CHAPTER 3

PROPOSED HYBRID OPTIMIZATION APPROACH FOR MULTIPROCESSOR TASK SCHEDULING

The present chapter proposes a hybrid approach (IPSO-SA) using Improved Particle Swarm Optimization (IPSO) and Simulated Annealing (SA) to further improve the performance of the multiprocessor task scheduling.

3.1 INTRODUCTION

In the previous chapter, IPSO is applied to a multiprocessor task scheduling problem and its performance is compared with GA, and to variants of PSO based approaches. To avoid the local trapping of PSO and to further improve the performance of the multiprocessor task scheduling in terms of minimization of total execution time a hybrid approach, IPSO-SA is proposed.

Modern meta-heuristics survive with exploration and exploitation search. The exploration search seeks new regions and once it finds a good region, the exploitation search comes in. However, since the two strategies are usually inter-wound, the search may be conducted to other regions before it reaches the local optima.

Currently, the research trend is to merge or combine the PSO with other heuristic computing techniques. Hence, here, IPSO is hybridized with metaheuristic algorithm SA. The details of the results of simulation illustrate
the influence of the proposed hybrid approach, IPSO-SA. A comparative study made between the proposed hybrid approach IPSO-SA with the previously proposed PSO based hybrid approaches are presented.

3.2 REVIEW OF LITERATURE ON HYBRID PARTICLE SWARM OPTIMIZATION

Lovbjerg et al (2001) presented two hybrid Particle Swarm Optimizers combining the idea of the particle swarm from Evolutionary Algorithms. The hybrid PSOs combine the traditional velocity and position update rules with the ideas of breeding and sub-populations. Both the hybrid models are tested and compared with the standard PSO and standard GA models. This is done to illustrate that the PSOs with the breeding strategies have the potential to achieve faster convergence and the potential to find a better solution.

Belal and Khalifa (2002) introduced a comparative study between swarm intelligence and Genetic Algorithms. The comparison is based on the convergence speed of each algorithm, and the variation of the parameters for each algorithm is also studied. The results of the simulation prove that swarm intelligence is faster than Genetic Algorithm in reaching the global optimum.

Zhang and Xie (2003) proposed a hybrid particle swarm with differential evolution operator, termed DEPSO, which provide the bell-shaped mutations with consensus on the populations diversity along with the evolution, while keeping the self-organized particle swarm dynamics. It is then applied to a set of benchmark functions and the experimental results illustrate its efficiency.

Naka et al (2003) proposed a hybrid Particle Swarm Optimization for a practical distribution state estimation. The proposed method considers
non-linear characteristics of the practical equipment and actual limited measurements in the distribution systems. The method can estimate load and distributed generation output values at each node by minimizing the difference between the measured and calculated voltages and currents. The feasibility of the proposed method is demonstrated and compared with an original Particle Swarm Optimization-based method on practical distribution system models. Effectiveness of the constriction factor approach of Particle Swarm Optimization is also investigated. The results indicate the applicability of the proposed state estimation method to the practical distribution systems.

Juang (2004) proposed a new evolutionary learning algorithm based on hybrid of Genetic Algorithm and Particle Swarm Optimization, called HGAPSO. In HGPSO, individuals in a new generation are created not only by crossover and mutation operations as in GA, but also by PSO. HGAPSO introduced the concept of maturing phenomenon in nature into the evolution of individuals originally modelled by GA. The maturing phenomenon is mimicked by PSO, where individuals enhance themselves based on social interaction and their private cognition. From the perspective of PSO, crossover operation is introduced into HGAPSO. The concept of elite strategy is adopted in HGAPSO, where the upper-half of the best performing individuals in population are regarded as elites. The performance of HGAPSO is compared to both GA and PSO, in these recurrent networks design problems, demonstrating its superiority.

Zhang and Wu (2012) proposed a hybrid Restarted Simulated Annealing Particle Swarm Optimization (RSAPSO) technique to find the global minima more efficiently and robustly. The proposed RSAPSO combines the global search ability of PSO and the local search ability of RSA, and offsets the weaknesses of each other. The four benchmark functions demonstrate the superiority of the algorithm.
Ge et al (2011) proposed a hybrid algorithm of Particle Swarm Optimisation (PSO) for solving grain logistics Vehicle Routing Problem (VRP), which integrates the PSO with Simulated Annealing (SA) algorithm by introducing SA conception. Simulation results indicate that the hybrid algorithm of PSO can find the optimal solution of grain logistics VRP with time window quickly. The cost of the grain distribution is therefore reduced.

Jiao and Yan (2011) proposed a novel intelligent algorithm (SACQPSO) mixed with Stimulated Annealing, Cooperative co-evolution thought, Quantum-behaved theory and Particle Swarm Optimization algorithm. The proposed algorithm enhances the capacity of searching the best solution and increases the diversity of particles owing to co-operative co-evolution thought and quantum-behaved theory. It also strengthens the ability of global searching as a result of simulated annealing. Large JSSPs are solved using the proposed algorithm.

Sun et al (2011) proposed an IPSO algorithm based on feasibility rules to solve constrained optimization problems. The average velocity of the swarm and the best history position in the particle's neighbourhood are introduced as two turbulence factors, which are considered to influence the fly directions of particles, into the algorithm so as not to converge prematurely. The performance of IPSO algorithm is tested on 13 well-known benchmark functions. The experimental results show that the proposed IPSO algorithm is simple, effective and highly competitive.

Ahmed et al (2005) presented an application of Hybrid Particle Swarm Optimization to loss power minimization, where the approach utilizes the local and global capabilities to search for optimal loss reduction. Notion of mutation is introduced from the field of Genetic Algorithm. The optima found by hybrid method are better than the standard PSO and the convergence speed is faster.
Montazeri et al (2006) proposed a hybrid algorithm to solve the multiprocessor task scheduling problem. The local search is done using the memetic algorithm and the global search is done using the Genetic Algorithm. The hybrid version performs better than the normal GA.

Yin et al (2006) proposed a Hybrid Particle Swarm Optimization (HPSO) algorithm which intends to minimize the cost and maximize the reliability simultaneously for executing programs in a distributed computing system. The HPSO initializes a swarm of particles, each of which corresponds to a candidate solution to the underlying problem. These particles iteratively improves their quality through collective experiences of personal cognition and social interactions. This is a positive feedback process such that the intelligence of the entire swarm is enriched. Penalty functions tailored to the system constraints are devised in order to deal with infeasible solutions. The HPSO embeds a local search heuristic into the evolutionary iterations for expediting the convergence. The performance of the proposed method is compared to a Genetic Algorithm and an exact algorithm. The experimental results manifest that the HPSO reports quality solutions on a large set of simulated instances involving different problem scales, task interaction densities and network topologies. The information gain and the worst-case analysis of the HPSO are theoretically and empirically conducted.

Attiya and Hamam (2006) presents a heuristic algorithm derived from SA to solve the task allocation in heterogeneous distributed systems with the goal of maximizing the system reliability. The performance of the proposed algorithm is evaluated through a large number of randomly generated instances and is compared with the Branch and Bound technique.

Hamam and Hindi (2000) presented Simulated Annealing approach to assign the modules of the program to processors in a distributed computer
system. He considered the resource requirements, and communication resources needed to schedule the programs. The computational results are reported.

Sivanandam et al (2009) proposed a hybrid method PSO-SA, which performs better with the constraint of cost reduction in distributed heterogeneous computing systems of static allocation of tasks in a multiprocessor system. Also, he implemented PSO with dynamically reducing inertia, which yields better results when compared with the fixed inertia.

Yazdani et al (2010) proposed a parallel variable neighbourhood search (PVNS) algorithm to solve the FJSP to minimize makespan time. The author uses various neighbourhood structures which carry the responsibility of making changes in assignment and sequencing of operations for generating neighbouring solutions. The results show that the proposed algorithm is a viable and effective approach for the FJSP.

Choong et al (2011) proposed two hybrid heuristic algorithms based on PSO with SA and TS. The algorithms were applied to flow shop scheduling problems. Experimental results reveal that the proposed algorithms effectively produce improved solutions over the conventional methods.

Jamili et al (2011) proposed a hybrid algorithm, namely EM-SA (Electromagnetism-like Mechanism and Simulated Annealing). The author evaluated the algorithm with some randomly constructed instances and compared with SA and Branch and Bound. The results infer that the proposed algorithm performs better.

Zhang and Wu (2011) discussed the mathematical programming model and its duality. When the processing order for each machine is fixed, the block based neighbourhood structure is designed. After this, simulated annealing algorithm is designed. The results show that the new neighborhood
promotes the searching considerably, and helps it to converge high quality solutions.

3.3 **PROPOSED HYBRID APPROACH (IPSO-SA)**

Hybrid Particle Swarm Optimization using IPSO and SA is proposed to solve the multiprocessor task scheduling problem. Simulated Annealing is a kind of global optimization technique based on annealing of metal. SA has a strong ability to find the local optimistic result, thus avoiding the problem being stuck at local optimum. Hence, Simulated Annealing has a strong ability to find the local optimistic result, thus avoiding the problem being stuck at local optimum. It can find the optimum value using stochastic search technology, from the means of probability. However, the speed of approximation remains the main shortcoming of SA.

To overcome the drawback of SA and PSO, and produce more accurate results, proposed IPSO is hybridized with Simulated Annealing, which leads to the combined effect of the good global search and local search algorithm.

3.3.1 **Simulated Annealing Algorithm for Scheduling**

The algorithm starts by randomly selecting an initial solution \( cs \) and computes the energy/cost \( E_{cs} \) at the current solution \( cs \). After setting an initial temperature \( T \), a neighbor finding strategy is invoked to generate a neighbor solution \( ns \) to the current solution \( cs \) and compute the corresponding Energy/cost \( E_{ns} \). If \( E_{ns} \) is lower than the current energy \( E_{cs} \), then the neighbor solution is accepted as a current solution. Otherwise a probability function \( \text{exe}(-\Delta/T) \) is evaluated to determine whether the neighbor solution may be accepted as a current solution, where \( \Delta = E_{ns} - E_{cs} \). After equilibrium is reached at the current temperature \( T \), the value of \( T \) is decreased by a cooling factor \( q \) and the number of inner
repetitions is increased by an increasing factor $\beta$. The algorithm continues from the current solution point searching for a thermal equilibrium at the new temperature level. The process terminates when either the lowest energy point is found or no upward/downward jumps have been taken for a number of successive thermal equilibrium. The algorithm is given as (Orsila et al 2008, Attiya et al 2006)

Randomly select an initial solution $cs$;
Compute the cost at this solution $E_{cs}$;
Select an initial temperature T;
Select a cooling factor $\alpha < 1$;
Select an initial chain $n_{rep}$;
Select a chain increasing factor $\beta > 1$;
Repeat
  Repeat
    Select a neighbor solution $ns$ to $cs$;
    Compute cost at $ns$, $E_{ns}$;
    $\Delta = E_{ns} - E_{cs}$;
    If $\Delta < 0$,
    $cs = ns$; $E_{cs} = E_{ns}$;
    Else
      Generate a random value x in the range (0,1);
      If $x < \text{exp} \left( - \frac{\Delta}{T} \right)$,
      $cs = ns$; $E_{cs} = E_{ns}$;
    End
  End
Until iteration= $n_{rep}$ (equilibrium state at T)
Set $T = \alpha * T$;
Set $n_{rep} = \beta * n_{rep}$;
Until stopping condition = true

\( E_{cs} \) is the cost and \( cs \) is the solution.

The initial temperature is set after executing a sufficiently large number of moves, such that a worst move would be allowed.

\( T \) be the initial temperature, \( c_r \) and \( c_i \) be numbers corresponding to cost reduction and cost increase respectively and \( c_a \) be the average cost increase value of \( c_i \) trials and \( a_o \) be the desired initial acceptance value. The following relation is then represented as,

\[
a_o = \frac{c_r + c_i e^{-\frac{c_i}{T}}}{c_r + c_i}
\]

(3.1)

The initial temperature is represented as,

\[
T = -\frac{c_a}{\log\left(\frac{c_a}{c_i (a_o - 1) + a_o}\right)}
\]

(3.2)

Thus, the simulated Annealing starts from a very high energy state and the energy is then reduced step by step until the minimum energy specified is reached. Due to this local search, the efficiency of the IPSO combined with Simulated Annealing can lead to a better result than the standard PSO.

Figure 3.1 depicts the working of the Proposed Hybrid Algorithm IPSO-SA.
A

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

Initialize temperature T

Compute the objective function

Invoke Hybrid algorithm

For each generation

For each particle

If E < best ‘E’ (P best) so far

Search is terminated optimal solution reached

Yes

No

Current value = new p best

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

Calculate particle velocity using (2.1)

Calculate particle position using (2.2)

Start

Figure 3.1 Flowchart for the proposed hybrid approach IPSO-SA
Update memory of each particle

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

Yes

Find a new particle using temperature

Accept a new particle as best with the probability as $\exp(-\frac{1}{T})$

Reduce the temperature $T$

End

End

Return by using Hybrid algorithm

Stop

Figure 3.1 (Continued)
3.4 SIMULATION PROCEDURE

The present section presents the details of the simulation carried out for the following datasets.

Benchmark datasets are taken from Eric Tailard’s site for dynamic task scheduling. Two datasets are taken for simulation. Dataset 1 involves 50 tasks and 20 processors. Dataset 2 involves 100 tasks with 20 processors. The data for the static scheduling are 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 is run with 30 independent trials with different values of random seeds and control parameters.

The optimal result is obtained for following parameter settings.

**Simulated Annealing**

- The initial solution is generated randomly
- Cooling factor \( \alpha \) = 0.9
- Increasing factor \( \beta \) = 1.05
- The number of inner loop repetitions = four times the number of tasks

**Proposed IPSO**

- The initial solution is generated randomly
- \( C_{lg}, C_{lb} \) and \( C_2 = 2, 2 \) and 2
• Swarm size = Twice the number of tasks (Salman et al 2002)
• \( W_{\text{min}} - W_{\text{max}} = 0.5 \)
• Iteration = 500

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

3.5  STATIC SCHEDULING

In this illustration, the tasks considered are independent and processors are homogeneous in nature. Hence any task can be executed in any processor and in any order. The objective function is the same as specified in the Equations (2.4) to (2.9).

3.5.1  Results and Discussion

The datasets defined in the simulation procedure have been tested with the proposed hybrid approach IPSO-SA. The obtained results are tabulated and shown in Table 3.1.

Table 3.1  Total finishing time and average waiting time using proposed hybrid approach IPSO-SA

<table>
<thead>
<tr>
<th>No of Processors</th>
<th>No of jobs</th>
<th>Proposed IPSO -SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average Waiting Time (AWT)</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>25.61</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>40.91</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>38.45</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>26.51</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>28.34</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>30.12</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>32.76</td>
</tr>
</tbody>
</table>
The proposed hybrid approach IPSO-SA produces total finishing time 65.40s and average waiting time 38.45s for 3 processors and 40 tasks. For 4 processors with 30 tasks, the total finishing time is 66.29s and average waiting time is 26.51s. For 5 processors with 60 tasks, the total finishing time is 69.13s and average waiting time is 32.76s.

3.5.2 Performance Comparison

In order to validate the performance of the proposed hybrid algorithm IPSO-SA, comparisons have been made with the approaches of standard PSO, IPSO for the same datasets and are reported in Table 3.2. The results are comparatively better for IPSO-SA than for the other approaches.

Table 3.2 Comparison of job total finishing time and average waiting time using PSO, IPSO and IPSO-SA

<table>
<thead>
<tr>
<th>No of Processors</th>
<th>No of jobs</th>
<th>PSO AWT</th>
<th>PSO TFT</th>
<th>IPSO AWT</th>
<th>IPSO TFT</th>
<th>IPSO-SA AWT</th>
<th>IPSO-SA TFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>30.10</td>
<td>60.52</td>
<td>29.12</td>
<td>57.34</td>
<td>25.61</td>
<td>54.23</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>45.92</td>
<td>56.49</td>
<td>45.00</td>
<td>54.01</td>
<td>40.91</td>
<td>50.62</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>42.09</td>
<td>70.01</td>
<td>41.03</td>
<td>69.04</td>
<td>38.45</td>
<td>65.40</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>30.65</td>
<td>72.18</td>
<td>29.74</td>
<td>70.97</td>
<td>26.51</td>
<td>66.29</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>32.79</td>
<td>74.24</td>
<td>30.06</td>
<td>70.62</td>
<td>28.34</td>
<td>68.01</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>34.91</td>
<td>70.09</td>
<td>33.65</td>
<td>68.04</td>
<td>30.12</td>
<td>66.43</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>39.61</td>
<td>75.42</td>
<td>36.56</td>
<td>72.31</td>
<td>32.76</td>
<td>69.13</td>
</tr>
</tbody>
</table>

For the dataset 3 processors with 40 tasks, PSO produces average waiting time 42.09s and Total finishing time is 70.01s, IPSO produces 41.03s as average waiting time and 69.04s as total finishing time and the proposed hybrid approach IPSO-SA produces a average waiting time of 38.45s and total finishing time of 65.40s. For the dataset 5 processors with 45 tasks, PSO produces average waiting time as 34.91s and total finishing time as 70.09s, IPSO produces 33.65s as average waiting time and 68.04s as total finishing time and the proposed hybrid approach IPSO-SA produces average waiting time as 30.12s and total finishing time as 66.43s.
The proposed hybrid approach IPSO-SA performs better and minimizes both Total finishing time and the waiting time simultaneously for small as well as large data sets. It is observed from Figures 3.2 to 3.8, that the proposed hybrid approach IPSO reduces the Total finishing time and average waiting time, compared with other algorithms such as standard PSO and IPSO.

**Figure 3.2**  Total finishing time and average waiting time for 2 processors with 20 jobs using PSO, IPSO and IPSO-SA

**Figure 3.3**  Total finishing time and average waiting time for 3 processors with 20 jobs using PSO, IPSO and IPSO-SA
Figure 3.4  Total finishing time and average waiting time for 3 processors with 40 jobs using PSO, IPSO and IPSO-SA

Figure 3.5  Total finishing time and average waiting time for 4 processors with 30 jobs using PSO, IPSO and IPSO-SA
Figure 3.6  Total finishing time and average waiting time for 4 processors with 50 jobs using PSO, IPSO and IPSO-SA

Figure 3.7  Total finishing time and average waiting time for 5 processors with 45 jobs using PSO, IPSO and IPSO-SA
Figure 3.8 Total finishing time and average waiting time for 5 processors with 60 jobs using PSO, IPSO and IPSO-SA

At the outset, the result reveals that the IPSO-SA performs better when compared to GA, standard PSO and the proposed IPSO.

3.6 DYNAMIC TASK SCHEDULING WITHOUT LOAD BALANCING

The main intention of dynamic task scheduling is to minimize the makespan of the schedule. To minimize the makespan, the objective function is the same as represented in Equations (2.10) to (2.12).

3.6.1 Results and Discussion

The results obtained using the proposed hybrid approach IPSO-SA is compared with standard PSO and IPSO are depicted in Table 3.3.
Table 3.3  Best cost, worst cost, average cost and convergence time using PSO, IPSO and IPSO-SA for dynamic task scheduling without load balancing

<table>
<thead>
<tr>
<th>Method</th>
<th>PSO</th>
<th>IPSO</th>
<th>Proposed IPSO-SA</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>2972</td>
<td>5552</td>
<td>2374</td>
</tr>
<tr>
<td>Worst Cost</td>
<td>3724</td>
<td>6018</td>
<td>3136</td>
</tr>
<tr>
<td>Average Cost</td>
<td>3187.5</td>
<td>5839.9</td>
<td>2755</td>
</tr>
<tr>
<td>Convergence Time in seconds</td>
<td>3.9774</td>
<td>5.7324</td>
<td>4.0521</td>
</tr>
</tbody>
</table>

The best, average and the worst cost obtained for dataset 1 and 2 using the proposed hybrid approach IPSO-SA is compared with the standard PSO and IPSO. For dataset 1, PSO produces the best cost as 2972, IPSO produces the best cost as 2374 and the proposed hybrid approach IPSO-SA produces the best cost as 2156 which is the best cost compared with all other approaches. For dataset 2, standard PSO produces the best cost as 5552, IPSO produces the best cost as 4527 and the proposed hybrid approach IPSO-SA produces the best cost as 4376. The time taken for convergence using the proposed hybrid approach is 4.2156s for dataset 1 and 5.8428s for dataset 2. The best and average cost obtained is better in the case of the proposed algorithm when compared to the other methods. The convergence time for the proposed hybrid algorithm IPSO-SA is slightly higher (0.15 times) than with IPSO, because of the extra calculation involved in the annealing schedule. The results infer that the IPSO-SA performs better than the other algorithms.

The best cost obtained using the proposed hybrid method IPSO-SA for data set 1 and 2 are compared with the standard PSO and IPSO and are shown in Figures 3.9 and 3.10.
3.6.2 Performance Comparison

The performance of the proposed IPSO is compared with the previously proposed (Visalakshi and Sivanandam 2009) hybrid PSO
algorithms, namely, PSO with Hill Climbing (PSO-HC) and PSO with Simulated Annealing (PSO-SA) for the same datasets and for multiprocessor dynamic task scheduling problem.

Table 3.4 Performance comparison 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-SA</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>2322</td>
<td>4621</td>
<td>2186</td>
</tr>
<tr>
<td>Worst cost</td>
<td>2994</td>
<td>5449</td>
<td>2916</td>
</tr>
<tr>
<td>Average cost</td>
<td>2658</td>
<td>5035</td>
<td>2551</td>
</tr>
<tr>
<td>Convergence time in seconds</td>
<td>4.9636</td>
<td>7.3588</td>
<td>6.4311</td>
</tr>
</tbody>
</table>

The performance of the proposed hybrid approach IPSO-SA is compared with the previously proposed hybrid methods such as PSO-HC and PSO-SA.

For dataset 1 PSO-HC produces the best cost as 2322, PSO-SA produces the best cost as 2186 and the proposed hybrid approach IPSO-SA produces the best cost as 2156. For dataset 2, PSO-HC produces the best cost as 4621, PSO-SA produces the best cost as 4496 and the proposed hybrid approach IPSO-SA produces the best cost as 4376. The convergence time for PSO-HC is 4.9636s for dataset 1 and 7.3588s for dataset 2. PSO-SA has taken 6.4311s for dataset 1 and 8.7349s for dataset 2. The proposed hybrid approach IPSO-SA has taken 4.2156s for dataset1 and 5.8428s for dataset 2.

Thus, the comparison reveals that the proposed hybrid approach IPSO-SA achieves better results faster than (2.2 to 2.9 times) the other hybrid approaches namely PSO-HC and PSO-SA.
3.7 DYNAMIC TASK SCHEDULING WITH LOAD BALANCING

An effective processor utilization is needed to support the concept of load balancing. 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). Table 3.5 illustrates the best cost, worst cost, average cost and convergence time for PSO, IPSO and the proposed hybrid approach IPSO-SA.

3.7.1 Results and Discussion

The best cost, average cost and worst cost values obtained using the proposed hybrid approach IPSO-SA are shown in Table 3.5.

Table 3.5  Best cost, worst cost, average cost and convergence time using PSO, IPSO and IPSO-SA for dynamic task scheduling with load balancing

<table>
<thead>
<tr>
<th>Method</th>
<th>PSO</th>
<th>IPSO</th>
<th>IPSO-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>19.6424</td>
<td>12.0042</td>
<td>21.4291</td>
</tr>
<tr>
<td>Best Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worst Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0603</td>
<td>6.8852</td>
<td>5.1176</td>
<td>6.9064</td>
</tr>
<tr>
<td>Convergence Time</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The best cost achieved using PSO for dataset 1 is 10.8634, IPSO produces the best cost as 12.0042 and the proposed hybrid algorithm IPSO-SA produces the best cost as 12.9961. For dataset 2, PSO produces the best cost as 19.6424, IPSO produces the best cost as 21.4291 and the proposed hybrid approach IPSO-SA produces the best cost as 22.0223. The average cost obtained is also improved in the proposed hybrid algorithm IPSO-SA. The convergence time for the proposed IPSO-SA method is
slightly higher (0.01s) than the proposed IPSO. The results reveal that the proposed hybrid approach IPSO-SA performs better than the IPSO.

The best cost obtained using the proposed hybrid method IPSO-SA for data set 1 and data set 2 are shown in Figures 3.11 and 3.12.

![Figure 3.11 Best cost for 50 tasks and 20 processors using PSO, IPSO and IPSO-SA](image)

![Figure 3.12 Best cost for 100 tasks and 20 processors using PSO, IPSO and IPSO-SA](image)
At the outset, the result reveals that the proposed hybrid approach IPSO-SA produces better results when compared to the standard PSO and IPSO approach to the dynamic task scheduling with load balancing concept.

### 3.7.2 Performance Comparison

The performance of the proposed hybrid approach IPSO-SA 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.

#### Table 3.6 Performance comparison 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-SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>50</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

For the dataset 1, PSO-HC produces 12.008 as best cost, PSO-SA produces 12.982 as best cost and the proposed hybrid approach IPSO-SA produces the best cost as 12.9961. For dataset 2, PSO-HC produces the best cost as 21.114, PSO-SA produces the best cost as 21.998 and the proposed hybrid approach produces the best cost as 22.0223. The proposed hybrid approach IPSO-SA performs better when compared with the other previously proposed hybrid methods PSO-HC and PSO-SA.
Thus, the comparison infers that the proposed hybrid approach IPSO-SA performs better than the hybrid approach PSO-HA and PSO-SA.

### 3.8 CONCLUSION

In chapter three, a hybrid approach IPSO-SA is presented for different types of multiprocessor task scheduling, namely, static task scheduling and dynamic scheduling with and without load balancing.

The proposed hybrid approach produces better results for both static and dynamic task scheduling problem in a multiprocessor system. In static task scheduling, the proposed hybrid approach IPSO-SA simultaneously reduces the total finishing time and the average waiting time. For the dataset 3 processors with 40 tasks, PSO produces the average waiting time as 42.09s and the total finishing time as 70.0s, IPSO produces the total finishing time as 69.04s and the average waiting time as 41.03s, and the proposed hybrid approach IPSO-SA produces the average waiting time as 38.45s and the total finishing time as 65.40s.

The proposed hybrid approach IPSO-SA is tested for dynamic task scheduling without load balancing problem and the results obtained are compared with the standard PSO and IPSO. For dataset 1, PSO produces the best cost as 2972, IPSO produces the best cost as 2374 and the proposed hybrid approach IPSO-SA produces the best cost as 2156. For dynamic task scheduling with load balancing, the best cost achieved using PSO for dataset 1 is 10.8634, IPSO produces as 12.0042 and the proposed hybrid algorithm IPSO-SA produces the best cost as 12.9961.

The results reveal that the proposed IPSO-SA technique has an improved performance when compared with the other hybrid methodologies such as PSO-HC and PSO-SA. The drawback of the proposed hybrid
approach IPSO-SA for dynamic task scheduling is slow convergence when compared with IPSO, because of the annealing schedule in Simulated Annealing algorithm. Hence, other hybrid technologies need to be tried to get a convergence time that is better than the methodologies tried out. Hence, new hybrid algorithms are proposed in the subsequent chapters to further refine the cost and convergence time achieved which is the main objective in task scheduling. The next chapter deals with the hybrid algorithm using Improved Particle Swarm Optimization with Artificial Immune System.