Asian Journal of Information Technology 15 (18): 3487-3500, 2016
ISSN: 1682-3915

Hybrid Optimization Techniques for Fuzzy Logic Controller Design in Parallel Job Scheduling Problems

S.V. Sudha
Department of Computer Science and Engineering ,
N.G.P Institute of Technology, 641004 Coimbatore, Tamil Nadu, India

Abstract: In this study attempt is taken to improve the performance of the fuzzy logic controller employed in finding solutions for the parallel job scheduling problems. The performance of the fuzzy logic controller depends on its knowledge base which consists of data base and the rule base. This paper proposes novel hybrid optimization techniques for performance improvement of the fuzzy logic controller by optimizing its knowledge base and a comparative analysis of the proposed optimization techniques are presented based on the computed simulation results. Scheduling of parallel jobs is one of the most challenging aspects with respect to analyzing the performance of the parallel system process. In a parallel system, if the application contains processes which are not co-scheduled together, then the performance of the parallel system starts degrading. Agile Scheduling algorithm classifies the grain sizes in a detailed manner for the real workloads and schedules them in an effective manner. Using the results obtained from the agile scheduling algorithm, a rule based system is generated which classifies all the scheduling states and assigns the appropriate scheduling class for the parallel jobs. The rule system is coded with the Mamdani Fuzzy model and to improve the modeled Fuzzy Logic Controller (FLC), the proposed optimization techniques are applied over the knowledge base of the fuzzy logic controller which involves optimization of both the database and rule base simultaneously. This paper employs optimization algorithms like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and River Formation Dynamics (RFD) for fuzzy logic controller design of parallel systems and as well hybrid technique along with the tabu search algorithm. Simulation results prove the effectiveness of the developed algorithms for fuzzy logic controller design of parallel job shop scheduling problems.

Key words: Job shop scheduling, parallel jobs, parallel systems, genetic algorithm, agile algorithm, particle swarm optimization, Tabu search, ant colony optimization, river formation dynamics, fuzzy logic controller

INTRODUCTION

Scheduling parallel jobs for execution in real time processing is similar to that of bin packing. Each job needs a certain number of processors for a certain time and the scheduler has to pack these jobs together so that most of the processors will be utilized most of the time. In job scheduling, synchronization overhead turns to be a key issue for utilization of the processors. If scheduling does not carefully address the synchronization overhead, the utilization of each processor in a parallel system can end up its usage comparatively lower than a single processor system.

The domain, considered is the scheduling of parallel jobs for execution in a parallel system and this type of scheduling is typically done by partitioning the machine's processors and running a job on each partition. This is similar to packing it into two dimensions. One dimension represents processors and the other represents time. A Parallel job is a rectangle, representing the use of a certain number of processors for certain duration of time. The scheduler has to pack these rectangles as tightly as possible within the space provided by the available resources. The sizes of the rectangles are known as and when each submitted job comes with a specification of how many processors to be used and an estimate of how long it will run. Due to the synchronization between processes in a job, the jobs do not pack perfectly; therefore holes are left in the schedule. If the processes are not co-scheduled properly, it will harm the performance of the parallel algorithm.

The earlier available co-scheduling algorithms include first come first serve, gang scheduling and flexible co-scheduling. The main drawback of first come first serve is the central queue which occupies a region of memory that must be accessed in a manner that enforces mutual exclusion. Thus, this becomes a bottle neck when several processors look for work at the

Table 1: Scheduling strategy results for fuzzy-GA, fuzzy-PSO, fuzzy-ACO and fuzzy-RFD for medium grain workload

Metrics/Scheduling strategy	Fuzzy-GA	Fuzzy-PSO	Fuzzy-ACO	Fuzzy-RFD	FuzzyGA with Tabu	FuzzyPSO with Tabu	FuzzyACO with Tabu	Fuzzy RFD with Tabu
Average waiting Time (h)	2:10:09	2:05:01	1:50:49	1:48:56	1:47:58	1:47:48	1:47:32	1:47:32
Mean response Time (h)	2:15:20	2:02:10	1:51:12	1:50:34	1:50:05	1:49:57	1:49:30	1:49:30
Turn Around Time (h)	0:05:32	0:04:21	0:03:12	0:02:45	0:02	0:01:54	0:01:33	0:01:33
Mean Reaction Time (h)	2:08:30	1:50:14	1:49:12	1:48:56	1:48:3	1:47:57	1:47:32	1:47:32
Mean Utilization	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.70
Mean Slowdown	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26

same time. When all threads are treated as a common pool of threads, it is unlikely that all the threads of a program gains access to processors at the same time.

On requiring high degree of coordination between the threads of a program, the process switches involve several serious compromise performances. In Gang scheduling, the schedule of communicating processes should be precomputed and this require an assumption about which processes gets communicated with each other. Flexible co-scheduling saturates at higher loads. Agile scheduling classifies a detailed granularity of processes and gives better results than the above-mentioned ones with the help of the scheduling metrics like average waiting time, turnaround time, mean response time, mean reaction time, mean slowdown and mean utilization. The agile algorithm is implemented over the processes and using the obtained results of scheduling of the parallel jobs, a rule based scheduling system is framed for a learning environment. The learning environment is designed employing the Mamdani model and the performance of the fuzzy logic controller is improved by optimizing the knowledge base. The optimization of the knowledge base module is carried out by the genetic algorithm, particle swarm optimization, ant colony optimization and river formation dynamics along with the tabu search algorithm (Dutot *et al*., 2011; Frachtenberg *et al*., 2005).

Genetic algorithm tuning for flc design in parallel processors: Genetic algorithm is a heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. Genetic algorithms are inspired by Darwin's theory about evolution survival of fittest. Genetic algorithm represents an intelligent exploitation of a random search employed to solve numerous optimization problems. In finding solutions to problems, there exists numerous solutions called as feasible solutions in search space and each point in the search space is a possible solution. Each possible solution is evaluated by the fitness function.

Binary coded population is used in this paper for the optimization of the database and the rule base of the fuzzy logic controller employed using parallel job scheduling. The binary coded chromosome contains 51 bits. First 15 bits represent the membership function of the input and the output variables of the FLC module. The values represent the base values of the isosceles triangle of the triangular membership function in FLC design. Employing genetic algorithm, the membership function values are modified and the same is applied to the fuzzy logic controller and the defuzzified values are noted.

The process is repeated until the difference between the absolute and the predicted values reaches a minimum value. Table 1 shows the sample chromosome (binary coded value) of the fine grain workload (Moratori *et al*., 2010; Sun *et al*., 2011). Figure 1 shows the process flow of the genetic algorithm. All the genetic fuzzy systems either encode single rules using Michigan approach (Bonarini, 1996) or complete rule bases using Pittsburg approach (Smith, 1980). In this research, each individual represents a whole rule base and thus it follows the Pittsburg approach. Figure 2-9 shows the computed optimization results for the fine grain workloads with respect to the metrics-average waiting time, turnaround time, mean response time, mean reaction time, mean slowdown and mean utilization.

Proposed GA based tuning algorithm for FLC design of parallel processors: The various steps involved in the genetic algorithm based fuzzy logic controller design of parallel processors are as follows:

Step 1: Generate the initial population P based on the binary coded value chromosomes of 51 bits. The population is used to represent the database and the rule base of the fuzzy logic controller employed for the parallel

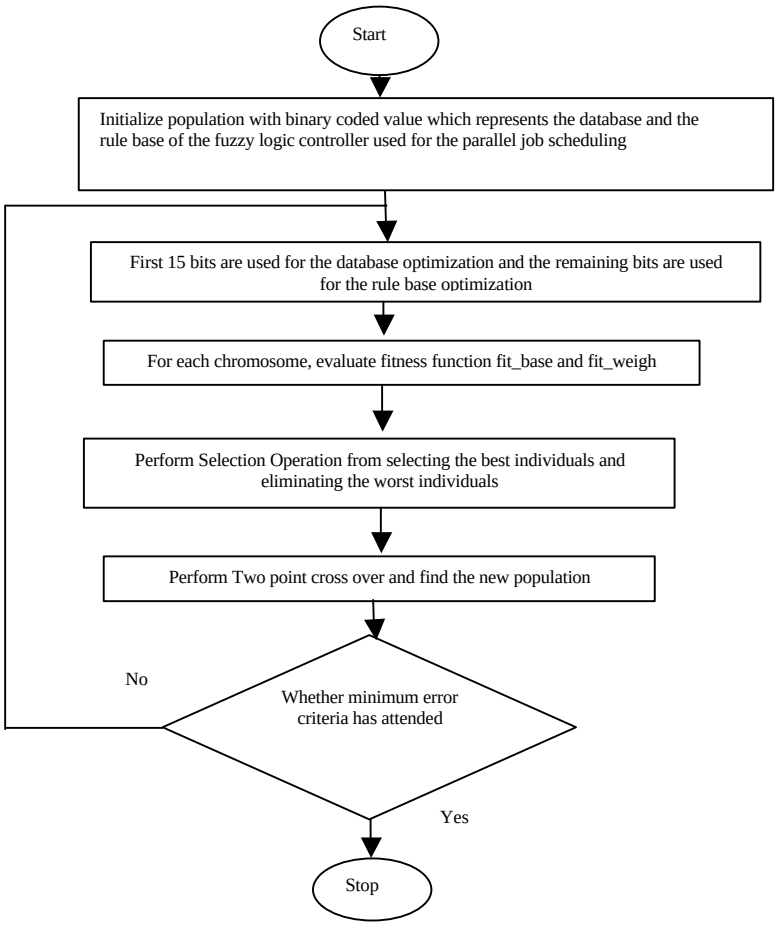

Fig. 1: Process flow of genetic algorithm process

job scheduling. Initialize the weight matrix (weigh_rule) which represents the values that specifies the increased weight factor component allocated to the rule used for FLC module in parallel job scheduling.

Step 2: When the current population P does not get converged do the following,

Step 2.1: Starting from the most significant bits of the binary coded value, every five bits are assigned to each of the base value of the two inputs and one output of the fuzzy controller. Totally 15 bits are used out of 51 bits of the population for optimization of the database of the FLC design.

Step 2.2: The remaining 36 bits are used to represent the rule base of the fuzzy controller.

Step 2.3: The base ranges for the input and the output of the fuzzy logic controller are defined.

Step 2.4: Linear mapping rule is used to determine the real values of the variables. The real values of base components are determined using:

$$b_i = b_i^{min} + (b_i^{max} - b_i^{min})/(2^l - 1) * dv \quad (1)$$

where 'dv' is the decoded value of the binary string, b_i^{min}, b_i^{max} are the minimum and the maximum base values, l is the number of bits used to code b_1, b_2 and b_3 and in this case it is 5 bits. 'b1' is the base of the interior isosceles triangle of the membership function distribution of the one of the input variable, 'b2' is the base of the interior isosceles triangle of the membership function distribution of one of the input variable and 'b3' specifies the base of the interior isosceles triangle of the membership function distribution of the output variable.

Step 2.5: Calculate the fitness value, fit_base =| absolute – predicted| for each of the chromosomes.

Step 2.6: For every last 36 bits of the chromosome, calculate fit_weight with the help of the weigh_rule matrix as follows:

$$\text{fit} - \text{weigh} = \sum_{i=36}^{51} \text{weightage}\ (\text{bit}_i) \quad (2)$$

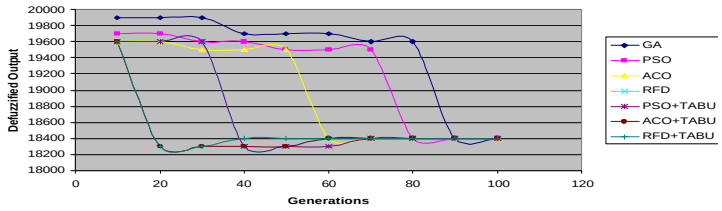

Fig. 2: Database optimization results of fine grain workload (average waiting time and turn around time)

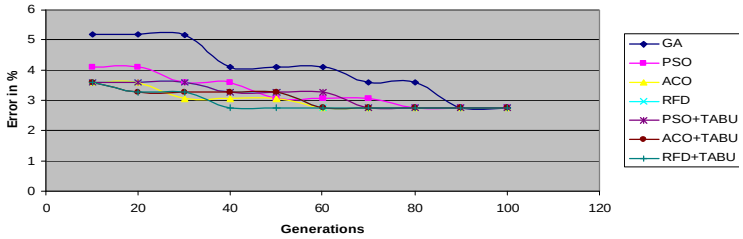

Fig. 3: Database optimization results of error calculations of fine grain workload (average waiting time and urnaround time)

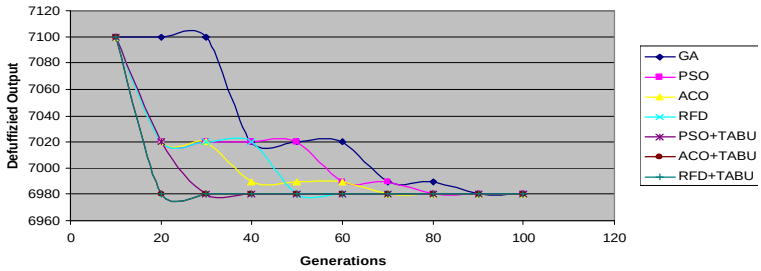

Fig. 4 Database optimization results of fine grain workload (Mean response time and mean reaction time)

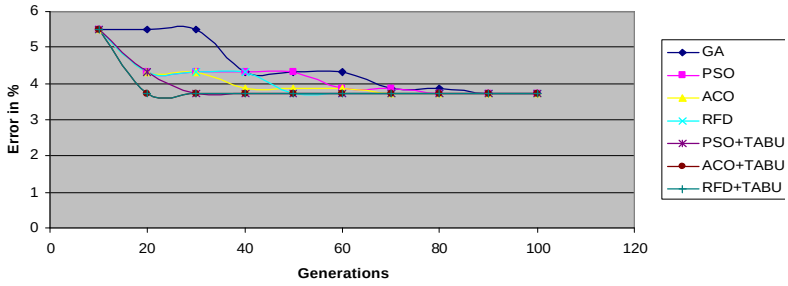

Fig. 5: Database optimization results of the error calculations of the fine grain workload (mean response time and mean reaction time)

Step 2.7: Keep the best of the individuals and terminate the worst of the individuals.

Step 2.8: Using the two points cross over operation, reproduce the offspring from the current population.

Step 3: Optimal solution is available and this is achieved with that of the best individuals.

Proposed hybrid PSO Tabu Search approach to tune FLC module in parallel processors: Particle swarm

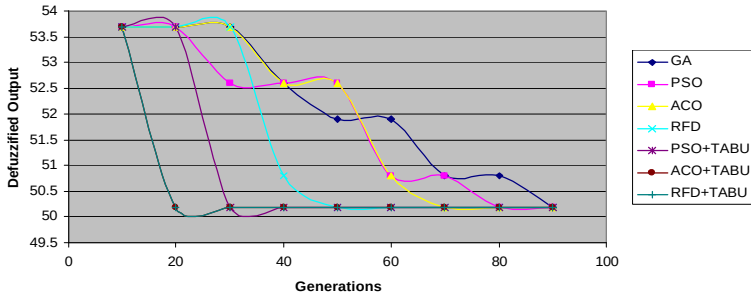

Fig. 6: Database optimization results of fine grain workload (mean slowdown)

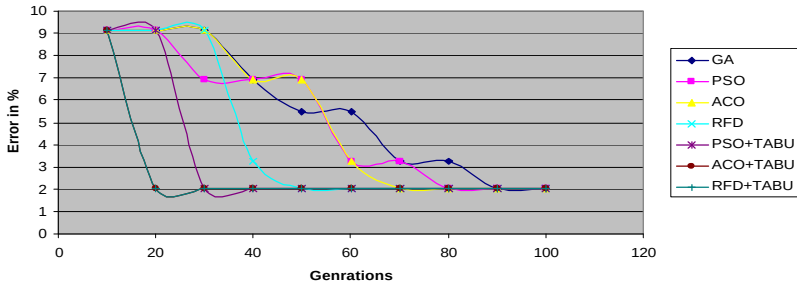

Fig. 7: database optimization results of the error calculations of the fine grain workload (mean slowdown)

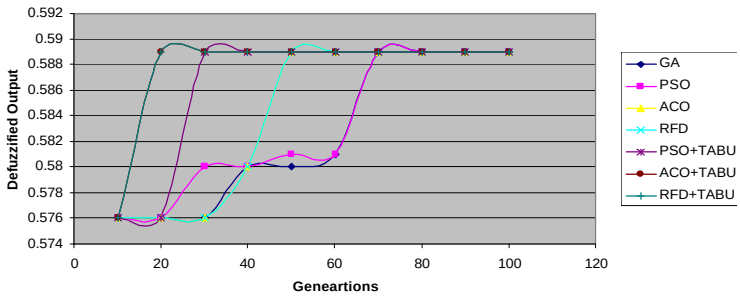

Fig. 8: Database optimization results of fine grain workload (mean utilization)

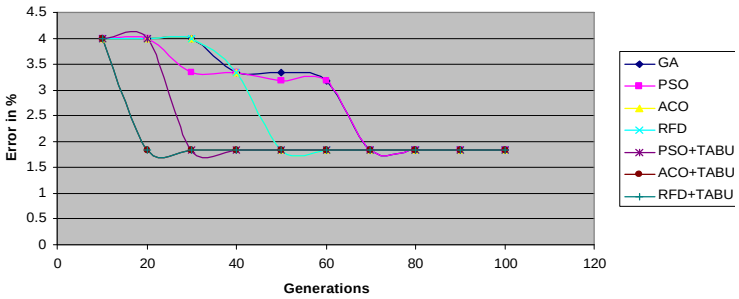

Fig. 9: Database optimization results of the error calculations of fine grain workload (mean utilization)

optimization is a population based stochastic optimization technique, inspired by social behavior of bird flocking or fish schooling. The particle swarm optimization (PSO) belongs to the class of direct search methods employed to find an optimal solution of any objective function. In PSO, each single solution is a bird in the search space and it is called as particle. All the particles have fitness values that are evaluated by the defined fitness function.

In this study, PSO takes the particles in the form of binary coded values with 51 bits. The first 15 bits are used to optimize the database of the fuzzy controller and the remaining bits are used for the rule base optimization of the fuzzy controller employed in the scheduling process. Tabu search is the meta-heuristic local search algorithm that is employed for solving combinatorial optimization problem. Tabu search uses a local or neighboring search procedure to iteratively move from one potential solution to an improved solution in the neighborhood search space, until the set stopping criteria is met. The tabu search employs a tabu list which is a memory structure used to filter the solutions that get admitted into the neighborhood search space.

This study hybridizes PSO and Tabu search approach. Particle swarm optimization results in a premature convergence and produces poor quality of the solution by considering the local optimum. Tabu search employ adaptive memory processes for guiding search mechanism and hence the hybrid method tends to produce better results. Figure 10 show the flow of hybrid Particle Swarm Optimization and the Tabu Search algorithm.

Proposed hybrid PSO Tabu Search Algorithm for FLC tuning: The various steps involved in the proposed hybrid PSO Tabu Search algorithm is as follows as: Step 1: Initialize the particles of binary coded value of 51 bits. The particle is used to represents the database and the rule base of the fuzzy logic controller used for the parallel job scheduling. Initialize the weight matrix weigh_rule, which represents the weight component of the rules used for the fuzzy logic controller for the parallel job scheduling.

Initialize the particle with position vector and velocity vector Initialize the size of the swarm, swarm size Initialize particle list to null. The base ranges for the inputs and the output of the fuzzy logic controller are defined.

Invoke PSO: Step 2: For each particle generated randomly, Step 2.1: Check the particle with the particle list Step 2.2: If the particle is present in the particle_list then Step 2.3: The position is already visited and makes the particle to move to its neighboring position that is not visited. Else starting from the most significant bits of the binary coded value, each five bits are assigned to each of the base value of the two inputs and one output of the fuzzy controller. Totally 15 bits are used out of 51 bits of the population for the database of the fuzzy logic controller. Step 2.4: The remaining 36 bits are used to represent the rule base of the fuzzy controller. Linear mapping rule is used to determine the real values of the variables. The real values of base part are determined by:

$$b_i = b_i^{min} + (b_i^{max} - b_i^{min}) / (2^l - 1) * dv \quad (3)$$

where 'dv' is the decoded value of the binary string, b_i^{min}, b_i^{max} are the minimum and the maximum base values, l is the number of bits used to code the base values and in this case it is 5 bits. Step 2.5: Calculate the fitness fit_base =| absolute – predicted| for each of the chromosomes. Step 2.6: For every last 36 bits of the chromosome, calculate fit_weight with the help of the weigh_rule matrix as follows:

$$\text{fit} - \text{weight} = \sum_{i=36}^{51} \text{weightage(bit)} \quad (4)$$

Step 2.7: If the fitness value is better than the best fitness value for both database and rule base optimization in history then set the current value as the new best value Step 2.8: Choose the particle with the best fitness value of the entire particle as the gBest1 and gBest2. Step 2.9: Set gbest1, gbest2 to particle_fit1, particle_fit2. Step 2.10: Input gbest1 to tabusearch1 and gbest2 to tabusearch2

Invoke tabu search 1 process: Step 3: Initialize Tabu list, candidate list to null, sbest =gbest1. Step 3.1: Generate the neighboring candidates. Step 3.2: For each candidate. Step 3.3: If candidate not present in the tabulist then add to the candidate list}. Step 4: Find the best candidate. Step 5: Update to the tabu list. Step 6: Update to the particle list till the candidates exist.

Invoke tabu search 2 process: Step 7: Initialize Tabu list, candidate list to null, sbest =gbest2. Step 7.1: Generate the neighboring candidates. Step 7.2: For each candidate. Step 7.3: If candidate not present in the tabu list then add to the candidate list. Step 8: Find the best candidate.

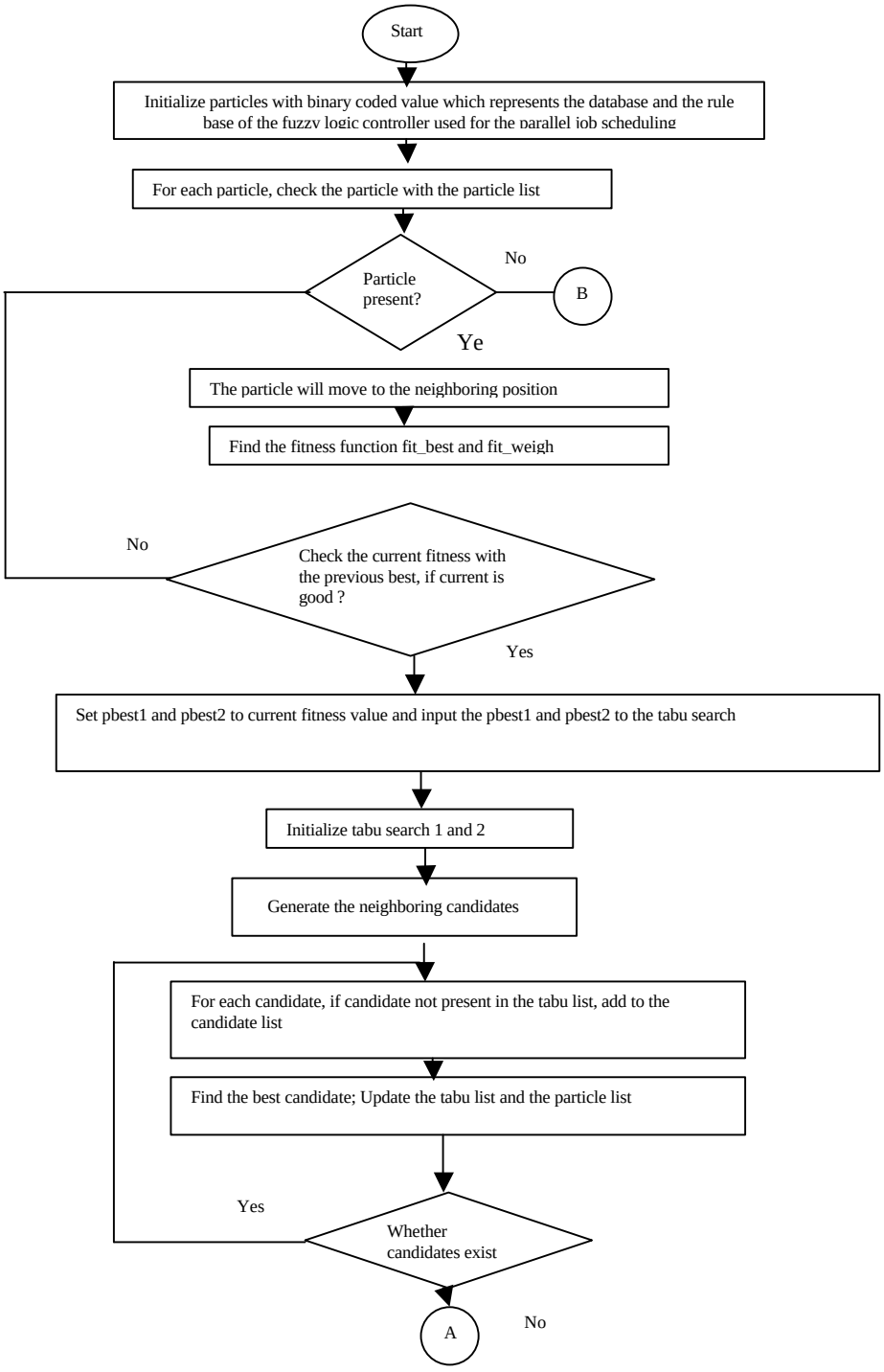

Fig. 10: Process flow of proposed hybrid PSO Tabu search

Step 9: Update to the tabu list
Step 10: Update to the particle list till the candidates exist
Step 11: Update swarm size until the particles exist
Step 12: Choose best of all particles and set as gbest 1 and gbest2 and Update the particle position and velocity,

v1[ ] = v1[ ] + c1* rand( )*(pbest1[ ]-present1[ ])+c2*rand()*(gbest1[ ]– present1[ ])
present1[ ] =present1[ ]+v1[ ]
v2[ ]=v2[ ]+c1*rand( )*(pbest2[ ]-present2[ ])+c2* rand( )*(gbest2 [ ]– present2[ ])

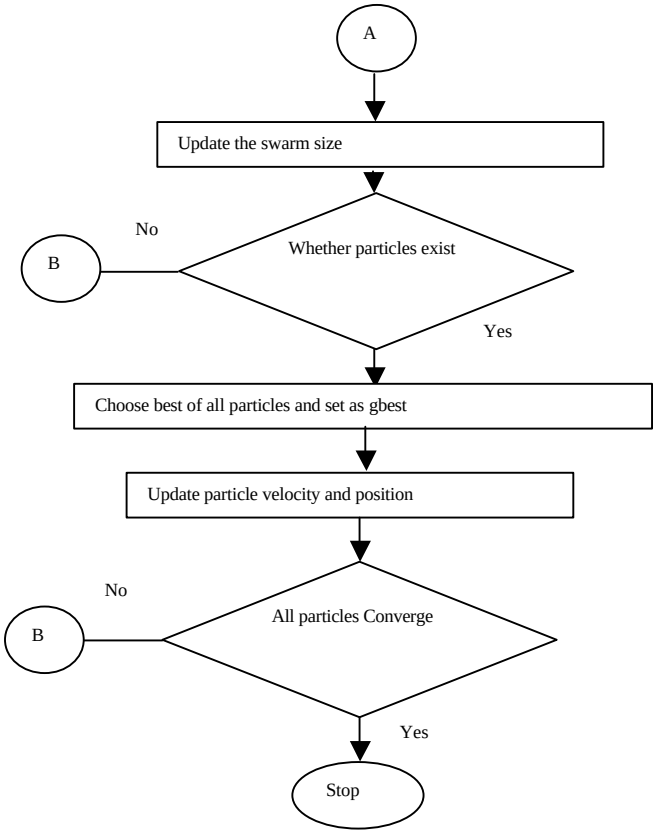

Fig. 11: Database optimization results of medium grain workload (average waiting time and turnaround time)

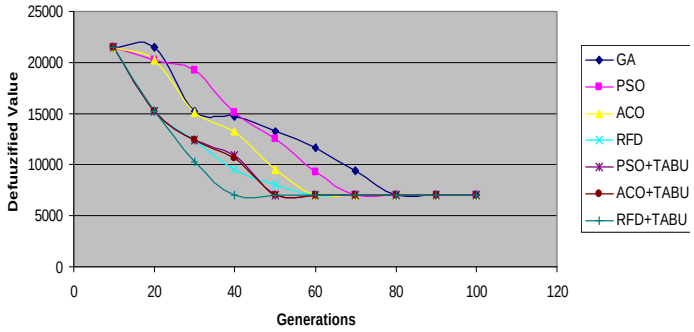

Fig. 12: Database optimization results of medium grain workload (mean response time and mean reaction time)

present2 [ ] =present2 [ ] +v2 [ ] where v1[ ] is the particle velocity, present1 [ ] is the current solution are the values considered for the database optimization and v2[ ] is the particle velocity, present 2[ ] is the current solution are the values considered for the rule base optimization of the fuzzy logic controller considered for the parallel job scheduling, rand( ) is a random number between (0,1), c1 and c2 are the learning factors usually takes the value.

Step 13: Perform the generations until all particles converge or maximum number of iterations are reached.

On implementing the proposed hybrid PSO Tabu search algorithm, Fig 11-18 are obtained that shows the optimization results for the medium grain workloads for the metrics-average waiting time, turnaround time, mean response time, mean reaction time, mean slowdown and mean utilization.

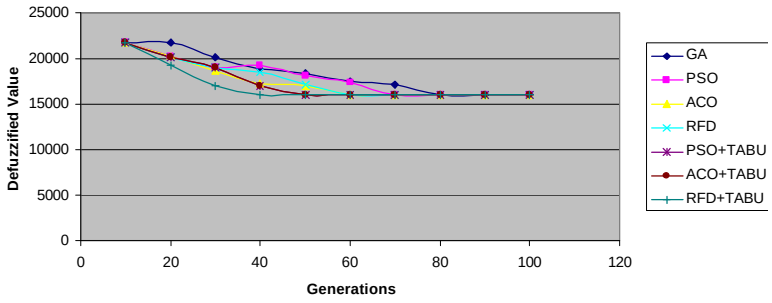

Fig. 13: Database optimization results of medium grain workload (mean slowdown)

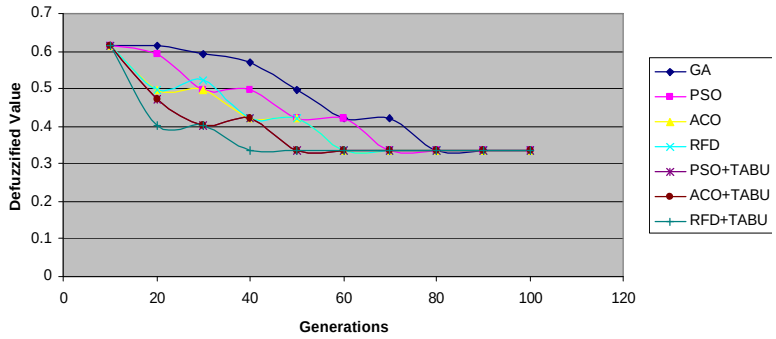

Fig. 14: Database optimization results of medium grain workload (mean utilization)

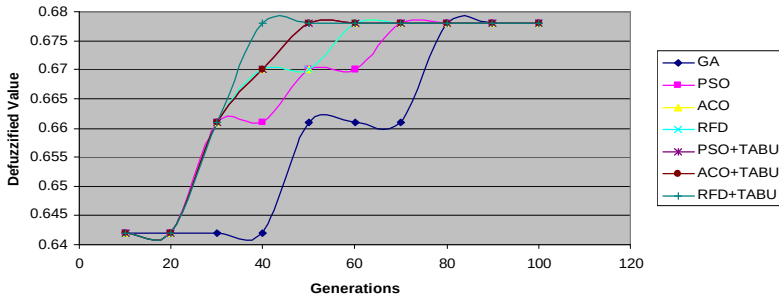

Fig. 15: Database optimization results of error calculations of fine grain workload (average waiting time and turnaround time)

Ant colony optimization for flc tuning in parallel processors: The inspiring source of the ant colony optimization is the food behavior of the real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. The ants deposit a chemical pheromone trail during its return trip to the nest. The other ants follow the pheromone laying on trails. Ant colony optimization constitutes a new family of global search bio inspired algorithm. ACO is applied in this paper to tune the knowledge base of FLC module in parallel job shop scheduling process. Figure 19-22 show the optimization results for the coarse grain workloads for the metrics-average waiting time, turnaround time, mean response time, mean reaction time, mean slowdown and mean utilization.

River formation dynamics for flc tuning in parallel processors: In River Formation Dynamics (RFD), ants substituted by drops and pheromone trails are replaced by altitude. Instead of associating the pheromone values to edges, the altitude value is associated with the nodes. Drops erode the ground or deposit the sediments as they

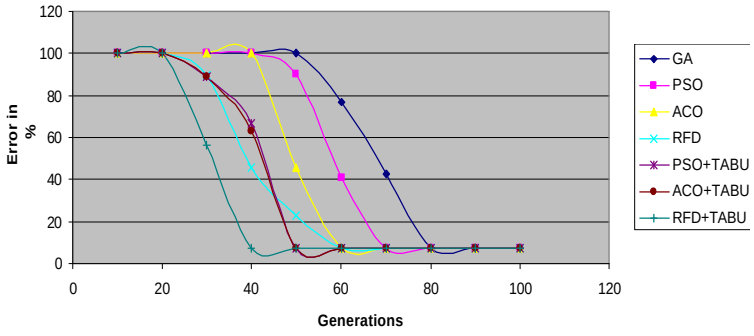

Fig. 16: Database optimization results of the error calculations of the medium grain workload (mean response time and mean reaction time)

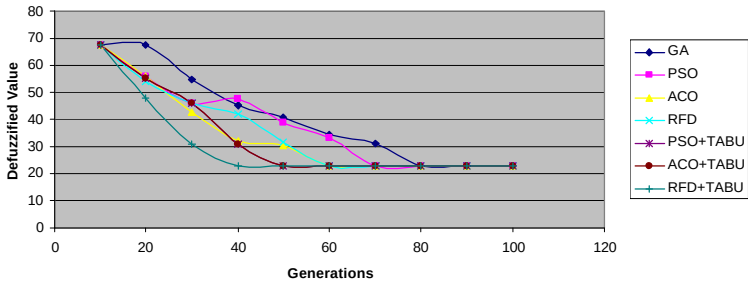

Fig. 17: Database optimization results of the error calculations of the medium grain workload (mean slowdown)

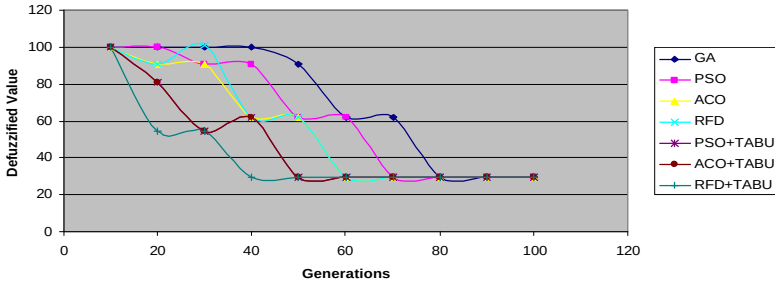

Fig. 18: Database optimization results of the error calculations of medium grain workload (mean utilization)

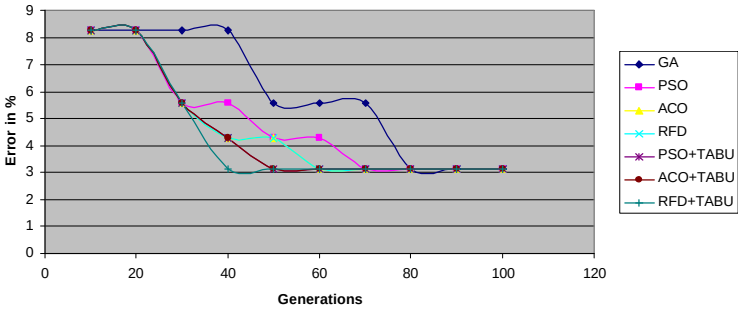

Fig. 19: Database optimization results of coarse grain workload (average waiting time and turnaround time)

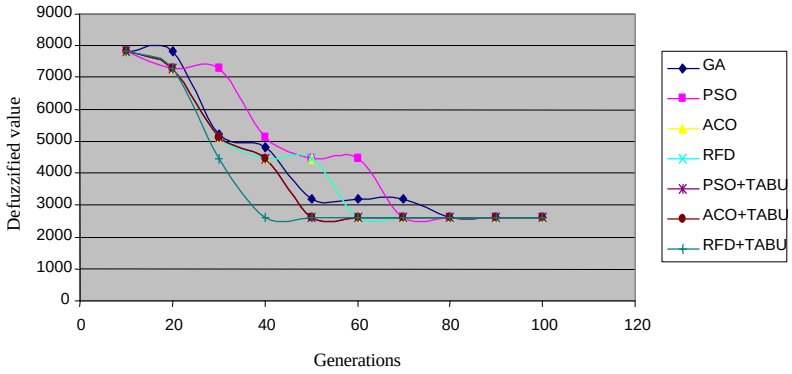

Fig. 20: Database optimization results of coarse grain workload (mean response time and mean reaction time)

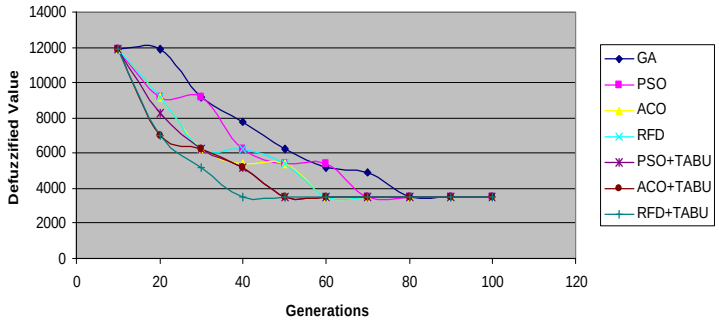

Fig. 21: Database optimization results of coarse grain workload (mean slowdown)

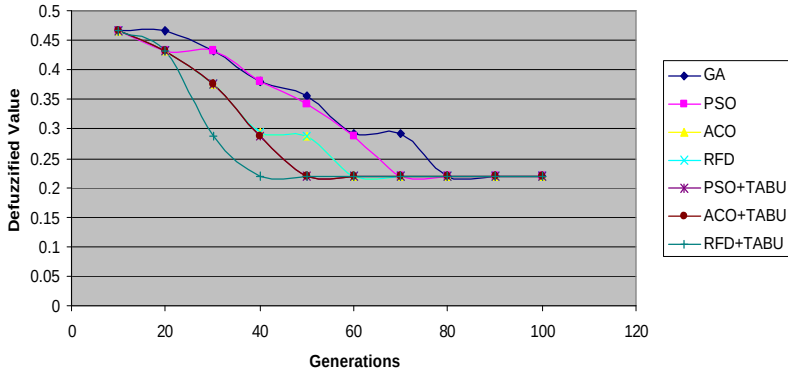

Fig. 22: Database optimization results of coarse grain workload (error calculations)

move. First the drops are initialized and continues execution the algorithm until all the drops find the same solution .The drops are made to move in a random way, While movement they deposit sediments, erode the ground and all the drops will follow the sediments and reach the destination node. RFD is applied in this study to tune the knowledge base of FLC module in parallel job shop scheduling process. Figure 23-26 shows the optimization results for the Independent grain workloads for the metrics like average waiting time, turnaround time, mean response time, mean reaction time, mean slowdown and mean utilization.

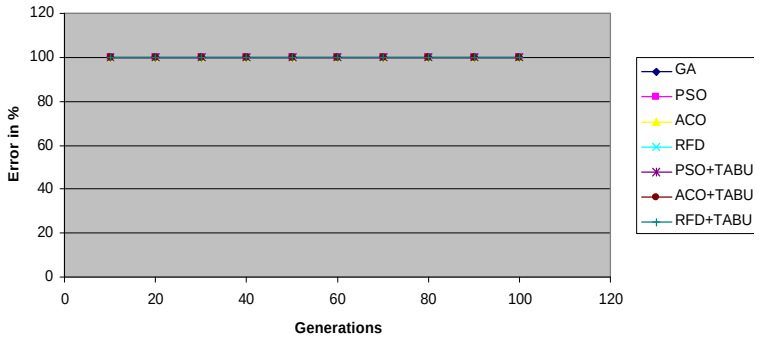

Fig. 23: Database optimization results of independent grain workload (average waiting time and turnaround time)

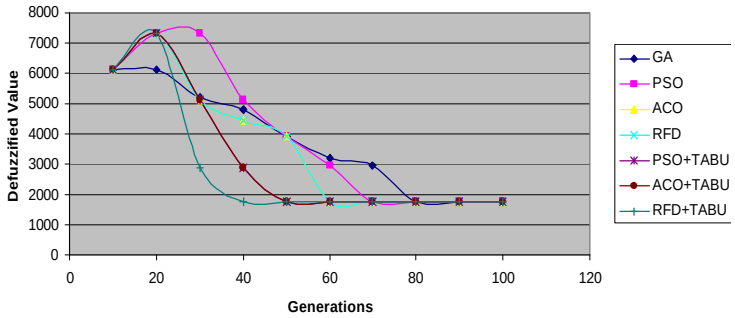

Fig. 24: Database optimization results of independent grain workload (mean response time and mean reaction time)

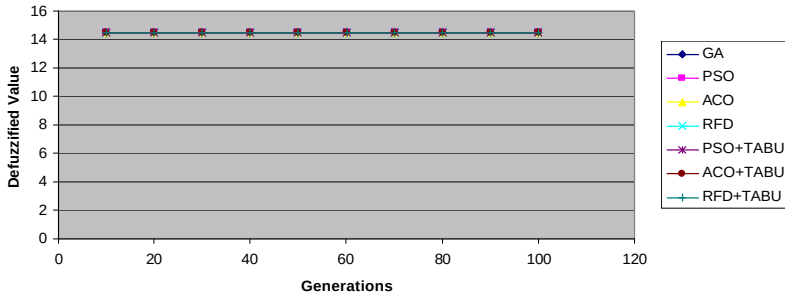

Fig. 25: Database optimization results of independent grain workload (mean slowdown)

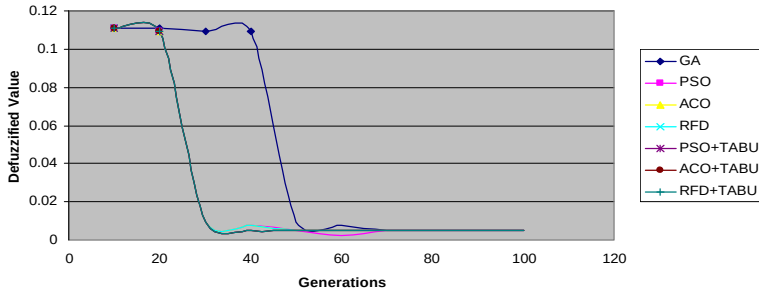

Fig. 26: Database optimization results of independent grain workload (error calculations)

RESULTS AND DISCUSSION

The problem considered here is to optimize the fuzzy controller used for the parallel job scheduling using genetic algorithm, particle swarm optimization, ant colony optimization, and river formation dynamics. Hybrid approaches of the above mentioned algorithms with the tabu search are also implemented and analyzed. This analysis is done to tune the fuzzy logic controller which is used as a learning step for a parallel job scheduling. Figure 2, 4, 6 and 8 represents the database optimization of the fine grain workloads for the scheduling metrics (average waiting time, turnaround time), (mean response time ,mean reaction time), mean slowdown and mean utilization. Figure 3, 5, 7, 9 show the error calculations of the database optimization of the fine grain workloads.

Figures 11-14 represent the database optimization of the medium grain workloads for the scheduling metrics (average waiting time, turnaround time), (mean response time ,mean reaction time), mean slowdown and mean utilization and Fig 15-18 shows the error calculations in the database optimization of the medium grain workloads. Figures 19, 20 and 21 represents the database optimization of the coarse grain workloads for the scheduling metrics (average waiting time, turnaround time), (mean response time, mean reaction time), mean slowdown. Figures 23, 24 and 25 represents the database optimization of the independent grain workloads for the scheduling metrics (average waiting time, turnaround time), (mean response time, mean reaction time); mean slowdown and Fig. 26 represent its error calculations.

The developed fuzzy scheduler results are compared to the classical scheduling strategies used for the parallel job scheduling. It is observed from Table 1 that the fuzzy based river formation dynamics produces good results for the medium grain workloads when compared to the other optimization techniques. Table 2 shows the actual scheduling results of the medium grain workload.

From Table 1 and 2, it is determined that the particle swarm optimization and ant colony optimization gives almost similar results in most of the cases with the hybridization of the tabu search. The best fitness values are obtained with the river formation dynamics with the tabu search. Error calculations are high for coarse and independent grain sizes since the defuzzified outputs are very small in value and the fitness value for these gives only a small correction from the predicted value. Thus a complete comparative analysis of the developed algorithms is done for the optimization of the fuzzy logic controller used for the parallel job scheduling. The hybrid approach produces effective improvements over the traditional techniques.

Table 2: Scheduling Results of the Co scheduling algorithm for the medium grain workload

Metrics/scheduling Algorithm	FCFS	Gang	FCS	Agile
Average waiting time (h)	12:32:43	3:08:11	2:30:33	1:47:32
Mean response time (h)	12:46:29	3:11:37	2:33:18	1:49:30
Turn around time (h)	7:11:34	1:42:18	1:00:00	0:01:33
Mean reaction time (h)	12:32:43	3:08:11	2:30:33	1:47:32
Mean utilization	0.5	0.5	0.60	0.70
Mean slowdown	1.83	0.46	0.37	0.26

CONCLUSION

In this study, an attempt is made to improve the performance of the fuzzy logic controller as a learning step for parallel job scheduling by optimizing the knowledge base and the rule base using various optimization techniques. The fuzzy logic controller was used due to its robustness and can be easily modified, but the controller fails to give the correct defuzzification results, thus tuning the parameter of the fuzzy controller are carried out in this paper. The proposed optimization techniques employ binary coded chromosomes for the database and the rule base optimization. The optimization of the data base and the rule base is made with the optimization algorithms like genetic algorithm, ant colony optimization, particle swarm optimization, river formation dynamics and the hybrid approaches of PSO, ACO and RFD with the tabu search. The results show that the hybrid approach of the river formation dynamics with the tabu search produces better results with that of the other proposed methodologies.

REFERENCES

Bonarini, A., 1996. Evolutionary Learning of Fuzzy Rules Competition and Cooperation. In: Fuzzy Modelling: Paradigms and Practice, Pedrycz, W. (Ed.). Kluwer Academic Press, Hebei, China, pp: 265-284.

Dutot, P.F., F. Pascual, K. Rzadca and D. Trystram, 2011. Approximation algorithms for the multiorganization scheduling problem. IEEE. Trans. Parallel Distrib. Syst., 22: 1888-1895.

Frachtenberg, E., G. Feitelson, F. Petrini and J. Fernandez, 2005. Adaptive parallel job scheduling with flexible coscheduling. IEEE. Trans. Paral. Distrib. Syst., 16: 1066-1077.

Moratori, P., S. Petrovic and R.J.A. Vazquez, 2010. Fuzzy approaches for robust job shop rescheduling. Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ), July 18-23, 2010, IEEE, New York, USA., ISBN:978-1-4244-6919-2, pp: 1-7.

Smith, S.F., 1980. A learning system based on genetic adaptive algorithms. Ph.D Thesis, Department of Computer Science, University of Pittsburgh, Pennsylvania.

Sun, H., O.Y. Ca and W.J. Hsu, 2011. Effective adaptive scheduling of multiprocessor with stable parallelism feedback. Parall. Distrib. Syst. IEEE. Trans., 22: 594-607.