A New Improved Particle Swarm Optimization Algorithm for Multiprocessor Job Scheduling

K.Thanushkodi[1], K.Deeba[2]

[1] Director, Akshaya College of Engineering and Technology
Coimbatore, Tamil nadu, India

[2] Department of Computer Science, Kalaignar Karunanidhi Institute of Technology
Coimbatore, Tamil nadu, India

Abstract

Job Scheduling in a Multiprocessor architecture is an extremely difficult NP hard problem, because it requires a large combinatorial search space and also precedence constraints between the processes. For the effective utilization of multiprocessor system, efficient assignment and scheduling of jobs is more important. This paper proposes a new improved Particle Swarm Optimization (ImPSO) algorithm for the job scheduling in multiprocessor architecture in order to reduce the waiting time and finishing time of the process under consideration. In the Improved PSO, the movement of a particle is governed by three behaviors, namely, inertia, cognitive, and social. The cognitive behavior helps the particle to remember its previous visited best position. This paper proposes to split the cognitive behavior into two sections .This modification helps the particle to search the target very effectively. The proposed ImPSO algorithm is discussed in detail and results are shown considering different number of processes and also the performance results are compared with the conventional techniques such as longest processing time, shortest processing time and Particle Swarm Optimization.

Keywords: *Multiprocessor job scheduling, finishing time, waiting time, PSO, Improved PSO (ImPSO)*

.

1. Introduction

Scheduling, in general, is concerned with allocation of limited resources to certain tasks to optimize few performance criterion, like the completion time, waiting time or cost of production. Job scheduling problem is a popular problem in scheduling area of this kind. The importance of scheduling has increased in recent years due to the extravagant development of new process and technologies. Scheduling, in multiprocessor architecture, can be defined as assigning the tasks of precedence constrained task graph onto a set of processors and determine the sequence of execution of the tasks at each processor. A major factor in the efficient utilization of multiprocessor systems is the proper assignment and scheduling of computational tasks among the processors. This multiprocessor scheduling problem is known to be Non-deterministic Polynomial (NP) complete except in few cases [1].

Several research works has been carried out in the past decades, in the heuristic algorithms for job scheduling and generally, since scheduling problems are NP- hard i.e., the time required to complete the problem to optimality increases exponentially with increasing problem size, the requirement of developing algorithms to find solution to these problem is of highly important and necessary. Some heuristic methods like branch and bound and prime and search [2], have been proposed earlier to solve this kind of problem. Also, the major set of heuristics for job scheduling onto multiprocessor architectures is based on list scheduling [3-9][16]. However the time complexity increases exponentially for these conventional methods and becomes excessive for large problems. Then, the approximation schemes are often utilized to find a optimal solution. It has been reported in [3, 6] that the critical path list scheduling heuristic is within 5 % of the optimal solution 90% of the time when the communication cost is ignored, while in the worst case any list scheduling is within 50% of the optimal solution. The critical path list scheduling no longer provides 50% performance guarantee in the presence of non-negligible intertask communication delays [3-6][16]. The greedy algorithm is also used for solving problem of this kind. In this paper a new Improved PSO (ImPSO) algorithm is used for solving job scheduling in multiprocessor architecture with the objective of minimizing the job finishing time and waiting time.

In the next section, the process of job scheduling in multiprocessor architecture is discussed. Section 3 will introduce the application of the existing optimization algorithms and proposed Improved optimization algorithm for the scheduling problem. Section 4 will show simulation results, and the importance of proposed ImPSO algorithm.

2. Job Scheduling in Multiprocessor architecture

Job scheduling, considered in this paper, is an optimization problem in operating system in which the ideal jobs are assigned to resources at particular times which minimizes the total length of the schedule. Also, multiprocessing is the use of two or more central processing units within a single computer system. This also refers to the ability of the system to support more than one processor and/ or the ability to allocate tasks between them. In multiprocessor scheduling, each request is a job or process. A job scheduling policy uses the information associated with requests to decide which request should be serviced next. All requests waiting to be serviced are kept in a list of pending requests. Whenever scheduling is to be performed, the scheduler examines the pending requests and selects one for servicing. This request is handled over to server. A request leaves the server when it completes or when it is preempted by the scheduler, in which case it is put back into the list of pending requests. In either situation, scheduler performs scheduling to select the next request to be serviced. The scheduler records the information concerning each job in its data structure and maintains it all through the life of the request in the system. The schematic of job scheduling in a multiprocessor architecture is shown in figure.1.

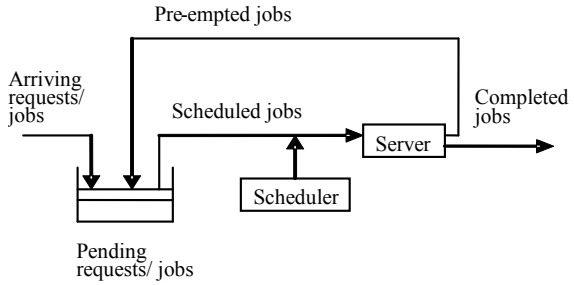

Fig 1. A Schematic of Job scheduling

2.1 Problem Definition

The job scheduling problem of a multiprocessor architecture is a scheduling problem to partition the jobs between different processors by attaining minimum finishing time and minimum waiting time simultaneously. If N different processors and M different jobs are considered, the search space is given by equation (1),

$$\text{Size of search space} = \frac{(M \times N)!}{(N!)^M}. \quad (1)$$

Earlier, Longest Processing Time (LPT), and Shortest Processing Time (SPT) and traditional optimization algorithms was used for solving these type of scheduling problems[10,18-21]. When all the jobs are in ready queue and their respective time slice is determined, LPT selects the longest job and SPT selects the shortest job, thereby having shortest waiting time. Thus SPT is a typical algorithm which minimizes the waiting time. Basically, the total finishing time is defined as the total time taken for the processor to completed its job and the waiting time is defined as the average of time that each job waits in ready queue. The objective function defined for this problem using waiting time and finishing time is given by equation (2),

$$\text{Minimize} \quad \sum_{n=1}^{m_n} \omega_n f_n(x) \quad (2)$$

3. Optimization Techniques

Several heuristic traditional algorithms has been used for solving the job scheduling in a multiprocessor architecture. In this paper a new improved PSO is suggested for the job scheduling NP-hard problem and its ouput is validated against the general particle swarm optimization. The following sections discuss on the application of these techniques to the considered problem.

3.1 Particle Swarm Optimization for Scheduling

The particle swarm optimization (PSO) technique appeared as a promising algorithm for handling the optimization problems. PSO is a population-based stochastic optimization technique, inspired by social behavior of bird flocking or fish schooling [10-15, 17]. PSO is inspired by the ability of flocks of birds, schools of fish, and herds of animals to adapt to their environment, find rich sources of food, and avoid predators by implementing an information sharing approach. PSO technique was invented in the mid 1990s while attempting to simulate the choreographed, graceful motion of swarms of birds as part of a socio cognitive study investigating the notion of collective intelligence in biological populations [10-15, 17].

The basic idea of the PSO is the mathematical modelling and simulation of the food searching activities of a swarm of birds (particles).In the multi dimensional space where the optimal solution is sought, each particle in the swarm is moved towards the optimal point by adding a velocity with its position. The velocity of a particle is influenced by three components, namely, inertial momentum, cognitive, and social. The inertial component simulates the inertial behaviour of the bird to fly in the previous direction. The cognitive component models the memory of the bird about its previous best position, and the social component models the memory of the bird about the best position among the particles.

PSO procedures based on the above concept can be described as follows. Namely, bird flocking optimizes a certain objective function. Each agent knows its best value so far (pbest) and its XY position. Moreover, each agent knows the best value in the group (gbest) among pbests. Each agent tries to modify its position using the current velocity and the distance from

the pbest and gbest. Based on the above discussion, the mathematical model for PSO is as follows,
Velocity update equation is given by

$$V_i = w \times V_i + C_1 \times r_1 \times (P_{best_i} - S_i) + C_2 \times r_2 \times (g_{best_i} - S_i) \quad (3)$$

Using equation (3), a certain velocity that gradually gets close to pbests and gbest can be calculated. The current position (searching point in the solution space) can be modified by the following equation:

$$S_{i+1} == S_i + V_i \quad (4)$$

Where, V_i : velocity of particle i, S_i: current position of the particle, w : inertia weight, C_1: cognition acceleration coefficient, C_2 : social acceleration coefficient, Pbest $_i$: own best position of particle i, gbest $_i$: global best position among the group of particles, r_1, r_2 : uniformly distributed random numbers in the range [0 to 1].
s_i : current position, s_{i+1} : modified position, v_i : current velocity, v_{i+1} : modified velocity, v_{pbest} : velocity based on pbest, v_{gbest} : velocity based on gbest .

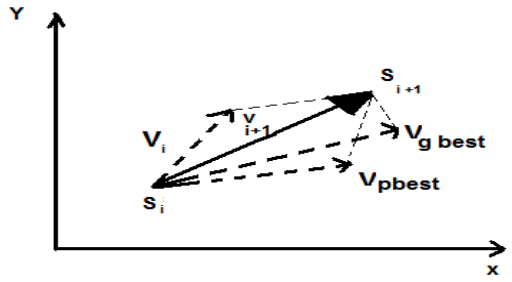

Fig. 2 Flow diagram of PSO

Fig.2 shows the searching point modification of the particles in PSO. The position of each agent is represented by XY-axis position and the velocity (displacement vector) is expressed by vx (the velocity of X-axis) and vy (the velocity of Y-axis). Particle are change their searching point from S_i to S_{i+1} by adding their updated velocity V_i with current position S_i. Each particle tries to modify its current position and velocity according to the distance between its current position S_i and V *pbest*, and the distance between its current position S_i and V *gbest* .
The General particle swarm optimization was applied to the same set of processors with the assigned number of jobs, as done in case of genetic algorithm. The number of particles-100, number of generations=250, the values of c1=c2=1.5 and ω=0.5. Table 2 shows the completed finishing time and waiting time for the respective number of processors and jobs utilizing PSO.

Table 1: PSO for job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	30.10	45.92	42.09	30.65	34.91
Finishing time	60.52	56.49	70.01	72.18	70.09

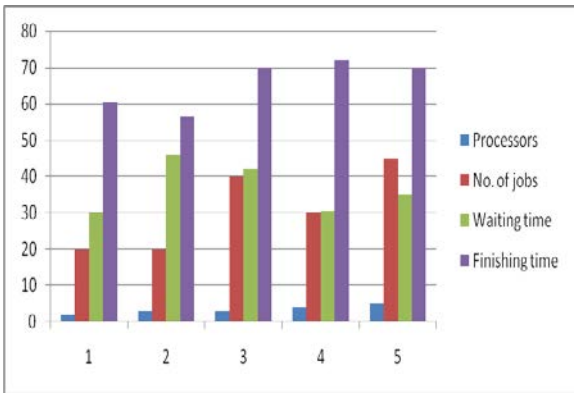

Fig. 3 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using PSO

Figure 3 shows the variation in finishing time and waiting time for the assigned number of jobs and processors using particle swarm optimization.

4. Proposed Improved Particle Swarm Optimization for Scheduling

In this new proposed Improved PSO (ImPSO) having better optimization result compare to general PSO by splitting the cognitive component of the general PSO into two different component. The first component can be called good experience component. This means the bird has a memory about its previously visited best position. This is similar to the general PSO method. The second component is given the name by bad experience component. The bad experience component helps the particle to remember its previously visited worst position. To calculate the new velocity, the bad experience of the particle also taken into consideration. On including the characteristics of Pbest and Pworst in the velocity updation process along with the difference between the present best particle and current particle respectively, the convergence towards the solution is found to be faster and an optimal solution is reached in comparison with conventional PSO approaches. This infers that including the good experience and bad experience component in the velocity updation also reduces the time taken for convergence.
The new velocity update equation is given by

$$V_i = w \times V_i + C_{1g} \times r_1 \times (P\ best_i - S_i) \times P\ best_i + C_{1b} \times r_2 \times (S_i - P_{worst\ i}) \times P_{worst\ i} + C_2 \times r_3 \times (Gbest_i - S_i) \quad (5)$$

Where,
C_{1g} : acceleration coefficient, which accelerate the particle towards its best position;
C_{1b} : acceleration coefficient, which accelerate the particle away from its worst position;
$P_{worst\ i}$: worst position of the particle i;

r_1, r_2, r_3 : uniformly distributed random numbers in the range [0 to 1];

The positions are updated using (4). The inclusion of the worst experience component in the behaviour of the particle gives the additional exploration capacity to the swarm. By using the bad experience component; the particle can bypass its previous worst position and try to occupy the better position. Figure 4 shows the concept of ImPSO searching points.

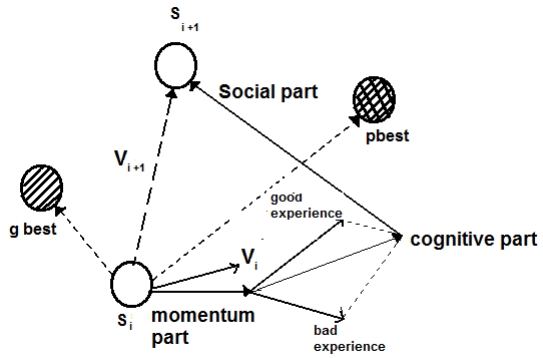

Fig. 4 Concept of Improved Particle Swarm Optimization search point

The algorithmic steps for the Improved PSO is as follows:

Step1: Select the number of particles, generations, tuning accelerating coefficients C_{1g} , C_{1b} , and C_2 and random numbers r_1, r_2, r_3 to start the optimal solution searching
Step2: Initialize the particle position and velocity
Step3: Select particles individual best value for each generation
Step 4: Select the particles global best value, i.e. particle near to the target among all the particles is obtained by comparing all the individual best values.
Step 5: Select the particles individual worst value, i.e. particle too away from the target
Step 6: Update particle individual best (p best), global best (g best), particle worst (P worst) in the velocity equation (5) and obtain the new velocity
Step 7: Update new velocity value in the equation (5) and obtain the position of the particle
Step 8: Find the optimal solution with minimum ISE by the updated new velocity and position

The flowchart for the proposed model formulation scheme is shown in Fig.5.

.

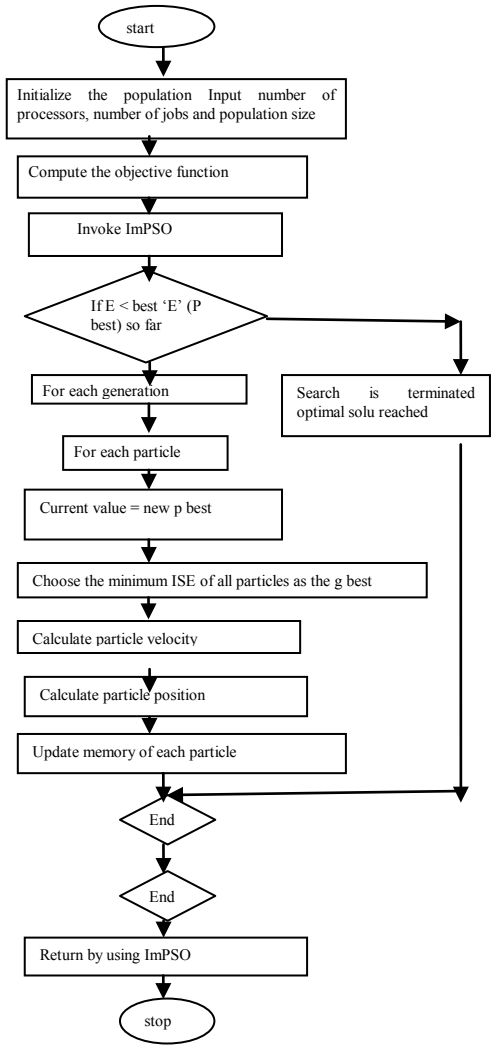

Fig. 5 Flowchart for job scheduling using Improved PSO

The proposed improved particle swarm optimization approach was applied to this multiprocessor scheduling problem. As in this case, the good experience component and the bad experience component are included in the process of velocity updation and the finishing time and waiting time computed are shown in table 2.

Table 2: Proposed Improved PSO for Job scheduling

Processors	2	3	3	4	5
No. of jobs	20	20	40	30	45
Waiting time	29.12	45.00	41.03	29.74	33.65
Finishing time	57.34	54.01	69.04	70.97	69.04

The same number of particles and generations as in case of general PSO is assigned for Improved PSO also. It is observed in case of proposed improved PSO, the finishing time and waiting time has been reduced in comparison with PSO. This is been achieved by the introduction of bad experience and good experience component in the velocity updation process. Figure 6 shows the variation in finishing time and waiting time for

the assigned number of jobs and processors using improved particle swarm optimization.

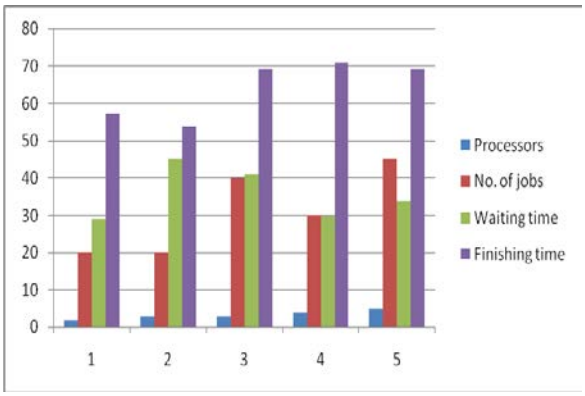

Fig. 6 Chart for job scheduling in multiprocessor with different number of processors and different number of jobs using ImPSO

5. Discussion

The growing heuristic optimization techniques have been applied for job scheduling in multiprocessor architecture. Table 3 shows the completed waiting time and finishing time for PSO, proposed Improved PSO and conventional longest processing time (LPT) and Shortest processing time (SPT) algorithm.

Table 3: Comparison of job using LPT,SPT, PSO and proposed Improved PSO

No of processors	*No of jobs*	*LPT*		*SPT*		*PSO*		*Improved PSO*	
		WT	*FT*	*WT*	*FT*	*WT*	*FT*	*WT*	*FT*
2	20	52.4	60.9	30.21	70.41	30.10	60.52	29.12	57.34
3	20	47.1	56.7	28.31	69.56	45.92	56.49	45.00	54.01
3	40	56.5	70.9	44.96	80.21	42.09	70.01	41.03	69.04
4	30	48.2	62.7	32.64	75.36	30.65	72.18	29.74	70..97
5	45	50.7	66.2	38.91	73.12	34.91	70.09	33.65	69.04

In LPT algorithm, it is noted that the waiting time is drastically high in comparison with the heuristic approached and in SPT with the heuristic approaches and in SPT algorithm, the finishing time is drastically high. Further the introduction of general PSO with the number of particles 100 and within 250 generations minimized the waiting time and finishing time in comparison with LPT and SPT algorithms. The proposed improved PSO with the good(pbest) and bad (pworst) experience component involved with the same number of particles and generations as in comparison with the general PSO, minimized the waiting time and finishing time of the processors with respect to the other considered algorithms.

Thus based on the results, it can be observed that the proposed improved PSO gives better results than the conventional methodologies LPT, SPT and other heuristic optimization technique General PSO. This work was carried out in Intel Pentium 2 core processors with 1 GB RAM.

6. Conclusion

In this paper, a new improved particle swarm optimization has been developed and applied to multiprocessor job shop scheduling. The proposed algorithm partitioned the jobs in the processors by attaining minimum waiting time and finishing time in comparison with the other algorithms, longest processing time, shortest processing time, and particle swarm optimization. The worst component being included along with the best component, tends to minimize the waiting time and finishing time, by its cognitive behaviour. Thus the proposed algorithm, for the same number of generations, has achieved better results.

References

[1] M .R.Garey and D.S. Johnson, Computers and Intractability: A Guide to the theory of NP completeness, San Francisco, CA, W.H. Freeman, 1979.

[2] L .Mitten, 'Branch and Bound Method: general formulation and properties', operational Research, 18, P.P. 24-34, 1970.

[3] T.L.Adam , K .M. Chandy, and J.R. Dicson, " A Comparison of List Schedules for Parallel Processing Systems", Communication of the ACM , Vol.17,pp.685-690, December 1974.

[4] C.Y. Lee, J.J. Hwang, Y. C. Chow, and F. D. Anger," Multiprocessor Scheduling with Interprocessor Communication Delays," Operations Research Letters, Vol. 7, No.3,pp.141-147, June 1998.

[5] S.Selvakumar and C.S. R. Murthy, " Scheduling Precedence Constrained Task Graphs with Non- Negligible Intertask Communication onto Multiprocessors," IEEE Trans. On Parallel and Distributed Computing, Vol, 5.No.3, pp. 328-336, March 1994.

[6] T. Yang and A. Gerasoulis, " List Scheduling with and without Communication Delays," Parallel Comuting, 19, pp. 1321-1344, 1993.

[7] J. Baxter and J.H. Patel, " The LAST Algorithm: A Heuristic- Based Static Task Allocation Algorithm," 1989 International Conference on parallel Processing, Vol.2, pp.217-222, 1989.

[8] G.C. Sih and E.A. Lee, " Scheduling to Account for Interprocessor Communication Within Interconnection-Constrained Processor Network," 1990 International Conference on Parallel Processing, Vol.1, pp.9-17,1990.

[9] M.Y. Wu and D. D. Gajski, " Hypertool: A Programming Aid for Message_Passing Systems," IEEE Trans on Parallel and Distributed Computing, Vol.1, No.3, pp.330-343, July 1990.

[10] K Deeba , K Thanushkodi, , "An Evolutionary Approach for Job Scheduling in a Multiprocessor Architecture", CiiTInternational Journal of Artificial Intelligent Systems and Machine Learning Vol 1, No 4, July 2009.

[11]. Kenedy, J., Eberhart R .C, " Particle Swarm Optimization" proc. IEEE Int. Conf. Neural Networks. Pistcataway, NJ(1995) pp. 1942-1948

[12] R.C. Eberhart and Y. Shi, Comparison between Genetic Algorithm and Particle Swarm Optimization", Evolutionary Programming VII 919980, Lecture Notes in Computer Science 1447, pp 611-616, Spinger

[13] Y. Shi and R. Eberthart: " Empirical study of particle swarm optimization," Proceeding of IEEE Congress on Evolutionary Computation, 1999, pp 1945-1950.

[14] X.D. Zhang, H. S. Yan, " Integrated optimization of production planning and scheduling for a kind of job-shop", International Journal Advanced Manufacture Technology(Spiringer), 2005.

[15] D.Y. Sha , Cheng-Yu Hsu, " A new particle swarm optimization for open shop scheduling problem ", Computers & Operations Research(Elsevier), 2007.
[16] Gur Mosheiov, Daniel Oron, " Open- shop batch scheduling with identical jobs", European Journal of Operations Research(Elsevier), 2006.
[17]. A.P. Engelbrecht, " Fundamentals of Computational Swarm Intelligence", John Wiley & Sons, 2005.
[18] Chen, B. A. Note on L PT scheduling , Operation Research Letters 14(1993), 139-142.
[19] Morrison, J. F.., A note on L PT scheduling, Operations Research Letters 7 (1998), 77-79.
[20] Dobson, G., Scheduling independent tasks on uniform processors, SIAM Journal on Computing 13 (1984), 705-716.
[21] Friesen, D. K., Tighter bounds for LPT scheduling on uniform processsors, SIAM Journal on C omputing 6(1987), 554-660.

Dr. K. Thanushkodi has got 30 ½ yrs of teaching experience in Government Engineering Colleges Has published 45 papers in International Journal and C onferences. Guided 1 P h.D and 1 MS (by Research), Guiding 15 R esearch Scholars for Ph.D Degree in the areas of Power System Engineering, Power Electronics, Computer Networking & Virtual Instrumentation and One Research Scholar for MS (Research). Principal in-charge and Dean, Government College of Engineering, Bargur Served as Senate Member, Periyar University, Salem. Served as member, Research Board, Anna University, Chennai. Served as Member, Academic Council, Anna University, Chennai Serving as Member Board of Studies in Electrical Engineering, Anna University, Chennai. Serving as Member, Board of Studies in Electrical and E lectronics & Electronics and Communication Engineering, Amritha Viswa Vidya Peetham, Deemed University, Coimbatore. Serving as Governing Council Member SACS MAVMM Engineering College, Madurai. Served as Professor and Head of E&I, EEE, CSE & IT Departments at Government College of Technology, Coimbatore.Presently he is the Director of Akshaya College of Engineering and Technology.

K.Deeba, has completed B.E in Electronics and communication in the year 1997, and c ompleted M.Tech (CSE) in National Institute of Technology, Trichy. She is having 11 Years of Teaching Experiencce. She has published 9 Papers in International and National Conferences. Currently working as a Associate Professor in Kalaignar Karunanidhi Institute of Technology, Coimbatore.