International Journal of Computer Science & Applications
Vol. 4 Issue 3, pp 95-106

Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia

S.N.Sivanandam[1] sns@mail.psgtech.ac.in
P.Visalakshi[2] visa_p@mail.psgtech.ac.in
A.Bhuvaneswari[3] bhuvan888@yahoo.com

[1] Professor and Head, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamilnadu, India – 641 004.
[2] Senior Lecturer, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamilnadu, India – 641 004.
[3] PG Student, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamilnadu, India – 641 004.

Abstract

The problem of task assignment in heterogeneous computing systems has been studied for many years with many variations. We have developed a new hybrid approximation algorithm. The proposed hybrid heuristic model involves Particle Swarm Optimization (PSO) Algorithm and Simulated Annealing (SA) algorithm. This PSO/SA performs static allocation of tasks in a heterogeneous distributed computing system in a manner that is designed to minimize the cost. Particle Swarm Optimization with dynamically reducing inertia is implemented which yields better result than fixed inertia. The experimental results manifest that the proposed hybrid method is effective and efficient in finding near optimal solutions.

Keywords: task assignment problem, distributed systems, hybrid strategy, particle swarm optimization, simulated annealing.

1 Introduction

The problem of scheduling a set of dependent or independent tasks in a distributed computing system is a well-studied area. In this paper, a static task allocation [1] in the heterogeneous computing system is examined which provides a variety of architectural capabilities, orchestrated to perform on application problems whose tasks have diverse execution requirements. Static allocation techniques can be applied to a large set of real-world applications that are able to be formulated in a manner which allows for deterministic execution. Some advantages of these techniques over dynamic ones, which determine the module assignment during runtime, are that, static techniques have no run time overhead and they can be designed using very complex algorithmic mechanisms which fully utilize the known properties of a given application. In this paper, a very fast and easily implemented hybrid algorithm is presented based on particle swarm optimization (PSO) [19] and simulated annealing (SA) algorithm. The proposed method assigns the tasks to processors and avoids becoming trapped in local optimum and also leads to faster convergence towards the targeted solution.

2 Related Work

Several research works have been carried out in Task Assignment Problem [TAP]. The traditional methods such as branch and bound, divide and conquer, and dynamic programming give the global optimum, but are often too time consuming or do not apply for solving typical real-world problems. The researchers [2,5,12] had 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. V.M.Lo [16] says that many of the heuristic algorithms use a graphical representation of the task-processor

system such that a Max Flow/Min Cut Algorithm can be utilized to find assignments of tasks to processors which minimize total execution and communication costs [18] and concludes that a measure of degree to which an algorithm achieves load balancing [3] can yield fairly unbalanced assignments. Traditional methods used in optimization are deterministic, fast, and give exact answers but often tends to get stuck on local optima. Also the time complexity from exponential to polynomial for traditional search algorithms on NP-hard problems cannot be changed.

Consequently, another approach is needed when the traditional methods cannot be applied. The modern heuristic approach helps in such situation. Modern heuristics are general purpose optimization algorithms. Their efficiency or applicability is not tied to any specific problem-domain. Available heuristics include Simulated Annealing algorithms [9], Genetic Algorithms [2,6] and Ant Colony algorithm [22]. Peng-Yeng et.al [11] had proposed a hybrid strategy using Hill Climbing algorithm as a local search method along with Particle Swarm Optimization. Hill Climbing heuristic has the problem of getting trapped in local optima.

The remainder of this Section formulates the TAP and discusses the existing methods for TAP. Section 3 defines the task assignment problem. Section 4 explains the Particle Swarm Optimization heuristic used in this paper. Section 5 illustrates the proposed methods of this paper in detail. Section 6 discusses the results obtained in this work. Finally Section 7 discusses the conclusions and directions for further research in this area.

3 Problem Definition

This paper considers the Task Assignment Problem with the following scenario. The system consists of a set of heterogeneous processors (n) 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 [18].

The objective is to minimize the total execution and communication cost encountered by the task assignment subject to the resource constraints. To achieve minimum cost for the TAP, the function is formulated as,

$$\text{Min } Q(X) = \sum_{i=1}^{r}\sum_{k=1}^{n} e_{ik}x_{ik} + \sum_{i=1}^{r-1}\sum_{j=i+1}^{r} c_{ij}\left(1 - \sum_{k=1}^{n} x_{ik}x_{jk}\right) \quad (1)$$

$$\sum_{k=1}^{n} x_{ik} = 1, \qquad \forall i = 1,2,...,r \quad (2)$$

$$\sum_{k=1}^{n} m_i x_{ik} \le M_k, \qquad \forall k = 1,2,...,n \quad (3)$$

$$\sum_{k=1}^{n} p_i x_{ik} \le P_k, \qquad \forall k = 1,2,...,n \quad (4)$$

$$x_{ik} \in \{0,1\}, \qquad \forall i,k \quad (5)$$

x_{ik} is set to 1 if task i is assigned to processor k. n denotes the number of processors, r denotes the number of tasks, e_{ik} denotes the incurred execution cost if tasks i is executed on processor k. c_{ij} is the incurred communication cost if tasks i and j are executed on different processors. m_i and p_i represents the memory requirements and processing requirements of task i respectively. M_k and P_k are the memory and processing capacity of processor k.

Q(X) is the objective function which combines the total execution cost and total communication cost as specified in 1st and 2nd terms of equation (1). The first constraint mentioned in equation (2) says that each task should be assigned to exactly one processor. Equation (3) and equation (4) are the 2nd and 3rd constraints respectively and they assure that the resource demand should never exceed the resource capacity. The final constraint as mentioned in equation (5) specifies that the x_{ik} is a binary decision variable.

4 Particle Swarm Optimization

PSO is a stochastic optimization technique [24] which operates on the principle of social behavior like bird flocking or fish schooling [19,21]. In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Rui mendes[13] 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. When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best position of the particle, and the resulting algorithm is referred to as the *gbest* PSO. 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 [22].

Each particle in the swarm is represented by the following characteristics. x_i is the *current position* of the particle, v_i is the *current velocity* of the particle and y_i is 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. 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 any of the particles in the entire swarm. Hence, the gbest model is a special case of the lbest model. During each PSO iteration, particle i adjusts its velocity v_{ij} and position vector $particle_{ij}$ through each dimension j by referring the random multipliers, either the personal best vector ($pbest_{ij}$) and the swarm's best vector ($gbest_j$, if the global version is adopted). If global version is adopted, the equations (6) and (7) are used.

$$v_{ij} = w * v_{ij} + c_1 * rand1(pbest_{ij} - particle_{ij}) + c_2 * rand2(gbest_j - particle_{ij}) \quad (6)$$

$$particle_{ij} = particle_{ij} + v_{ij} \quad (7)$$

If local version is adopted, then the following equations (8) and (9) are used:

$$v_{ij} = w * v_{ij} + c_1 * rand1(pbest_{ij} - particle_{ij}) + c_2 * rand2(lbest_j - particle_{ij}) \quad (8)$$

$$particle_{ij} = particle_{ij} + v_{ij} \quad (9)$$

where c_1 and c_2 are the cognitive coefficients and rand1 and rand2 are random real numbers drawn from U (0, 1). Thus, the particle flies through potential solutions toward $pbest_i$ and gbest in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optima. If c1 = c2, each particle will be attracted to the average of pbest and gbest[1]. Since c1 expresses how much the particle trusts its own past experience, it is called the cognitive parameter, and since c2 expresses how much it trusts the swarm, it is called the social parameter. Most implementations use a setting with c1 roughly equal to c2[2]. The inertia weight, w controls the momentum of the particle. The inertia weight can be dynamically varied by applying an annealing scheme for the w-setting of the PSO, where w decreases from w = 0.9 to w = 0.4 over the whole run. In general the inertia weight w is set according to the following equation (10).

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \quad (10)$$

A significant performance improvement is seen by varying the inertia.

4.1 Neighborhood Topology

The selection of the neighborhood plays a major role in reaching the optimal solution faster. The various topological structures are illustrated in Fig. 1 The all topology represents a fully connected graph, and, based on all the statistics, it is conjectured that information spreads quickly [13]. Sociologically, it could represent a small and closed community where decisions are taken in consensus[19].

The ring topology represents a regular graph with a minimum number of edges between its nodes. The graph statistics show that information travels slowly along the graph. This allows for different regions of the search space to be explored at the same time, as information of successful regions takes a long time to travel to the other side of the graph. It is called k best topology in general, (each node connected with k nodes). It is a circle topology if k=2. The four clusters topology represents four cliques connected among

themselves by several gateways. Sociologically, it resembles four mostly isolated communities, where a few individuals have an acquaintance outside their group.

The pyramid represents a three dimensional wire-frame pyramid. It has the lowest average distance of all the graphs and the highest first and second degree neighbors. The square is a graph representing a rectangular lattice that folds like a torus. This structure, is commonly used to represent neighborhoods in the Evolutionary Computation and Cellular Automata communities, and is referred to as the von Neumann neighborhood (Hypercube topology).

The wheel topology, in which the only connections are from one central particle to the others. Based on the available topologies it can be concluded that the shorter the average topological distance between any two particles is, the faster the convergence gets and less connected topologies do not seem to prevent premature convergence [7].

Figure 1: Fully connected, Ring and Master-Slave Topology in PSO

It must be noted that the results only indicate (although with some certainty) that the best topology is gbest in general.

5 Proposed Methodology

This section discusses Simple PSO and the proposed Hybrid PSO. In PSO, each particle corresponds to a candidate solution of the underlying problem. In the proposed method each particle represents a feasible solution for task assignment using a vector of r elements, and each element is an integer value between 1 to n. Fig. 2 shows an illustrative example where each row represents the particles which correspond to a task assignment that assigns five tasks to three processors, and $Particle_{particle\,3,T4}$=P1 means that in particle 3 the Task 4 is assigned to Processor 1.

Particle Number	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

Figure 2: Representation of particles

In hybrid version, hybridization is done by performing simulated annealing at the end of an iteration of simple PSO [17]. Generally the Task Assignment Problem is to assign the n tasks to m processors so that the load is shared and also balanced. Here the proposed system considers n=20 and r=5 i.e. 20 tasks should be shared among 5 processors. The system calculates the fitness value of each assignment and selects the optimal assignment from the set of solutions. The system compares the memory and processing capacity of the processor with the memory and processing requirements of the tasks respectively. If capacity is enough then the task is assigned, else a penalty is added to the calculated fitness value.

5.1 Fitness Evaluation

The initial population is generated randomly and checked for the consistency [15]. Then each particle must be assigned with the velocities obtained randomly and it lies in the interval [0, 1]. Each solution vector in the solution space is evaluated by calculating the fitness value for each vector. The objective value of Q(X) in

Equation (1) can be used to measure the quality of each solution vector. In modern heuristics the infeasible solutions are also considered since they may provide a valuable clue to targeting optimal solution [20]. A penalty function as shown in equation (11) is devised to estimate the infeasibility level of a solution. The penalty function is only related to constraints (3) and (4), and it is given by

$$Penalty(X) = \max\left(0, \sum_{i=1}^{r} m_i x_{ik} - M_k\right) + \max\left(0, \sum_{i=1}^{r} p_i x_{ik} - P_k\right). \quad (11)$$

This penalty is added to the objective function whenever the resource requirement exceeds the capacity. Hence the fitness function of the particle vector can finally be defined as in equation (12),

$$Fitness(X) = (Q(X) + Penalty(X))^{-1} \quad (12)$$

Hence, as the fitness value increases the total cost is minimized which is the objective of the problem.

5.2 Simple PSO

As we had seen earlier in Section 2 the classical PSO is very simple. There are two versions for keeping the neighbors' best vector, namely lbest and gbest. The global neighborhood 'gbest' is the most intuitive neighborhood. In the local neighborhood 'lbest', a particle is just connected to a fragmentary number of processes [21]. The best particle is obtained from, the best particle in each fragment.

5.2.1 gbest PSO

In the global version, every particle has access to the fitness and best value so far of all other particles in the swarm.

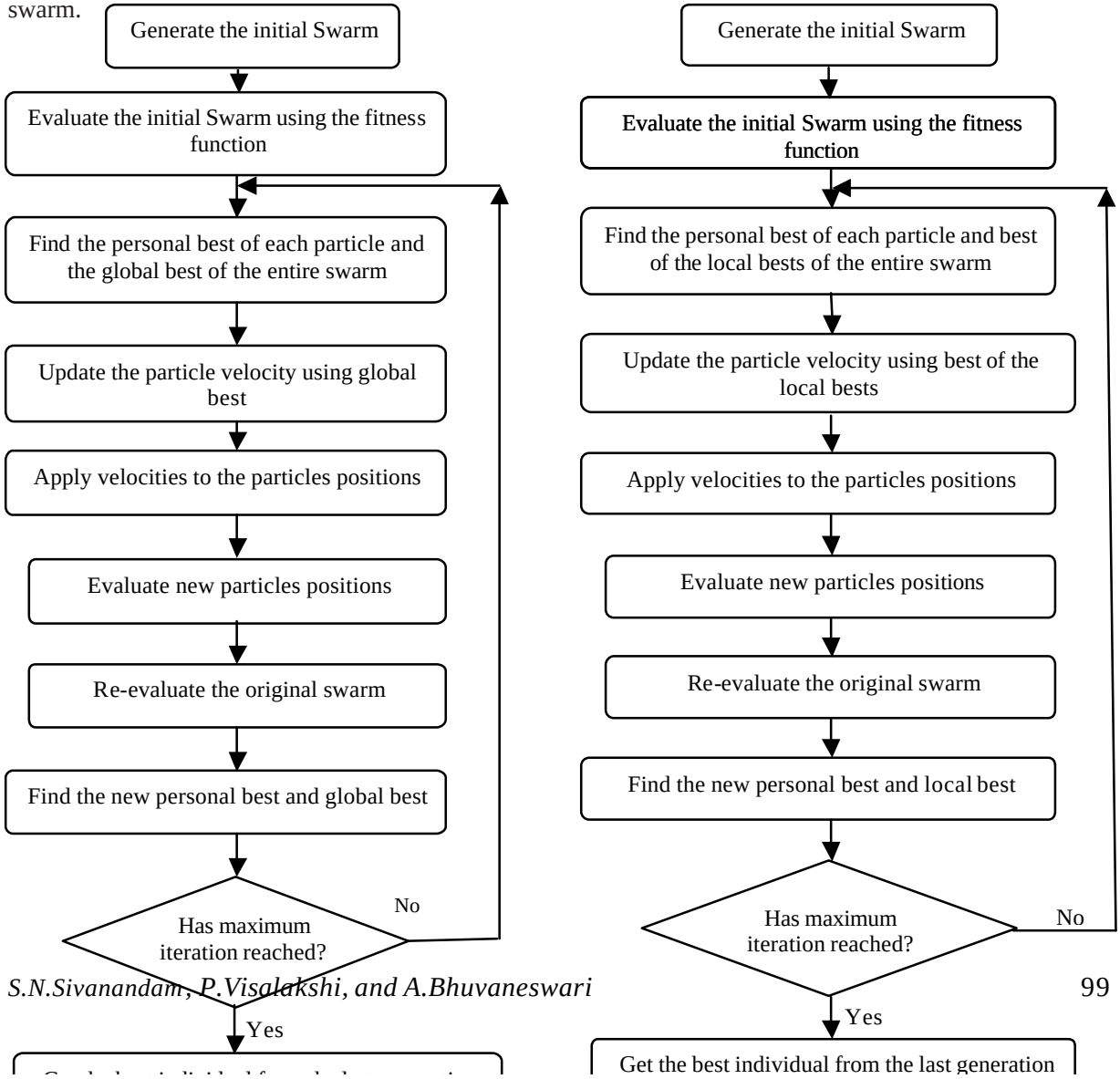


Figure 3: Global best PSO Figure 4: Local best PSO

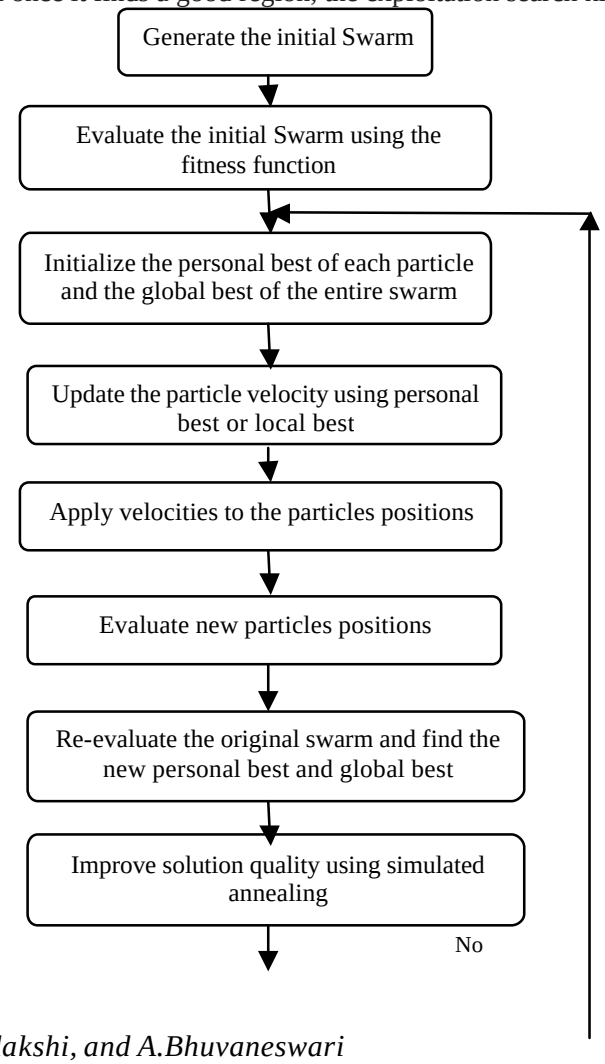
Each particle compares its fitness value with all other particles. This method implements the star topology. It has exploitation of solution spaces but exploration is weak [8]. The implementation can be depicted as a flow chart as shown in Fig. 3

5.2.2 lbest PSO

In the local version, each particle keeps track of the best vector lbest attained by its local topological neighborhood of particles. Each particle compares with its neighbors decided based on the size of the neighborhood. The groups exchange information about local optima. Here the exploitation of solution space is weakened and exploration becomes stronger. The implementation can be depicted as a flow chart as shown in Fig. 4

5.3 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.



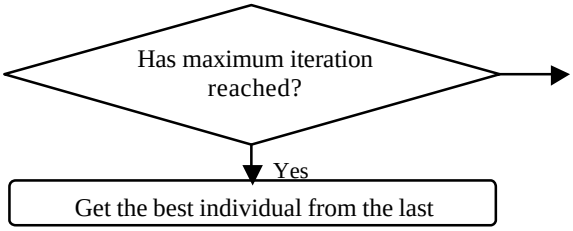


Figure 5: Hybrid PSO

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 [4,11].

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 and all the particles are examined. The computation for the fitness value due to the element updating can be maximized. Since a value change in one element affects the assignment of exactly one task, we can save the fitness computation by only recalculating the system costs and constraint conditions related to the reassigned task. The flow is shown in Fig. 5

6 Results and Discussion

This section describes the results of simulations conducted to gain insight into the performance of the PSO algorithm implementation. Various versions of PSO algorithm like the simple PSO, the global PSO and Hybrid PSO with Simulated Annealing were implemented in MATLAB 7.0.1 and run with E.Taillard's benchmark data. The experimental results clearly demonstrate the effectiveness of the Hybrid version of PSO. The value of (r, n) is set to (20, 5). The values of the other parameters are generated randomly as specified in [11]. The results of this experiment are obtained by varying the number of particles, number of iterations and topology of neighborhood particles.

6.1 Cost Evaluation

The task incurs the execution cost and communication cost when executed on different machines. Our objective is to minimize this total cost. This paper discusses the evaluation of cost in various versions of PSO in three methods. The three methods are Cost Vs Number of iterations, Cost Vs Population Size and Fixed inertia Vs Varying inertia. For each version the number of iterations was increased upto 100 and the results were recorded.

6.1.1 Cost Evaluation in Global Best PSO

In the first method the cost is compared with an increase in the number of iterations. The cost obtained initially was 1011. As illustrated in Fig. 6, the cost reduces as we increase the number of iterations and Gbest PSO converges to the minimum cost of 857 at the 28th iteration and remains the same till the last iteration.

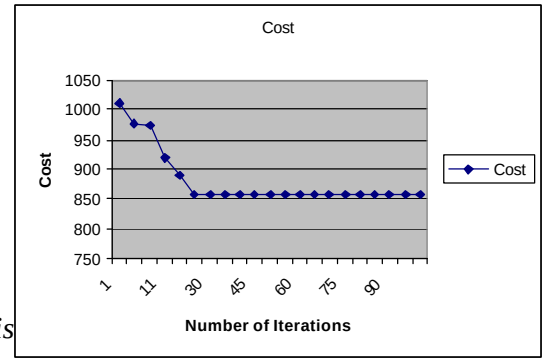


Figure 6: Global best PSO with varying number of iterations

In the second method the cost obtained is compared with an increase in the population size. The cost initially was 883. As illustrated in Fig. 7, the cost reduces as we increase the population size and Gbest PSO converges to the minimum cost of 783 for the population size of 500 and remains the same albeit the increase in the population size.

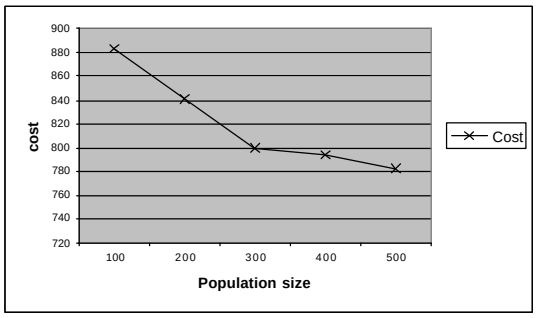


Figure 7: Decreasing Cost in Gbest PSO with varying population size

In the third method, a comparison of global PSO in terms of fixed inertia and varying inertia is considered. The global PSO with fixed inertia starts with a cost of 870 and reduces to only 857. But in global PSO with varying inertia, the initial cost is 883 and it reduces to 783 for a population size of 500 and is depicted in Fig. 8

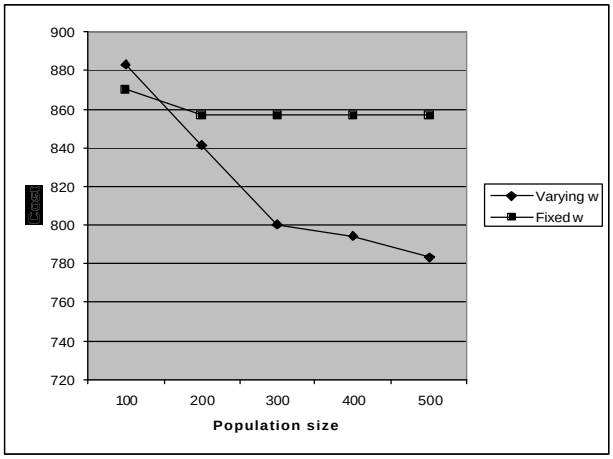


Figure 8: Comparison of global PSO for fixed and varying inertia

Since the global best topology compares each particle with every other particle, the convergence is faster. When the population size is increased the cost is further minimized because the exploration is high. Since the dynamically varying inertia is implemented a balance between global and local exploration is achieved thus requiring less iteration on an average to find the optimal solution compared to fixed inertia.

6.1.2 Cost Evaluation in Local Best PSO

In the first method the cost is compared with an increase in the number of iterations. Fig. 9 depicts the cost obtained initially as 1035. The cost reduces as we increase the number of iterations and Lbest PSO converges

to the minimum cost of 857 at the 100th iteration only. This is because it gets the information only from its neighbors.

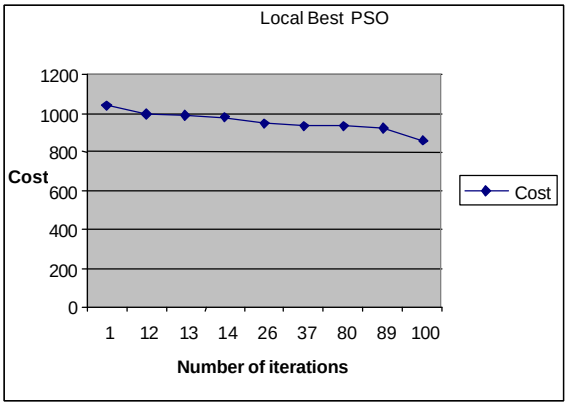


Figure 9: Local best PSO with varying number of iterations

In the second method the cost obtained is compared with an increase in the population size. Fig. 10 depicts the cost obtained initially as 914. The cost reduces as we increase the population size and Lbest PSO converges to the minimum cost of 818 for the population size of 500. The increase in cost is because of the each particle getting the information only from its neighbors.

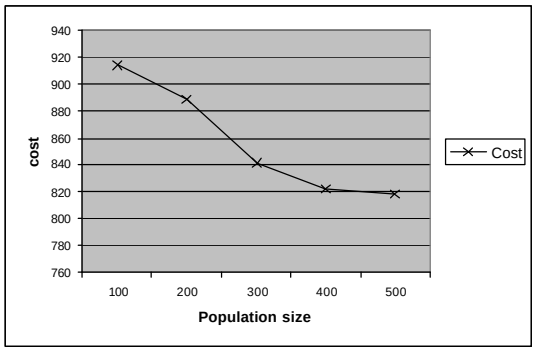


Figure 10: Decreasing Cost in Lbest PSO with varying population size

In the third method, a comparison of local PSO with fixed inertia and varying inertia is considered. The local PSO with fixed inertia starts with a cost of 857 and remains constant. But in local PSO with varying inertia, the initial cost is 914 and it reduces to 818 for a population size of 500 which is depicted in Fig. 11.

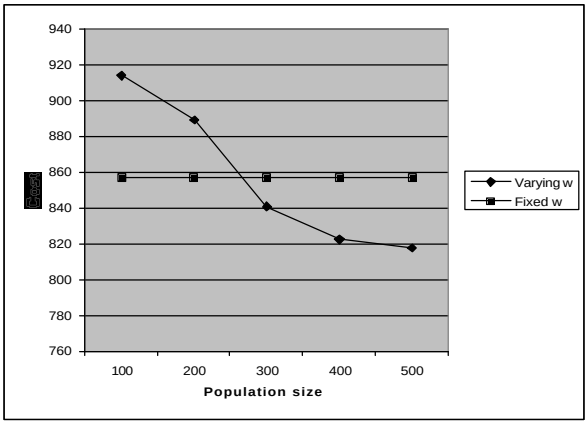


Figure 11: Comparison of local PSO for fixed and varying inertia

Since the local best topology compares each particle with its immediate neighborhood, the convergence is slower compared to global best. When the population size is increased the cost is further minimized because the exploration is high. Since the dynamically varying inertia is implemented a balance between global and local exploration is achieved thus requiring less iteration on an average to find the optimal solution compared to fixed inertia.

6.1.3 Cost Evaluation in Hybrid PSO

In the first method the cost is compared with an increase in the number of iterations. In this method the cost obtained initially was 940 as shown in Fig. 12. The cost reduces as we increase the number of iterations and Hybrid PSO converges to the minimum cost of 857 at the 21st iteration and remains the same till the last iteration. The faster convergence is due to the hybridization with simulated annealing.

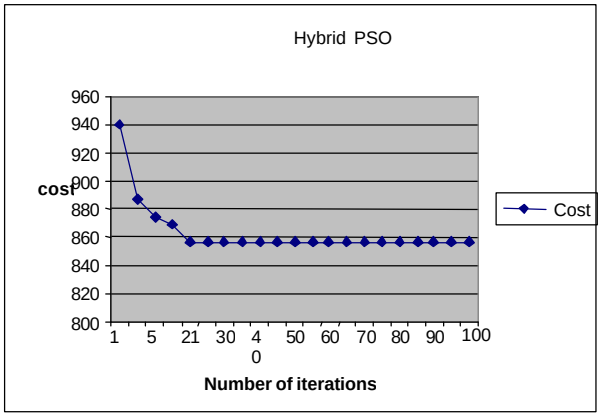


Figure 12: Hybrid PSO with varying number of iterations

In the second method the cost obtained is compared with an increase in the population size. In this method the cost obtained initially was 885 as shown in Fig. 13. The cost reduces as we increase the population size and Hybrid PSO converges to the minimum cost of 783 for a population size of 500 and remains the same for any increase in the population size. The faster convergence is due to the hybridization with simulated annealing.

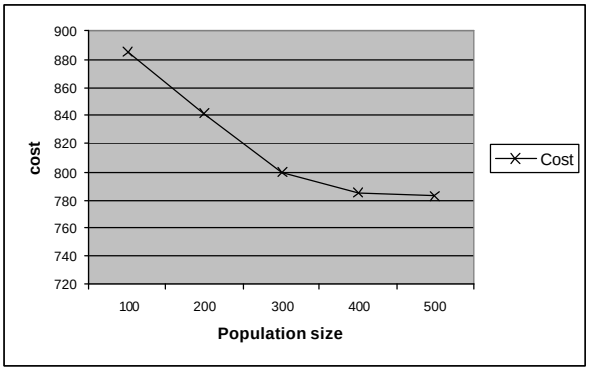


Figure 13: Hybrid PSO with varying population size

In the third method, a comparison of Hybrid PSO with fixed inertia and varying inertia is considered. The Hybrid PSO with fixed inertia starts with a cost of 857 and remains constant. But in Hybrid PSO with varying inertia, the initial cost is 885 and it reduces to 783 for a population size of 500 which is depicted in Fig. 14.

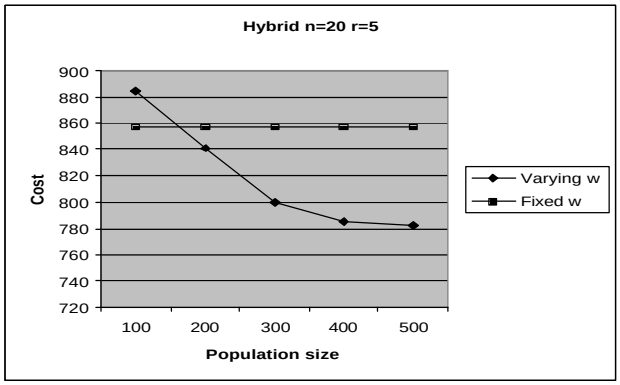


Figure 14: Comparison of Hybrid PSO for fixed and varying inertia

Since the hybrid PSO involves simulated annealing at the end of an iteration the search is refined only towards the feasible solution which leads to faster convergence compared to other methods. When the population size is increased the cost is further minimized because the exploration is high. Since the dynamically varying inertia is implemented a balance between global and local exploration is achieved thus requiring less iteration on an average to find the optimal solution compared to fixed inertia.

6.2 Time Taken For Convergence

Next this paper considers the time taken for the convergence of the particles. The hybrid version of PSO converges faster than all other versions. This is checked for varying population.

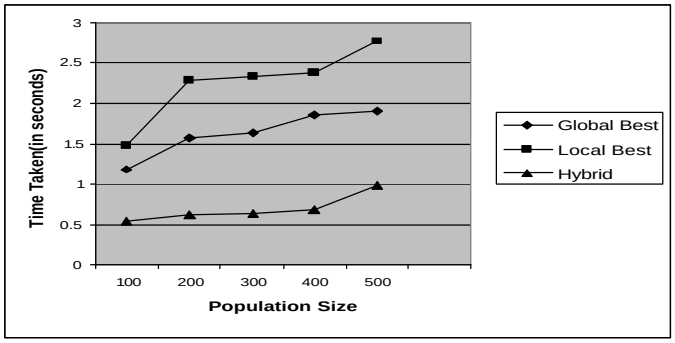


Figure 15: Time Taken for convergence

As the population increases the global and local best versions takes longer time for the convergence. But hybrid PSO performs better than the local and global PSO because the search in the negative direction is prevented. This can be inferred from Fig. 15

7 Conclusion

In many problem domains, the assignment of the tasks of an application to a set of distributed processors such that the incurred cost is minimized and the system throughput is maximized. Several versions of the task assignment problem (TAP) have been formally defined but, unfortunately, most of them are NP-complete. In this paper, we have proposed a particle swarm optimization/simulated annealing (PSO/SA) algorithm which finds a near-optimal task assignment with reasonable time. The Hybrid PSO performs better than the local PSO and the Global PSO. We are currently conducting our research for using PSO to solve another version of the TAP with dependent tasks and the problem objective is to minimize the cost for accomplishing the task execution in a dynamic environment.

References

[1] Abdelmageed Elsadek.A, 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.

[2] Annie S. Wu, Shiyun Jin, Kuo-Chi Lin and Guy Schiavone, Incremental Genetic Algorithm Approach to Multiprocessor Scheduling, *IEEE Transactions on Parallel and Distributed Systems*, 2004.

[3] Batainah.S and Al-Ibrahim.M, Load management in loosely coupled multiprocessor systems, *Journal of Dynamics and Control*, Vol.8, No.1, pp. 107-116, 1998.

[4] Chen Ai-ling, YANG Gen-ke, Wu Zhi-ming, Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem, *Journal of Zhejiang University*, Vol.7, No.4, pp.607-614, 2006

[5] Dar-Tzen Peng, Kang G. Shin, Tarek F. Abdelzaher, Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems, *IEEE Transactions on Software Engineering*, Vol. 23, No. 12, 1997.

[6] 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.

[7] Fatih Tasgetiren.M & Yun-Chia Liang, A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem, *Journal of Economic and Social Research*, Vol.5 No.2, pp. 1-20.

[8] 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.

[9] Osman, I.H., Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, *Annals of Operations Research*, Vol.41, No.4, pp.421-451, 1993

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

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

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

[13] Rui Mendes, James Kennedy and José Neves, The Fully Informed Particle Swarm: Simpler, Maybe Better, *IEEE Transactions of Evolutionary Computation*, Vol. 1, No. 1, January 2005.

[14] 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, pp.2296–2315, 2004

[15] 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.

[16] Virginia Mary Lo, Heuristic algorithms for task assignment in distributed systems, *IEEE Transactions on Computers*, Vol. 37, No. 11, pp. 1384– 1397, 1998.

[17] Yskandar Hamam , Khalil S. Hindi, Assignment of program modules to processors: A simulated annealing approach, *European Journal of Operational Research* 122 509-513 2000

[18] Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters*, Vol. 85, pp. 317–325, 2003.

[19] James Kennedy, Russell Eberhart, Particle Swarm Optimization, *Proc. IEEE Int'l. Conference on Neural Networks*, Vol.4, pp.1942-1948.

[20] Shi.Y, and Eberhart.R, Parameter Selection in Particle Swarm Optimization, Evolutionary Programming VII, *Proceedings of Evolutionary Programming*, pp. 591-600, 1998

[21] Kennedy.J and Russell C. Eberhart, Swarm Intelligence, pp 337-342, Morgan-Kaufmann, 2001.

[22] Graham Ritchie, Static Multi-processor scheduling with Ant Colony Optimization and Local search, *Master of Science thesis*, University of Edinburgh, 2003.

[23] Van Den Bergh.F, Engelbrecht.A.P., A study of particle swarm optimization particle trajectories, *Information Sciences*, PP. 937–97, 2006.

[24] Yuhui Shi, Particle Swarm Optimization, *IEEE Neural Network Society*, 2004