A Genetic Based Neuro Fuzzy Technique for Process Grain Sized Scheduling of Parallel Jobs

Sadasivam Vijayakumar Sudha and Keppanagowder Thanushkodi
1Department of Information Technology, Kalaignar Karunanidhi Institute of Technology, Anna University of Technology, Coimbatore, 641 402, India, 2Akshaya College of Engineering, Anna University of Technology, Coimbatore, India

Abstract: Problem statement: In this study, we present the development of genetic algorithm based neuro fuzzy technique for process grain sized in scheduling of parallel jobs with the help of real life workload data. Approach: The study uses the rule based scheduling strategy for the scheduling and classifies all possible scheduling strategies. The rule bases are developed with the help of the neuro fuzzy system and with the genetic fuzzy system. From the comparison of the two classifiers of the fuzzy systems, it is found that the neuro fuzzy system results higher error rate when compared to the genetic fuzzy system. Hence the study concentrates on reducing the error rate of the results of the neuro fuzzy system by using the genetic algorithm for improving the parameters to the neuro fuzzy system. Results: The study shows that improving parameter like weights in the layers of the neuro fuzzy system using genetic algorithm reduces the error rate and comparative results of the neuro fuzzy, genetic fuzzy and the genetic based neuro fuzzy technique are shown for the parallel job scheduling. Conclusion: The study confirmed that the Genetic Based Neuro Fuzzy Technique can be used as a better optimization tool for optimizing any scheduling algorithm and this optimization tool is used in this study for agile algorithm which is used for process grain scheduling of parallel jobs.

Key words: Parallel system, agile algorithm, genetic algorithm, mean response time, mean reaction time, neural network, neuro fuzzy technique, genetic fuzzy system, scheduling algorithm, optimization tool

INTRODUCTION

Scheduling is a key concept of computer multitasking and multiprocessing operating system design. It refers to the way that the processes are assigned priorities in a priority queue and this assignment is carried out by software known as scheduler. Various algorithms were available to schedule the related threads or processes to run simultaneously on different processors. In parallel job scheduling, scheduling the parallel jobs for execution needs a certain number of processors for a certain time and the schedule have to pack the jobs together. In parallel job scheduling, the grain size based scheduling plays an important role and the synchronization overhead also affects the performance of the parallel system. A good way of characterizing a parallel job scheduling is to consider the synchronization granularity between the processes in a system. The term grain size refers to the size of an independent set of calculations within a parallel algorithm. If the set of processes are dependent with each other, we say that the process grain is fine grain sized jobs and these jobs cannot be schedules separately. If the processes are slightly dependent with each other, we say medium grain sized scheduling. If the processes are not related with each other and then the scheduling is an independent grain size. When scheduling the parallel jobs in a parallel system, we need to consider all the grain sizes. The agile algorithm concentrates on the detailed scheduling of the parallel jobs by classifying the grain size in detail. The agile algorithm compared with the traditional algorithms like first come first served, gang scheduling and flexible co scheduling (Minh and Wolters, 2010).

The agile algorithm (Sudha and Thanushkodi, 2008) is optimized using the two classifiers. The two classifiers are genetic fuzzy system and the neuro fuzzy system. Soft computing is an innovative approach to construct computationally intelligent systems. It is now
realized that complex real world problems require intelligent systems that combine knowledge, techniques and methodologies from various sources. Soft computing consists of computing paradigms including neural network, fuzzy sets, approximate reasoning and derivative free optimization methods such as genetic algorithm and simulation annealing. The integration of these methodologies forms the core of the soft computing era fuzzy technique gives better optimization results for the agile algorithm (Dutot et al., 2011).

MATERIALS AND METHODS

Fuzzy system: Fuzzy theory is the mathematical tool to express and interpret the linguistic character of the natural language. The use of natural language in the expression of a knowledge form known as rule based systems. A fuzzy variable should be identified among the variables of the problem such that there is significantly uncertainty involved in characterizing its behavior. Using a fuzzy variable yields a realistic representation of the problem, than using the crisp counterpart. The solution of the problem using fuzzy variable becomes most practical and cost effective compared to the solution using a crisp variable (Jintao et al., 2010).

Scheduling strategy based on genetic based neuro fuzzy technique: The fuzzy approach uses the six feature definitions as objective functions. A complete rule base is developed using the features already defined and the different classes are defined for the scheduling situation classes. Each rule contains a conditional and a consequence part. The conditional part describes the features for firing the rule and the consequence part describes the scheduling state. In order to specify all scheduling states in an appropriate fashion, each rule defines certain partitions of the feature space within the conditional part. Our neuro fuzzy system uses the mamdani’s fuzzy model. Each feature in the rule is modeled from a Gaussian membership function. The rules used for the optimization are (Sun et al., 2011) and Norozi et al. (2010):

1. If awt is small and tat is small then the Scheduling Algorithm is class A
2. If awt is small and tat is medium then the Scheduling Algorithm is class B
3. If awt is small and tat is large then the Scheduling Algorithm is class C
4. If awt is small and tat is v_large then the Scheduling Algorithm is class D
5. If awt is medium and tat is small then the Scheduling Algorithm is class A
6. If awt is medium and tat is medium then the Scheduling Algorithm is class B
7. If awt is medium and tat is large then the Scheduling Algorithm is class C
8. If awt is medium and tat is v_large then the Scheduling Algorithm is class D
9. If awt is large and tat is small then the Scheduling Algorithm is Class A
10. If awt is large and tat is medium then the scheduling algorithm is class B
11. If awt is large and tat is large then the scheduling algorithm is class C
12. If awt is v_large and tat is small then the scheduling algorithm is class B
13. If awt is v_large and tat is medium then the scheduling algorithm is class B
14. If awt is v_large and tat is large then the scheduling algorithm is class C
15. If awt is v_large and tat is v_large then the scheduling algorithm is class D
16. If awt is large and tat is v_large then the scheduling algorithm is class D
17. If mrt is small and mret is small then the Scheduling Algorithm is class A
18. If mrt is small and mret is medium then the Scheduling Algorithm is class B
19. If mrt is small and mret is large then the Scheduling Algorithm is class C
20. If mrt is small and mret is v_large then the Scheduling Algorithm is class D
21. If mrt is medium and mret is small then the Scheduling Algorithm is class A
22. If mrt is medium and mret is medium then the Scheduling Algorithm is class B
23. If mrt is medium and mret is large then the Scheduling Algorithm is class C
24. If mrt is medium and mret is v_large then the Scheduling Algorithm is class D
25. If mrt is large and mret is small then the Scheduling Algorithm is class A
26. If mrt is large and mret is medium then the Scheduling Algorithm is class B
27. If mrt is large and mret is large then the Scheduling Algorithm is class C
28. If mrt is large and mret is v_large then the Scheduling Algorithm is class D
29. If mrt is v_large and mret is small then the Scheduling Algorithm is class D
30. If mrt is v_large and mret is medium then the Scheduling Algorithm is class D
31. If mrt is v_large and mret is large then the Scheduling Algorithm is class D
32. If mrt is v_large and mret is v_large then the Scheduling Algorithm is class D
33. If msl is small and mu is small then the scheduling algorithm is class A
34. If msl is small and mu is medium then the scheduling algorithm is class A
35. If msl is small and mu is large then the scheduling algorithm is class A
36. If msl is small and mu is v_large then the scheduling algorithm is class A
37. If msl is medium and mu is small then the scheduling algorithm is class B
38. If msl is medium and mu is medium then the scheduling algorithm is class B
39. If msl is medium and mu is large then the scheduling algorithm is class B
40. If msl is medium and mu is v_large then the scheduling algorithm is class B
41. If msl is large and mu is small then the scheduling algorithm is class C
42. If msl is large and mu is medium then the scheduling algorithm is class C
43. If msl is large and mu is large then the scheduling algorithm is class C
44. If msl is large and mu is v_large then the scheduling algorithm is class C
45. If msl is v_large and mu is small then the scheduling algorithm is class D
46. If msl is v_large and mu is medium then the scheduling algorithm is class D
47. If msl is v_large and mu is large then the scheduling algorithm is class D
48. If msl is v_large and mu is v_large then the scheduling algorithm is class D

Where awt is the average waiting time, tat is the turn around time, mrt is the mean response time, mret is the mean reaction time, msl is the mean slowdown and mu is the mean utilization. The scheduling classes are Class A is the Agile Algorithm, Class B is the Flexible co scheduling, Class C is the Gang Scheduling and the Class D is the First Come First Serve scheduling. Totally 48 rules are used for the optimization. We can also extend the rules to 108,192 for even better results.

**Feed forward back propagation network:** Back Propagation neural network is a multilayer feed forward network using a rule based back propagation of error rule. Back propagation provides a computationally efficient method for changing the weights in a feed forward network with differentiable activation function units to learn a training set of input-output pairs. The training algorithm of back propagation involves four stages:

- Initialization of weights
- Feed forward
- Back propagation of errors
- Updating of the weights and trains

During the feed forward stage, each input receives an input signal and transmit this signal to each of the hidden units. Each hidden units then calculates the activation function and sends its signal to each output unit. The output unit calculates the activation function to form the response of the network for the given input pattern.

**Algorithm:**

Step 1: Initialize the weights to small random values.
Step 2: Perform step 2-4 for each input vector.
Step 3: Set the activation of input unit for xi (i=1 to n)
Step 4: Calculate the net input to hidden unit and its output:

\[
Z_{in} = \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i + x_{ji})
\]

\[
Z_h = f(Z_{in})
\]

Step 5: Now compute the output:

\[
Y_{ino} = W_1 + \sum_{i=1}^{n} (Z_h)
\]

\[
Y_0 = f(Y_{ino})
\]

Where:

- x = Input training vector. The various parameters of x are turnaround time, mean response time, mean reaction time, mean slowdown, average waiting time and mean utilization
- Zinph = Sum of all the inputs from the input layer
- Zh = Input to the hidden layer
- Yino = Sum of all the inputs to the output layer from the hidden layer
- Yo = Input to the output layer
- w0 = Bias on the hidden unit
- w1 = Bias on the output unit

The Eq. 1-4 is the activation functions.
Tuning of neuro fuzzy system using genetic algorithm: The back propagation feed forward network uses the fuzzy logic rules whose scheduling parameters and the consequents are the fuzzy singletons which represents the class of the scheduling algorithm. In order to obtain a standard output signal, the trapezoidal function is used to force the neural network to be within the range of (0, 1). GAs is global optimization technique that simulates the procedures of natural evolution, which ensures the quality solution via a systematic information exchange that depends on a probabilistic decision. GAs are very robust due to the global searching and the neural network can be combined with GA for good optimization. We can see this type of optimization in the following two phases

Phase 1:

Step 1: Fuzzy logic rules for the various scheduling parameters are done
Step 2: The Population is represented as a real number string, rather than a binary string. The population involves the weight parameters of the neural network
Step 3: Initial population for the string is generated
Step 4: Feed Forward calculation of the neural network is done
Step 5: Calculate the fitness function for each chromosome. The fitness function for the GA is defined as the average squared system error of the corresponding NFN (Ghedjati, 2010)
Step 6: Continue the cycle initiated in step 5 until convergence is achieved (Moratori et al., 2010)

Phase 2:

Step 1: Initialize the weights using the chromosome with the smallest fitness function value
Step 2: Training data are inputs to the neural network.
Step 3: The feed forward calculation of the fuzzy neural network is done
Step 4: Tuning of the real number (i.e.,) weight of the network is performed by using the gradient learning
Step 5: The objective function or the inference error is calculated and the steps 3 and 4 are repeated until the error is less than a desired threshold value

In phase I, GA is used to learn the algorithm process. The GA performs a global search and seeks an optimal point for the second stage. In this process, each chromosome returned will be considered as the initial weights of the fuzzy neural network in the second stage. Next the gradient algorithm performs the learning process until the terminal condition is satisfied.

RESULTS

The following tables and the figures show the results of the scheduling results of the various algorithms and the results with the optimization ones. Tables 1-4 shows the results of the running time of the calculations of the objective functions using the traditional algorithms and the Agile Algorithm. The same is optimized using the Neuro Fuzzy, Genetic Fuzzy and Genetic based Neuro Fuzzy systems. The Table 1-4 also shows the % of error calculations in the optimizations. The genetic Fuzzy based optimization shows the better results when compared to the other optimizations. Table 5-8 shows the results of the objective values for the four workloads. We have considered four objective functions to measure the result of the optimization. The Table 5-8 shows that the genetic fuzzy system gives better results when compared to the neuro fuzzy system and the neuro fuzzy system results are improved by the genetic based optimization. Figure 1-8 shows the error calculations of the workloads.
Fig. 3: Error in running time of the coarse grain workloads

Fig. 4: Error in running time of the independent grain workloads

Fig. 5: Error in the objective value of the fine grain workloads

Fig. 6: Error in the objective value of the medium grain workloads

Fig. 7: Error in the objective value of the coarse grain workloads

Fig. 8: Error in the objective value of the independent grain workloads

Table 1: Running time results for the scheduling of the fine grain workloads

| Algo/schpar | FCFS | GANG | FCS | AGILE | NF | GF | GN | NF (error %) | GF (error %) | GNF (error %) |
|------------|------|------|-----|-------|----|----|----|-------------|-------------|--------------|
| AWT        | 53002| 13251| 663 | 221   | 222,6590| 222,431| 222,5980| 0.751 | 0.648 | 0.723 |
| TAT        | 17110| 4278 | 214 | 71    | 72,6588 | 72,431 | 72,5979 | 2.336 | 2.015 | 2.251 |
| MRESP      | 53794| 13448| 672 | 224   | 225,6590| 225,418| 225,5980| 0.741 | 0.633 | 0.713 |
| MREAT      | 53002| 13251| 663 | 221   | 222,6590| 222,418| 222,5980| 0.751 | 0.642 | 0.723 |

Table 2: Running time Results for the Scheduling of the medium grain workloads

| Algo/schpar | FCFS | GANG | FCS | AGILE | NF | GF | GN | (%) of error in NF | (%) of error in GF | (%) of error in GNF |
|------------|------|------|-----|-------|----|----|----|-------------------|-------------------|-------------------|
| AWT        | 45163| 11291| 9033| 6452  | 6768.60 | 6468.81 | 6468.81 | 4.907 | 0.261 | 0.261 |
| TAT        | 25894| 6138 | 3600| 93    | 409.60  | 109.81  | 109.81  | 340.430 | 18.075 | 18.075 |
| MRESP      | 45989| 11947| 9198| 6570  | 6627.52 | 6573.35 | 6573.35 | 0.875 | 0.051 | 0.051 |
| MREAT      | 45163| 11291| 9033| 6452  | 6509.52 | 6455.35 | 6455.35 | 0.892 | 0.052 | 0.052 |
### Table 3: Running time results of the coarse grain workloads

| Alg/ | FCFS | GANG | FCSC | AGILE | NF  | GF  | GNF  | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|------|------|------|------|-------|-----|-----|------|-------------------|-------------------|-------------------|
| AWT  | 6    | 0    | 0    | 0     | 1.66| 1.43| 1.60 | 1.66              | 1.66              | 1.43              |
| TAT  | 56590| 801  | 174  | 16    | 17.66| 17.43| 17.60 | 10.37             | 8.96              | 9.99              |
| MRESP| 23   | 0    | 0    | 0     | 1.66| 1.44| 1.60 | 1.66              | 1.44              | 1.60              |
| MREAT| 6    | 0    | 0    | 0     | 1.66| 1.44| 1.60 | 1.66              | 1.44              | 1.60              |

### Table 4: Running time results of the independent workloads

| Algo/ | schpar | FCFS | GANG | FCSC | AGILE | NF  | GF  | GNF  | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|-------|--------|------|------|------|-------|-----|-----|------|-------------------|-------------------|-------------------|
| AWT   | 2936   | 2110 | 2110 | 2110 | 2111.4936 | 2111.431 | 2111.513 | 0.071 | 0.068 | 0.072 |
| TAT   | 31660  | 16810| 16810| 16810| 6740.0 | 6710.00 | 6711.000 | 0.458 | 0.486 | 0.2972 |
| MRESP | 5752   | 3350 | 3350 | 3350 | 3351.5646 | 3351.477 | 3351.5095 | 0.047 | 0.044 | 0.048 |
| MREAT | 4120   | 3380 | 3380 | 3380 | 3381.5646 | 3381.477 | 3381.5095 | 0.046 | 0.044 | 0.047 |

### Table 5: Optimization results of the fine grain workloads

| Objective | Objective values | Best value (NF) | Best value (GF) | Best value (GNF) | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|-----------|------------------|-----------------|-----------------|------------------|-------------------|-------------------|-------------------|
| AWT, TAT  | 18920.00         | 19000.00        | 18915.00        | 105.4795         | 1.3699            | 6.1644            |
| MRESP, MREAT | 6730.00         | 6700.00         | 6650.00         | 34.8315          | 0.2247            | 10.1124           |
| Mean util | 0.6              | 0.61            | 0.6             | 0.605            | 1.4286            | 0.0000            |
| Mean slow | 49.2             | 50.00           | 49.0            | 49.000           | 1.6260            | 0.4065            |

### Table 6: Optimization Results of the medium grain workloads

| Objective | Objective values | Best value (NF) | Best value (GF) | Best value (GNF) | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|-----------|------------------|-----------------|-----------------|------------------|-------------------|-------------------|-------------------|
| AWT, TAT  | 292.000          | 600.00          | 296.000         | 310.00           | 105.4795          | 6.1644            |
| MRESP, MREAT | 445.000         | 600.00         | 446.000         | 400.00           | 34.8315          | 0.2247            |
| Mean util | 0.700            | 0.71            | 0.70            | 0.70             | 1.4286            | 0.0000            |
| Mean slow | 0.013            | 0.100           | 0.014           | 0.01             | 669.2308         | 7.6923            |

### Table 7: Optimization results of coarse grain workloads

| Objective | Objective values | Best value (NF) | Best value (GF) | Best value (GNF) | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|-----------|------------------|-----------------|-----------------|------------------|-------------------|-------------------|-------------------|
| AWT, TAT  | 6545.000         | 7000.00         | 6547.00         | 6650.00          | 6.952             | 0.311             |
| MRESP, MREAT | 13022.00         | 14000.00        | 13021.00        | 13000.00         | 7.510             | 0.008             |
| Mean util | 0.700            | 0.71            | 0.70            | 0.70             | 1.429             | 0.000             |
| Mean slow | 0.26             | 0.36            | 0.27            | 0.28             | 38.462            | 3.846             |

### Table 8: Optimization results of independent grain workloads

| Objective | Objective values | Best value (NF) | Best value (GF) | Best value (GNF) | (% of error in NF) | (% of error in GF) | (% of error in GNF) |
|-----------|------------------|-----------------|-----------------|------------------|-------------------|-------------------|-------------------|
| AWT, TAT  | 16.000000        | 20.00           | 17.000000       | 17.000000        | 25.000            | 6.250             |
| MRESP, MREAT | 0.000000         | 1.00            | 1.000000        | 0.1000           | 100.000           | 100.000           |
| Mean util | 0.600000         | 0.61            | 0.600000        | 0.610            | 1.667             | 0.000             |
| Mean slow | 0.001036         | 0.10            | 0.00104         | 0.001            | 9552.510          | 0.386             |

**DISCUSSION**

The study analyzed parallel job scheduling algorithms in detail and new scheduling algorithm called agile algorithm was discussed. The agile algorithm was compared with the traditional algorithms like first come first served, gang scheduling, flexible co scheduling with the help of six performance metrics like mean response time, mean reaction time, mean slowdown, turn around time, average waiting time and mean utilization. The study concentrates on the optimized technique like neuro fuzzy and genetic fuzzy for the agile algorithm. The fuzzy rules were framed based on the results of the agile algorithm. The optimized results were compared with the agile algorithm and the results shows that the genetic fuzzy technique gives better results than the neuro fuzzy. The study concentrates on improving the values of the neuro fuzzy optimized values from genetic algorithm approach. The genetic algorithm is used to improve the weights of the neural network systems used for the optimization.
CONCLUSION

The agile algorithm is optimized using two fuzzy classifiers, genetic fuzzy systems and neuro fuzzy systems. Genetic fuzzy algorithm is proved to perform better when compared to the neuro fuzzy algorithm. Fuzzy system provides robust inference mechanisms with no learning and adaptability, while genetic algorithms provide an efficient data modification in optimizing the objectives functions of the given application. Neuro fuzzy algorithm is superior as it inherits adaptability and learning, but suffers in terms of lacking of optimal nature. From the results, it has been shown that the error percentage is least in genetic fuzzy algorithms than the neuro fuzzy algorithms for running time results and the objective value analysis. The neuro fuzzy system is improved by means of the genetic system. The results show that the genetic based neuro fuzzy system produced better results than the optimization in the neuro fuzzy system. The study shows all the comparative results of the traditional algorithms and the optimization results of the classifiers.

REFERENCES

Dutot, P., F. Pascual, K. RZadca and D. Trystram, 2011. Approximation Algorithms for the multi organization scheduling problem. IEEE Trans. Parall. Distrib. Syst., 99: 1-1. DOI: 10.1109/TPDS.2011.47

Ghedjati, F., 2010. Heuristic and a hybrid meta-heuristic for a generalized job shop scheduling problem, evolutionary computing. Proceedings of the IEEE Conference Evolutionary Computation, July 18-23, IEEE Xplore Press, Barcelona, pp: 1-8. DOI: 10.1109/CEC.2010.5586004

Jintao, M., Y. Jun and L. Xiaoxu, 2010. Parallel-batching scheduling problem with family jobs for minimizing makespan. Proceedings of the International Conference on Industrial and Information systems, July 10-11, IEEE Xplore Press, Dalian, pp: 159-162. DOI: 10.1109/INDUSIS.2010.5565887

Minh, T.N. and L. Wolters, 2010. Using historical data to predict application runtimes on backfilling parallel systems. Proceeding of 18th Euromicro International Conference on Parallel, Distributed and Network based Processing, Feb. 17-19, IEEE Xplore Press, Pisa, pp: 246-252. DOI: 10.1109/PDP.2010.18

Moratori, P., S. Petrovic. And J.A. Vazquez-Rodriguez, 2010. Fuzzy approach for robust job shop rescheduling. Proceedings of the IEEE International Conference Fuzzy Systems, July 18-23, IEEE Xplore Press, Barcelona, pp: 1-7. DOI: 10.1109/FUZZY.2010.5584722

Sudha, S.V. and K. Thanushkodi, 2008. An approach for parallel job scheduling using nimble algorithm. Proceeding of the International Conference on Computing, Communication and Networking, Dec. 18-20, IEEE Xplore Press, St. Thomas, VI, pp: 1-9. DOI: 10.1109/ICCCNET.2008.4787750

Sun, H., Y. Cao and W.J. Hsu, 2011. Effective adaptive scheduling of multiprocessor with stable parallelism feedback. Trans. Parall. Distrib. Syst., 22: 594-607. DOI: 10.1109/TPDS.2010.121

Norozi, A., M.K.A. ArIffin and N. Ismail, 2010, Application of intelligence based genetic algorithm for job sequencing problem on parallel mixed-model assembly line. Am. J. Eng. Applied Sci., 3: 15-24. DOI: 10.3844/AJEASSP.2010.15.24