CHAPTER 3

HYBRID PARTICLE SWARM OPTIMIZATION TECHNIQUE FOR TASK SCHEDULING

3.1 INTRODUCTION

Efficient application scheduling is critical for achieving high performance in homogeneous cluster computing systems. Because of its prime importance, the task scheduling problem has been extensively studied and various heuristics techniques have been proposed. To improve the efficiency of the heuristics based approach, there are metaheuristics like Simulated Annealing, Tabu Search, Particle Swarm Optimization (PSO), Genetic Algorithm, etc. Recently, Metaheuristics techniques have received a lot of attention. The common features of these algorithms are starting with the initial solution and updating through a successive iteration process to improve the solution.

PSO is one of the population-based search techniques. It optimizes an objective function by iteratively improving a swarm of solution vectors called particles, based on the special management of memory (Shi 2004). Each particle is modified by referring to the memory of individual and swarm's best information. Due to the collective intelligence of these particles, the swarm is able to repeatedly improve its best observed solution and to converge on an optimum. It also has fewer algorithm parameters than the Genetic Algorithm (GA) and Simulated Annealing (SA).

The idea of applying Tabu Search (TS) is inspired from its solution structure and works with more than one solution (neighborhood solution) at a time (Glover and Laguna 1997). Moreover, TS has the memory structure, and can store the past information of a certain period. It prevents from repeating the same search process or forbids movement to the best solution at the present for a certain period. The TS metaheuristic is proposed to the static task scheduling problem whose makespan is improved approximately by 25% with respect to the schedule generated by the Best Greedy Algorithm (Porto and Ribeiro 1995b).

The work in this chapter proposes a new algorithm, named PSO-TS, which combines PSO with the concept of TS. That has been taken as a hint for the improvement of the task scheduling problem with precedence as constraint, and task as non-preemptive. The experimental results show that the proposed hybrid approach outperforms the GA and the existing list scheduling heuristics.

3.2 PARTICLE SWARM OPTIMIZATION (PSO)

3.2.1 Fundamentals

PSO is a novel Evolutionary Algorithm and a swarm intelligence computation technique (Kennedy and Eberhart 1995). It is somewhat similar to the Genetic Algorithm in that the system is initialized with a population of random solutions. Unlike other algorithms, however, each potential solution is also assigned a randomized velocity and then flown through the problem hyperspace. The Particle Swarm Optimization has been found to be extremely effective in solving a wide range of engineering problems.

3.2.2 Historical Background

The implicit rules adhered to by the members of the bird flocks and fish schools, that enable them to move synchronized, without colliding, resulting in an amazing choreography, were studied and simulated by several scientists (Heppner and Grenander 1990), (Reynolds 1987). In simulations, the movement of flocks was an outcome of the individuals (Eberhart and Shi 1998). The social behavior of animals, and in some cases of humans, is governed by similar rules (Wilson 1975). However human social behavior is more complex than a flock's movement. Besides physical motion, humans adjust their beliefs, moving, thus in a belief space. Although two persons cannot occupy the same space of their physical environment, they can have the same beliefs, occupying the same position in the belief space, without collision. This abstractness in human social behavior is intriguing and has constituted the motivation for developing simulations of it. There is general belief, and numerous examples coming from nature enforces the view, that social sharing of information among the individuals of a population may provide an evolutionary advantage. This was the core idea behind the development of PSO (Eberhart and Shi 1998).

3.2.3 Working Principle of PSO

PSO is basically developed through the simulation of bird flocking in two dimensional spaces. The position of each agent is represented by the XY axis position and also the velocity is expressed as vx (the velocity of X axis) and vy (the velocity of Y axis). The modification of the agent position is realized by the position and the velocity information. Bird flocking optimizes a certain objective function. Each agent knows its best value so far (pbest) and its XY position. This information is an analogy of personal experiences of each agent. Moreover, each agent knows the best value so far in the group (gbest) among (pbest). This information is the analogy of the knowledge of how other agents around them have performed. Each agent tries to modify its position using the following information:

- The current position (x,y)
- The current velocity (vx,vy)
- The distance between the current position and pbest
- The distance between the current position and gbest

In a PSO system, a particle or an individual depicted by its position vector X and its velocity vector V is a candidate solution to the problem. Considering D dimensions and NP population as an example, the velocity and position for the i th ($i \in [1, NP]$) particle is represented as $V_i = (v_{i1}, \dots v_{id}, \dots v_{iD})$ and $X_i = (x_{i1}, \dots x_{id}, \dots x_{iD})$ respectively, where $x_{id} \in [ld, ud]$, $d \in [1, D]$, ld, ud are the lower and upper bounds for the d th dimension. During the search process, the particle successively adjusts its position according to two factors: one is the best position found by itself (pbest), denoted by $p_{id} = (p_{i1}, p_{i2}, \dots p_{id})$; the other is the best position found so far by its neighbors (gbest), denoted by $p_{gd} = (p_{i1}, p_{i2}, \dots p_{gd})$. The personal best position (pbest) of the particle 'i' is the best position visited by particle 'i' so far.

In a standard PSO algorithm, a population of initialized solutions search through a multidimensional solution space. With each member of the population, it continually adjusts its position and velocity through learning its own experience and the experience of the other members of the population. The particles are evolved according to the following Equation (3.1).

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{3.1}$$

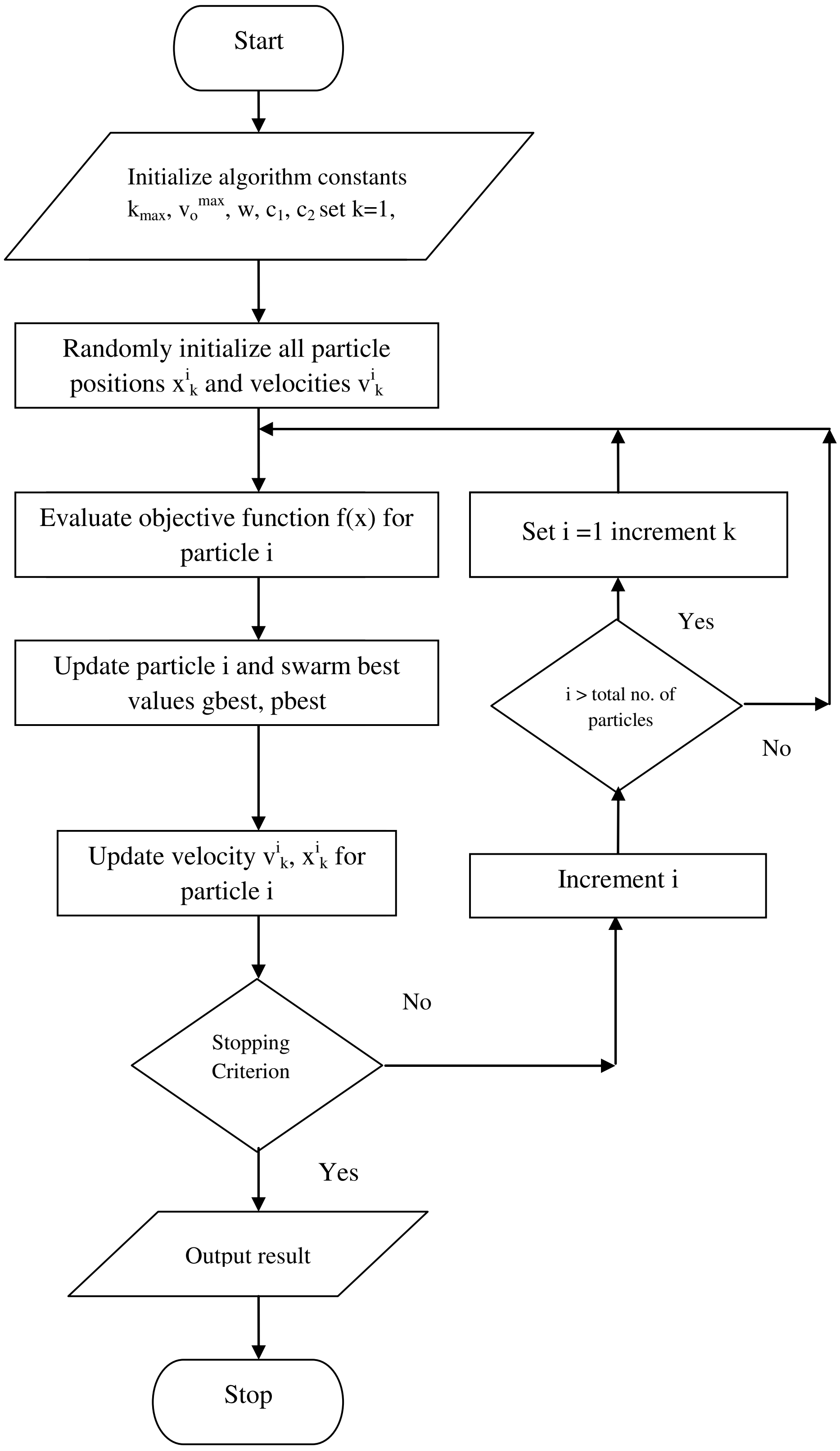
$$x_{id} = x_{id} + v_{id} \tag{3.2}$$

where c_1 and c_2 are acceleration constants; r_1 and r_2 are two random numbers in the range [0, 1]. This is the basic form of the PSO algorithm which is shown in Figure 3.1.

```
Step 1:  Initialize swarm with random positions and velocities;
Step 2:  Begin
           Repeat For each particle
                 Evaluate the fitness i.e. makespan of the scheduler;
                 If current fitness of particle is better than $P_{id}$ then set $P_{id}$ to
                   current value;
                    If $P_{id}$ is better than global best then set $P_{gd}$ to current
                       particle fitness value;
                      Change the velocity and position of the particle;
                    End If;
                 End If;
          Until termination = True
         End
```

Figure 3.1 The PSO Algorithm

In Equation (3.1), the first part represents the inertia of the previous velocity, the second part is the "cognition" part, representing the private experience by itself; the third part is the "social" part, representing the cooperation among the particles (Dorigo and Gambardella 1997). If the sum of the accelerations causes the velocity v_{id} on that dimension to exceed $v_{max,d}$ then v_{id} is limited to $v_{max,d}$ which determines the region between the present position and the target positions. The main steps of the PSO are given as a flowchart in Figure 3.2.



Start
Initialize algorithm constants k_{max}, v_o^{max}, w, c_1, c_2 set k=1,
Randomly initialize all particle positions x^i_k and velocities v^i_k
Evaluate objective function f(x) for particle i
Update particle i and swarm best values gbest, pbest
Update velocity v^i_k, x^i_k for particle i
Stopping Criterion
No
Yes
Output result
Stop
Increment i
i > total no. of particles
Yes
No
Set i =1 increment k

Figure 3.2 Main Steps of PSO

3.2.4 Metaheuristic Approaches

3.2.4.1 Tabu Search

Tabu Search (TS) is one of the most efficient heuristic techniques in the sense that it finds quality solutions in relatively short running time. It applies restrictions to guide the search to diverse regions.

Tabu search (Glover 1995) is a neighborhood search technique that tries to avoid the local minima and attempts to guide the search towards a global minimum. Tabu search starts with an initial solution, which can be obtained by applying a simple one-pass heuristic, and scans the neighborhood of the current solution-that is, all the solutions that differ from the current one by a single move. For the multiprocessor task-scheduling problem, a move consists of moving a task from one processor to some other processor, or changing the order of execution of a task within the list of tasks scheduled to a processor. This technique considers all the moves in the immediate neighborhood, and accepts the move which results in the best makespan.

The local search is usually a simple greedy strategy which finds an improved solution in the immediate neighborhood of a current solution. In order to stop the search repeating itself, recently visited solutions are added to the tabu list, and if a solution found by the local search already exists in the tabu list, it is forbidden, or ‘taboo’, and an alternative, non-tabu, solution is used. To ensure that the search doesn’t quickly exclude all neighbors, solutions are only held in the tabu list for some period of time-the tabu tenure. The mechanism used to alter the search path varies from problem to problem, but a common approach is to simply pick a random solution in the neighborhood of the current solution. Sometimes, the TS makes use of non-improving solutions for the sake of avoiding the local optima.

Porto and Ribeiro (1995a) applied the Tabu Search metaheuristics to the static task scheduling problem under precedence constraints. The results obtained by Tabu Search considerably improved by approximately 25% the makespan (the overall completion time) of the parallel applications, with respect to the schedule generated even by the best greedy algorithm. The pseudocode for TS procedure is shown in Figure 3.3.

```
Tabu Search ( )
Step 1: Begin
            T: = [ ];
            s: = initial solution;
            s* = s;
            Repeat
                  Find the best admissible s'ϵ N (s);
                    If f (s') > f (s*) then s* = s';
                    s: = s';
                    Update Tabu list, T;
             Until Stopping criterion;
       End;
```

Figure 3.3 Tabu Search Algorithms

where T is the tabu list and N (s) is the set of neighborhood solutions.

Since the Tabu Search algorithm found (Burke et al 1997 and Glover 1990) to be an extremely efficient algorithm for this work, it has been used as a local search. The generic flowchart for Tabu Search is shown in Figure 3.4.

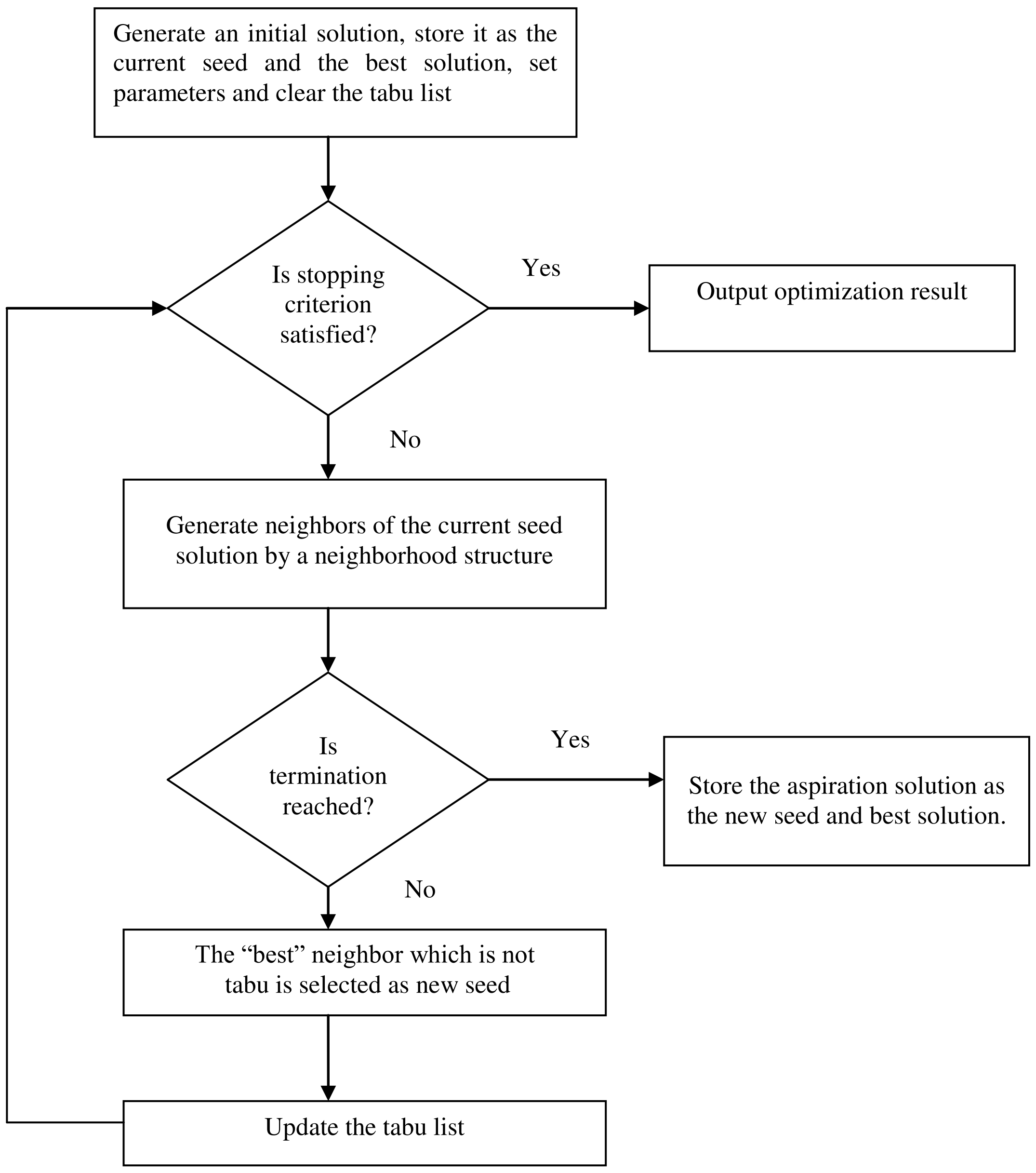

Generate an initial solution, store it as the current seed and the best solution, set parameters and clear the tabu list
Is stopping criterion satisfied?
Yes
Output optimization result
No
Generate neighbors of the current seed solution by a neighborhood structure
Is termination reached?
Yes
Store the aspiration solution as the new seed and best solution.
No
The "best" neighbor which is not tabu is selected as new seed
Update the tabu list

Figure 3.4 Generic Flowchart of Tabu Search

3.2.4.2 Simulated Annealing

Simulated Annealing (SA) is a Monte Carlo (Eliasi et al 1990) approach for the optimization functions. This approach derives from the roughly analogous physical process of heating and then cooling a substance to obtain a strong crystalline structure. The Simulated Annealing process lowers the temperature by slow stages until the system "freezes" and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state or equilibrium. Like the GA, SA is also a non-deterministic algorithm.

SA is a stochastic heuristic algorithm in which the solutions are searched in hill climbing processes constantly commenced by random moves. It is an extremely popular method for solving large-sized and practical problems like job-shop scheduling, timetabling and traveling salesman because of its ease of use. However, for various reasons, like many other search algorithms, SA may become trapped by any local minima, which does not allow moving up or down, or take a long time to find a reasonable solution, which makes the method unpreferrable sometimes. For these reasons, many SA implementations have been done as a part of a hybrid method. The structure of the SA is shown in Figure 3.5.

Its main advantages over other local search methods are its flexibility and its ability to approach global optimality. The algorithm is quite versatile since it does not rely on any restrictive properties of the model. The SA methods are easily "tuned". For any reasonably difficult nonlinear or stochastic system, a given optimization algorithm can be tuned to enhance its performance and since it takes time and effort to become familiar with a given code, the ability to tune a given algorithm for use in more than one problem should be considered an important feature of an algorithm.

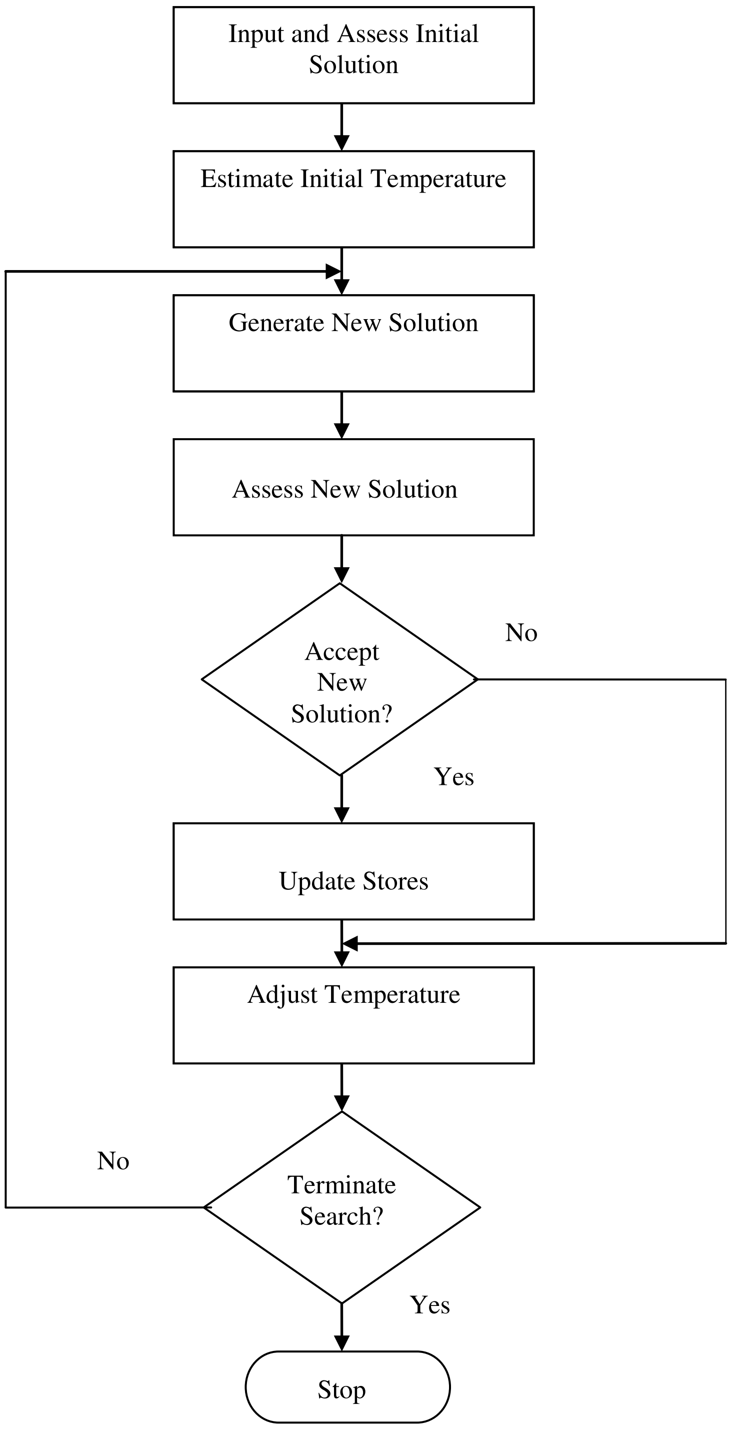


Figure 3.5 The structure of the Simulated Annealing Algorithm

3.3 HYBRID STRATEGY

Modern metaheuristics manage to combine exploration and exploitation search. The exploration search seeks new regions, and once it finds a good region, the exploitation search kicks in. However, since the two strategies are usually inter-wound, the search may be conducted to other regions before it reaches the local optima. As a result, many researchers suggest employing a hybrid strategy which embeds a local optimizer such as Hill Climbing heuristic in between the iterations of the metaheuristics. It has been shown in many applications that a hybrid strategy that combines a metaheuristic algorithm with a local heuristic can obtain a better result (Michalewicz and Fogel 2002, Cristina Boeres 2003) .

Both SA and TS are local methods searching for better solutions through neighborhood move on the current solution. Defining a neighborhood move as transforming the current solution into another solution, a subset of solutions called the neighbor can be generated. The TS algorithm starts with a feasible solution and keeps improving it in successive iterations by exploring its neighborhood, finally the best candidate instead of the current solution (Glover 1989). A key in Tabu Search is constructing the tabu list which follows the track of the recently visited solutions to avoid cycling or getting trapped in a local optimum. The Tabu algorithm has gradually been utilized in the scheduling domain (Porto et al 2000). In addition, the metaheuristic methods generally outperformed the exact or heuristic methods.

3.4 THE HPSO ALGORITHM

Although PSO is very robust and has a good global exploration capability, it has the tendency of being trapped in local minima and slow convergence. In order to improve its performance, many researchers experimented with hybrid PSO algorithms. Poli et al (2007) gave a review of the variations and the hybrids of the PSO. The basic idea of the hybrid algorithm presented in this work is to run the PSO first and then improve the result by employing a Tabu Search heuristics.

The details of the proposed Hybrid PSO (referred to as HPSO) algorithm are presented in Figure 3.6. The proposed PSO-TS algorithm starts with an initial swarm of K particles. Each particle vector corresponds to a candidate solution of the underlying problem. Then, all of the particles repeatedly move until a maximal number of iterations have been passed. During each iteration, the particle's individual best and swarm's best positions are determined. The particle adjusts its position based on the individual experience ($pbest_i$) and the Swarm's intelligence (gbest) as described in Equations (3.1) and (3.2). The global best (gbest) is calculated using the PSO. The particle that gives global best is set as the initial solution for Tabu Search. Subsets of neighbors are generated and evaluated to find the best solution.

To expedite the convergence speed, all of the particles are further updated using the TS before entering the next iteration. When the algorithm is terminated, the incumbent gbest and the corresponding fitness value as output are considered as the optimal task assignment with the minimum cost.

3.4.1 HPSO Implementation

If the PSO algorithm is designed to a discrete problem, a direct correlation must be found between the particle vector and the solution of a scheduling problem. Hence, the position and velocity of particles as well as operations considering the precedence relations between tasks are redefined. One of the key issues in designing a successful PSO algorithm is the representation step, (i.e.) finding a suitable mapping between problem solution and PSO particle.

Encoding position: In this work, the position vector of a particle represents a feasible solution. That is the position is encoded as a node list satisfying the DAG graph topology order which is based on task permutation (Tandon, 2000). Here the particle is represented as a string of length which is equal to the number of nodes in the DAG.

Defining Swap Operator and Swap Sequence (SS): For n tasks, a normal solution S is a sequence of n nodes. If a new solution S ' is obtained when exchanging node n_i and node n_j in solution S, the operation is defined as Swap Operator, denoting SO (n_i, n_j) and the process as S' = S+ SO(n_i, n_j).

For example,

$$\{n_1, n_2, n_3, n_4, n_5, n_6\}+SO\,(n_1,n_3)=\{n_3, n_2, n_1, n_4, n_5, n_6\} \tag{3.3}$$

when Swap Operators SO_1, SO_2 , SO_3, ... SO_n are imposed to a solution continuously, the action is defined as the Swap Sequence (SS), denoting

$$SS = (SO_1, SO_2, SO_3 ... SO_n) \tag{3.4}$$

Here the order of Swap Operators acting is very important.

Encoding the velocity: According to Equation (3.1), a new position is updated when velocity V is imposed on old position, so the velocity can be defined as SS acting on a scheduling list. In a specific scheduling environment,

$$V= \{(n^k_i, n^k_j), i, j \ \varepsilon \ \{1, 2 \ldots n\}, k \ \varepsilon \ \{1, 2 \ldots n\}\} \tag{3.5}$$

which represents that node n^1_i, n^1_j are swapped first, and then n^2_i and n^2_j are swapped second and so forth.

The evolution equation of PSO in task scheduling problem: Using the concept described above, the following evolution Equations (3.6) and (3.7) in task scheduling problem can be obtained.

$$V_i(t+1)=V_i(t)\oplus c_1r_1(P_i-X_i(t))\oplus c_2r_2(P_g-X(t)) \tag{3.6}$$

$$X_i(t+1)=X_i(t)+V_i(t+1) \tag{3.7}$$

In the approach of this thesis, c_1 and c_2 are fixed to the value of 0.5 and r_1 and r_2 are a random number in the range [0,1].

The particle competes for the best solution during the evolution according to a measure of solution quality called fitness. Then the swarm evolution is navigated towards the optimal solution by the best particles.

Fitness is used to evaluate the performance of particles in the swarm. Generally, choosing a proper objective function as the fitness function to represent the corresponding superiority of each particle is one of the key factors for the successful resolution of the relevant problem using the HPSO algorithm. The fitness of the particle is measured with the maximum computation time of all the tasks. The mapping of the tasks to the nodes in cluster is given in Figure 3.7. A particle with the lowest completion time is a good solution which has to be kept in the search process.

3.5 PSO-TS ALGORITHM

The proposed PSO-TS method combines the excellence of both PSO and TS. In the case of the PSO, when a particle discovers a good solution, other particles gather around the solution “global best” (gbest), too. Therefore they cannot escape from a local optimal solution. Consequently, the PSO cannot achieve global searches. With a view to achieve global searches, the new algorithm based on PSO for getting hints from the concept of the TS is proposed. It is named PSO-TS. A subset of neighbors is generated and evaluated to find the best solution.

In the search process of PSO-TS, the best solution at the present i.e. the pbest and the gbest which are obtained in the search so far might not be updated for a certain tabu period, T_1. At this time, there are two possibilities. One is the occasion that the particles have reached the global optimal solution and another is that they have been caught in a local optimal solution. To search for the solution space effectively, it is hoped that the particles search intensively around the best solution at present when they have reached around the global optimal solution, and search for getting away from the solution at present when they have reached a local optimal solution. That is, in the later case, approaching the best solution at the present is assumed to be tabu. In the search process, however, particles cannot make judgment whether the solution at the present is local or global. So the particles are divided into two swarms, one is swarm1 which plays the role of local searches, and the other is swarm2 which plays the role of global searches. These two swarms can search efficiently for the solutions space.

Swarm1 playing the role of intensification searches around the area around the best solution at the present, and swarm2 plays the role of diversification, intends to avoid the local optimal solutions and to find global optimal too. The particles of swarm1 search under the condition of fixing their position in the value of gbest at the dimension for which the solutions have not been updated, and the particles of swarm2 search by moving away from the pbest or gbest at the present at that dimension. By this movement, particles have possibilities of discovering a better solution different from the solution at the present i.e., PSO-TS can achieve both operations of intensification and diversification by the action of these two kinds of swarms. If the tabu periods T_2 would pass, then the tabu states are released and returns to the former, the PSO algorithm. Then the search process is repeated.

```
Step 1:  Initialize
           Generate K particles either by permutation or by topological sorting.
           Generate velocities $v_{ij}$, $1 \leq i \leq K$ and $1 \leq j \leq r$,
              where $v_{ij}$ is randomly drawn from [0.0, 1.0].
Step 2:  **Repeat until a given maximal number of iterations (or) stopping criteria**
          is achieved.
              Evaluate the fitness i.e. makespan of each particle.
                  **If** current fitness of particle is better than $P_{id}$ then set $P_{id}$ to
                    current value;
                      **If** $P_{id}$ is better than global best then set $P_{gd}$ to current
                         particle fitness value;
                      Update velocities $v_{ij}$ restricted by a maximum threshold $v_{max}$.
                      Update particles' vectors.
                      **End If;**
                  **End If;**
                Improve the solution quality of each particle using tabusearch.
```

Figure 3.6 The PSO-TS Algorithms for the Task Scheduling Problem

3.6 EXPERIMENTAL RESULTS AND DISUCSSIONS

In order to evaluate the performance of the proposed algorithm, the proposed approach was applied for the task scheduling problem in the cluster of workstations and is implemented using C and run on a Pentium IV 3.2 GHz machine.

Ten sets of experiments have been completed with an increment of 10 resources which vary from 10 to 100 resources. In each experiment, a DAG is generated automatically in which the size varies from 10 to 100 nodes. The weight of each node was randomly selected from a normal distribution which is equal to the mean of the specified average workload. The weight of each edge was also randomly selected from a normal distribution with mean equal to the specified value. Each individual in the population is called a particle. Here the particle is represented as a string of length which is equal to the number of nodes in the DAG. Each particle represents a candidate solution.

2	1	0	1	3	2	1	0	0	3
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}

Figure 3.7 Particle Representation

Figure 3.7 represents the particle representation for four nodes in a cluster for the DAG in Figure 2.2. The entries represent the processor id. The vector shows that task 1 is given to the processor whose id = 2 and task 2 is allotted to processor whose id = 1 and so on. The makespan of this schedule is 210.

Consider a swap operation, SO (t_3, t_6). Now the new particle is shown in Figure 3.8. The makespan of this new particle is 199 which is less compared to the previous value. Hence the corresponding particle will be updated.

2	1	**2**	1	3	**0**	1	0	0	3
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}

Figure 3.8 Particle Representation after Swap Operation SO (t_3, t_6)

The parameter values used in both the PSO-TS and the GA for comparison only are optimally tuned by intensive preliminary experiments to let the competing algorithms perform at the best level. The GA parameters used are (Population size = 20, Crossover probability = 0.8; Mutation probability = 0.09; Maximum Iterations = 50). The parameters used in the proposed algorithm are (Swarm size = 20; inertia weight = 0.9 to 0.1; $c_1=c_2=1.49$; Maximum iterations = 50). The criteria of performance considered were to minimize the application makespan (schedule length), which is the time elapsed between the first application task starts executing and the last task is completed. The average makespan values for 10 trails for different CCRs are illustrated in Table 3.1.

Table 3.1 Performance Comparison

Algorithm	**Average Makespan**
GA	38.04
PSO	37.48
PSO-TS	36.66

The performance of the proposed algorithm in terms of the speedup by varying the number of processors in the cluster is shown in Figure 3.9. The speedup increases as the number of tasks in a DAG gets increased.

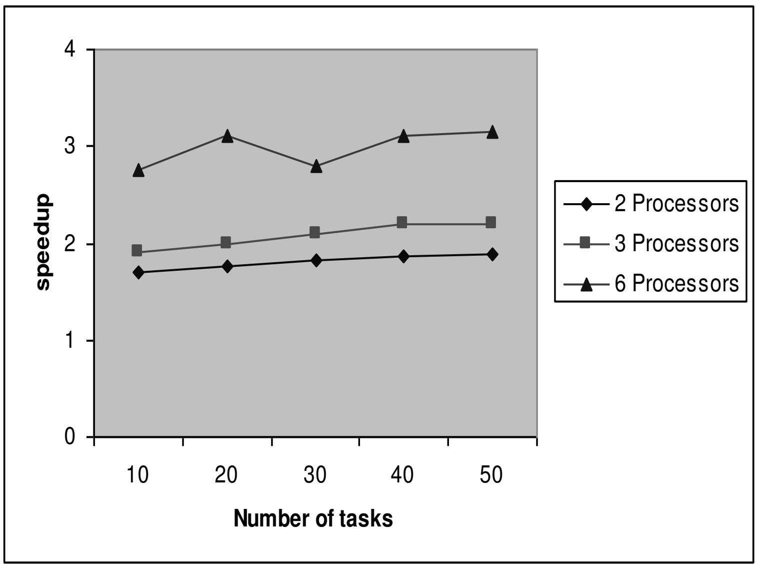


Figure 3.9 Speedup vs. Number of Tasks

The proposed approach outperforms the GA which is illustrated in Figure 3.10. It shows the variation in the makespan generated by the PSO-TS and the GA as the number of tasks in a DAG is increased. The proposed method spent shorter time to complete the scheduling than GA. PSO has been proved to be an efficient method for many global optimization problems and in some cases it does not suffer the difficulties encountered by other evolutionary computation techniques.

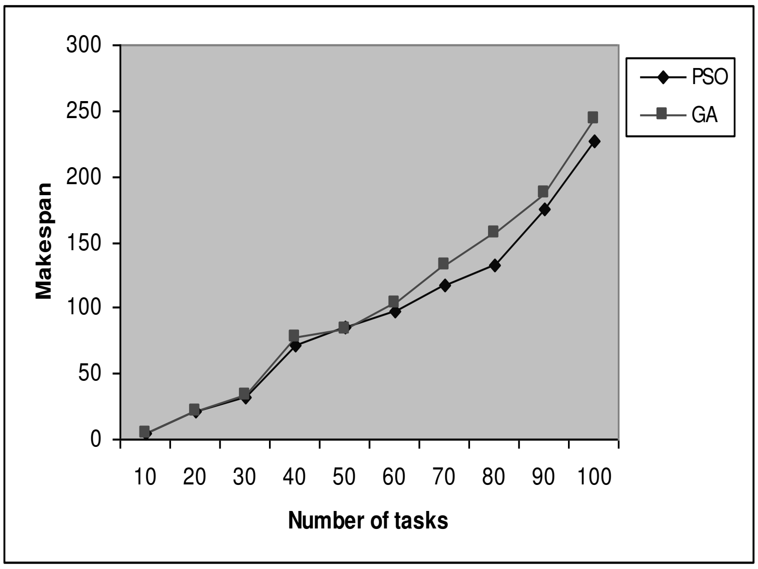


Figure 3.10 Makespan vs. Number of Tasks

The proposed approach performs exactly as the GA for a smaller number of tasks but when tasks are increased then considerable variations occur in the makepsan. This shows that the proposed approach gives better result for medium to coarse grain applications.

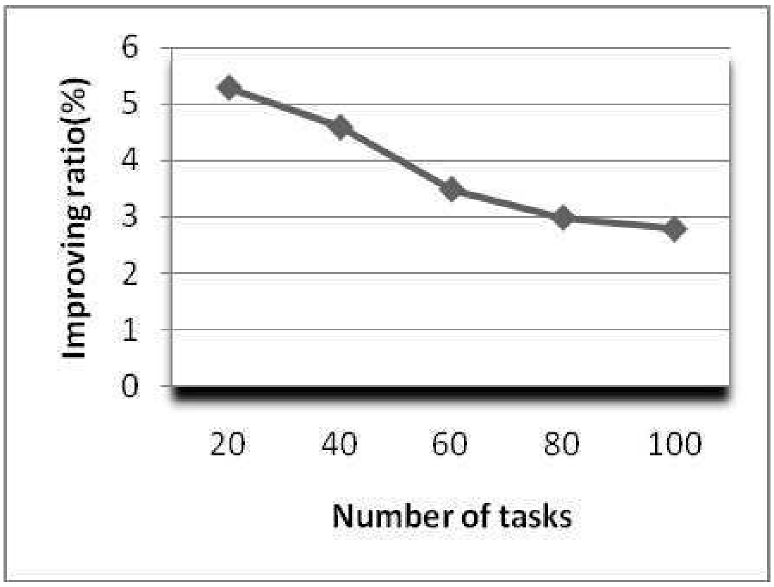


Figure 3.11 Average Improving Ratio of Proposed Method

The percentage improvement in the makespan of the proposed approach is shown in Figure 3.11. The improving ratio is defined as the ratio of the best result of the proposed approach to the worst result of the other algorithms. It is also defined as the decreasing ratio of the makespan of the PSO-TS to the GA under the identical scheduling problem. With increasing task size, the improving ratio decreases because the large scale problem requires more time to search for the solution.

3.7 SUMMARY OF THE CHAPTER

In the proposed method, the essential factors of TS are focused such as i) Movement away form a present best solution (taking a change for the worse) is allowed in the search process; ii) The algorithm memorizes information on the past solutions in tabu lists; iii) If the tabu periods pass, tabu states are released and the memory is deleted. All possible task scheduling schemes are assumed as a DAG. Thus the problem is mapped into a graph optimal selection problem, and to find the optimal solution quickly and accurately, a hybrid PSO with Tabu search technique is proposed. The experimental results show that the proposed approach is effective in solving the task scheduling problem. The average makespan of the proposed hybrid metaheuristic approach is 36.66 while the average makespan of the GA is 38.04 for different CCRs. The scheduling model in this work assumes negligible communication cost. Contention for communication resources is not considered, yet it has a strong influence on the execution time of an application. Further work investigates the contention awareness in task scheduling through edge scheduling.