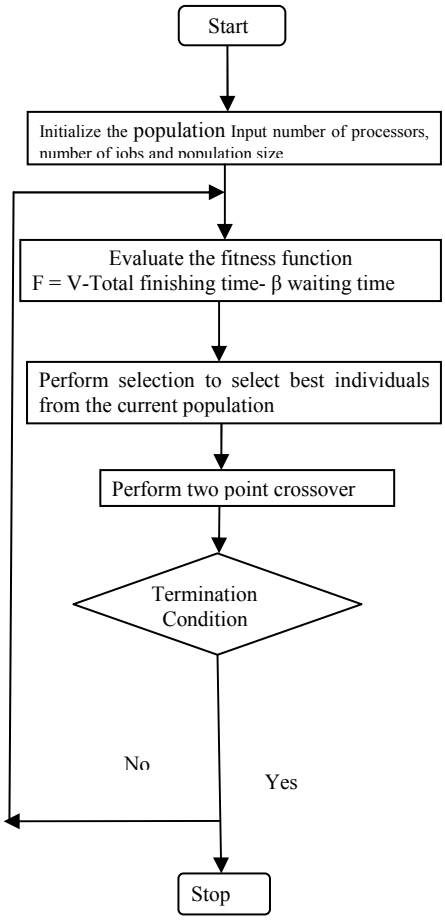


Fig.2 Flowchart for genetic algorithm to JSP

Genetic Algorithm was invoked with the number of populations to be 100 and 900 generations. The crossover rate was 0.1 and the mutation rate was 0.01. Randomly the populations were generated and for various trials of the number of processors and jobs, the completed fitness values of waiting time and finishing time as shown in Table.1. The experimental set up considered possessed 2-5 processors and number of jobs as shown in Table. 1 were assigned to each of the processors.

Table. 1: GA for job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	31.38	47.01	44.31	32.91	38.03
Finishing time	61.80	57.23	70.21	74.26	72.65

From the Table.1, it can be observed that for equal no of jobs for different processors, the finishing time has got reduced. The finishing time and waiting time is observed based on the number of jobs allocated to each processors. Figure 3 shows the variation in finishing time and waiting time for the assigned number of jobs and processors

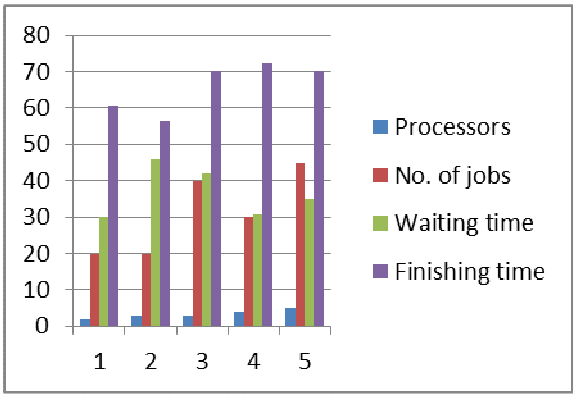


Fig. 3 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using GA

3.2 Particle Swarm Optimization for Scheduling

The particle swarm optimization (PSO) technique appeared as a promising algorithm for handling the optimization problems. PSO is a population-based stochastic optimization technique, inspired by social behavior of bird flocking or fish schooling [10]-[15], [17]. PSO is inspired by the ability of flocks of birds, schools of fish, and herds of animals to adapt to their environment, find rich sources of food, and avoid predators by implementing an information sharing approach. PSO technique was invented in the mid 1990s while attempting to simulate the choreographed,

graceful motion of swarms of birds as part of a socio cognitive study investigating the notion of collective intelligence in biological populations [10]-[15], [17].

The basic idea of the PSO is the mathematical modelling and simulation of the food searching activities of a swarm of birds (particles).In the multi dimensional space where the optimal solution is sought, each particle in the swarm is moved towards the optimal point by adding a velocity with its position. The velocity of a particle is influenced by three components, namely, inertial momentum, cognitive, and social. The inertial component simulates the inertial behaviour of the bird to fly in the previous direction. The cognitive component models the memory of the bird about its previous best position, and the social component models the memory of the bird about the best position among the particles.

PSO procedures based on the above concept can be described as follows. Namely, bird flocking optimizes a certain objective function. Each agent knows its best value so far (pbest) and its XY position. Moreover, each agent knows the best value in the group (gbest) among pbests. Each agent tries to modify its position using the current velocity and the distance from the pbest and gbest. Based on the above discussion, the mathematical model for PSO is as follows,

Velocity update equation is given by

$$V_i = w \times V_i + C_1 \times r_1 \times (P_{best_i} - S_i) + C_2 \times r_2 \times (g_{best_i} - S_i) \quad (4)$$

Using equation (4), a certain velocity that gradually gets close to pbests and gbest can be calculated. The current position (searching point in the solution space) can be modified by the following equation:

$$S_{i+1} == S_i + V_i \quad (5)$$

Where, V_i : velocity of particle i, S_i: current position of the particle, w : inertia weight, C_1: cognition acceleration coefficient, C_2 : social acceleration coefficient, Pbest $_i$: own best position of particle i,
gbest $_i$: global best position among the group of particles, r_1, r_2 : uniformly distributed random numbers in the range [0 to 1].

s_i : current position, s_{i+1} : modified position, v_i : current velocity, v_{i+1} : modified velocity, v_{pbest} : velocity based on pbest, v_{gbest} : velocity based on gbest .

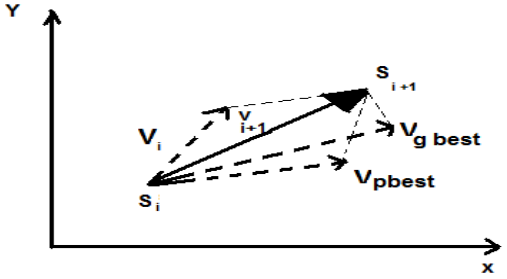


Fig. 4 Flow diagram of PSO

Fig.4 shows the searching point modification of the particles in PSO. The position of each agent is represented by XY-axis position and the velocity (displacement vector) is expressed by vx (the velocity of X-axis) and vy (the velocity of Y-axis). Particle are change their searching point from S_i to S_{i+1} by adding their updated velocity V_i with current position S_i. Each particle tries to modify its current position and velocity according to the distance between its current position S_i and V *pbest*, and the distance between its current position S_i and V *gbest* .

The General particle swarm optimization was applied to the same set of processors with the assigned number of jobs, as done in case of genetic algorithm. The number of particles-100, number of generations=250, the values of c1=c2=1.5 and ω=0.5. Table.2 shows the completed finishing time and waiting time for the respective number of processors and jobs utilizing PSO.

Table. 2 : PSO for job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	30.10	45.92	42.09	30.65	34.91
Finishing time	60.52	56.49	70.01	72.18	70.09

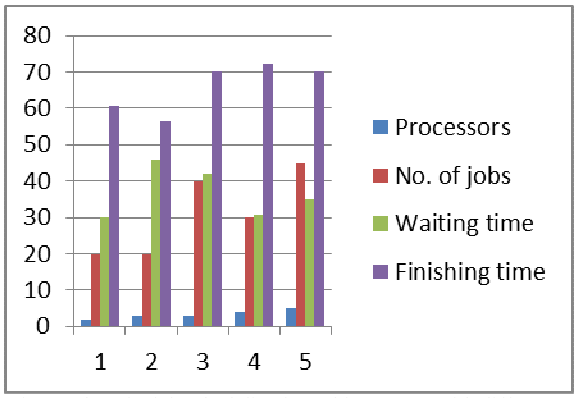


Fig. 5 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using PSO

It is noted from Table.2 that for the same number of processors and jobs , the waiting time and finishing time using PSO has constructively reduced with less number of generations in comparison with GA. . Fig.5 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using particle swarm optimization.

4. Simulated Annealing

Annealing is an operation in metal processing [30]-[35]. Metal is heated up very strongly and then cooled slowly to get a very pure crystal structure with a minimum of energy so that the number of fractures and irregularities becomes minimal. first the high temperature accelerates the movement of the particles. During the cooling time they can find an optimal place within the crystal structure. While the temperature is lowered the particles subsequently lose the energy they were supplied with in the first stage of the process. Because of a thermodynamic, temperature-dependent random component some of them can reach a higher energy level regarding the level they were on before. These local energy fluctuations allow particles to leave local minima and reach a state of lower energy.

Simulated annealing is a relatively straight forward algorithm through which includes metropolis Monte Carlo method .the metropolis Monte Carlo algorithm is well suited for simulated annealing, since only energetically feasible states will be sampled at any given temperature. The simulated annealing algorithm is therefore a metropolis Monte Carlo simulation that starts at a high temperature. The temperature is slowly reduced so that the search space becomes smaller for the metropolis simulation, and when the temperature is low enough the system will hopefully have settled into the most favorable state. Simulated Annealing can also be used to search for the optimum solution of the problems by properly determining the initial (high) and final (low) effective temperatures which are used in place of kT (where k is a Boltzmann's constant) in the acceptance checking, and deciding what constitutes a Monte Carlo step [30]-[35]. The initial and final effective temperatures for a given problem can be determined from the acceptance probability. In general, if the initial Monte Carlo simulation allows an energy (E) increase of dEi with a probability of Pi, the initial effective temperature is $kTi = -dEi/ln(Pi)$. If at the final temperature an increase in the cost of 10 should only be accepted with a probability of 0.05 (5%), the final effective temperature is $kTf = -10/\ln(0.05) = 3.338$.

4.1 Algorithm

Start with the system in a known configuration, at known energy E
T=temperature =hot; frozen=false;
While (! frozen) {
repeat {
Perturb system slightly (e.g., moves a particle)
Compute E, change in energy due to perturbation
If($\Delta E < 0$)
Then accept this perturbation, this is the new system config
Else
accept maybe, with probability = exp(-$\Delta E/KT$)
} until (the system is in thermal equilibrium at this T)
If(ΔE still decreasing over the last few temperatures)
Then T=0.9T //cool the temperature;
do more perturbations
Else frozen=true
}
return (final configuration as low-energy solution)

5. Artificial Immune System

Biological immune systems can be viewed as a powerful distributed information processing systems, capable of learning and self-adaptation. AIS is rapidly emerging, which is inspired by theoretical immunology and observed immune functions, principles, and models. An immune system is a naturally occurring event response system that can quickly adapt to changing situations. The efficient mechanisms of immune system, including clonal selection, learning ability, memory, robustness and flexibility, make AIS s useful in many applications. AIS appear to offer powerful and robust information processing capabilities for solving complex problems.[36]-[40] The AIS based algorithm is built on the principles of clonal selection, affinity maturation, and the abilities of learning and memory.

5.1 AIS-Based Scheduling Algorithm

The brief outline of the proposed algorithm based on AIS can be described as follows.

Step 1) Initialize pop_size antibodies (PS_A) as an initial population by using the proposed initialization algorithm, where pop_size denotes the population size.

Step 2) Select m antibodies from the population by the proportional selection model and clone them to a clonal library.

Step 3) Perform the mutation operation for each of the antibodies in the clonal library.

Step 4) Randomly select s antibodies from the clonal library to perform the operation of vaccination.

Step 5) Replace the worst s antibodies in the population by the best s antibodies from the clonal library.

Step 6) Perform the operation of receptor editing if there is no improvement of the highest affinity degree for a certain number of generations G.

Step 7) Stop if the termination condition is satisfied; else, repeat Steps 2 to7.

In this paper, the parameters are taken as *pop_size =50, m =30, s =10, and G =80.*

6. Proposed Improved Particle Swarm Optimization for Scheduling

In this new proposed Improved PSO (ImPSO) having better optimization result compare to general PSO by splitting the cognitive component of the general PSO into two different component. The first component can be called good experience component. This means the bird has a memory about its previously visited best position. This is similar to the general PSO method. The second component is given the name by bad experience component. The bad experience component helps the particle to remember its previously visited worst position. To calculate the new velocity, the bad experience of the particle also taken into consideration. On including the characteristics of Pbest and Pworst in the velocity updation process along with the difference between the present best particle and current particle respectively, the convergence towards the solution is found to be faster and an optimal solution is reached in comparison with conventional PSO approaches. This infers that including the good experience and bad experience component in the velocity updation also reduces the time taken for convergence.
The new velocity update equation is given by, equation (6)

$$V_i = w \times V_i + C_{1g} \times r_1 \times (P\ best_i - S_i) \times P\ best_i + C_{1b} \times r_2 \times (S_i - P_{worst\ i}) \times P_{worst\ i} + C_2 \times r_3 \times (Gbest_i - S_i) \quad (6)$$

Where,

C_{1g} :acceleration coefficient, which accelerate the particle towards its best position;

C_{1b} :acceleration coefficient, which accelerate the particle away from its worst position;

$P_{worst\ i}$:worst position of the particle i;

r_1, r_2, r_3 : uniformly distributed random numbers in the range [0 to 1];

The positions are updated using equation (5). The inclusion of the worst experience component in the behaviour of the particle gives the additional exploration capacity to the swarm. By using the bad experience component; the particle can bypass its previous worst position and try to occupy the better position. Fig.6 shows the concept of ImPSO searching points.

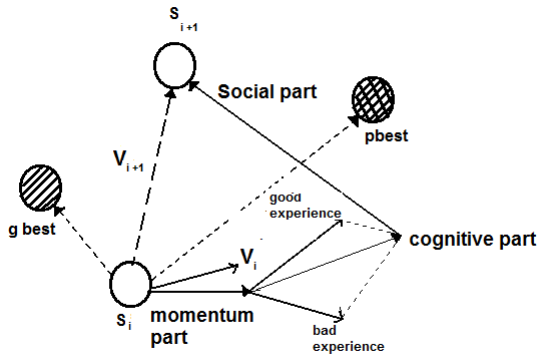


Fig. 6 Concept of Improved Particle Swarm Optimization search point

The algorithmic step for the Improved PSO is as follows:

Step1: Select the number of particles, generations, tuning accelerating coefficients C_{1g} , C_{1b} , and C_2 and random numbers r_1, r_2, r_3 to start the optimal solution searching

Step2: Initialize the particle position and velocity.

Step3: Select particles individual best value for each generation.

Step 4: Select the particles global best value, i.e. particle near to the target among all the particles is obtained by comparing all the individual best values.

Step 5: Select the particles individual worst value, i.e. particle too away from the target.

Step 6: Update particle individual best (p best), global best (g best), particle worst (P worst) in the velocity equation (6) and obtain the new velocity.

Step 7: Update new velocity value in the equation (5) and obtain the position of the particle.

Step 8: Find the optimal solution with minimum ISE by the updated new velocity and position.

The flowchart for the proposed model formulation scheme is shown in Fig.7.

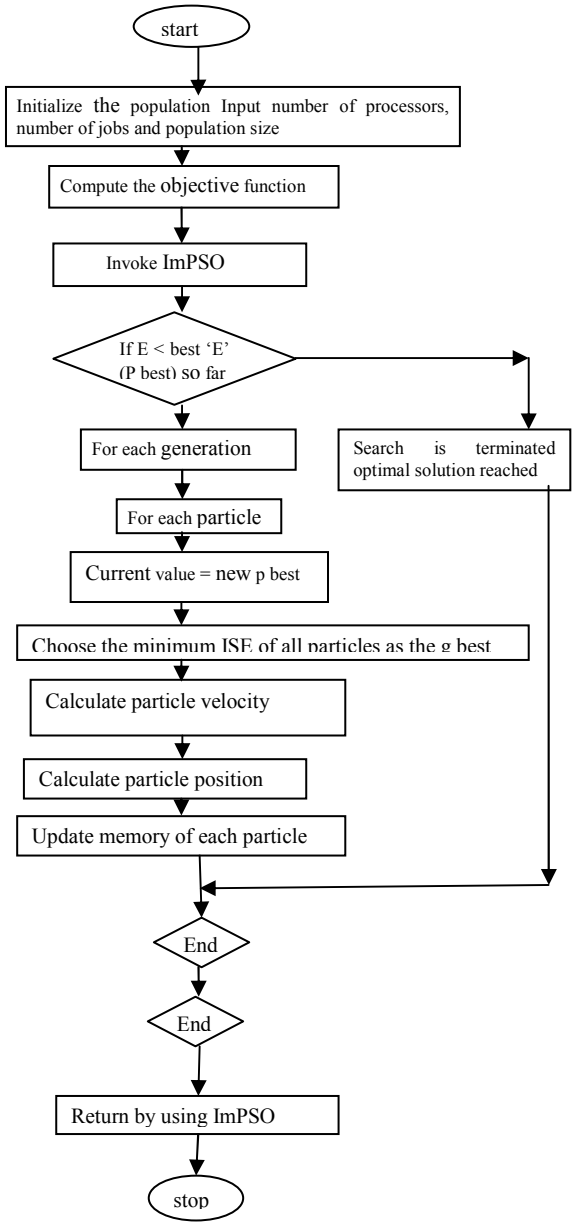


Fig. 7 Flowchart for job scheduling using Improved PSO

The proposed improved particle swarm optimization approach was applied to this multiprocessor scheduling problem. As in this case, the good experience component and the bad experience component are included in the process of velocity updation and the finishing time and waiting time computed are shown in Table. 3.

Table. 3: Proposed Improved PSO for Job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	29.12	45.00	41.03	29.74	33.65
Finishing time	57.34	54.01	69.04	70.97	69.04

The same number of particles and generations as in case of general PSO is assigned for Improved PSO also. It is observed in case of proposed improved PSO, the finishing time and waiting time has been reduced in comparison with GA and PSO. This is been achieved by the introduction of bad experience and good experience component in the velocity updation process. Fig.8 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using improved particle swarm optimization.

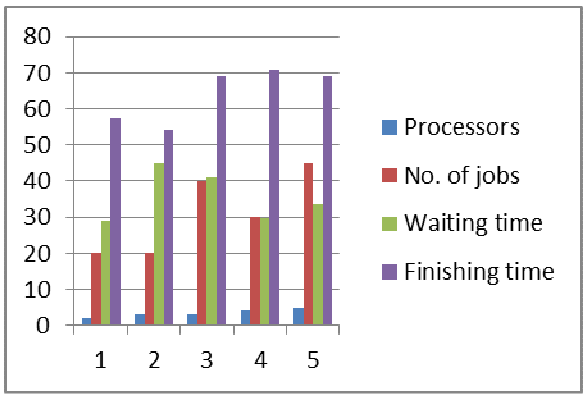


Fig.8 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using ImPSO

7. Proposed Hybrid Algorithm for job scheduling (ImPSO with SA)

The proposed improved PSO algorithm is independent of the problem and the results obtained using the improved PSO can be further improved with the simulated annealing. The probability of getting trapped in a local minimum can be minimized in simulated annealing.

The steps involved in the proposed hybrid algorithm is as follows

Step1: Initialize temperature T to a particular value.

Step2: Initialize the number of particles N and its value may be generated randomly. Initialize swarm with random positions and velocities.

Step3: Compute the finishing time for each and every particle using the objective function and also find the " pbest " i.e.,
If current fitness of particle is better than
" pbest" the set " pbest" to current value.
If "pbest" is better than "gbest then set "gbest" to current particle fitness value.

Step4: Select particles individual "pworst" value i.e., particle moving away from the solution point.

Step5: Update velocity and position of particle as per equation (5) , (6).

Step6: If best particle is not changed over a period of time,
a) find a new particle using temperature.

Step7: Accept the new particle as best with probability as exp-($\Delta E/T$). In this case, ΔE is the difference between current best particles fitness and fitness of the new particle.

Step8: Reduce the temperature T.

Step 9: Terminate the process if maximum number of iterations reached or optimal value is obtained . else go to step 3.

The flow chart for the hybrid algorithm is shown in Fig.9

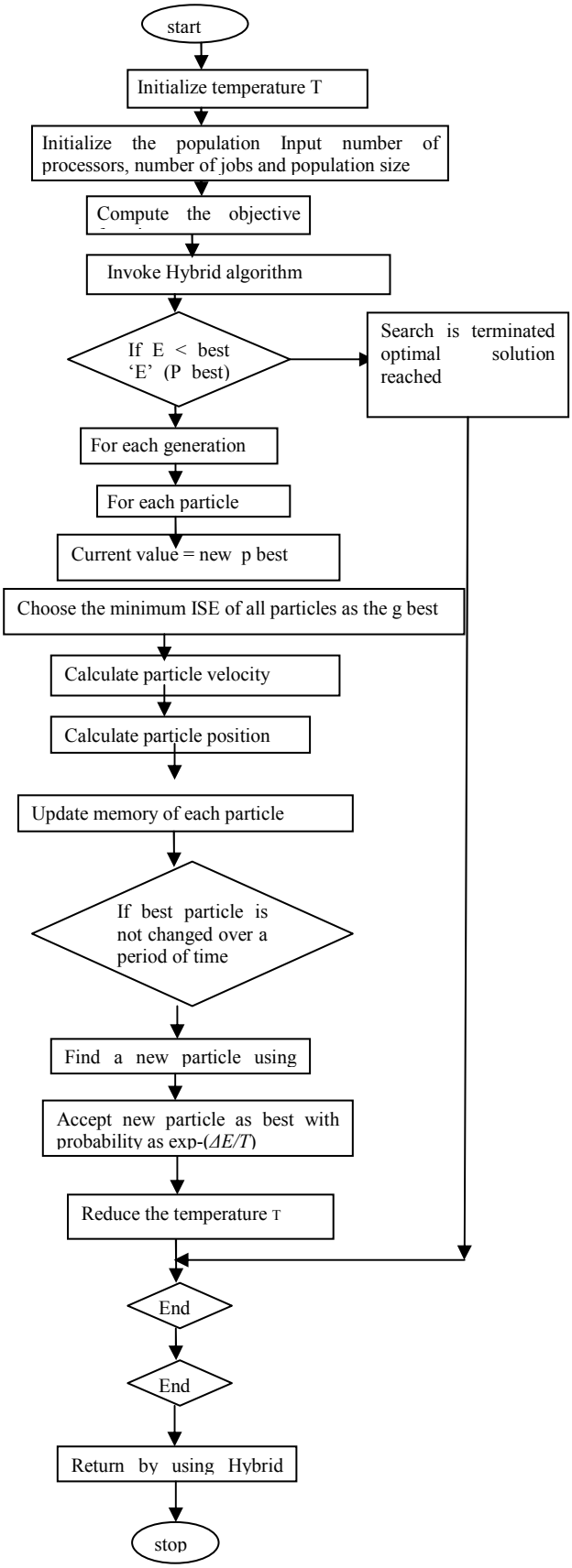


Fig. 9 Flowchart for job scheduling using Hybrid algorithm

The proposed hybrid algorithm is applied to the multiprocessor scheduling algorithm. In this algorithm 100 particles are considered as the initial population and temperature T as 5000. The values of C1 and C2 is 1.5. The finishing time and waiting time completed for the random instances of jobs are as shown in Table. 4

Table 4: Proposed Hybrid algorithm for Job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	25.61	40.91	38.45	26.51	30.12
Finishing time	54.23	50.62	65.40	66.29	66.43

The same number of generations as in the case of improved PSO is assigned for the proposed hybrid algorithm. It is observed, that in the case of proposed hybrid algorithm, there is a drastic reduction in the finishing time and waiting time of the considered processors and respective jobs assigned to the processors in comparison with the general PSO and improved PSO. Thus combining the effects of the simulated annealing and improved PSO, better solutions have been achieved. Fig.10 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using Hybrid algorithm.

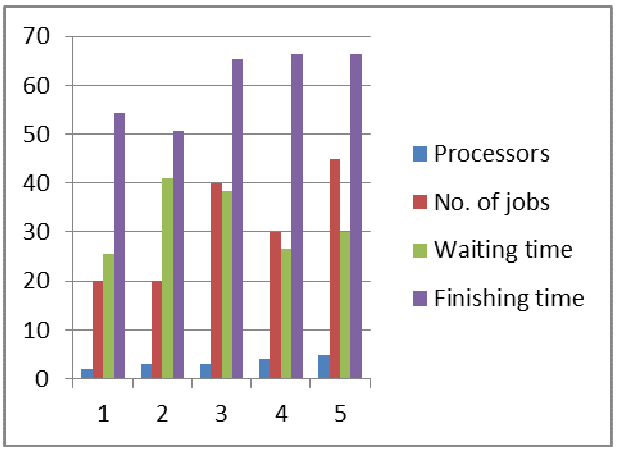


Fig. 10 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using Hybrid algorithm(Improved PSO with Simulated Annealing)

8. Proposed Hybrid Algorithm for job scheduling (ImPSO with AIS)

The proposed improved PSO algorithm is independent of the problem and the results obtained using the improved PSO can be further improved with the AIS.

The steps involved in the proposed hybrid algorithm is as follows

Step 1) Initialize Population size of the antibodies as PS_A.

Step 2) Initialize the number of particles N and its value may be generated randomly. Initialize swarm with random positions and velocities.

Step 3) Compute the finishing time for each and every particle using the objective function and also find the "pbest " i.e., If current fitness of particle is better than " pbest" the set " pbest" to current value.
If "pbest" is better than "gbest then set "gbest" to current particle fitness value.

Step 4) Select particles individual "pworst" value i.e., particle moving away from the solution point.

Step 5) Update velocity and position of particle as per equation (5), (6).

Step 6) If best particle is not changed over a period of time,
a) Select 'm' antibodies out of the population PS_A by the proportional selection model and clone them to a colonal library.

Step 7) Select m antibodies from the population by the proportional selection model and clone them to a clonal library.

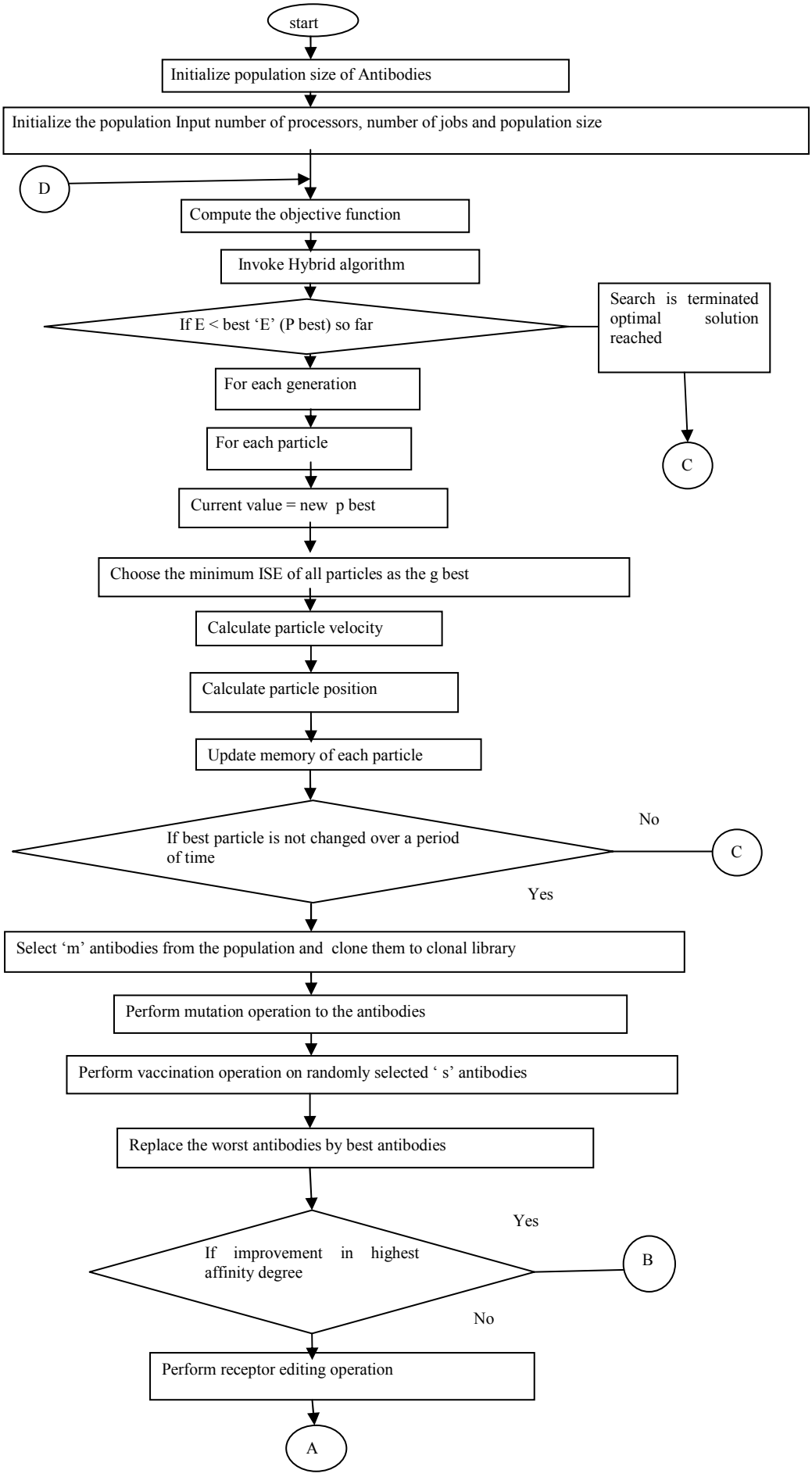
Step 8) Perform the mutation operation for each of the antibodies in the clonal library.

Step 9) Randomly select s antibodies from the clonal library to perform the operation of vaccination.

Step 10) Replace the worst s antibodies in the population by the best s antibodies from the clonal Library

Step 11)Terminate the process if maximum number of iterations reached or optimal value is obtained , . else go to step 3.

The flow chart for the hybrid algorithm is shown in Fig 11.


start
Initialize population size of Antibodies
Initialize the population Input number of processors, number of jobs and population size
D
Compute the objective function
Invoke Hybrid algorithm
If E < best 'E' (P best) so far
Search is terminated optimal solution reached
C
For each generation
For each particle
Current value = new p best
Choose the minimum ISE of all particles as the g best
Calculate particle velocity
Calculate particle position
Update memory of each particle
If best particle is not changed over a period of time
No
C
Yes
Select 'm' antibodies from the population and clone them to clonal library
Perform mutation operation to the antibodies
Perform vaccination operation on randomly selected ' s' antibodies
Replace the worst antibodies by best antibodies
If improvement in highest affinity degree
Yes
B
No
Perform receptor editing operation
A

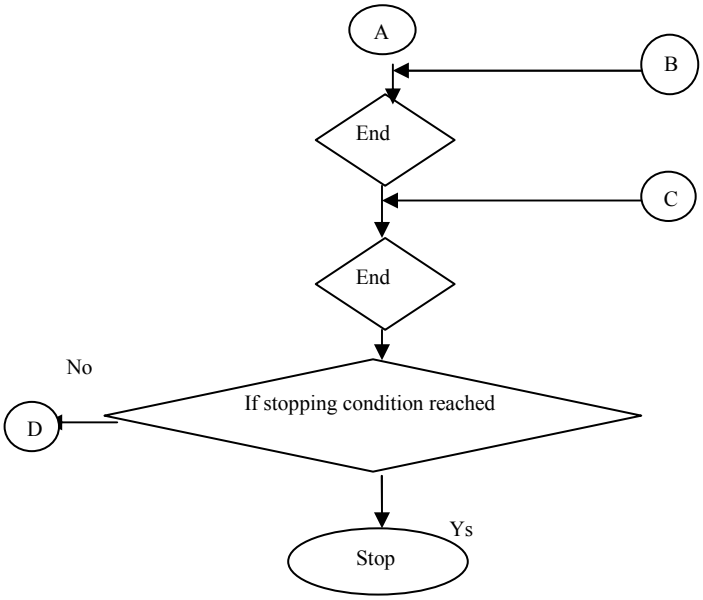


Fig. 11 Flowchart for job scheduling using Hybrid algorithm (Improved PSO with AIS) for job scheduling in Multiprocessor Architecture

The proposed hybrid algorithm is applied to the multiprocessor scheduling algorithm. In this algorithm *100* particles are considered as the initial population. The values of C1 and C2 are 1.5. The finishing time and waiting time completed for the random instances of jobs are as shown in Table.5

Table 5**:** Proposed Hybrid algorithm (ImPSO with AIS) for Job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	22.16	38.65	34.26	23.92	27.56
Finishing time	52.64	48.37	61.20	65.47	64.96

The same number of generations as in the case of improved PSO is assigned for the proposed hybrid algorithm. It is observed, that in the case of proposed hybrid algorithm, there is a drastic reduction in the finishing time and waiting time of the considered processors and respective jobs assigned to the processors in comparison with the general PSO and improved PSO. Thus combining the effects of the AIS and improved PSO, better solutions have been achieved. Fig.12 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using Hybrid algorithm.

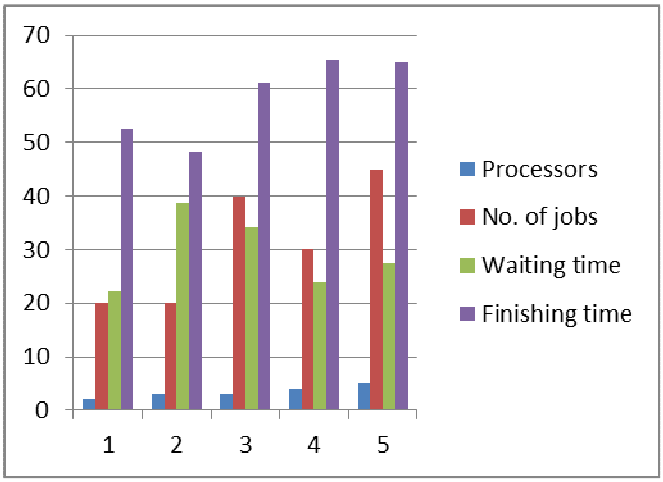


Fig. 12 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using Hybrid algorithm (Improved PSO with AIS)

9. Discussion

The growing heuristic optimization techniques have been applied for job scheduling in multiprocessor architecture. Table.6 shows the completed waiting time and finishing time for GA, PSO, proposed Improved PSO, Proposed Hybrid algorithm and conventional longest processing time (LPT) and Shortest processing time (SPT) algorithm.

Table 6: Comparison of job using LPT,SPT, GA, PSO, Proposed Improved PSO and Proposed Hybrid Algorithm

No of processors	No of jobs	GA		PSO		Proposed Improved PSO		Proposed Hybrid(Improved with SA)		Proposed Hybrid(Improved with AIS)	
		WT	FT	WT	FT	WT	FT	WT	FT	WT	FT
2	20	31.38	61.80	30.10	60.52	29.12	57.34	25.61	54.23	22.16	52.64
3	20	47.01	57.23	45.92	56.49	45.00	54.01	40.91	50.62	38.65	48.37
3	40	44.31	70.21	42.09	70.01	41.03	69.04	38.45	65.40	34.26	61.20
4	30	32.91	74.26	30.65	72.18	29.74	70..97	26.51	66.29	23.92	65.47
5	45	38.03	72.65	34.91	70.09	33.65	69.04	30.12	66.43	27.56	64.96

In LPT algorithm [25],[26],[28], it is noted that the waiting time is drastically high in comparison with the heuristic approached and in SPT with the heuristic approaches and in SPT algorithm, the finishing time is drastically high. Genetic algorithm process was run for about 900 generations and the finishing time and waiting time has been reduced compared to LPT and SPT algorithms. Further the introduction of general PSO with the number of particles 100 and within 250 generations minimized the waiting time and finishing time considerably with GA. The proposed improved PSO with the good(pbest) and bad (pworst) experience component involved with the same number of particles and generations as in comparison with the general PSO, minimized the waiting time and finishing time of the processors with respect to all the other considered algorithms. Further, taking the effects of Improved PSO and combining it with the concept of simulated annealing and deriving the proposed hybrid algorithm it can be observed that it has reduced the finishing time and waiting time drastically. Thus the Temperature coefficient, good experience component and bad experience component of the hybrid algorithm has reduced the waiting time and finishing time .

In AIS, the colonal library consists of the pool of antibodies are identified and replaced with the best antibodies in a manner of how best and worst particles are included in PSO. Further, taking the effects of Improved PSO and combining it with the concept of AIS has reduced the finishing time and waiting time drastically, compared with hybrid algorithm using improved PSO with Simulated Annealing. Thus, when independently the Improved PSO takes more convergence time, the hybrid Improved PSO along with AIS, reduces the finishing and waiting time of the jobs.

Thus based on the results, it can be observed that the proposed hybrid algorithm (ImPSO with AIS) gives better results than the conventional methodologies LPT, SPT and other heuristic optimization techniques GA, General PSO and Proposed Improved PSO. This work was carried out in Intel Pentium i3 core processors with 2 GB RAM.

10. Conclusion

In this paper, a new hybrid algorithm based on the concept of simulated annealing and hybrid algorithm based on AIS was compared. The proposed hybrid algorithm using Improved

PSO with AIS attaining minimum waiting time and finishing time in comparison with the other algorithms, longest processing time, shortest processing time, genetic algorithm, particle swarm optimization, the proposed particle swarm optimization and also Improved PSO with SA. The worst component being included along with the best component and AIS, tends to minimize the waiting time and finishing time, by its cognitive behaviour drastically. Thus the proposed algorithm, for the same number of generations, has achieved better results.

References

[1] M.R.Garey and D.S. Johnson, Computers and Intractability: *A Guide to the theory of NP completeness*, San Francisco, CA, W.H. Freeman, 1979.

[2] L.Mitten, 'Branch and Bound Method: general formulation and properties*', operational Research*, 18, P.P. 24-34, 1970.

[3] T.L.Adam , K.M. Chandy, and J.R. Dicson, " A Comparison of List Schedules for Parallel Processing Systems", *Communication of the ACM,* Vol.17,pp.685-690, December 1974.

[4] C.Y. Lee, J.J. Hwang, Y. C. Chow, and F. D. Anger," Multiprocessor Scheduling with Interprocessor Communication Delays," *Operations Research Letters*, Vol. 7, No.3,pp.141-147, June 1998.

[5] S.Selvakumar and C.S. R. Murthy, " Scheduling Precedence Constrained Task Graphs with Non- Negligible Intertask Communication onto Multiprocessors," *IEEE Trans. On Parallel and Distributed Computing*, Vol, 5.No.3, pp. 328-336, March 1994.

[6] T. Yang and A. Gerasoulis, " List Scheduling with and without Communication Delays," *Parallel Computing*, 19, pp. 1321-1344, 1993.

[7] J. Baxter and J.H. Patel, " The LAST Algorithm: A Heuristic- Based Static Task Allocation Algorithm," *1989 International Conference on parallel Processing*, Vol.2, pp.217-222, 1989.

[8] G.C. Sih and E.A. Lee, " Scheduling to Account for Interprocessor Communication Within Interconnection-Constrained Processor Network," *1990 International Conference on Parallel Processing*, Vol.1, pp.9-17,1990.

[9] M.Y. Wu and D. D. Gajski, " Hypertool: A Programming Aid for Message_Passing Systems," *IEEE Trans on Parallel and Distributed Computing,* Vol.1, No.3, pp.330-343, July 1990.

[10] S.N.Sivanandam and S.N. Deepa, "Introduction to Genetic Algorithm", *Springer verlog* , 2007.

[11] K. Deeba , K. Thanushkodi, , "An Evolutionary Approach for Job Scheduling in a Multiprocessor Architecture", *CiiT International Journal of Artificial Intelligent Systems and Machine Learning* Vol 1, No 4, July 2009.

[12] Kenedy, J., Eberhart R.C, " Particle Swarm Optimization" proc. IEEE Int. Conf. Neural Networks. Pistcataway, NJ(1995) pp. 1942-1948

[13] R.C. Eberhart and Y. Shi, Comparison between Genetic Algorithm and Particle Swarm Optimization", *Evolutionary Programming VII 919980, Lecture Notes in Computer Science 1447*, pp 611-616, Spinger

[14] Y. Shi and R. Eberthart: " Empirical study of particle swarm optimization," *Proceeding of IEEE Congress on Evolutionary Computation*, 1999, pp 1945-1950.

[15] Elnaz Zafarani Moattar, Amir Masoud Rahmani, Mohammad Reza Feizi Derakhshi, " Job Scheduling in Multiprocessor Architecture Using Genetic Algorithm", *Proc. IEEE 2008*, pp. 248-250.

[16] Ali Allahverdi, C. T. Ng, T.C.E. Cheng, Mikhail Y. Kovalyov, " A Survey of Scheduling Problems with setup times or costs", *European Journal of Operational Research(Elsevier),* 2006.

[17] Gur Mosheiov, Uri Yovel, " Comments on " Flow shop and open shop scheduling with a critical machine and two operations per job", *European Journal of Operational Research(Elsevier),* 2004.

[18] K.S. Amirthagadeswaran, V.P. Arunachalam, " Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction", *International Journal Advanced*

Manufacture Technology(Spiringer), 2005.

[19] Tung-Kuan Liu, Jinn- Tsong Tsai, Jyh-Hong, Chou, " Improved genetic algorithm for the job-shop scheduling problem", *International Journal Advanced Manufacture Technology(Spiringer)*, 2005.

[20] X.D. Zhang, H. S. Yan, " Integrated optimization of production planning and scheduling for a kind of job-shop", *International Journal Advanced Manufacture Technology(Spiringer),* 2005.

[21] D.Y. Sha , Cheng-Yu Hsu, " A new particle swarm optimization for open shop scheduling problem ", *Computers & Operations Research(Elsevier),* 2007.

[22] Gur Mosheiov, Daniel Oron, " Open-shop batch scheduling with identical jobs", *European Journal of Operations Research(Elsevier),* 2006.

[23] A.P. Engelbrecht, " Fundamentals of Computational Swarm Intelligence", *John Wiley & Sons,* 2005.

[24] Zhou, M., Sun, S.d. " Genetic algorithms: theory and application " *National Defense Industry Pres*s, Beijing, China, pp. 130-138, 1999.

[25] Chen, B. A. "Note on LPT scheduling" , *Operation Research Letters* 14(1993), 139-142.

[26] Morrison, J. F.., A note on LPT scheduling, *Operations Research Letters 7* (1998), 77-79.

[27] Dobson, G., Scheduling independent tasks on uniform processors, *SIAM Journal on Computing 13* (1984), 705-716.

[28] Friesen, D. K., Tighter bounds for LPT scheduling on uniform processsors, *SIAM Journal on Computing* 6(1987), 554-660.

[29] Coffman, Jr., E.G. and Graham, R. L.., Optimal scheduling for two-processor systems, *Acta Informatica* 1(1972), 200-213.

[30] Bozejko W., Pempera J. and Smuntnicki C. 2009."Parallel simulated annealing for the job shop scheduling problem", *Lecture notes in computer science*, *Proceedings of the 9th International Conference on Computational Science*, Vol.5544, pp. 631-640.

[31] Ge H.W., Du W. and Qian F. 2007. "A hybrid algorithm based on particle swarm optimization and simulated annealing for job shop scheduling", *Proceedings of the Third International Conference on Natural Computation*, Vol. 3, pp. 715–719.

[32] Weijun Xia and Zhiming Wu "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J" *Computers & Industrial Engineering*, 2005,48(2)_409-425

[33]Yi Da, Ge Xiurun. "An improved PSO-based ANN with simulated annealing technique[J]". *Neurocomputing*, 2005, 63 (1): 527-533.

[34] Kirkpatrick S., Gelatt C.D. and Vecci M.P. 1983. Optimization by simulated annealing, *Science, New Series*, Vol. 220, No. 4598, pp. 671-680.

[35] Wang X. and Li J. 2004. Hybrid particle swarm optimization with simulated annealing, *Proceedings of Third International Conference on Machine Learning and Cybernetics*, Vol.4, pp. 2402-2405.

[36] H. B. Yu and W. Liang, "Neural network and genetic algorithm-based hybrid approach to expanded job- shop scheduling," Comput. Ind. Eng., vol. 39, no. 3/4, pp. 337–356, Apr. 2001.

[37] S. Yang and D. Wang, "A new adaptive neural network and heuristics hybrid approach for job-shop scheduling," Comput. Oper. Res., vol. 28, no. 10, pp. 955–971, Sep. 2001.

[38] S. Yang and D. Wang, "Constraint satisfaction adaptive neural network and heuristics pproaches for generalized job-shop scheduling," IEEE Trans. Neural Netw., vol. 11, no. 2, pp. 474–486, Mar. 2000.

[39] C. A. C. Coello, D. C. Rivera, and N. C. Cortes, "Use of an artificial immune system for job Shop scheduling," in Proc. 2nd Int. Conf. Artificial Immune Syst., 2003, vol. 2787, pp. 1–10.

[40] Hong-Wei Ge, Liang Sun, Yan-Chun Liang, and Feng Qian,' An Effective PSO and AIS-Based Hybrid Intelligent Algorithm for Job-Shop Scheduling" IEEE Transactions On Systems, Man, and Cybernetics—Part A: Systems And Humans, VOL. 38, NO. 2, March 2008.

[41] K. Thanushkodi, K. Deeba, "On Performance Comparisons of GA, PSO and proposed Improved PSO for Job Scheduling in Multiprocessor Architecture." International Journal of Computer Science and Network Security, May, 2011.
[42] K. Thanushkodi, K. Deeba " A Comparative study of proposed improved PSO algorithm with proposed Hybrid Algorithm for Multiprocessor Job Scheduling ", International Journal of Computer Science and Information Security, Vol. 9 No. 6, June, 2011.
[43] K. Thanushkodi, K. Deeba, " A New Improved Particle Swarm Optimization Algorithm for Multiprocessor Job Scheduling", International Journal of Computer Science and Issues , Volume 8, Issue 4 July, 2011.

Dr.K. Thanushkodi.
He has got 30.5 Years of Teaching Experience in Government Engineering Colleges. Has Published 45 Papers in International Journal and Confernces. Guided 3 Ph.D and 1 MS(by Research), Guiding 15 Research Scholars for Ph.D Degree in the area of Power Electronics, Power System Engineering, Computer Networking, Parallel and Distributed Systems & Virtual Instrumentation and One Research Scholar in MS(Reaearch). Principal in_charge and Dean, Government College of Engineering, Bargur, Served as Senate member, Periyar University, Salem. Served as member, Research Board, Anna University, Chennai. Served as Member, Academic Council, Anna University, Chennai. Serving as Member, Board of Studies in Electrical and Electronics and Communication Engineering in Amirta Viswa Vidhya Peetham, Deemed University, Coimbatore. Serving as Governing Council Member SACS MAVMM Engineering College, Madurai. Served as Professor and Head of E&I, EEE, CSE & IT Departments at Government College of Technology, Coimbatore. Presently he is the Director of Akshaya College of Engineering and Technology.

K. Deeba, has completed B.E in Electronics and communication in the year 1997, and completed M.Tech (CSE) in National Institute of Technology, Trichy. She is having 12 Years of Teaching Experiencce. She has published 13 Papers in International and National Conferences and Journals. Currently she is working as a Associate Professor and Head, Department of Computer Science and Engineering in Kalaignar Karunanidhi Institute of Technology, Coimbatore.