CHAPTER 5

PROPOSED HYBRID HEURISTIC APPROACH FOR MULTIPROCESSOR SCHEDULING

The present chapter proposes a hybrid approach using Improved Particle Swarm Optimization algorithm (IPSO) with Ant Colony Optimization (ACO) for multiprocessor scheduling problem with static and dynamic tasks.

5.1 INTRODUCTION

Ant colony metaheuristic is a concurrent algorithm in which a colony of artificial ants cooperate to find optimized solutions of a given problem. The Ant Colony Optimization (ACO) algorithm was introduced by Dorigo which is a probabilistic technique for solving computational problems.

The Ant Colony Optimization is helpful in guiding the heuristic algorithms to obtain a good solution to NP-hard problems. Hence, a new hybrid algorithm based on the concept of IPSO and ACO has been developed and applied to multiprocessor scheduling problem. Ant Colony Optimization has positive feedback for rapid discovery of good solutions and a simple implementation of pheromone-guided will improve the performance of the Improved Particle Swarm Optimization (IPSO).

Hybrid heuristic approach IPSO-ACO for solving the multiprocessor scheduling problem is dealt with in the present chapter. The objective of applying the ACO with IPSO algorithm is to enhance the performance of multiprocessor scheduling problem. The ACO concept is used
to generate an initial population of points that are scattered uniformly over the feasible solution space, so that the algorithm can evenly scan the feasible solution space to locate good points for further exploration in subsequent iterations. Pheromone evaporation in ACO also has the advantage of avoiding the convergence to a locally optimal solution.

ACO algorithms perform better than Simulated Annealing and GA approaches for the dynamic task scheduling problems, since the Ant Colony algorithm can be run continuously and adapt to changes in real time.

5.2 REVIEW OF LITERATURE

Dorigo et al (1996) proposed a new computational paradigm called Ant System (AS) to stochastic combinatorial optimization. The proposed methodology is applied to the classical Travelling Salesman Problem (TSP). The obtained results are compared with Tabu Search and Simulated Annealing. The Ant System (AS) can be applied to other optimization problems like the asymmetric travelling salesman, the quadratic assignment and the job-shop scheduling to demonstrate the robustness of the approach.

Dorigo and Gambardella (1997) introduces the ant colony system (ACS), a distributed algorithm that is applied to the Travelling Salesman Problem (TSP). In the proposed method, a set of cooperating agents called ants cooperate to find good solutions to TSP’s. The results show that the ACS outperforms other nature-inspired algorithms such as Simulated Annealing and Evolutionary Computation, and comparing ACS-3-opt, a version of the ACS augmented with a local search procedure, to some of the best performing algorithms for symmetric and asymmetric TSP’s.

Ritchie (2003) proposed the Ant Colony Optimization algorithm with the combination of the local search operators for solving two important
versions of static scheduling of independent jobs onto homogeneous and heterogeneous processors.

Ying and Liao (2003) presented an Ant Colony System (ACS) approach to continuously improve the constructive heuristics. The developed ACS approach is verified on the single machine total weighted tardiness problem. The results show that the proposed method can effectively improve the robustness of various constructive heuristics, and outperform the other existing heuristic algorithms for well-known benchmark problems.

Liang and Smith (2004) used an Ant Colony meta-heuristic optimization method to solve the Redundancy Allocation Problem (RAP). Meta-heuristic methods offer a practical way to solve large instances of the relaxed RAP where different components can be placed in parallel. He tested the proposed algorithm on a well-known suite of problems from the literature, and the results show that the ant colony method performs with little variability over problem instance or random number seed. It is competitive with the best known heuristics for redundancy allocation.

Rajendran and Ziegler (2004) proposed two Ant Colony algorithms for permutation flow shop scheduling problem to minimize the makespan. Holden and Freitas (2005) proposed a hybrid PSO/ACO algorithm for hierarchical classification, where the classes to be predicted are arranged in a tree-like hierarchy. The performance of the algorithm is evaluated on a challenging biological data set, involving the hierarchical functional classification of enzymes. The proposed algorithm is compared with an existing PSO for classification, which was also adapted for hierarchical classification.

Gómez-Cabrero et al (2005) proposed a hybrid algorithm based on Ant Colony System (ACS), which is a distributed (agent based) algorithm.
The optimum parameters for the proposed algorithm have to be found by trial and error. Hence, he used a Particle Swarm Optimization algorithm (PSO) to optimize the ACS parameters working in a designed subset of TSP instances. The hybrid PSO-ACS algorithm is applied on a single instance to find the optimum parameters and optimum solutions for the instance. The algorithm is then used to analyze those sets of optimum parameters, in relation to instance characteristics. Computational results have shown good quality solutions for single instances though with high computational times and that there may be sets of parameters that work optimally for a majority of instances.

Heinonen and Pettersson (2007) applied the hybrid Ant Colony Optimization algorithm to job-shop scheduling problem. ACO tries to preserve and improve the existing solutions, and a post processing algorithm is applied to the tour of an ant upon its completion.

Chang et al (2008) developed an ant colony system to address the multistage job shop scheduling problem and also multiple objective scheduling. The author considered the marketing criteria in addition to the production objectives. The system has been developed with the necessitated ACO components redeveloped or adapted and multiple quantitative and qualitative objectives, soft constraint, and penalty function. The ACS provides a more computationally efficient result.

Chen et al (2008) presented a modified local search ant colony optimization algorithm on the basis of alternative ant colony optimization algorithm for solving flow shop scheduling problems, to improve the quality of solutions. From the simulation rule, random order transition rule used in ACO with local search integrated is an effective scheme for the flow shop scheduling problem.
Arnaout et al (2008) introduced a two stage Ant Colony Optimization (ACO) algorithm to solve non-preemptive unrelated parallel machine scheduling problem with machine dependent setup times, with the objective of makespan minimization. The performance of the proposed algorithm is evaluated by comparing its solutions to the solution of Tabu Search and an existing heuristic for the same problem. The results show that ACO outperformed the other algorithms.

Nada and Salami (2009) proposed a Hybrid method to solve a combinatorial optimization problem by using Ant Colony and Genetic programming algorithms. Evolutionary process of Ant Colony Optimization algorithm adapts genetic operations to enhance ant movement towards the solution state. The algorithm converges to the optimal final solution, by accumulating the most effective sub-solutions.

Zhou et al (2009) identified the appropriate domains of ACO in the area of dynamic job shop scheduling problem. He has tested the algorithm in a shop floor scenario with three levels of machine utilisations, three different processing time distributions and three different performance measures for intermediate scheduling problems. Two series of experiments are conducted to identify the best ACO strategy and best performing dispatching rule. Those two approaches are compared with different processing times. The experiments show that the ACO outperforms the other approaches.

Yumiaozhou et al (2009) presented a new 2opt called PC-2opt based on the general model of ant colony algorithm for solving the Resource-Constrained Project Scheduling Problem (RCPSP) optimization problem. The proposed method PC-2opt, need not to calculate the location of successors, could be directly used to solve the RCPSP, and improve the time efficiency.
It is proved by experiments that the local search mechanism presented is marginally more feasible, effective and better than other similar algorithms.

Boveiri (2010) proposed a new approach named ACO-MTS to multiprocessor task scheduling based on Ant Colony Optimization, which is a multi-agent approach, in which agents (artificial ants) try to find the shortest path for solving the given problem using an indirect communication. The proposed ACO-MTS is evaluated in comparison with not only the traditional heuristics but also the Genetic Algorithm. However, the genetic algorithm examines two more solutions to achieve the best scheduling compared to ACO-MTS. The presented results demonstrate that the proposed approach is very successful in multiprocessor task scheduling.

Zhang et al (2010) developed an Ant Colony Optimization based approach for the satellite control resource scheduling problem. The solution space of the problem is considered as sparse, two pheromone updating methods, i.e., the reinitialize-guidance updating and current-guidance-updating methods, are proposed to avoid the trapping in local optima, by changing the distribution of pheromone trails by updating them with a guidance solution once the algorithm stagnates. The proposed algorithm was compared with several other heuristics. The experimental results demonstrate that the new approach is competitive in terms of exploration capability of reaching the near-global optimal solution and adaptability to the future situations.

Surekha et al (2010) presents a Genetic Algorithm and Ant Colony Optimization algorithm for solving Job Shop Scheduling Problem (JSSP). The sequences of jobs are scheduled using Fuzzy Logic and optimized using GA and ACO. The makespan, algorithm efficiency, elapsed time and completion time for the Genetic Algorithm and Ant Colony Optimization are evaluated
and compared. The parameters used for the improvement in the performance are also discussed. Computational results are analyzed using benchmark instances.

Yagmahan and Yenisey (2010) proposed an algorithm which combines Ant Colony Optimization approach with local search strategy in order to solve the objectives of makespan and flowtime. The proposed algorithm is tested with well-known problems from literature. The computational results show that the proposed algorithm is more effective.

Senthilkumar et al (2011) proposed a hybrid algorithm based on Particle Swarm Optimization algorithm (PSO) and Ant Colony Optimization algorithms (ACO) to generate optimal solutions for different weighted earliness and tardiness measures for unrelated parallel machine earliness-tardiness non-common due date sequence-dependent set-up time scheduling problem. Fuzzy Logic approach has been used to select the optimal weighted earliness tardiness combinations in an unrelated parallel machine environment. The hybrid algorithm identifies the best sequence for the different weighted combinations of earliness and tardiness measures for each given set of jobs. The algorithm has been evaluated using benchmark parallel machine earliness tardiness scheduling problems. The performance of the combined objective function obtained by the proposed hybrid technique has been compared with the solutions yielded by the Genetic Algorithm techniques. The comparison shows that the proposed hybrid technique outperforms GA-Fuzzy techniques.

Kim and Kang (2011) proposed an algorithm for off-line communication-aware task scheduling and voltage selection using Ant Colony Optimization, which minimizes the total energy consumption of an application executing on a homogeneous multiprocessor system. The artificial
ants explore the search space, for which voltage selection and dependencies between tasks are considered. The pheromone trails represent the global heuristic information in order to utilize the entire energy consumption information from the previous evaluated solutions. Experimental results show that the proposed algorithm outperforms in terms of total energy consumption.

Ahmadizar et al (2012) developed a new Ant Colony Optimization algorithm for solving the permutation flowshop problem such as minimization of completion time. In the proposed novel mechanism, the pheromone trails are based on an initial sequence. Also, pheromone intensities are limited between lower and upper bounds which change dynamically. After the complete sequence of jobs are constructed, a local search is performed to improve the performance quality of the solution. The proposed algorithm is applied to Taillard’s benchmark problems. The results show that the proposed algorithm is better.

Tavares et al (2012) describe how the current literature uses the ACO approach to solve scheduling problems. The author concluded that ACO is a hugely viable approach to solve scheduling problems. On the basics of the review of literature, the authors determined the possible directions for future research.

5.3 BASIC ACO ALGORITHM FOR TASK SCHEDULING

The algorithm is as follows (Boveiri 2010),

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

Consider a \( n \times n \) matrix named \( \tau \) as pheromone variables, in which \('n' is number of jobs. \( \tau(i, j) \) is desirability of selecting job \( j \) just after job \( i \). Initially all the elements of the matrix have the same, small value (initial pheromone = 0.1)

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

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

\[
P^k(i, j) = \frac{\tau(i, j)}{\sum_{k \in N} \tau(i, k)}
\]

(5.1)

where \( N \) is set of ready jobs.

A random number is then generated and the next job is selected with respect to the generated number. It is clear that the jobs which have a bigger pheromone value have bigger chance to be selected. The selected job is appended to the ant list and removed from the 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, the tasks are extracted one by one from the ant's list. They are committed to the processor that supplies the earliest start time. Maximum finish time of the last task on all processors is calculated which is the desirability of the obtained scheduling of the ant. With respect to this desirability, the measure of pheromone depositing on the visited states is calculated by,

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

(5.2)

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

That is, $\Delta ij$ will be deposited on $\tau(i, j)$ only if job j would be selected just after task i. otherwise, $\tau(i, j)$ will be remain unchanged. In the fourth stage, the best ant until now (Ant$^{\text{min}}$), is selected as the best solution. Extra pheromone is deposited on states visited by the best ant by using,

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

(5.3)

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

(5.4)

where, $\rho$ is evaporation rate in the range of $[0, 1]$.

In the last stage, by using Equation (5.4), pheromone variables are decreased simulating pheromone evaporation in real environments. It must be taken to avoid premature convergence (stagnation) because of local minima.
5.4 PROPOSED HYBRID HEURISTIC APPROACH

The steps involved in the proposed hybrid algorithm is 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”, 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 means the particle is moving away from the solution point.

**Step 5** : Update the velocity and position of particle as per Equations (2.1) and (2.2).

**Step 6** : If best particle is not changed over a period of time,

a) Generate solutions from each ant’s random walk

**Step 7** : Update pheromone intensities

**Step 8** : Terminate the process if the maximum number of iterations are reached or optimal value is obtained, else go to step 3.

The flow chart for the hybrid algorithm is shown in Figure 5.1.
Start

Initialize ACO parameters

Initialize the population, number of processors, number of jobs and population size

Compute the objective function

Invoke Hybrid algorithm

If E < best ‘E’ (P best) so far

Yes

Search is terminated optimal solution reached

No

For each generation

For each particle

Current value = new p best

Choose the minimum ‘F’ of all particles as the g best

Figure 5.1 Flowchart for the proposed hybrid heuristic approach IPSO-ACO
Calculate the particle velocity using (2.1)

Calculate the particle position using (2.2)

Update memory of each particle

If the best particle is not changed over a period of time

Yes

Generate solutions from each ant’s random walk

Perform pheromone updation

End

If stopping condition reached

Yes

Stop

No

Figure 5.1 (Continued)
The proposed hybrid heuristic algorithm IPSO-ACO is applied to the multiprocessor scheduling of two cases namely static and dynamic task scheduling problems.

5.5 SIMULATION PROCEDURE

The details of the simulation carried out for implementing the proposed heuristic hybrid algorithm is presented here.

Benchmark datasets are taken from EricTailard’s site for dynamic task scheduling. Two datasets are taken for simulation. Data set 1 involves 50 tasks and 20 processors. Data set 2 involves 100 tasks with 20 processors. The datasets for static scheduling is randomly generated such as 2 processors with 20 tasks, 3 processors with 20 tasks, 3 processors with 40 tasks, 4 processors with 30 tasks, 4 processors with 50 tasks, 5 processors with 45 tasks and 5 processors with 60 tasks.

To demonstrate the effectiveness of the proposed hybrid algorithm, the proposed approach IPSO-ACO is run 30 independent trials with different values of random seeds and control parameters. The optimal result is obtained for the following parameter settings.

Ant Colony Optimization

- Number of ants : Based on the number of Processors
- Number of Iterations : 500
- Pheromone evaporation \( \rho \) : 0.7
Improved Particle Swarm Optimization

- The initial solution is generated randomly
- \( C_{1g}, C_{1b} \) and \( C_2 \) = 2, 2 and 2
- Population size = Twice the number of tasks (Salman et al. 2002)
- \( W_{\text{min}} - W_{\text{max}} \) = 0.5
- Max. Iteration = 500

The proposed hybrid approach IPSO-ACO is developed using MATLAB R2009 and executed in a PC with Intel core i3 processor with 3 GB RAM and 2.13 GHz speed.

5.6 STATIC TASK SCHEDULING

The independent tasks are considered for static scheduling. To reduce the total finishing time and average waiting time, the objective function is the same as specified in the Equations (2.4) to (2.9). This section gives the details of the hybrid heuristic approach IPSO-ACO for static multiprocessor task scheduling.

5.6.1 Results and Discussion

The hybrid heuristic approach IPSO-ACO is tested with static independent task scheduling problem and the obtained results are shown in Table 5.1.
Table 5.1 Total finishing time and waiting time using IPSO-ACO

<table>
<thead>
<tr>
<th>No of processors</th>
<th>No of jobs</th>
<th>Proposed Hybrid approach</th>
<th>IPSO-ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AWT</td>
<td>TFT</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>18.02</td>
<td>48.92</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>35.12</td>
<td>48.00</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>30.16</td>
<td>57.32</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>21.87</td>
<td>62.45</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>24.63</td>
<td>67.45</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>26.21</td>
<td>60.87</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>28.42</td>
<td>64.26</td>
</tr>
</tbody>
</table>

The proposed hybrid heuristic approach IPSO-ACO produces the average waiting time as 30.16s and total finishing time as 57.32s for the dataset 3 processors with 40 tasks. For the dataset 5 processors with 45 tasks, IPSO-ACO produces 26.21s as average waiting time and 60.87s as total finishing time.

5.6.2 Performance Comparison

In order to validate the performance of the heuristic hybrid approach IPSO-ACO, comparisons have been made with the approaches IPSO-SA and IPSO-AIS for the same problem with same datasets, and are reported in Table 5.2. The results obtained are comparatively better for IPSO-ACO than the other approaches.

IPSO algorithm is hybridized with the heuristic approach ACO to further reduce the average waiting time and the total finishing time of all the jobs in static task scheduling. Based on the convergence to the solutions and the minimization of the objective function, the validity of the considered algorithms is enforced.
For the dataset 2, processors with 20 tasks, IPSO-SA produces average waiting time of 25.61s, total finishing time of 54.23s, IPSO-AIS produces average waiting time of 22.16s, total finishing time of 52.64s and the proposed hybrid heuristic approach IPSO-ACO produces average waiting time of 18.02s, total finishing time of 48.92s.

Table 5.2  Comparison of job total finishing time and average waiting time using IPSO-SA, IPSO-AIS and the proposed hybrid heuristic approach IPSO -ACO

<table>
<thead>
<tr>
<th>No of processors</th>
<th>No of jobs</th>
<th>IPSO-SA</th>
<th>IPSO-AIS</th>
<th>Proposed Hybrid heuristic approach (IPSO-ACO)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AWT</td>
<td>TFT</td>
<td>AWT</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>25.61</td>
<td>54.23</td>
<td>22.16</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>40.91</td>
<td>50.62</td>
<td>38.65</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>38.45</td>
<td>65.40</td>
<td>34.26</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>26.51</td>
<td>66.29</td>
<td>23.92</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>28.34</td>
<td>68.01</td>
<td>25.96</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>30.12</td>
<td>66.43</td>
<td>27.56</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>32.76</td>
<td>69.13</td>
<td>30.19</td>
</tr>
</tbody>
</table>

For the dataset 3 processors with 40 tasks, IPSO-SA produces average waiting time of 38.45s, total finishing time of 65.40s, IPSO-AIS produces average waiting time of 34.26s, total finishing time of 61.20s and the proposed hybrid heuristic approach IPSO-ACO produces 30.16s as average waiting time and 57.32s as total finishing time. For the dataset 5 processors with 45 tasks, IPSO-SA produces 30.12s as average waiting time, 66.43s as total finishing time, IPSO-AIS produces average waiting time of 27.56s, 64.96s as total finishing time and the proposed IPSO-ACO produces 26.21s as average waiting time, 60.87s as total finishing time.
The variations of total finishing time and waiting time for the hybrid approaches, IPSO-SA, IPSO-AIS and IPSO-ACO are shown from Figures 5.2 to 5.8.

Thus, the results reveal that the proposed heuristic approach IPSO-ACO comparatively performs better and simultaneously minimises both average waiting time and total finishing time. It is observed, that in the case of proposed hybrid algorithm IPSO-ACO, 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 IPSO-SA and IPSO-AIS.

![Total Finishing time and Average waiting time](image)

**Figure 5.2** Total finishing time and average waiting time for 2 processors with 20 jobs using IPSO-SA, IPSO-AIS and IPSO-ACO
Figure 5.3  Total finishing time and average waiting time for 3 processors with 20 jobs using IPSO-SA, IPSO-AIS and IPSO-ACO

Figure 5.4  Total finishing time and average waiting time for 3 processors with 40 jobs using IPSO-SA, IPSO-AIS and IPSO-ACO
Figure 5.5  Total finishing time and average waiting time for 4 processors with 30 jobs using IPSO-SA, IPSO-AIS and IPSO-ACO

Figure 5.6  Total finishing time and average waiting time for 4 processors with 50 jobs using IPSO-SA, IPSO-AIS and IPSO-ACO
It has been proven that IPSO-ACO is able to determine reasonable quality solutions much faster than other hybrid algorithms such as IPSO-SA and IPSO-AIS. The cause for the improvement in the results is that, the worst
component is being included along with the best component and is also combined with Ant Colony Optimization. This tends to minimize the average waiting time and total finishing time, simultaneously. Thus, the proposed hybrid heuristic approach IPSO-ACO has achieved better results.

5.7 DYNAMIC TASK SCHEDULING WITHOUT LOAD BALANCING

The main aim of the task scheduling is to achieve the minimum total execution cost for dynamic task scheduling. The objective function is represented in the Equations (2.10) to (2.12).

5.7.1 Results and Discussion

The proposed hybrid heuristic approach IPSO-ACO is applied to Dynamic Task Scheduling without load balancing and the obtained results such as best cost, worst cost, average cost and convergence time have been tabulated and shown in Table 5.3.

Table 5.3 Best cost, worst cost, average cost and convergence time for IPSO-SA, IPSO-AIS and the proposed hybrid heuristic approach IPSO-ACO for dynamic task scheduling without load balancing

<table>
<thead>
<tr>
<th>Method</th>
<th>IPSO-SA</th>
<th>IPSO-AIS</th>
<th>Proposed hybrid heuristic approach IPSO-ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>50</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Best Cost</td>
<td>2156</td>
<td>4376</td>
<td>2136</td>
</tr>
<tr>
<td>Worst Cost</td>
<td>2901</td>
<td>4908</td>
<td>2886</td>
</tr>
<tr>
<td>Average Cost</td>
<td>2528.5</td>
<td>4624</td>
<td>2511</td>
</tr>
<tr>
<td>Convergence Time in seconds</td>
<td>4.2156</td>
<td>5.8428</td>
<td>4.9124</td>
</tr>
</tbody>
</table>
The obtained results of best, average and worst cost for dynamic task scheduling using the proposed hybrid heuristic approach IPSO-ACO has been compared with hybrid approaches IPSO-SA and IPSO-AIS.

For dataset1, the best cost produced by IPSO-SA is 2156, IPSO-AIS produced best cost as 2136 and the proposed hybrid approach IPSO-ACO produced best cost as 2131. For the dataset 2, the best cost produced by IPSO-SA is 4376, the best cost produced by IPSO-AIS is 4309 and the best cost obtained by the proposed hybrid heuristic approach IPSO-ACO is 4226. From the results, the best cost and average cost obtained are better in the case of the proposed hybrid approach IPSO-ACO, when compared to the other methods. The convergence time for the proposed Hybrid algorithm IPSO-ACO is slightly higher (approximately1 time) than with the hybrid approaches IPSO-SA and IPSO-AIS, because of the extra calculation involved in the ant searching method to find the best solution.

The best cost obtained using the proposed hybrid heuristic approach IPSO-ACO for data set 1 and data set 2 are shown in Figures 5.9 and 5.10.

![Best cost for 50 tasks and 20 processors using IPSO-SA, IPSO-AIS and IPSO-ACO](image)

Figure 5.9  Best cost for 50 tasks and 20 processors using IPSO-SA, IPSO-AIS and IPSO-ACO
Figure 5.10 Best costs for 100 tasks and 20 processors using IPSO-SA, IPSO-AIS and IPSO-ACO

The results infer that the IPSO-ACO perform better than the other hybrid approaches IPSO-SA and IPSO-AIS.

5.7.2 Performance Comparison

The performance of the proposed hybrid heuristic approach IPSO-ACO is compared with the previously proposed ((Visalakshi and Sivanandam 2009) hybrid PSO algorithms PSO-HC and PSO-SA for the same datasets and for multiprocessor dynamic task scheduling problem.

For dataset 1, the best cost achieved by PSO-HC is 2322, PSO-SA achieves the best cost of 2186 and the proposed hybrid approach IPSO-ACO achieves a best cost of 2131 which is better than the approaches compared. For dataset 2, the best cost achieved by PSO-HC is 4621, PSO-SA achieves the best cost as 4496 and the proposed hybrid heuristic approach IPSO-ACO achieves best cost as 4226 which is better than that of PSO-HC and PSO-SA. The convergence time of the proposed hybrid approach IPSO-ACO is
0.6 times faster than the PSO-SA for dataset 2 and 0.45 times faster than the PSO-SA for dataset 1.

Table 5.4  Performance comparisons of various PSO based hybrid approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>PSO-HC (Visalakshi and Sivanandam 2009)</th>
<th>PSO-SA (Visalakshi and Sivanandam 2009)</th>
<th>Proposed IPSO-ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Best cost</td>
<td>2322</td>
<td>2186</td>
<td>2131</td>
</tr>
<tr>
<td>Worst cost</td>
<td>2994</td>
<td>2916</td>
<td>2853</td>
</tr>
<tr>
<td>Average cost</td>
<td>2658</td>
<td>2551</td>
<td>2492</td>
</tr>
<tr>
<td>Convergence time in seconds</td>
<td>4.9636</td>
<td>6.4311</td>
<td>5.9822</td>
</tr>
</tbody>
</table>

Thus, the result infers that the proposed hybrid heuristic approach IPSO-ACO performs well, when compared with the other previously proposed hybrid methods such as PSO-HC and PSO-SA.

5.8 DYNAMIC TASK SCHEDULING WITH LOAD BALANCING

To improve the performance of the processor utilization, the load of all the processors have to be balanced. Thus, the concept of load balancing is dealt, in which the objective function is the same as represented in the Equations (2.13) to (2.16).

5.8.1 Results and Discussion

Table 5.5 illustrate the best cost, worst cost, average cost and convergence time obtained using the proposed hybrid heuristic approach
IPSO-ACO and the comparison with hybrid approaches IPSO-SA and IPSO-AIS for scheduling dynamic tasks in a multiprocessors system.

Table 5.5  Best cost, worst cost, average cost and convergence time using IPSO-SA, IPSO-AIS and the proposed hybrid heuristic approach IPSO-ACO for dynamic task scheduling with load balancing

<table>
<thead>
<tr>
<th>Method</th>
<th>IPSO-SA</th>
<th>IPSO-AIS</th>
<th>Proposed hybrid heuristic approach IPSO-ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>50</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Best Cost</td>
<td>12.9961</td>
<td>22.0223</td>
<td>13.0014</td>
</tr>
<tr>
<td></td>
<td>22.132</td>
<td></td>
<td>22.132</td>
</tr>
<tr>
<td>Worst Cost</td>
<td>11.4832</td>
<td>20.9313</td>
<td>11.4881</td>
</tr>
<tr>
<td></td>
<td>20.9474</td>
<td></td>
<td>20.9474</td>
</tr>
<tr>
<td>Average Cost</td>
<td>12.2396</td>
<td>21.4768</td>
<td>12.2448</td>
</tr>
<tr>
<td></td>
<td>21.5397</td>
<td></td>
<td>21.5397</td>
</tr>
<tr>
<td>Convergence Time</td>
<td>5.1284</td>
<td>6.9205</td>
<td>6.2154</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.3992</td>
<td>7.5695</td>
</tr>
<tr>
<td>in seconds</td>
<td></td>
<td></td>
<td>10.6314</td>
</tr>
</tbody>
</table>

For dataset 1, the best cost achieved using IPSO-SA is 12.9961, the best cost achieved using IPSO-AIS is 13.0014 and the best cost achieved by the proposed hybrid heuristic approach IPSO-ACO is 13.0582. For dataset 2, the best cost achieved by IPSO-SA is 22.0223, the best cost achieved by IPSO-AIS is 22.132 and the best cost achieved by the proposed hybrid heuristic approach is 22.1531. The convergence time for the proposed hybrid heuristic approach IPSO-ACO is higher (1.3 to 2.2s) than the hybrid approach IPSO-AIS.

The best cost obtained using the proposed hybrid heuristic approach IPSO-ACO for data set 1 and data set2 are shown in Figures 5.11 and 5.12.
Thus, from the results it is concluded that the PSO in combination with Ant Colony Optimization flicker when compared with the standard PSO, IPSO and hybrid intelligent algorithms such as IPSO-SA and IPSO-AIS.
when applied to the dynamic task scheduling with load balancing concept. However, the time taken for convergence is slightly higher when compared to the hybrid algorithm IPSO-SA and IPSO-AIS, because the fixing of ants and the tweaking of parameters in the ants schedule.

5.8.2 Performance comparison

The performance of the proposed hybrid heuristic approach IPSO-ACO is compared with the previously proposed (Visalakshi and Sivanandam 2009) hybrid PSO algorithms PSO with Hill Climbing and PSO with SA for the same datasets and for the multiprocessor dynamic task scheduling problem shown in Table 5.6.

Table 5.6 Performance comparisons of various PSO based hybrid approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>PSO-HC (Visalakshi and Sivanandam 2009)</th>
<th>PSO-SA (Visalakshi and Sivanandam 2009)</th>
<th>Proposed hybrid heuristic approach IPSO-ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>50          100</td>
<td>50          100</td>
<td>50          100</td>
</tr>
</tbody>
</table>

For the dataset 1, best cost achieved by PSO-HC is 12.008, best cost achieved by PSO-SA is 12.982 and the best cost achieved by the proposed hybrid heuristic approach IPSO-ACO is 13.0582 which are better than the compared approaches. For the dataset 2, the best cost achieved by the PSO-HC is 21.114, the best cost achieved by PSO-SA is 21.998 and the best cost achieved by the proposed hybrid heuristic approach IPSO-ACO is 22.1531.
Thus, the comparison study reveals that the proposed hybrid approach IPSO-ACO achieves better results than the other hybrid approaches that was proposed, namely, PSO-HC and PSO-SA for the same datasets. The proposed hybrid approach IPSO-ACO is faster (0.01 to 0.09 times) than the other approaches.

5.9 CONCLUSION

The chapter five has dealt with the application of the proposed hybrid heuristic approach IPSO-ACO. The proposed hybrid heuristic approach IPSO-ACO is applied to multiprocessor task scheduling of static independent tasks and dynamic task scheduling with and without load balancing. Based on the convergence to the solutions and the minimization of the objective function, the validity of the considered algorithms is enforced. The proposed approach yields better results for both static and dynamic task scheduling problem. The proposed approach IPSO-ACO simultaneously reduces both the average waiting time and total finishing time when applied to static independent task scheduling. For the dataset 5 processors with 45 tasks, IPSO-SA produces total finishing time as 66.43s, average waiting time as 30.12s, IPSO-AIS produces the total finishing time as 64.96s, average waiting time as 27.56s and the proposed hybrid heuristic approach IPSO-ACO produces total finishing time as 60.87s and average waiting time as 26.21. Based on the results obtained, it can be observed that the proposed hybrid approach IPSO-ACO provides better solutions.

The proposed hybrid approach IPSO-ACO is applied to dynamic task scheduling without load balancing. The results achieved by the proposed IPSO-ACO approach is compared with hybrid approaches proposed earlier namely PSO-HC and PSO-SA. For dataset1, the best cost achieved by PSO-HC is 2322, PSO-SA achieves the best cost as 2186 and the proposed hybrid approach IPSO-ACO achieves the best cost as 2131 which is better
than the approaches compared. For dataset 2, the best cost achieved by PSO-HC is 4621, PSO-SA achieves the best cost as 4496 and the proposed hybrid heuristic approach IPSO-ACO achieves best cost as 4226 which is better than PSO-HC and PSO-SA. The convergence time of the proposed hybrid approach IPSO-ACO is 0.6 times faster than PSO-SA for dataset 2 and 0.45 times faster than PSO-SA for dataset 1.

The proposed hybrid approach IPSO-ACO is applied to dynamic task scheduling with load balancing. The results achieved by the proposed IPSO-ACO approach is compared with the hybrid approaches proposed earlier, namely, PSO-HC and PSO-SA. For the dataset 1, the best cost achieved by PSO-HC is 12.008, the best cost achieved by PSO-SA is 12.982 and the best cost achieved by the proposed hybrid heuristic approach IPSO-ACO is 13.0582 which are better than the compared approaches. For dataset 2, the best cost achieved by PSO-HC is 21.114, best cost achieved by PSO-SA is 21.998 and the best cost achieved by the proposed hybrid heuristic approach IPSO-ACO is 22.1531.

In the proposed hybrid heuristic approach IPSO-ACO a simple pheromone guided search mechanism of ant colony is 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 hybrid approach achieved better results for static and dynamic task scheduling, except slow convergence compared with the hybrid approaches IPSO-SA and IPSO-AIS when applied to dynamic task scheduling, because of the tweaking of parameters in ants schedule. Hence other hybrid technologies need to be tried so that the convergence time is better than the methodologies tried out. Hence, new parallel algorithms are proposed in the subsequent chapters to further refine the total execution cost and convergence time achieved which are the main objectives of task scheduling. The next chapter deals with the Parallel IPSO approaches for multiprocessor task scheduling.