Scientific Research and Essays Vol. 7(20), pp. 1935-1953, 30 May, 2012
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE12.056
ISSN 1992-2248 ©2012 Academic Journals

Full Length Research Paper

Hybrid intelligent algorithm [improved particle swarm optimization (PSO) with ant colony optimization (ACO)] for multiprocessor job scheduling

K. Thanushkodi[1] and K. Deeba[2]*

[1]Akshaya College of Engineering, Anna University of Technology, Coimbatore, India.
[2]Department of Computer Science and Engineering, Kalignar Karunanidhi Institute of Technology, Anna University of Technology, Coimbatore, India.

Accepted May 3, 2012

Efficient multiprocessor scheduling is essentially the problem of allocating a set of computational jobs to a set of processors to minimize the overall execution time. The main issue is how jobs are partitioned in which total finishing time and waiting time is minimized. Minimization of these two criteria simultaneously, is a multi objective optimization problem. There are many variations of this problem, most of which are NP-hard problem, so we must rely on heuristics to solve the problem instances. Many heuristic-based approaches have been applied to finding schedules that minimize the execution time of computing tasks on parallel processors. Particle swarm optimization (PSO) is currently employed in several optimization and search problems due to its ease and ability to find solutions successfully. A variant of PSO, called as improved particle swarm optimization (ImPSO) has been developed in this paper and is hybridized with the ant colony optimization (ACO) to achieve better solutions. The proposed hybrid algorithm effectively exploits the capabilities of distributed and parallel computing of swarm intelligence approaches. In addition hybrid algorithm using improved particle swarm optimization (ImPSO) with artificial immune system (AIS) is also implemented for the same set of problems to compare with the proposed hybrid algorithm (ImPSO with ACO). It was observed that the proposed hybrid approach (Improved PSO with ACO) gives better results in experiments and reduces finishing and waiting time simultaneously.

Key words: Particle swarm optimization (PSO), improved particle swarm optimization (ImPSO), ant colony optimization (ACO), job scheduling, finishing time, waiting time.

INTRODUCTION

Scheduling, in general, is concerned with allocation of limited resources to certain tasks to optimize few performance criterions, like the completion time, waiting time or cost of production. Job scheduling problem is a popular problem in scheduling area of this kind. The main objective of job scheduling problem is to find optimal scheduling of the jobs to processors such that the overall finishing time is reduced. The importance of scheduling has increased in recent years due to the extravagant development of new process and technologies. Multiprocessors have been accepted in vehicles for improving computing speeds, cost/performance and enhanced reliability or availability. The main reason for using a multiprocessor system is to improve the performance and to achieve high scalability. 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

*Corresponding author. E-mail: deeba.senthil@gmail.com.

proper assignment and scheduling of computational tasks among the processors.

Total finishing time and waiting time are two computable criteria in multiprocessor architecture which can be used to evaluate efficiency of proposed algorithms. Total finishing time is defined as maximum of each processor's finishing time which is the time that the processor finishes its job. Waiting time is defined as average of time that each job waits in ready queue. Most of the algorithms reduce only finishing time not the finishing and waiting time simultaneously. Simultaneously minimizing these two criteria is a multi objective optimization (MOO) problem (Elnaz et al., 2008).

The main objective of MOO algorithms is to find a set of solutions which optimally balances the trade-offs among the objectives of a multi objective problem (MOP). The task is to find a set of non-dominated solutions, referred to as the Pareto-optimal set. In the following, domination and Pareto-optimal set concepts will be described, but first we should define our notations (Engelbrecht, 2005).

Let $P \subseteq Q^{m_x}$ denote the m_x –dimensional search space. The search space, P is also referred to as the decision space and F $\subseteq P$ the feasible space. With no constraints, "the feasible space is the same as the search space." Let $X = (x_1, x_2, ...x_{m_x})$ referred to as a decision vector. A single objective function, $f_n(X)$ is defined as $f_n : Q^{m_n} \to Q$.

Let $f(X) = (f_1(X), f_2(X), ... f_{m_n}(X)) \in O \subseteq Q^{m_x}$ be an objective vector containing m_n objective function evaluations, O is referred as the objective space.

Domination

A Decision vector, X_1 dominates a decision vector, X_2 (denoted by $X_1 \prec X_2$) if and only if

X_1 is not worse than X_2 in all objectives, that is, $f_n(X_1) \leq f_n(X_2), \forall n = 1.....m_n$ and X_1 is strictly better than X_2 in at least one objective, that is, $\exists n = 1.....m_n : f_n(X_1) < f_n(X_2)$

Similarly, an objective vector, f_1 dominates another objective vector f_2, if f_1 is not worse than f_2 in all objective values and f_1 is better than f_2 in at least one of the objective values. Objective vector dominance is denoted by $f1 \prec f_2$.

Pareto-optimal

A Decision vector $X^* \in F$ is pareto optimal if there does not exist a decision vector, $X \neq X^* \in F$ that dominates it. That is $\nexists n : f_n(X) < f_n(X^*)$. An objective vector $f^*(X)$ is pareto optimal if X is pareto- optimal.

Pareto-optimal set

The set of all pareto optimal decision vectors form the pareto-optimal set P^* that is $P^* = \{X^* \in F \mid \nexists X \in F : X \prec X^* \}$

Related works on multiprocessor scheduling

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 that is, 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, Mitten (1970), 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 Adam et al. (1974), Lee et al. (1998), Baxter et al. (1989), Sih et al. (1990) and Wu et al. (1990). 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 Adam (1974), Baxter (1989) 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. The greedy algorithm is also used for solving problem of this kind.

Lin and Hsu (1990) proposed a stochastic heuristic algorithm, simulated annealing for the problem of static task assignment scheduling in distributed computing systems. The purpose of task assignment scheduling is to assign modules of programs over a set of interconnected tasks in order to reduce the job turnaround time as well as to obtain the best system performance.

Selvakumar and Siva Ram Murthy (1994) presented an efficient heuristic algorithm for scheduling precedence constrained task graphs with non-negligible inter-task communication onto multiprocessors taking contention in the communication channels into consideration. The proposed algorithm for obtaining satisfactory suboptimal

schedules is based on the classical list scheduling strategy. It simultaneously exploits the schedule holes generated in the processors and in the communication channels during the scheduling process in order to produce better schedules. The effectiveness of the proposed algorithm is demonstrated by comparing with two competing heuristic algorithms.

Ahmad and Dhodhi (1996) used a problem-space genetic algorithm (PSGA) that combines the list scheduling with the genetic algorithm for the static scheduling of directed acyclic graphs. Lee et al. (1997) considered the problem of assigning the tasks of a distributed application to the processors of a distributed system such that the sum of execution and communication costs is minimized. A modeling technique is developed that transforms the assignment problem in an array or tree into a minimum-cut maximum-flow problem. The assignment problem is the solved for a general array or tree network in polynomial time.

Graham (2003) proposed the ant colony optimization algorithm for solving the static multiprocessor scheduling problem along with the combination of the local search operators. Wu et al. (2004) presented a representation for the task scheduling using Genetic Algorithm in which each individual consists of series of cells. Each cell is a task-processor pair (t, p) for which task 't' is assigned to processor 'p'. The number of cells can vary from one individual to the next and therefore different individuals can have different lengths. If the prerequisites are not satisfied, a penalty is assigned to the respective individual.

A new genetic algorithm called genetic convex cluster algorithm is proposed by Sanchez and Trystram (2005) to solve the task assignment with large communication delays. It uses the convex cluster property and is well suited for parallel systems like cluster of computers with hierarchical communications.

Tzu et al. (2006) proposed a solution to the constrained scheduling problem in display system operation, using the particle swarm optimization. In particle encoding, the authors used a one- dimensional 0-1 array mapping of a three- dimension matrix of a candidate solution for each particle and then used sigmoid function to produce probability threshold for velocity updating in each particle. The results show that the proposed approach is capable of obtaining higher quality solution efficiently in constrained scheduling problems.

Lei et al. (2008) adopted a heuristic approach based on particle swarm optimization for solving task scheduling problem in a grid environment. Each particle is represented by a possible solution and the position vector is transformed from the continuous variable. The approach aims to generate an optimal schedule so as to get the minimum completion time while completing the tasks. The results of simulated experiments show that the particle swarm optimization algorithm is able to get the better schedule than genetic algorithm.

Visalakshi et al. (2009) proposed a hybrid particle swarm optimization (HPSO) method for solving the task assignment problem. HPSO yields better results compared with normal PSO and also the proposed method is compared with GA.

Gagne et al. (2002) proposed an Ant Colony Optimization Algorithm for the scheduling problem and showed that it performs competitively with the best results of previous methods like genetic algorithm and simulated annealing.

Gupta and Smith (2006) proposed two heuristics, a greedy randomized adaptive search procedure (GRASP) and a problem space-based local search heuristics performs equally well when compared to ant colony optimization of Gagne et al. (2002) while taking much less computational time.

In the past decades basic PSO and ACO have used to solve the scheduling problem. Basic PSO easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction, for this reason to improve the speed of convergence and quality of solution found by the PSO many variant PSO have been developed. Similarly, ACO also takes more convergence time. Hence, blending PSO with other intelligent optimization algorithm, that is, combining the advantages of the PSO with the advantages of other intelligent optimization algorithms produces results with minimum convergence time.

In this paper a new hybrid algorithm based on improved PSO (ImPSO) and ACO is developed to solve job scheduling in multiprocessor architecture with the objective of minimizing the job finishing time and waiting time.

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 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,

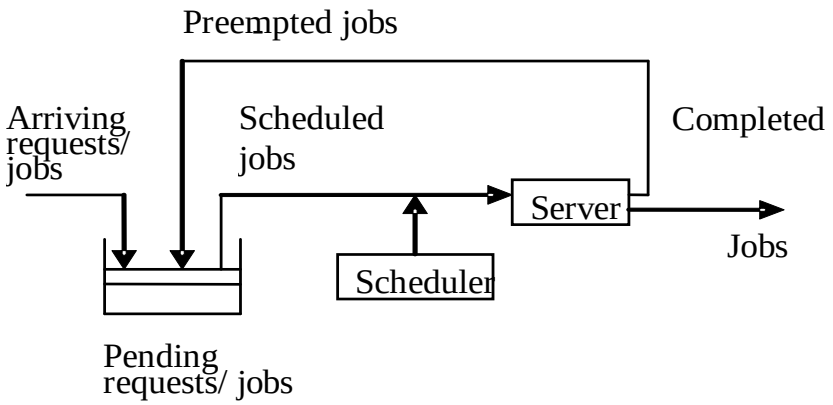


Figure 1. A Schematic of job scheduling.

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 Figure 1.

Problem definition

The objective function defined for this problem as a weighted sum of the objectives. Using aggregation method MOP is redefined as,

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

Subject to

$$g_t(X) \le 0, \quad t = 1,.....m_g$$
$$h_t(X) = 0, \quad t = m_g + 1,.....m_g + m_h$$
$$X = [X_{\min}, X_{\max}]^{m_x}$$
$$\omega_n \ge 0, \quad n = 1,.....m_n$$

Where g_t and h_t are the inequality and equality constraints. [X_{min}, X_{max}] represents the boundary constraints.

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 2, Liu et al. (2005).

$$\text{Size of search space} = \frac{(M \times N)!}{(N!)^M} \quad (2)$$

Earlier, longest processing time (LPT), and shortest processing time (SPT) and traditional optimization algorithms was used for solving these type of scheduling problems Sivanandam et al. (2007), Amirthagadeswaran et al. (2005), Sha et al. (2007), Dobsonet et al. (1984), Coffmanet al. (1972).

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 complete its job and the waiting time is defined as the average of time that each job waits in ready queue. LPT and SPT reduces finishing time and Waiting time respectively. Minimization of finishing time and waiting time simultaneously is a multi objective optimization problem. Two objective of this problem can be unified by using waited sum, Zhou et al. (1999) as is presented in the following, which is a derivative of Equation 1.

f=Total finishing Time + β Waiting Time (3)

'β' is a weight coefficient which can be adjusted to compromise between two minimization objectives. A smaller β attaches relatively more emphasis on the minimization of total finishing time. While a larger β gives relatively more emphasis on the minimization of waiting time. In this paper an attempt is made to solve job scheduling biobjective optimization problem by using a hybrid algorithm using improved particle swarm optimization and ant colony optimization.

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 ant colony optimization (ACO) is suggested for the job scheduling NP-hard problem. The following sections discuss on the application of these techniques to the considered problem Thanushkodi et al. (2011a and b).

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

Gur et al. (2004) introduces a polynomial time solution algorithm for the problem which appears to be surprisingly simple.

Amirthagadeswaran et al. (2005) used Genetic Algorithm to solve job –based , operation-based and proposed methods of representation and schedule deduction with the makespan objective. Computational experiments show better results with appreciable reduction in computer processing time.

Tung-Kuan et al (2005), suggests an improved genetic algorithm, called the hybrid Taguchi-genetic algorithm (HTGA), is proposed to solve the job-shop scheduling problem (JSP). The HTGA approach is a method of combining the traditional genetic algorithm (TGA), which has a powerful global exploration capability, with the Taguchi method, which can exploit the optimal offspring. The Taguchi method is inserted between crossover and mutation operations of a TGA. Then, the systematic reasoning ability of the Taguchi method is incorporated in the crossover operations to systematically select the better genes to achieve crossover, and consequently enhance the genetic algorithm. Therefore, the proposed HTGA approach possesses the merits of global exploration and robustness. The computational experiments show that the proposed HTGA approach can obtain both better and more robust results than other GA-based methods reported recently.

Qing-dao-er-ji et al (2010) Job shop scheduling problem is a typical NP-hard problem. In this paper, new designed crossover and mutation operators based on the characteristic of the job shop problem itself are specifically designed. Based on these, an improved genetic algorithm is proposed. The computer simulations are made on a set of benchmark problems and the results indicate the effectiveness of the proposed algorithm. The algorithm used to solve scheduling problem is thus explained in steps.

Step 1

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

Step 2

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

$$F = \begin{cases} V - Total\ Finishing\ Time & f < V \\ \quad - \beta Waiting\ Time & \\ 0 & f \geq V \end{cases} \quad (4)$$

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.

Step 3

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.

Step 4

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.

Step 6

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

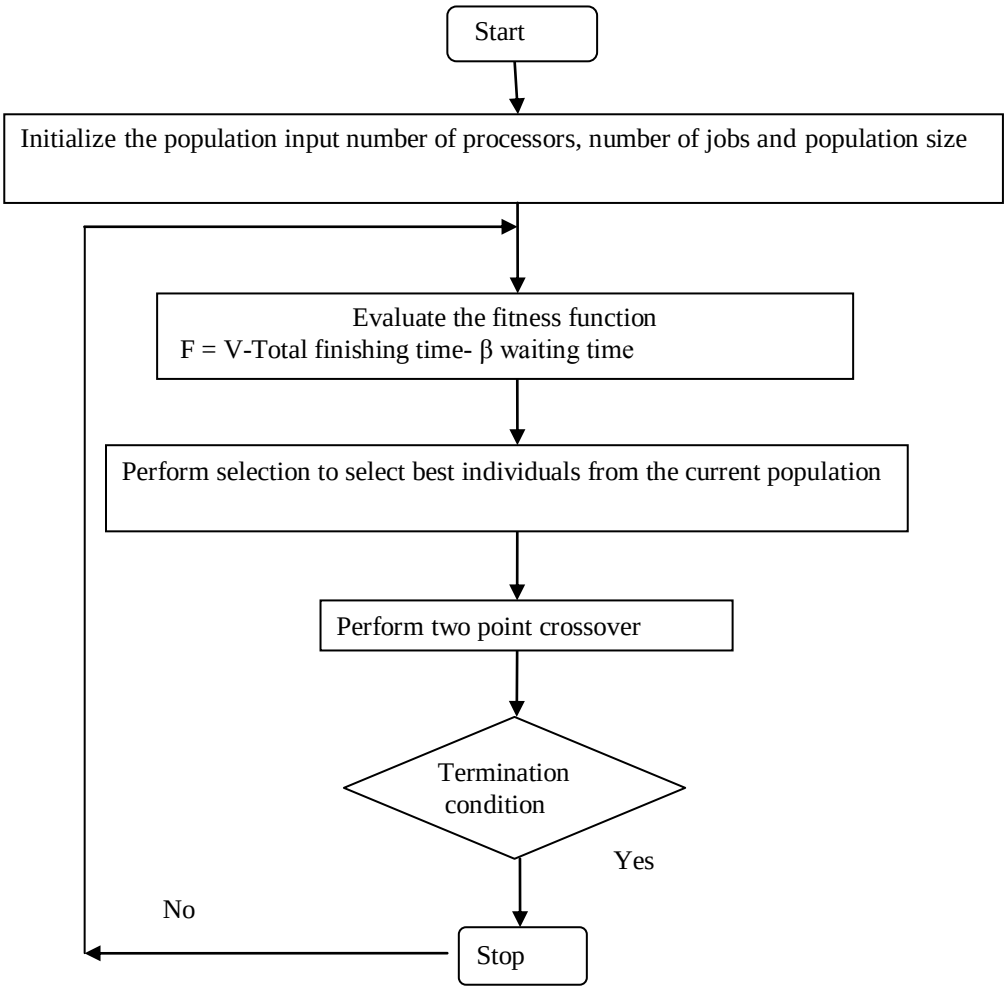


Figure 2. Flowchart for genetic algorithm to JSP.

stopping condition satisfied then go to step 7 else go to step 2.

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 Figure 2.

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.

From the Figure 3, 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 processor. Also, shows the variation in finishing time and waiting time for the assigned number of jobs and processors.

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 Thanushkodi et al. (2009), Shi et al. (1999) and Gur et al. (2004). 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

Table 1. Comparison of job using GA, PSO, Proposed Improved PSO and Proposed Hybrid Algorithm (ImPSO with ACO).

No of processors	No of jobs	GA		PSO		Proposed improved PSO (ImPSO)		Proposed hybrid (ImPSO with AIS)		Proposed hybrid (ImPSO with ACO)	
		WT	FT	WT	FT	WT	FT	WT	FT	WT	FT
2	20	31.38	61.80	30.10	60.52	29.12	57.34	22.16	52.64	18.02	48.92
3	20	47.01	57.23	45.92	56.49	45.00	54.01	38.65	48.37	35.12	48.00
3	40	44.31	70.21	42.09	70.01	41.03	69.04	34.26	61.20	30.16	57.32
4	30	32.91	74.26	30.65	72.18	29.74	70..97	23.92	65.47	21.87	62.45
4	50	35.72	76.21	32.79	71.20	30.06	70.62	25.96	69.83	24.63	67.45
5	45	38.03	72.65	34.91	70.09	33.65	69.04	27.56	64.96	26.21	60.87
5	60	42.93	77.29	39.61	75.42	36.56	72.31	30.19	69.21	28.42	64.26

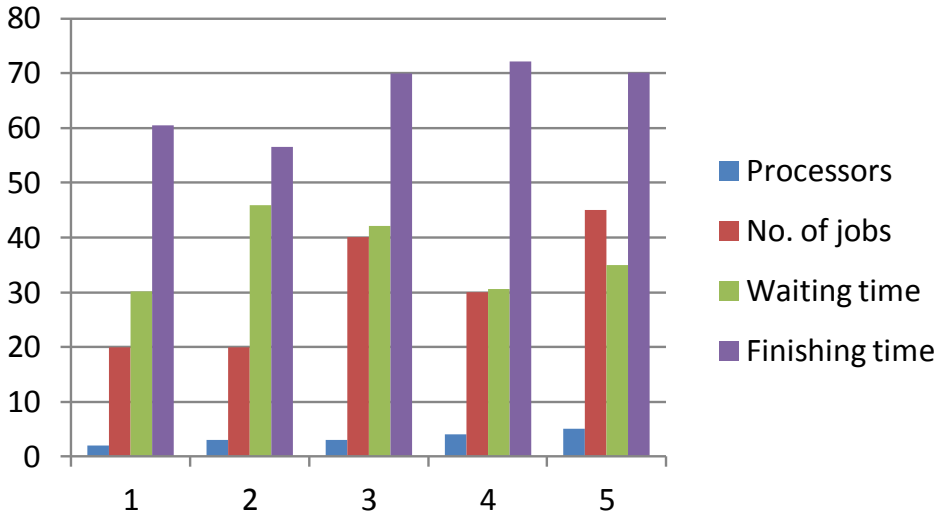


Figure 3. Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using GA.

biological populations.

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:

$$V_i = w \times V_i + C_1 \times r_1 \times (P_{best_i} - S_i) + C_2 \times r_2 \times (g_{best_i} - S_i) \quad (5)$$

Using Equation 5, 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:

$$S_{i+1} == S_i + V_i \quad (6)$$

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

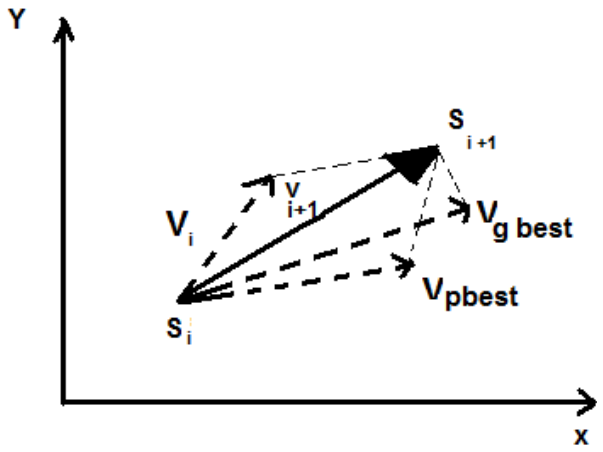


Figure 4. Flow diagram of PSO.

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.

Figure 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. Figure 3 shows the completed finishing time and waiting time for the respective number of processors and jobs utilizing PSO.

It is noted from Figure 5 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 Also, Figure 5 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using particle swarm optimization.

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 Yu et al. (2001), Yang et al. (2001), Yang et al. (2000), Coello et al. (2003) and Hong-Wei et al. (2008). The AIS based algorithm is built on the principles of clonal selection, affinity maturation, and the abilities of learning and memory.

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.

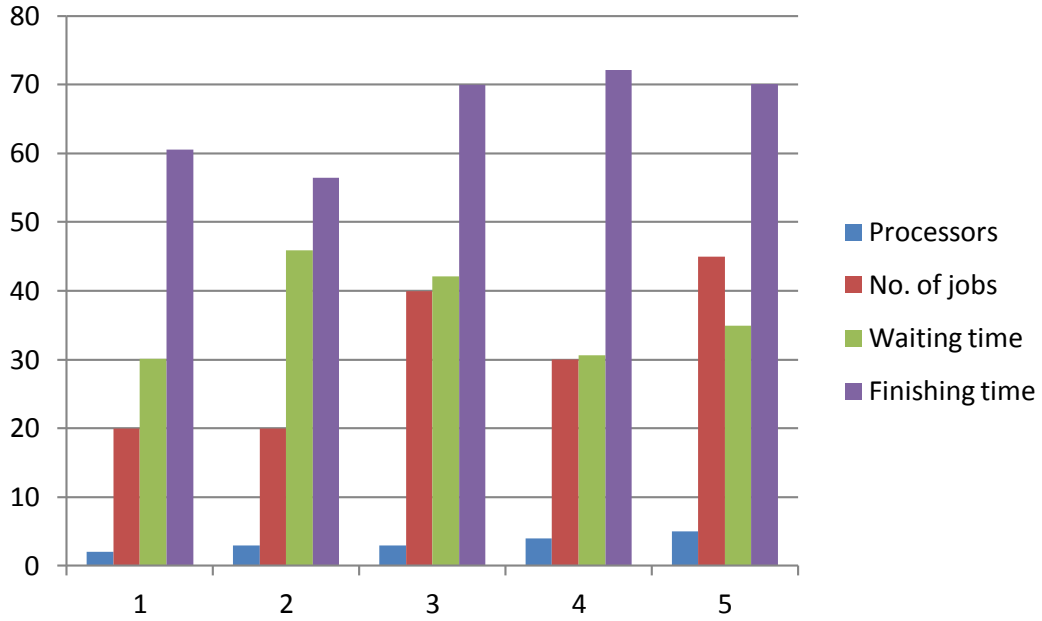


Figure 5. Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using PSO.

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.

ANT COLONY OPTIMIZATION

Ant colony metaheuristic is a concurrent algorithm in which a colony of artificial ants cooperates to find optimized solutions of a given problem. The ant colony optimization (ACO) algorithm was introduced by Dorigo et al. (2006), Dorigo et al. (2007), Dorigo et al. (2002) and Shelokar (2002). It is a probabilistic technique for solving computational problems, which can be reduced to finding good paths through graphs.

They are inspired by the behavior of ants in finding paths from the colony to the food. In the real world, ants initially wander randomly, and upon finding food, they return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but rather follow the trail, returning and reinforcing it if they eventually find food. However, the pheromone trail starts to evaporate over time, therefore reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the quicker it takes for the pheromones to evaporate. A short path, by comparison, gets marched over faster and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the ants following ones. In that case, the exploration of the solution space is constrained. Thus, when one ant finds a short path from the colony to a food source (that is, a good solution), other ants are more likely to follow that path, and positive feedback eventually leaves all the ants following a single path.

ACO algorithms have an advantage over simulated annealing and GA approaches when the graph may change dynamically, since the ant colony algorithm can be run continuously and adapt to changes in real time.

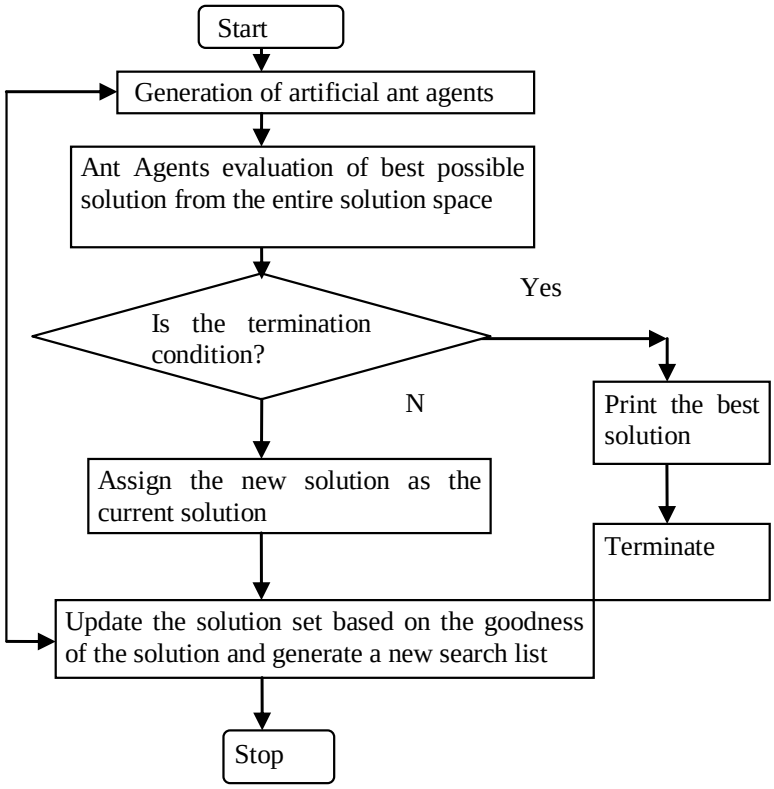


Figure 6. Flow chart for ant colony optimization algorithm.

General ant colony optimization (ACO) algorithm

Step 1: Initialize ACO parameters;
Step 2: Generate solutions from each ants' random walk;
Step 3: Update pheromone intensities;
Step 4: go to step 3, and repeat until convergence or a stopping condition is satisfied. The flow chart for the ant colony optimization (ACO) algorithm is depicted in Figure 6.

ACO for job scheduling

Consider a n× n matrix named τ as pheromone variables, in which 'n' is number of jobs.
τ (i, j) is desirability of selecting job j just after job i. Initially all elements of matrix have the same and small value.

Then iterative ant colony algorithm is executed:

1. Generate ant (or ants)
2. Loop for each ant (until complete scheduling of tasks) - Select next task with respect to pheromone variables of ready tasks.
3. Deposit pheromone on visited states
4. Daemon activities
5. Evaporate pheromone

At first, a list with length of n, is created as ant. Initially it is empty and will be completed during next stage. In the second stage, there is a loop for each ant. At each iteration, ant must select a job from ready list with regard to values of pheromone variables of jobs in the ready list using a probabilistic decision making.

For ant k, probability of selecting job j just after selecting job i is obtained by using:

$$P^k(i,j) = \frac{\tau(i,j)}{\sum_{k \in N} \tau(i,k)} \quad (7)$$

where N is set of ready jobs.

Then a random number is generated and next job is selected with respect to the generated number. It is clear that jobs which have bigger pheromone value, have bigger chance to select. Selected job is appended to the ant list, removed from ready list. These operations are repeated until complete scheduling of all jobs, in the other words, completing the ant's list. In the third stage, tasks are extracted one by one from ant's list, are committed to processor that supplies the earliest start time. Maximum finish time of the last tasks on all over processors is calculated which it is desirability of obtained scheduling of the ant. With respect to this desirability, measure of

pheromone depositing on the visited states is calculated by:

$$\Delta\tau_{ij}^{k} = \frac{1}{L^{k}} \text{ if (i, j) } \in T^{k} \quad (8)$$

where L^k is finish time obtained by ant k and T^k is executed tour of this ant.

That is, Δij will be deposited on τ (i, j) only if job j would be selected just after task i. otherwise, τ (i, j) will be remained unchanged. In the forth stage, the best ant until now (Ant^{min}), is selected as the best solution. Extra pheromone is deposited on states visited by the best ant by using:

$$\Delta\tau_{ij}^{\min} = \frac{1}{L^{\min}} \quad if\,(i, j) \in T^{\min} \quad (9)$$

In the last stage, by using (9), pheromone variables are decreased simulating pheromone evaporation in real environments. It must be taken to avoid premature convergence (stagnation) because of local minima:

$$\tau(i, j) = (1-\rho)\,\tau(i, j) \quad (10)$$

where, ρ is evaporation rate in the range of [0, 1].

PROPOSED IMPROVED PARTICLE SWARM OPTIMIZATION (IMPSO) 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 11:

$$V_i = w \times V_i + C_{1g} \times r_1 \times (P\ best_i - S_i) \times P\ best_i + C_{1b} \times r_2 \times (S_i - P_{worst\,i}) \times P_{worst\,i} + C_2 \times r_3 \times (Gbest_i - S_i) \quad (11)$$

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. Figure 7 shows the concept of ImPSO searching points.

The algorithmic steps for the Improved PSO is as follows:

Step 1: 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.

Step 2: Initialize the particle position and velocity.

Step 3: 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 11 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 'F' by the updated new velocity and position.

The flowchart for the proposed model formulation scheme is shown in Figure 8. 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 1.

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. Figure 9

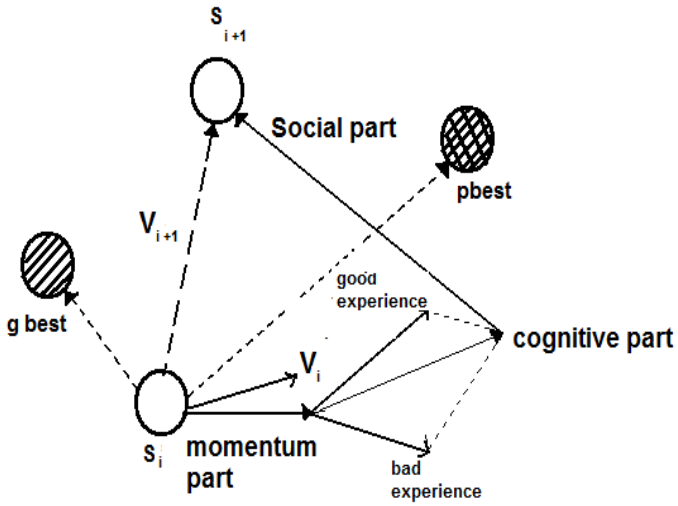


Figure 7. Concept of Improved Particle Swarm Optimization search point.

shows the variation in finishing time and waiting time for the assigned number of jobs and processors using improved particle swarm optimization.

HYBRID ALGORITHM (IMPSO WITH AIS) FOR JOB SCHEDULING

The proposed improved PSO algorithm is independent of the problem and the results obtained using the improved PSO can be improved with 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", that is, 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, that is, particle moving away from the solution point.

Step 5: Update velocity and position of particle as per Equation 5, 10.

Step 6: If best particle is not changed over a period of time, 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 Figure 10.

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 C1g, C1b and C2 are 1.5, w = 0.5. The values are utilized as per the previous literatures and as suggested by the author of PSO (Kenedy and Eberhart, 1995). The finishing time and waiting time completed for the random instances of jobs are as shown in Table 1.

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, with improved PSO, better solutions have been achieved. Figure 11 shows the variation in finishing time and waiting

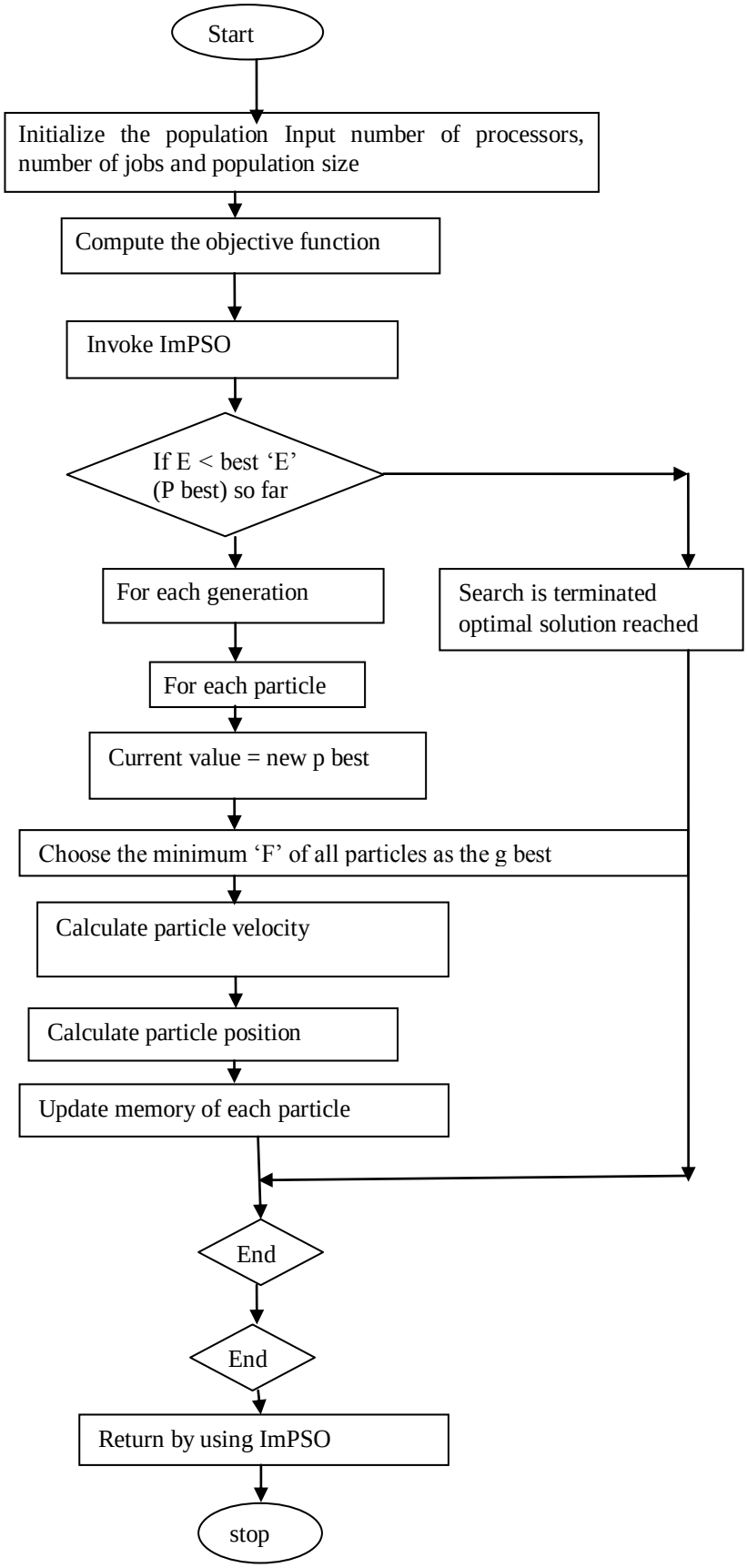


Figure 8. Flowchart for job scheduling using Improved PSO.

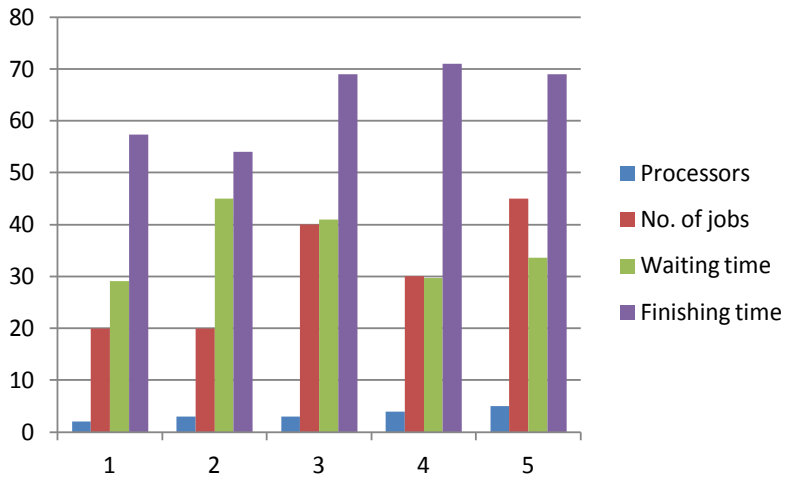


Figure 9. Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using ImPSO.

time for the assigned number of jobs and processors using Hybrid algorithm.

PROPOSED HYBRID ALGORITHM (IMPSO WITH ACO) FOR JOB SCHEDULING

The proposed improved PSO algorithm is independent of the problem and the results obtained using the improved PSO can be further improved with Ant Colony Optimization (ACO).

The steps involved in the proposed hybrid algorithm are as follows:

Step 1: Initialize ACO parameters.

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 “ thar is, 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 that is, particle moving away from the solution point.

Step 5: Update velocity and position of particle as per Equation 5, 11.

Step 6: If best particle is not changed over a period of time, (a) Generate solutions from each ants’ random walk.

Step 7: Update pheromone intensities.

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 Figure 12.

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 1.

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 Ant Colony Optimization (ACO) and improved PSO, better solutions have been achieved. Figure 13 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using Hybrid algorithm.

DISCUSSION

The growing heuristic optimization techniques have been applied for job scheduling in multiprocessor architecture. Table 1 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.

In this paper, the four intelligent algorithms (GA, PSO, AIS, and ACO) have been involved. These intelligent

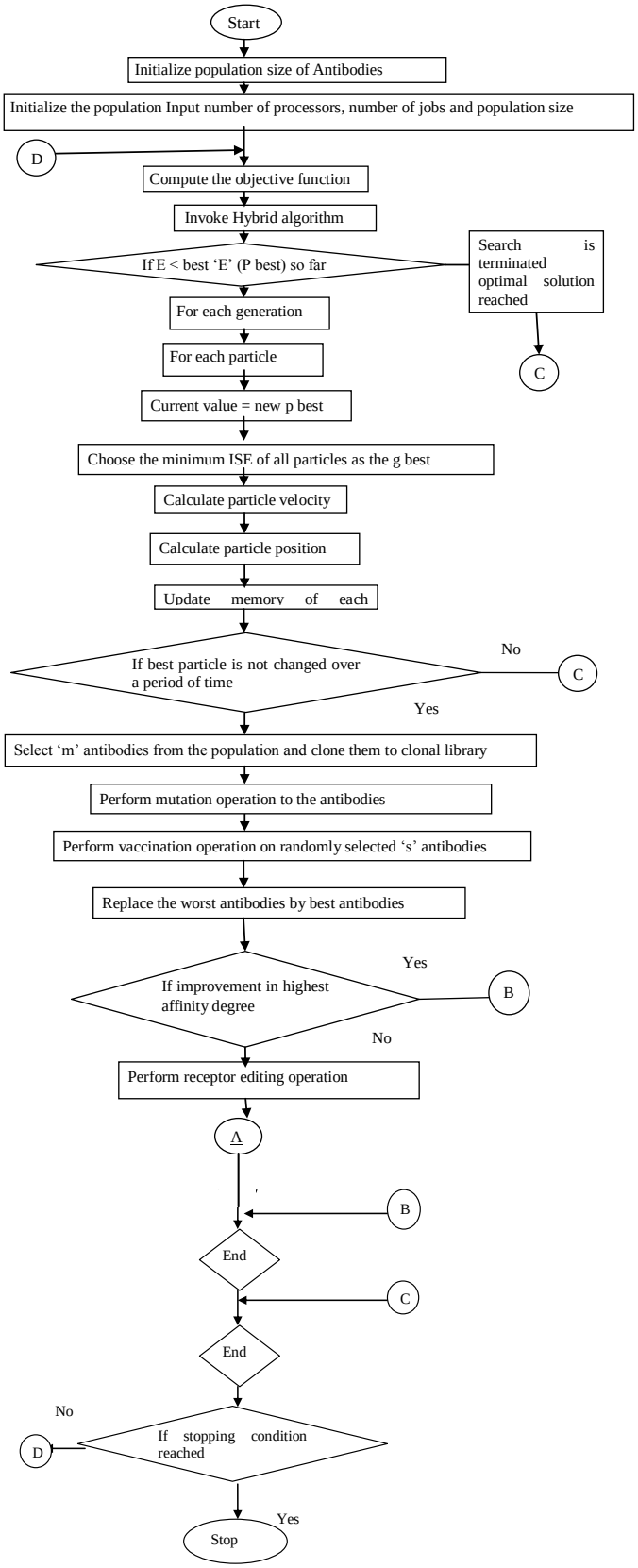


Figure 10. Flowchart for job scheduling using Hybrid algorithm (Improved PSO with AIS) for job scheduling in Multiprocessor Architecture.

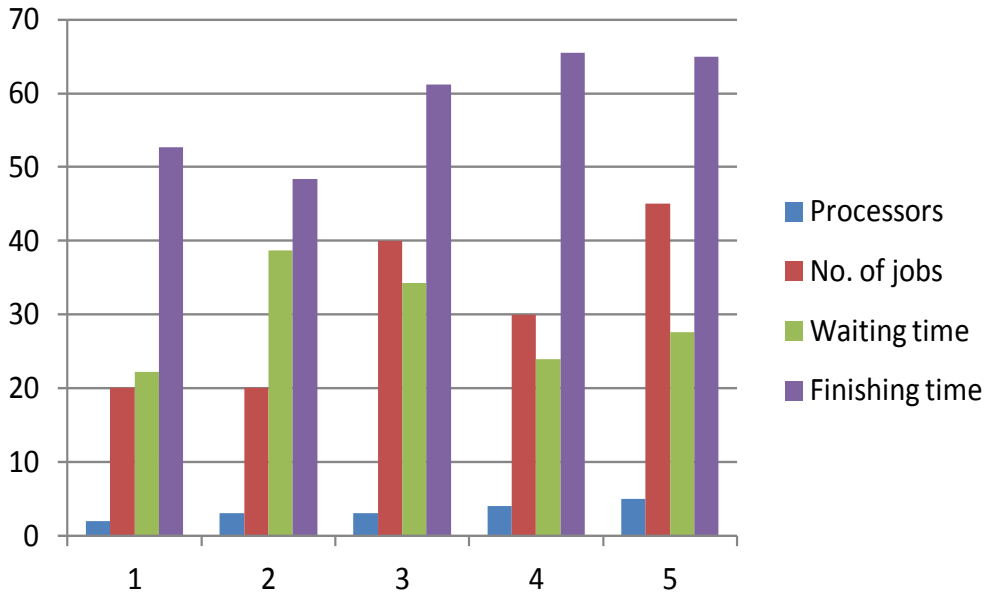


Figure 11. Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using Hybrid algorithm (Improved PSO with AIS).

algorithms, has been applied to the mentioned scheduling problem either individually (GA or PSO) or hybridized (AIS, ACO). Based on the convergence to the solutions and the minimization of the objective function, the validity of the considered algorithms is enforced. GA and PSO involve more time in comparison with that of the two said hybridized approaches. Also due to the behavioral and convergence characteristics, the proposed improved PSO with ACO provides a better solution. The problems values are randomly generated.

In LPT algorithm, Friesen et al. (1987,) Coffmanet al. (1972) 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 Genetic Algorithm (GA). The proposed improved Particle Swarm Optimization (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 Particle Swarm Optimization (PSO), minimized the waiting time and finishing time of the processors with respect to all the other considered algorithms.

The Improved Particle Swarm Optimization (ImPSO) combined with the concept of Artificial Immune System, a hybrid algorithm was proposed and it has reduced the finishing time and waiting time. In Artificial Immune System (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 Particle Swarm Optimization.

Further, Improved Particle Swarm Optimization (ImPSO) is combined with Ant Colony Optimization (ACO) and a new hybrid algorithm proposed. The new hybrid algorithm reduces finishing and waiting time drastically. In ACO each ant incrementally constructs a solution to the problem. When an ant complete solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. Ants deposit a certain amount of pheromone on the components; that is, either on the vertices or on the edges that they traverse. The amount of pheromone deposited may depend on the quality of the solution found. Subsequent ants use the pheromone information as a guide toward promising regions of the search space. Ants adaptively modify the way the problem is represented and perceived by other ants, but they are not adaptive themselves.

Thus based on the results, it can be observed that the proposed hybrid algorithm gives better results than the conventional methodologies LPT, SPT and other heuristic optimization techniques Genetic Algorithm (GA), General Particle Swarm Optimization (PSO) and Proposed Improved Particle Swarm Optimization (ImPSO). This work was carried out in Intel Pentium i3core processors with 2 GB RAM and it took around 68 seconds by the CPU for the complete simulation of 451 generations.

Thus, when independently the Improved PSO takes more convergence time, the hybrid Improved Particle Swarm Optimization along with Ant Colony Optimization,

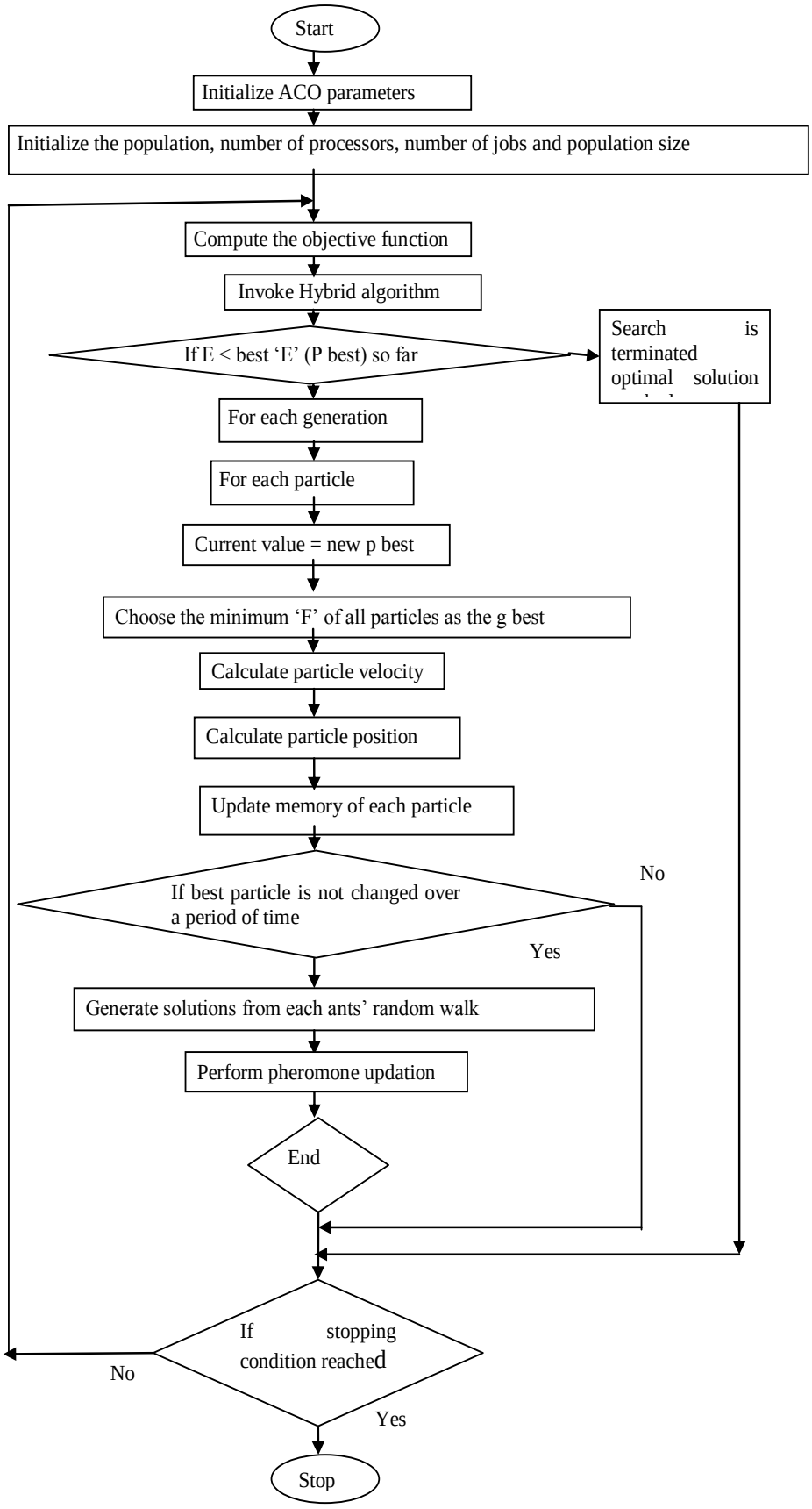
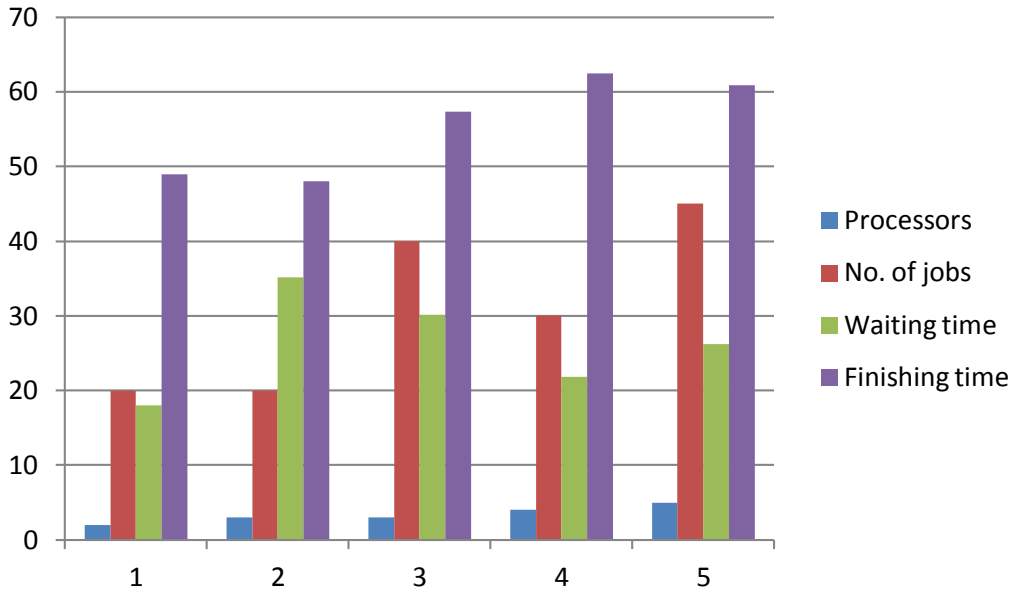


Figure 12. Flowchart for job scheduling using Improved PSO with ACO.



Figurer 13. Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using Hybrid algorithm (Improved PSO with ACO).

reduces the finishing and waiting time of the jobs.

CONCLUSION

In this paper, a new hybrid algorithm based on the concept of Ant Colony Optimization and proposed improved particle swarm optimization has been developed and applied to multiprocessor job shop scheduling. Ant Colony Optimization has positive feedbacks for rapid discovery of good solutions and a simple implementation of pheromone-guided will improve the performance of Improved Particle Swarm Optimization (ImPSO). It has been proven that Improved Particle Swarm Optimization was able to discover reasonable quality solutions much faster than other evolutionary algorithms. However, Particle Swarm Optimization does not possess the ability to improve upon the quality of the solutions as the number of generations is increased.

In our application of Improved Particle Swarm Optimization / Ant Colony Optimization approach, a simple pheromone-guided search mechanism of ant colony was implemented which acted locally to synchronize positions of the particles in Particle Swarm Optimization to attain the feasible domain of the objective function faster.

The proposed algorithm partitioned the jobs in the processors by attaining minimum waiting time and finishing time in comparison with the other algorithms, longest processing time, shortest processing time, genetic algorithm, particle swarm optimization and also the proposed particle swarm optimization. The worst component being included along with the best component and Ant Colony Optimization tends to minimize the waiting time and finishing time, by its cognitive behavior drastically. Thus the proposed algorithm (ImPSO with ACO), for the same number of generations, has achieved better results.

REFERENCES

Adam TL, Chandy KM, Dicson JR (1974). A Comparison of List Schedules for Parallel Processing Systems. Commun. ACM, 17: 685-690.

Ahmad I, Dhodhi MK (1996). Multiprocessor scheduling in a genetic paradigm. Parallel Comput., 22: 859-866.

Amirthagadeswaran KS, Arunachalam VP (2005). Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. International Journal Advanced Manufacture Technology (Spiringer), pp. 532-540.

Baxter J, Patel JH (1989). The LAST Algorithm: A Heuristic- Based Static Task Allocation Algorithm. Int. Conf. Parallel Process., 2: 217-222.

Chen BA (1993). Note on LPT scheduling. Oper. Res. Lett., 14: 139-142.

Coello CAC, Rivera DC, Cortes NC (2003). Use of an artificial immune system for job Shop scheduling, in Proc. 2nd Int. Conf. Artif. Immune Syst., 2787: 1-10.

Coffman Jr EG, Graham RL (1972. Optimal scheduling for two-processor systems. Acta Inf., 1: 200-213.

Dobson G (1984). Scheduling independent tasks on uniform processors. SIAM J. Comput., 13: 705-716.

Dorigo M, Birattari M, Stützle T (2006). Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique. IEEE Computational Intelligence Magazine.

Dorigo M, Socha K (2007). An Introduction to Ant Colony Optimization. Gonzalez TF, Approximation Algorithms and Metaheuristics. CRC Press.

Dorigo M, Stützle T (2002). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. Handbook of Metaheuristics.

Elnaz ZM, Amir MR, Mohammad R, Feizi D (2008). Job Scheduling in Multiprocessor Architecture Using Genetic Algorithm. Proc. IEEE, pp. 248-250.

Engelbrecht AP (2005). Fundamentals of Computational Swarm Intelligence. John Wiley & Son.

Friesen DK (1987). Tighter bounds for LPT scheduling on uniform processors. SIAM J. Comput., 6: 554-660.

Gagne C, Price WL, Gravel M (2002). Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence dependent setup times. J. Oper. Res. Soc., 53: 895-906.

Graham R (2003). Static Multi-processor scheduling with Ant Colony Optimization and Local search. Master of Science thesis, University of Edinburgh, pp. 1-101.

Gur M, Daniel O (2006). Open- shop batch scheduling with identical jobs. Eur. J. Oper. Res. (Elsevier), pp. 1282-1292.

Gur M, Uri Y (2004). Comments on Flow shop and open shop scheduling with a critical machine and two operations per job. European Journal of Operational Research(Elsevier), pp. 257-261.

Hong-Wei G, Liang S, Yan-Chun L, Feng Q (2008). An Effective PSO and AIS-Based Hybrid Intelligent Algorithm for Job-Shop Scheduling. IEEE Transactions On Systems, Man, And Cybernetics-Part A: Syst. And Hum., 38(2).

Kenedy J, Eberhart RC (1995). Particle Swarm Optimization. proc. IEEE Int. Conf. Neural Networks. Pistcataway, pp. 1942-1948.

Lee CY, Hwang JJ, Chow YC, Anger FD (1998).Multiprocessor Scheduling with Interprocessor Communication Delays. Oper. Res. Lett., 7(3): 141-147.

Lei Z, Yuehui C, Runyuan S, Shan J, Bo Y (2008). A Task Scheduling Algorithm Based on PSO for Grid Computing. Int. J. Comput. Intell. Res., 4(1): 37-43.

Lin FT, Hsu CC (1990). Task assignment Scheduling by simulated annealing', in proceed. conf. Comput. Commun. Syst. Hong Kong, 1: 279-283.

Mitten L (1970). Branch and Bound Method: general formulation and properties. Oper. Res., 18: 24-34.

Morrison JF (1998). A note on LPT scheduling. Oper. Res. Lett., 7: 77-79.

Qing-dao-er-ji R, Yuping W, Xiaojing S (2010). Improved genetic Algorithm for Job shop scheduling problem Sch. of Comput. Sci. & Technol., Xidian Univ., Xi'an, China , Computational Intelligence and Security (CIS), 2010 Int. Conf., p. 113.

Sanchez JEP, Trystram D (2005). A New Genetic Convex Clustering Algorithm for Parallel Time Minimization with Large Communication Delays. in Proceedings of the International Conference on Parallel Computing, ParCo , Malaga, Spain, 33: 709-716.

Selvakumar S, Siva Ram Murthy C (1994), Scheduling Precedence Constrained Task Graphs with Non-Negligible Intertask Communication onto Multiprocessors. IEEE Trans. Parallel Distrib. Syt., 5(3): 328-336.

Sha DY, Cheng-Yu Hsu (2007). A new particle swarm optimization for open shop, pp. 3243-3261.

Shelokar PS (2007). Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continues Optimization. Appl. Math. Comput., 188: 129-142.

Shi Y, Eberthart R (1999). Empirical study of particle swarm optimization. Proc. IEEE Congr. Evol. Comput., pp. 1945-1950.

Sih GC, Lee EA (1990). Scheduling to Account for Interprocessor Communication Within Interconnection- Constrained Processor Network. Int. Conf. Parallel Process., 1: 9-17.

Sivanandam SN, Deepa SN (2007). Introduction to Genetic Algorithm. Springer verlog.

Gupta SR, Smith JS (2006). Algorithms for single machine total tardiness scheduling with sequence dependent setups. European Journal of Operational Research (In Press).

Thanushkodi K, Deeba K (2009). An Evolutionary Approach for Job Scheduling in a Multiprocessor Architecture. CiiT Int. J. Artif. Intell. Syst. Mach. Learn., 1(4).

Thanushkodi K, Deeba K (2011). A New Improved Particle Swarm Optimization Algorithm for Multiprocessor Job Scheduling. Int. J. Computer Sci. Issues. 8(4).

Thanushkodi K, Deeba K (2011). A Comparative study of proposed improved PSO algorithm with proposed Hybrid Algorithm for Multiprocessor Job Scheduling. International Journal of Computer Science and Information Secur. 9(6).

Thanushkodi K, Deeba K (2011). On Performance Comparisons of GA, PSO and proposed Improved PSO for Job Scheduling in Multiprocessor Architecture. Int. J. Comput. Sci. Netw. Secur., pp. 27-34.

Tung-Kuan L, Jinn- Tsong T, Jyh-Hong C (2005). Improved genetic algorithm for the job-shop scheduling problem. International Journal Advanced Manufacture Technology (Spiringer), pp. 1021-1029.

Tzu-Chiang C, Po-Yin C, Yueh-Min H (2006). Multiprocessor Tasks with Resource and Timing Constraints Using Particle Swarm Optimization. Int. J. Comput. Sci. Netw. Secur. 6(4): 71-77.

Wu MY, Gajski DD (1990). Hypertool: A Programming Aid for Message_Passing Systems. IEEE Trans Parallel Distrib. Comput., 1(3): 330-343.

Yang S, Wang D (2000). Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling. IEEE Trans. Neural Network, 11(2): 474-486.

Yang S, Wang D (2001). A new adaptive neural network and heuristics hybrid approach for job-shop scheduling. Comput. Oper. Res., 28(10): 955-971.

Yang T Gerasoulis A (1993). List Scheduling with and without Communication Delays. Parallel Comput., 19: 1321-1344.

Yu HB, Liang W (2001). Neural network and genetic algorithm-based hybrid approach to expanded job- shop scheduling, Comput. Ind. Eng., 39(3/4): 337-356.

Zhang XD, Yan HS (2005). Integrated optimization of production planning and scheduling for a kind of job-shop. Int. J. Adv. Manuf. Technol. (Spiringer), pp. 876-886.

Zhou M, Sun SD (1999). Genetic algorithms: theory and application. National Defense Industry Press. Beijing, China, pp. 130-138.