Int. J. Open Problems Compt. Math., Vol. 2, No. 3, September 2009
ISSN 1998-6262; Copyright ©ICSRS Publication, 2009
www.i-csrs.org

Dynamic Task Scheduling with Load Balancing using Hybrid Particle Swarm Optimization

P Visalakshi[1] , S N Sivanandam[2]

Senior Lecturer, Dept. of CSE, PSG College of Technology, Coimbatore, India
Professor& Head, Dept. of CSE, PSG College of Technology, Coimbatore, India
e-mail : visapsg@gmail.com , sns@mail.psgtech.ac.in

Abstract

This paper presents a Hybrid Particle Swarm Optimization (HPSO) method for solving the Task Assignment Problem (TAP) which is an np-hard problem. Particle Swarm Optimization (PSO) is a recently developed population based heuristic optimization technique. The algorithm has been developed to dynamically schedule heterogeneous tasks on to heterogeneous processors in a distributed setup. Load balancing which is a major issue in task scheduling is also considered. The nature of the tasks are independent and non pre-emptive. The HPSO yields a better result than the Normal PSO when applied to the task assignment problem. The results Of PSO and HPSO is also compared with another popular heuristic optimization technique namely Genetic Algorithm (GA). The results infer that the PSO performs better than the GA.

Keywords: *GA, HPSO, Inertia, PSO, TAP, Topology.*

1 Introduction

Multiprocessor Scheduling is an np-hard problem. The problem of scheduling a set of dependent or independent tasks in a distributed computing system is a well-studied area. In this paper, the dynamic task allocation methodology is examined in a heterogeneous computing environment. Dynamic allocation techniques can be applied to large sets of real-world applications that are able to be formulated in a manner which allows for deterministic execution. Some advantages of the dynamic techniques over static techniques are that, static techniques should always have a prior knowledge of all the tasks to be executed but dynamic techniques do not require that. The traditional methods such as branch and bound, divide and conquer, and dynamic programming gives the global optimum, but is often time consuming or do not apply for solving typical real-world problems. The researchers [1] have derived optimal task assignments to minimize the sum of task execution and communication costs with the branch-and-bound method and evaluated the computational complexity of this method using simulation techniques. Traditional methods used in optimization are deterministic, fast, and give exact answers but often tends to get stuck on local optima [2]. The dynamic task scheduling considering the load balancing issue is an np-hard problem [3] [4].

Consequently, another approach is needed when the traditional methods cannot be applied. Modern heuristics [5] are general purpose optimization algorithms. Their efficiency or applicability is not tied to any specific problem-domain. Multiprocessor scheduling methods can be divided into list heuristics and Meta heuristics. In List heuristics, the tasks are maintained in a priority queue in decreasing order of priority. The tasks are assigned to the free processors in a First in First Out manner. A meta-heuristic is a heuristic method for solving a very general class of computational problems by combining user-given black-box procedures, usually heuristics themselves, in a hopefully efficient way.

Available meta heuristics include Simulated Annealing algorithms, Genetic Algorithms [6] [7] [8] [9][10] [11], Hill Climbing, Tabu Search, Neural Networks [12], Particle Swarm Optimization [13][14] and Ant Colony algorithm [15]. PSO yields faster convergence when compared to GA, because of the balance between exploration and exploitation in the search space. In this paper, a very fast and easily implemented dynamic algorithm is presented based on particle swarm optimization (PSO) and its variants. In this paper, a scheduling strategy is presented which uses PSO to schedule heterogeneous tasks on to heterogeneous processors to minimize the total execution time. It operates dynamically, which allows the new tasks to arrive at any time interval. The remaining of the paper is organized as follows, Section 2 deals with the problem definition and Section 3 illustrates the working of Particle Swarm Optimization Algorithm. The proposed methodologies are explained in Section 4. Results and discussion is depicted in Section 5. Conclusion and future work is dealt in Section 6.Open Problem is highlighted in Section 7.

2 Problem Description

This paper considers the Task Assignment Problem (TAP) with the following scenario. The system consists of a set of heterogeneous processors (m) having different memory and processing resources, which implies that tasks (r), executed on different processor encounters different execution cost. The communication links are assumed to be identical, however communication cost between two tasks will be encountered when executed on different processors. A task will make use of the resources from its execution processor.

Load Balancing algorithms [16] are designed essentially to equally spread the load on processors and maximize their utilization while minimizing the total task execution time. In order to achieve these goals, the load-balancing mechanism should be fair enough in distributing the load across the processors.

The objective is to minimize the total execution and communication cost encountered by the task assignment, subject to the resource constraints. A particle is evaluated by calculating its fitness function. Fitness function indicates the goodness of the schedule. The objective function calculates the total execution time of the set of tasks allocated to each processor. The fitness function calculates the average of the total execution time of the set of tasks allocated to the processors.

Definition 2.1

The fitness function is given as,

$$fitness(P_i) = (1/\,max\,span)\ x\ average\ utilization \tag{1}$$

where $fitness(P_i)$ is the fitness function of processor P_i. The fitness function is used to evaluate the quality of the task assignment. Effective processor utilization is needed to support the concept of load balancing. If all the processors are used to their maximum, then the loads, which are the measures of idleness of processors are effectively reduced.

Definition 2.2

The average utilization is calculated based on the individual performance of the processor. The utilization of the individual processor is given by,

$$utilization(P_x) = completion\_time(P_x)/\,max\,span \tag{2}$$

Then dividing the sum of all processors utilization by the total number of processors gives the average processor utilization. When the average processor utilization is optimized, then the processors being idle for long time are avoided.

Definition 2.3

The objective function is represented as shown in equation 3. The objective function calculates the average of the total execution time of the set of tasks allocated to the processors.

$$\text{Objective function} = max\{ \frac{\sum_{i=1}^{m} fitness\ (P_i)}{m} \} \tag{3}$$

where m is the number of processors.

The objective is the maximization of the equation (3) mentioned. The value clearly indicates the optimum schedule along with the balance in the processor utilization.

3 Particle Swarm Optimization

PSO is a stochastic optimization technique [17] which operates on the principle of social behavior like bird flocking or fish schooling. In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Rui Mendes [18] discusses the complete information about the Particle Swarm Optimization. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself i.e. its own experience and the position of the best particle in its neighborhood i.e. the experience of neighboring particles [19]. When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as the *gbest* PSO [20]. When smaller neighborhoods are used, the algorithm is generally referred to as the *lbest* PSO. The performance of each particle is measured using a fitness function that varies depending on the optimization problem [21].

Each particle in the swarm is represented by the following characteristics: x_{id}: The *current position* of the particle; v_{id}: The *current velocity* of the particle; p_{id}: The *personal best position* of the particle. The personal best position of particle *i* is the best position visited by particle *i* so far. There are two versions for keeping the neighbors' best vector, namely lbest and gbest [22]. In the local version, each particle keeps track of the best vector lbest attained by its local topological neighborhood of particles. For the global version, the best vector gbest is determined by all particles in the entire swarm. Hence, the gbest model is a special case of the lbest model. The equations (4) and (5) are used for the velocity updation and the position updation.

$$v_{id} = w\, v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \tag{4}$$

$$x_{id} = x_{id} + v_{id} \tag{5}$$

where c_1 and c_2 are the cognitive coefficients and rand() and Rand() are random real numbers drawn from U (0, 1). Several topologies exist in literature for the particles to communicate with one another. The topologies are ring, star, pyramid and master-slave topologies [23]. The star topology is adopted where each particle communicates with every other particle in the population to achieve the optimal solution. Among the topologies, the star is the best topology .The inertia w is used to achieve a balance in the exploration and exploitation of the search space. The inertia dynamically reduces during a run which facilitates a balance in the exploration and exploitation of the search space.

4 Proposed Methodology

This section discusses the proposed dynamic task scheduling using Particle Swarm Optimization. Table 1 shows an illustrative example where each row represents the particles which correspond to a task assignment that assigns five tasks to three processors, and [Particle 3, T4] =P1 means that, in particle 3, the task 4 is assigned to Processor 1.

Table 1 *A PSO Particle Representation*

	T1	T2	T3	T4	T5
particle 1	P3	P2	P1	P2	P2
particle 2	P1	P2	P3	P1	P1
particle 3	P1	P3	P2	P1	P2
particle 4	P2	P1	P2	P3	P1
particle 5	P2	P2	P1	P3	P1

The Algorithm used is for the dynamic task scheduling is as follows: The particles are generated based on the number of processors used, number of tasks that have arrived at a particular point of time and the population size specified. Initially the particles are generated at random and the fitness is calculated which decides the goodness of the schedule. The pbest and gbest values are calculated. Then the velocity updation and the position updation are done. The same procedure is repeated for the maximum number of iterations specified. The global solution which is the optimal solution is obtained. When a new task arrives, it is compared with the tasks that are in the waiting queue and a new schedule is obtained. Thus the sequence keeps on changing with time based on the arrival of new tasks.

Each particle corresponds to a candidate solution of the underlying problem. Thus, each particle represents a decision for task assignment using a vector of r elements, and each element is an integer value between 1 to n. The algorithm terminates when the maximum number of iterations is reached. The near optimal solution is thus obtained using Particle Swarm Optimization.

4.1 Dynamic Task Scheduling using Genetic Algorithm

Holland proposed Genetic Algorithm (GA) as a heuristic method based on "Survival of the fittest". GA was discovered as a useful tool for search and optimization problem. GA handles a population of possible solutions. Each solution is called a chromosome. The selection of chromosomes is done by evaluating the fitness function. The procedure for dynamic task scheduling using Genetic Algorithm is as follows,

1. Generate an initial population of chromosomes randomly.
2. Evaluate the fitness of each chromosome in the population
3. Create a new population by repeating the following steps until the new population is complete,

Selection

Select two parent chromosomes from a population according to their fitness. (The better the fitness, the higher is the chance for getting selected).

Crossover

With a crossover probability, do cross over operations on the parents to form a new offspring. If no crossover is performed, offspring is the exact copy of the parents.

Mutation

With a mutation probability, mutate new offspring at each locus (position in chromosome)

Acceptance

Place the new offspring in the new population.

4. Using the newly generated population for a further sum of the algorithm.
5. If the test condition is satisfied, stop and return the best solution in the current population.
6. Repeat Step 3 until the target is met.
7. Finally obtain the optimal solution.

The dynamic task scheduling using Genetic Algorithm is done to compare its performance with the PSO and its variants.

4.2 Scheduling using PSO with fixed (PSO-fi) and variable inertia (PSO-vi)

The inertia weight, w controls the momentum of the particle in equation (4). The inertia used is the dynamically varying inertia which decreases gradually during the long run. The equation used for implementing varying inertia is,

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{6}$$

where w_{min} = 0.4 and w_{max} = 0.9, $iter_{max}$ is the maximum number of iteration specified and *iter* is the current iteration in progress. The inertia weight is introduced in equation (4) to balance between the global and local search abilities. The large inertia weight facilitates global search while the small inertia weight facilitates local search. The introduction of the inertia weight also eliminates the requirement of carefully setting the maximum velocity. Two versions of PSO algorithm were carried out namely PSO with fixed inertia and PSO with varying inertia. In PSO with fixed inertia, the inertia value is fixed to a constant value of 0.8 during the whole run of the algorithm. Another version was the PSO with dynamically reducing inertia in which the inertia decreases from a maximum value of 0.9 to a minimum value of 0.4 [17] during the whole run of the algorithm. A significant cost reduction is achieved in variable inertia than with the fixed inertia.

4.3 Scheduling using PSO with Elitism (EPSO)

Elitism is the process of selecting the better individuals, or more to the point, selecting individual with a bias towards the better ones. Elitism is important since it allows the solutions to get better over time. The best particle in PSO is copied to the population in the next generation. The rest are chosen in the classical way. Elitism can very rapidly increase performance of PSO, because it prevents losing the best found solution to date. A variation is to eliminate an equal number of the worst solutions, i.e. for each "best particle” carried over a "worst particle” is deleted.

In this method, the strategy of replacing the worst string of the new population with the best string of the current population is adopted. Particle Swarm Optimization Algorithms with this strategy are referred as Elitism Particle Swarm Optimization Algorithm or EPSO. The basic steps in an EPSO are described as follows:

1. Generate an initial population *P* of size *M* and calculate the fitness of each particle *S* of *P* using the objective function.
2. Find the best particle pbest of P. If the best particles are not unique, then call anyone of the best particle in P as pbest.
3. Perform the velocity updation and the position updation on the particles in the mating pool according to the PSO equation given in (4) and (5) and obtain a population Ptmp.
4. Compare the fitness of each particle *S* of Ptmp with pbest. Replace the worst particle of Ptmp with pbest if the fitness of each particle in Ptmp is less than the fitness of pbest. Otherwise no replacement takes place in Ptmp. Rename Ptmp as P.
5. Go to step 3.

A significant improvement in the result is obtained when EPSO is used. Repeat the steps until the maximum number of iterations is specified.

4.4 Dynamic Task Scheduling using Hybrid PSO

Modern meta-heuristics manage to combine exploration and exploitation search. The exploration search seeks for 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 in between the iterations of the meta-heuristics. Hybrid PSO was proposed in [24] which makes use of PSO and the Hill Climbing technique and the author has claimed that the hybridization yields a better result than normal PSO. This paper uses the hybridization of PSO and the Simulated Annealing Algorithm.

PSO is a stochastic optimization technique which operates on the principle of social behavior like bird flocking or fish schooling. PSO has a strong ability to find the most optimistic result. The PSO technique can be combined with some other evolutionary optimization technique to yield an even better performance. Simulated Annealing is a kind of global optimization technique based on annealing of metal. It can find the global optimum using stochastic search technology from the means of probability. Simulated Annealing (SA) algorithm has a strong ability to find the local optimistic result. And it can avoid the problem of local optimum, but its ability of finding the global optimistic result is weak. Hence it can be used with other techniques like PSO to yield a better result than used alone.

The HPSO shown in Fig 1 is an optimization algorithm combining the PSO with the SA. PSO has a strong ability in finding the most optimistic result. Meanwhile, at times it has a disadvantage of local optimum. SA has a strong ability in finding a local optimistic result, but its ability in finding the global optimistic result is weak. Combining PSO and SA leads to the combined effect of the good global search algorithm and the good local search algorithm, which yields a promising result.

The embedded simulated annealing heuristic proceeds as follows. Given a particle vector, its r elements are sequentially examined for updating. The value of the examined element is replaced, in turn, by each integer value from 1 to n, and retains the best one that attains the highest fitness value among them. While an element is examined, the values of the remaining r -1 elements remain unchanged. A neighbor of the new particle is selected. The fitness values for the new particle and its neighbor are found. They are compared and the minimum value is selected. This minimum value is assigned to the personal best of this particle. The heuristic is terminated if all the elements of the particle have been examined for updating

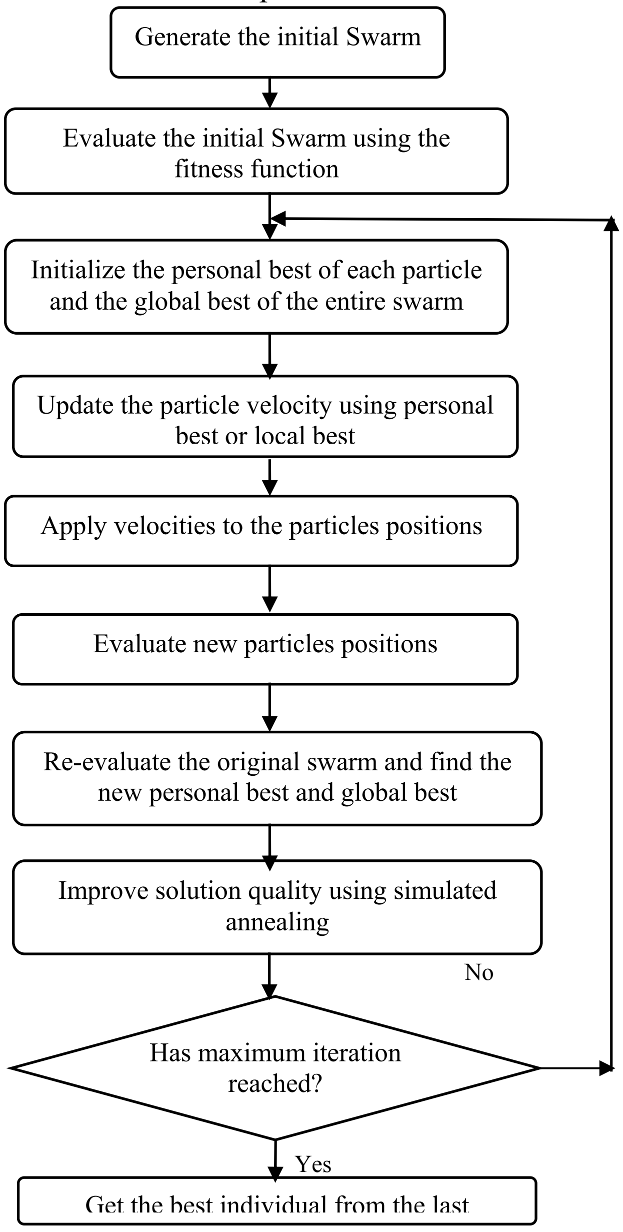
Generate the initial Swarm
Evaluate the initial Swarm using the fitness function
Initialize the personal best of each particle and the global best of the entire swarm
Update the particle velocity using personal best or local best
Apply velocities to the particles positions
Evaluate new particles positions
Re-evaluate the original swarm and find the new personal best and global best
Improve solution quality using simulated annealing
No
Has maximum iteration reached?
Yes
Get the best individual from the last

Fig 1 Hybrid PSO for dynamic task scheduling

and all the particles are examined. The computation for the fitness value due to the element updating can be maximized. The procedure for Hybrid PSO is as follows,

1. Generate the initial swarm.
2. Initialize the personal best of each particle and the global best of the entire swarm.
3. Evaluate the initial swam using the fitness function.
4. Select the personal best and global best of the swarm
5. Update the velocity and the position of each particle using the equations (4) and (5).
6. Obtain the optimal solution in the initial stage.
7. Apply Simulated Annealing algorithm to further refine the solution.
8. Repeat step 3- step 7 until the maximum number of iterations specified.
9. Obtain the optimal solution at the end.

5 Evaluation and Experimental Results

An effective scheduling algorithm has been developed to schedule the tasks onto processors in a distributed computing system using PSO and its variants. The scheduling algorithm has been implemented and applied to benchmark data from the etaillard and a number of different experiments have also been performed to demonstrate the effectiveness of the scheduling algorithm. For these experiments each task was assumed to have a fixed arrival time and execution time. All the scheduling algorithms were presented with the same set of tasks for scheduling and all schedulers have the same information available to them.

Performance of the PSO method and its variants with varying Population Sizes

Dynamic task scheduling is performed in the global scheduler using Particle Swarm Optimization. Each possible solution is represented as the particle. It represents the order in which the tasks are to be executed for a particular processor. This paper involves 20 processors and 50 jobs. The experimental data has been taken from the etaillard benchmark. During the experiment period, the maximum number of iteration was set to 100. The results are compared for varying population size, where the size ranges from 10 to 250.

Table 2 and Fig 2 gives a comparative analysis of the performance of the global scheduler implemented using PSO algorithm and its variants for varying population sizes. PSO with fixed inertia, PSO with dynamically varying inertia, PSO with Elitism (EPSO) and Hybrid PSO (HPSO) are considered. The results of PSO and its variants are also compared with Genetic Algorithm (GA). From the results it can be inferred that the PSO with varying inertia performs better than fixed inertia because there is a balance in exploitation and exploration in search space when the inertia is varied. The result for fixed inertia is 4.6894 in fixed inertia where as it is 4.7392 for variable inertia for a population size of 250.

Table 2 PSO and its variants for varying population sizes

Particle Size	GA	PSO-fi	PSO-vi	EPSO	HPSO
10	4.4812	4.5702	4.5826	4.5826	4.6078
20	4.5001	4.5826	4.6312	4.6894	4.6851
50	4.5521	4.5826	4.6312	4.7392	4.6954
100	4.5864	4.6312	4.6894	4.7392	4.7681
250	4.6012	4.6894	4.7392	4.7392	4.8002

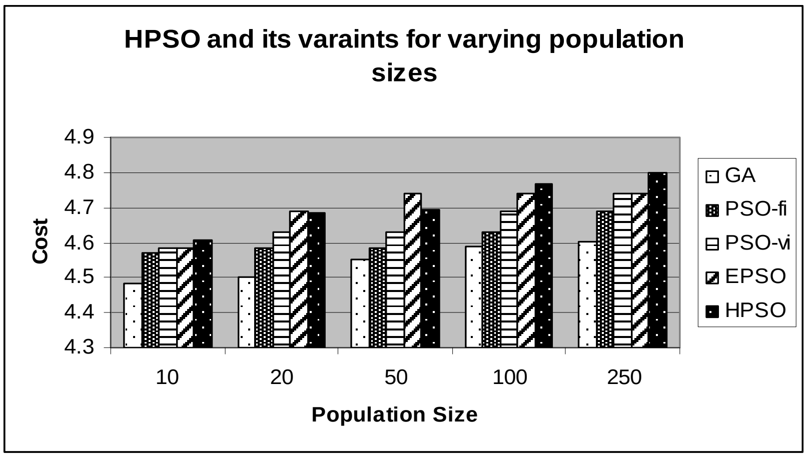

Fig 2Cost for HPSO and its variants for varying population sizes

From Table 2 and fig 2, it can be inferred that the HPSO outperforms all other PSO variants and GA because of the balance in the exploitation and exploration of the search space.Thus a cost of 4.8002 is obtained for HPSO which is comparatively better than the other methods tested.

Performance of the PSO method and its variants for Varying Number of Iterations

The results of PSO and its variants are compared for varying number of iterations which varies from 10 to 250 in Table 3 and Figure 3

Table 3 PSO and its variants for varying number of iterations

Iterations	GA	PSO-fi	PSO-vi	EPSO	HPSO
10	4.267	4.32	4.4647	4.4647	4.6255
20	4.28	4.3761	4.535	4.6023	4.6193
50	4.3761	4.4134	4.6023	4.6193	4.7324
100	4.4134	4.4647	4.6352	4.6833	4.818
200	4.483	4.4985	4.6635	4.6833	4.8196
250	4.5122	4.535	4.6833	4.6833	4.8113

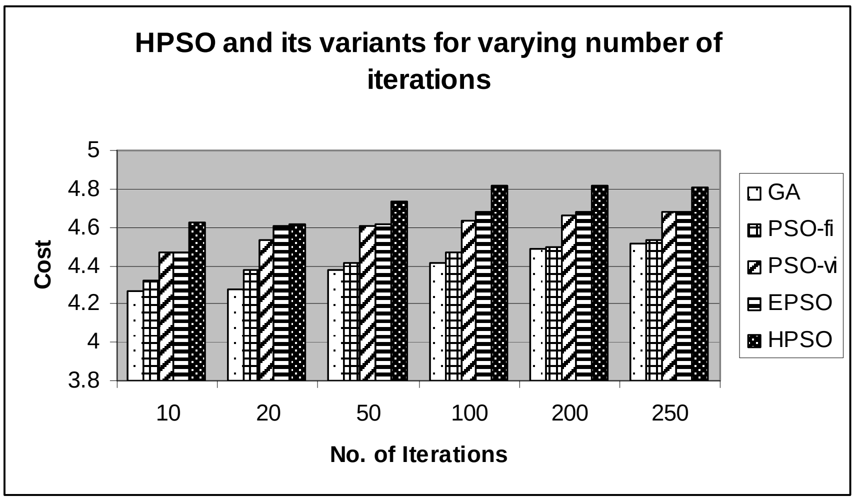

Fig 3 Cost for HPSO and its variants for varying number of iterations

From Table 3and Fig 3, it can be inferred that the HPSO outperforms all other methodologies tested. Thus the load balancing is better in HPSO than other methods.

6 Conclusion and Future work

The objective of this paper is to dynamically schedule the tasks in a heterogeneous environment. The tasks are independent and non-preemptive in nature. PSO is chosen as the optimization technique because it has enormous advantages when compared to other heuristic optimization techniques. Different approaches for solving the dynamic task scheduling using PSO has been tried namely PSO with fixed inertia, PSO with variable inertia, PSO with elitism, and Hybrid PSO. The experimental results show that the Hybrid PSO is cost effective when compared to other variants of PSO. The PSO results are also compared with Genetic Algorithm which is another popular heuristic technique. The results show that the PSO and its variants perform better than the GA. The future work can involve scheduling tasks which are dependent and pre-emptive in nature. Other hybridization techniques can also be used with PSO to achieve a better result.

7 Open Problem

Active research is going on in the optimization of the multiprocessor scheduling using various techniques. The traditional methods are often time consuming and do not provide exact solutions. Lots of work is being undertaken to solve the problem using heuristic approaches. There are open problems of the nature of the tasks involved. We have considered only non-preemptive tasks which are dynamic in nature. The future work could deal with preemptive task scheduling. The parameters in PSO applied for dynamic task scheduling may also be analyzed and updated to achieve an even better performance.

Acknowledgements

The authors would like to thank the Management and the Principal, PSG College of Technology, Coimbatore, India for having provided all the resources for successfully carrying out this research work.

References

[1] Dar-Tzen Peng, Kang G. Shin and Tarek F. Abdelzaher, " Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems", *IEEE Transactions On Software Engineering*, Vol. 23, No. 12(1997), pp. 745-758.

[2] Tzu-Chiang Chiang, Po-Yin Chang, and Yueh-Min Huang, "Multi-Processor Tasks with Resource and Timing Constraints Using Particle Swarm Optimization", *IJCSNS International Journal of Computer Science and Network Security*, Vol.6 No.4 (2006), pp. 71-77.

[3] P. Brucker, "Scheduling Algorithms", Springer, Berlin (2001), 3rd edition.

[4] Hans-Ulrich Heiss and Michael Schmitz, "Decentralized Dynamic Load Balancing: The Particles Approach", Information Sciences, Vol. 84, No.2 (1995), pp.115 - 128.

[5] Abdelmageed Elsadek A and Earl Wells B, "A Heuristic model for task allocation in heterogeneous distributed computing systems", *The International Journal of Computers and Their Applications*, Vol. 6, No. 1(1999), pp. 1 – 36.

[6] Page A.J and Naughton T.J, "Framework for task scheduling in heterogeneous distributed computing using genetic algorithms", *In 15th Artificial Intelligence and Cognitive Science Conference, Ireland*(2004), pp. 137–146.

[7] Page, A.J and Naughton, T.J, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing", *Proceedings of the 19th*

IEEE/ACM International Parallel and Distributed Processing Symposium, Denver, USA(2005), pp. 1530-2075.

[8] Annie S. Wu, Han Yu, “Shiyuan Jin, Kuo-Chi Lin and Guy Schiavone, “ An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No.9(2004), pp. 824 – 834.

[9] Zomaya A.Y and The Y.H, “Observations on using genetic algorithms for dynamic load-balancing”, *IEEE Transactions on Parallel and Distributed Systems*, Vol 12. No.9 (2001), pp.899-911.

[10] Edwin S. H., Hou Ninvan Ansari and Hong Ren, “A genetic algorithm for multiprocessor scheduling”, *IEEE Transactions On Parallel And Distributed Systems*, Vol. 5, No. 2(1994), pp. 113-120.

[11] Manimaran G and Siva Ram Murthy C, “A Fault-Tolerant Dynamic Scheduling Algorithm for Multiprocessor Real-Time Systems and Its Analysis”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No.11 (1998), pp. 1137 – 1152.

[12] Ruey-Maw Chen, and Yueh-Min Huang, “Multiprocessor Task Assignment with Fuzzy Hopfield Neural Network Clustering Techniques”, *Journal of Neural Computing and Applications*, Vol.10, No.1 (2001), pp. 12 -21.

[13]Chunming Yang, Simon D, “A new particle swarm optimization technique”, *Proceedings of the International Conference on Systems Engineering*(2005), pp.164-169.

[14]Van Den Bergh, F, Engelbrecht, A.P, “A study of particle swarm optimization particle trajectories”, *Information Sciences* (2006), pp. 937–997.

[15] Graham Ritchie, “Static Multi-processor scheduling with Ant Colony Optimisation and Local search”, *Master of Science thesis, University of Edinburgh* (2003), pp. 1- 101.

[16]Xie, T, Andrew Sung , Xiao Qin, Man Lin and Laurence Yang, “ Real Time Scheduling with quality of security constraints”, *International Journal of High Performance Computing and Networking,* Vol.4, No.3(2006), pp. 188-197.

[17] Yuhui Shi, “Particle Swarm Optimization”, *IEEE Neural Network Society*, (2004), pp.8-13.

[18]Rui Mendes, James Kennedy and José Neves, “The Fully Informed Particle Swarm: Simpler, Maybe Better”, *IEEE Transactions of Evolutionary Computation*, Vol. 8, No. 3(2004), pp. 204- 210.

[19] Maurice Clerc and James Kennedy, “The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space”, *IEEE Transactions On Evolutionary Computation,* Vol. 6, No. 1(2002), pp. 58 -73.

[20] Parsopoulos K.E, and Vrahatis M.N, “Recent approaches to global optimization problems through particle swarm optimization”, *Natural Computing* , Vol.1(2002), pp. 235 – 306.

[21] Schutte J.F, Reinbolt J.A, Fregly B.J, Haftka R.T and George A.D, "Parallel global optimization with the particle swarm algorithm", *International Journal for Numerical Methods in Engineering*, Vol 6(2004), pp.2296–2315

[22] Kennedy J and Russell C Eberhart, "Swarm Intelligence", *Morgan-Kaufmann* (2001), pp 337-342.

[23] Fatih Taşgetiren M and Yun-Chia Liang, "A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem", *Journal of Economic and Social Research*, Vol.5 No.2 (2004), pp. 1-20.

[24] Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, and Yi-Te Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems", *Computer Standards & Interfaces* , Vol.28(2006), pp. 441-450.