Process Grain Sized Based Scheduling of Parallel Jobs using Genetic Fuzzy Systems

S.V.Sudha and K.Thanushkodi

Abstract—**Fuzzy Systems have shown their utility for solving a wide range of problems in different application domains. The use of Genetic Algorithms for designing fuzzy systems allows us to introduce the learning and adaptation capabilities. This has attracted considerable attention in parallel job scheduling. In this paper, we present a methodology for automatically generating online scheduling strategies for different types of process granularity in parallel job scheduling. The scheduling problem includes all the synchronization granularity of parallel jobs.In order to allow a wide range of objective functions, we use a rule based scheduling strategy. The rule system classifies all possible scheduling strategies for the process grains and assigns an appropriate scheduling strategy based on the process grains. The rule bases are developed with the help of a genetic fuzzy system that uses workloads from Logs of real parallel workloads from production systems. http://www.cs.hiji.ac.il/labs/parallel/workload/logs.html. We have already developed a new scheduling algorithm called Agile Algorithm which schedules the jobs according to the synchronization granularity and this paper focuses on the good optimized results for the Agile Algorithm using Genetic Fuzzy System.**

Index Terms—**Genetic Algorithm, Fuzzy Systems, Parallel Jobs, Performance metrics..**

I. Introduction

Computational Intelligence Techniques such as Artificial Neural Networks, fuzzy logic and genetic algorithms are popular research subjects ,since they can deal with complex Engineering Problems which are difficult to solve by classical methods.

Hybrid Approaches have attracted considerable attention in the computational intelligence community .One of the most popular approaches in the hybridization between fuzzy logic and GAs leading to genetic fuzzy systems (GFSs).A GFSs is basically a fuzzy system augmented by a learning process based on GA.GAs are search algorithms based on natural genetics that provide robust search capabilities in complex spaces and thereby offer a valid approach to problems requiring efficient and effective search processes.Fuzzy Systems are one of the most important areas for the application of the fuzzy set theory. Fuzzy Systems have been successfully applied to solve different kinds of problems in various applications domains. The Genetic fuzzy systems are genetic fuzzy rule based systems whose genetic process learns or tunes different components of a fuzzy rule based systems. Within GFRBSs, it is possible to distinguish between either parameter optimization or rule generation processes (ie) adaptation and learning. In this paper, we address the development of a methodology to automatically generate scheduling strategies for parallel jobs under the consideration of process grain size synchronization .The scheduling problem consists of different process grain jobs that are submitted to the machines over time. The scheduling strategy needs to incorporate the various grains criteria during the scheduling process. A new Scheduling Algorithm Agile Algorithm was already implemented and executed with the various grain workloads and the algorithm was compared with the First Come First Served, Gang Scheduling, and Flexible Co scheduling. The new algorithm with the performance metrics like turnaround time, mean response time ,mean reaction time ,mean slowdown ,average waiting time and utilization. The development of scheduling strategies of the parallel job system is based on workload traces originating from real workload achieve. With such a complex scheduling objective ,the paper focuses on a methodology to automatically generate a rule based scheduling system that is able to produce good quality schedules with respect to a given complex provider objective. The whole rule base is only evaluated after the complete scheduling of all jobs belonging to a workload trace. This has a significant influence on the learning method which generate rule base as this typr of evaluation prevents the application of a supervised learning algorithm.[1][3]

II. Evaluation Methodology

A. *Scheduling Algorithm*

Scheduling parallel jobs for execution is similar to 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 could turn to be key issue for utilizations of the processors .Such Scheduling is typically done by partitioning the machine's processors and running a job on each partition. The various scheduling algorithm are [2][6]

1) *First Come First Served*

Load Sharing is perhaps the simplest approach and the one that carries over most directly from a uniprocessor environment. First Come First Served is a version of load sharing. When a job arrives ,each of its thread is placed

Manuscript received October 9, 2009.

S.V.Sudha ,working as Assistant Professor in the Department of Information Technology,Kalignar Karunanidhi Institute of Technology ,Coimbatore 641402 ,Tamil Nadu ,India (e-mail: svsudha@rediffmail.com)

K.Thanushkodi ,Principal ,of Akshaya College of Engineering and Technology, Coimbatore -642 109,Tamil Nadu,India

consecutively at the end of the shared queue. When a processor becomes idle, it picks the next ready thread ,which it executes until completion or blocking.[4][3]

The major disadvantage of First Come First Served is the central queue occupies a region of memory that must be accessed in a manner that enforces mutual exclusion. Thus ,it may become a bottleneck if many processors look for work at the same time .When there is only a small number of processors ,this is unlikely to be a noticeable problem .However, when the multiprocessor consists of dozens or even hundreds of processors ,the potential for bottleneck is real.[9][10]

2) Gang Scheduling

A set of related threads is scheduled to run on a set of processors at the same time on a one to one basis. The concept of scheduling a set of processes simultaneously on a set of processors predates the use of threads, which is refer to group scheduling.[7][11]

Gang scheduling is for medium grain to fine grain parallel applications whose performance severely degrades when any part of the application is not running while other parts are ready to run.

One obvious way in which gang scheduling improves the performance of a single application is that process switches are minimized. If closely related processes execute in parallel, synchronization blocking may be reduced, less process switching may be necessary and performance will increase.

Scheduling overhead may be reduced because a single decision affects a number of processors and processes

at one time. Gang scheduling is used so that if two threads or processes communicate with each other, they will all be ready to communicate at the same time.[5][8]

If they were not gang schedules, then one could wait to send or receive a message to another while it is sleeping and vice versa. If gang scheduling is not used within a group of processes or threads which communicate with each other, it can lead to starvation. Gang Scheduling ensures that constructing a static global list of the order in which jobs should be scheduled coordinates the scheduling of communicating jobs. Gang scheduling requires the schedule of communicating processes be precomputed which complicates the co scheduling of client server applications and require pessimistic assumptions about which processes communicate with one another.

3) Flexible Co scheduling

Flexible co scheduling is used to improve overall system performance in the presence of heterogeneous hardware or software by using dynamic measurement of applications, communication patterns and classification of application.[9]

FCS employs dynamic process classification and schedules processes using this class information .Processes are categorized as

CS –CS processes should be co schedules and should not be preempted.

F –F processes should be co scheduled but can be preempted when synchronization is not effective.

DC –D processes impose no restrictions on scheduling

4) Agile Algorithm

The Algorithm concentrates on detailed classification of the frequency of synchronization between processes in a system. The processes are classified as [10][11]

Fine Grain

Fine Grained Parallelism represents a much more complex use of parallelism .The processes communicate often and must be co scheduled effectively due to their demanding synchronization.

Medium Grain

Medium Grain Parallelism represents enough synchronization between the processes and the scheduling algorithms should take care of the performance evaluation of the system.

Coarse Grain

With Coarse Grain, there is synchronization among processes, but at a very gross level. This kind of situation is easily handles as a set of concurrent processes running on a multiprogrammed uniprocessor and can be supported on a multiprocessor with little or no change to the software.

Independent

With Independent parallelism, there is no explicit synchronization among processes. Each represents a separate, independent application or job

B. Feature Definition

The synthetic workload generated Feitelson's archive are used as input to the simulation of various scheduling strategies. We monitor the following parameters the arrival time, start time, execution time; finish time etc .Different Scheduling algorithms have different properties and may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for scheduling algorithms. The criteria includes the following

Mean Utilization:

We want to keep the CPU as busy as possible. CPU Utilization may range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily loaded system). The mean utilization is the ratio of CPU busy time to the number of processors multiplied with Total time for execution.

$$\text{Mean Utilization} = \frac{\Sigma\ \text{CPU Busy Time}}{\text{Number of Processors} * \text{Total Time}}$$

Mean Response Time

Another measure is the time from the submission of a request until the first response is produced. This measure is called response time

$$\text{Mean Response Time} = \frac{\Sigma\ \text{Job Finish Time} - \text{Job Submit Time}}{\text{Number of Jobs}}$$

Mean Reaction Time

The mean job reaction time defined as the mean time interval between the submission and the start of the job.

$$\text{Mean Reaction Time} = \frac{\Sigma \text{ Job Start Time-Job Submit Time}}{\text{Number of Jobs}}$$

Mean Slowdown

Mean Slowdown is the sum of jobs response times divided by the job's execution times. This metric emerges as a solution to normalize the high variation of the jobs response time.

$$\text{Mean Slow down} = \frac{\Sigma \text{ Job Response Time/ Job Execution}}{\text{Number of Jobs}}$$

Turn Around Time

From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.

Waiting Time

The scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

C. Scheduling Objective

This section introduces several scheduling features and objectives which have been used to construct more complex evaluation function for the whole scheduling procedure. We have given many possible scheduling situations within the rule based scheduling system. The objectives evaluate the whole scheduling process and at the end of a simulation the features describes the current state of the system. Both the objective and the feature set refers to the properties and to the overall performance of the whole system. During the development of the scheduling system and to do comparison of the agile algorithm with that of the traditional one, an evaluation function is needed in order to describe the achieved quality. We have generated four scheduling objectives from the scheduling features and we are in a focus to optimize the Agile algorithm with all features and we have optimized with Genetic Fuzzy system.

The four Objective functions are

1) FObjective 1=AWT+TAT
2) FObjective=MRT+MRET
3) FObjective=MSL
4) FObjective=Mean Utilization

III. Rule Base Scheduling Systems

A rigid rule based scheduling system has the advantage of a simple implementation and easy interpretation. The selected scheduling algorithm for a certain scheduling situation can directly be extracted from the corresponding rules..Rule bases are generated by assigning potential scheduling strategies to rule in a random fashion such that each scheduling strategy is assigned to each rule. The conditional part is rigid and does not vary. A single rule describes a single scheduling situation class complexity. We use those rules bases to produce schedules for the given process grain workload datas and evaluate those scheduling objective. We have used four objective functions ,out of which three of which are for the minimization of the objective function and one for the maximization of the objective function. The performance can be increased by generating more rule bases.

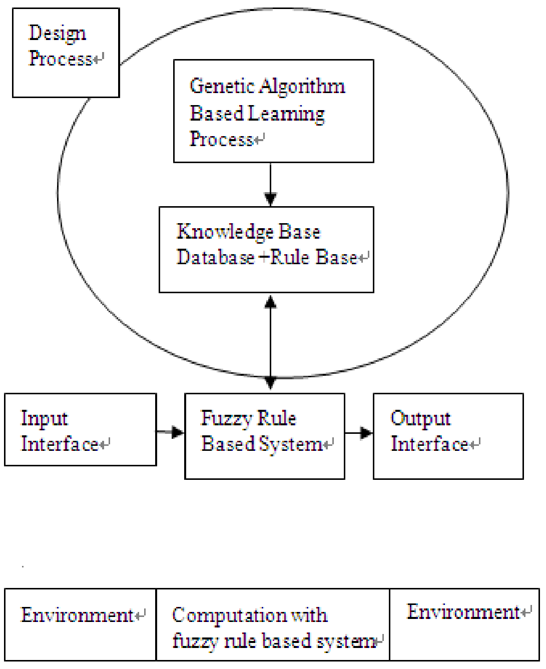
Design Process
Genetic Algorithm Based Learning Process
Knowledge Base Database +Rule Base
Input Interface
Fuzzy Rule Based System
Output Interface
Environment
Computation with fuzzy rule based system
Environment

Fig 1:Genetic Fuzzy System

IV. Evaluation

The Agile Algorithm which was already developed executes and schedules all the parallel jobs according to the grain sizes and corresponding algorithm was executed for the grains.The Agile Algorithm compared with the three other algorithms like FCFS, Gang and FCS. The results are compared with the performance metrics like turnaround time, mean response time, mean reaction time, mean slowdown, utilization and average waiting time. The results are noted and presented in the following tables. The values which are got are analyzed and optimized using the Genetic Fuzzy systems. The results using the Genetic Fuzzy systems are also presented in the tables.

The Following tables present the scheduling results of all the workloads along with the optimization results. Four Objective functions are considered for the analysis. All the analysis is made from the Genetic Fuzzy Systems.

Our Gentic Fuzzy systems are based on the mamdani's model for fuzzy systems.The rule's feature is modeled from aGaussian membership function.Based on the featrure description ,a single rule can be described by

Ri={G1,G2,G3…………………….Gi}

By representing a rule with several GMFs, it will automatically coverage the possible feature space. The scheduling features are turn around time ,mean response time, mean reaction time, mean slowdown, utilization ,average waiting time. We choose four objective functions to choose the scheduling strategy. We represent the results for the grain process .It is observed that the online rule based scheduling system produce ,schedules almost as good as those achieved in the offline case. We have developed 22 rules for each grain and totally 88 rules are developed for a total of four objective functions.

The various Objective functions used are

1) fobj1 =AWT+TAT
2) fobj2=MRT+MRET
3) fobj3=MSL
4) fobj4=MU

The Scheduling algorithm which produces the minimum value for the objective 1,2,3 and maximum value for the objective 4 is selected.

V. Optimization Results

1) Function Objective 1 =AWT+TAT

Where AWT is Average Waiting Time and TAT is Turn Around Time.

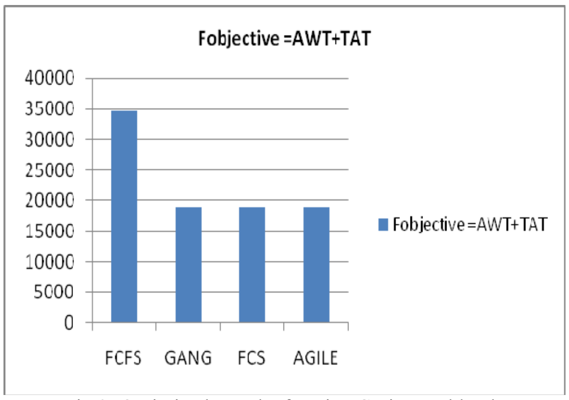

Fig 2: Optimized Results for Fine Grain Workloads

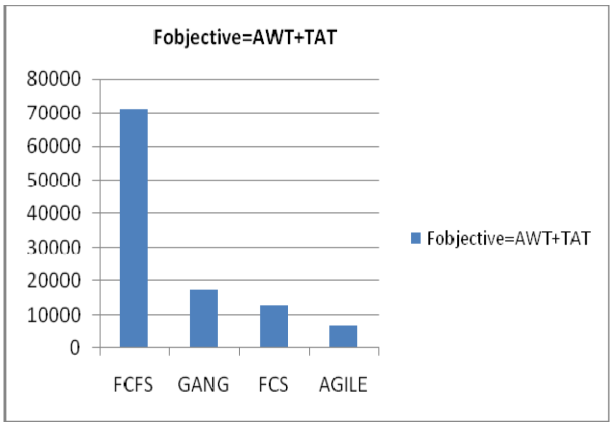

Fig 3: Optimized Results for Medium Grain Workloads

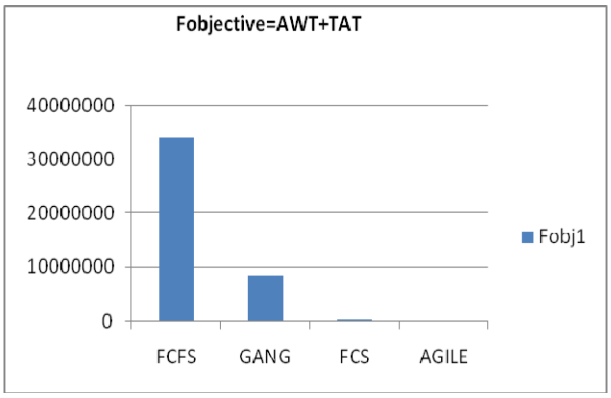

Fig 4: Optimized Results for Coarse Grain Workloads

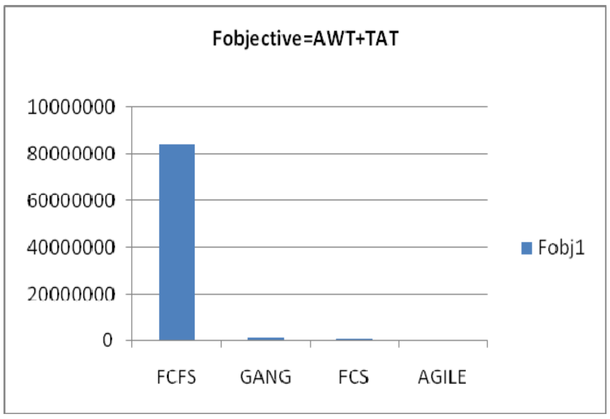

Fig 5: Optimized Results for Independent Workloads

2) Function Objective 2
Fobj2=MRT+MRET

Where MRT is Mean Response Time and MRET is Mean Reaction Time.

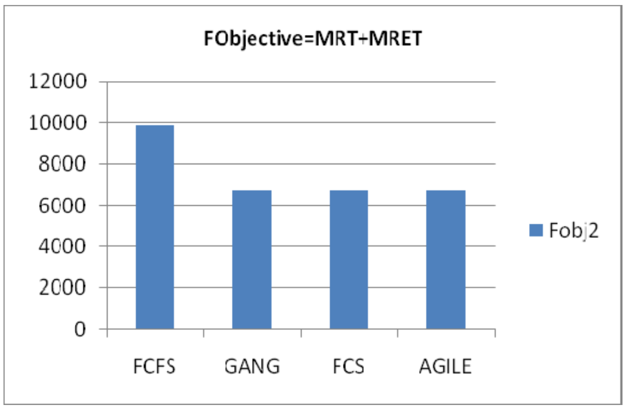

Fig 6: Optimized Results for Fine Grain Workloads

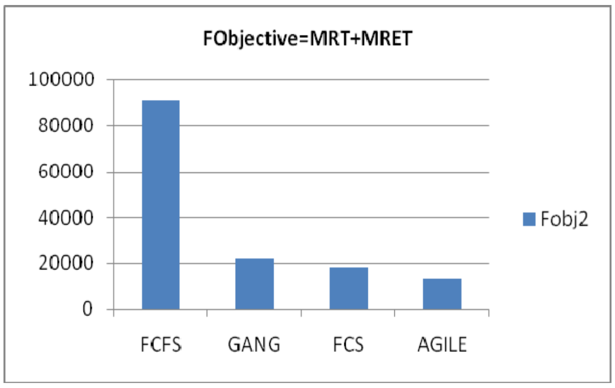

Fig 7: Optimized Results for Medium Grain Workloads

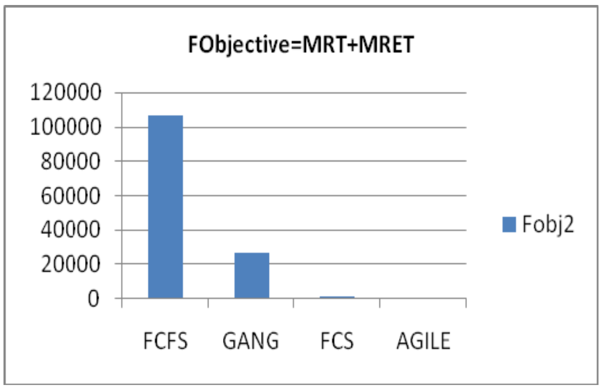

Fig 8: Optimized Results for Coarse Grain Workloads

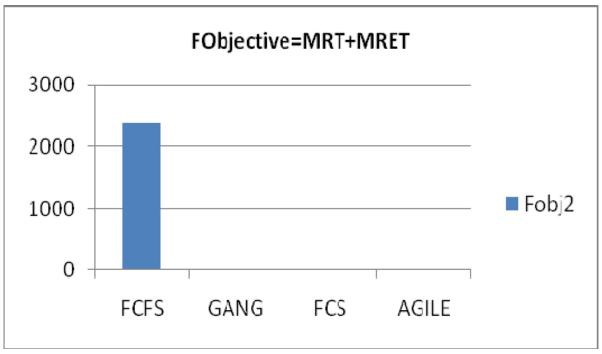

Fig 9: Optimized Results for Independent Workloads

3) Function Objective
Fobj3 =MSL where MSL is Mean Slowdown

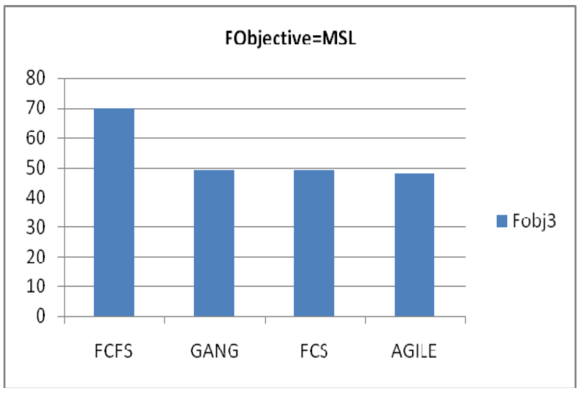

Fig 10:Optimized Results for Fine Grain Workloads

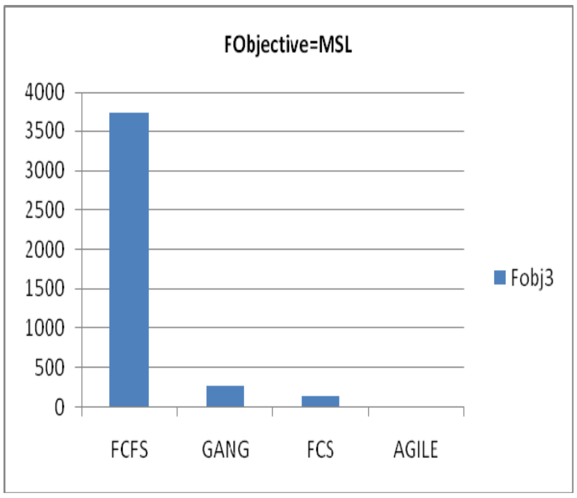

Fig 11: Optimized Results for Medium Grain Workloads

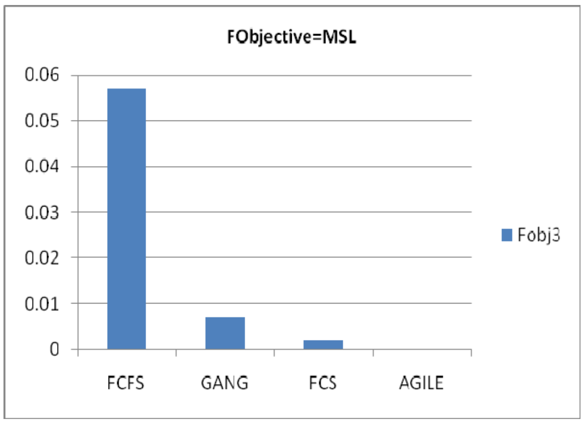

Fig 12: Optimized Results for Coarse Grain Workloads

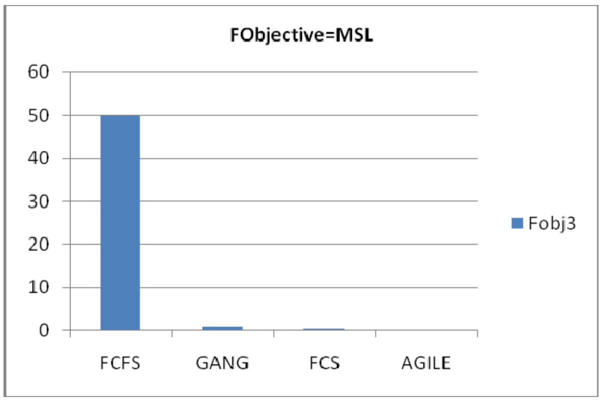

Fig 13: Optimized Results for Independent Workloads

4) Function Objective
Fobj4=Mean Utilization

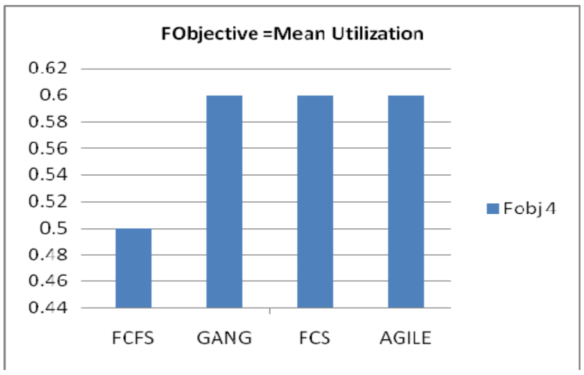

Fig 14:Optimized Results for Fine Grain Workloads

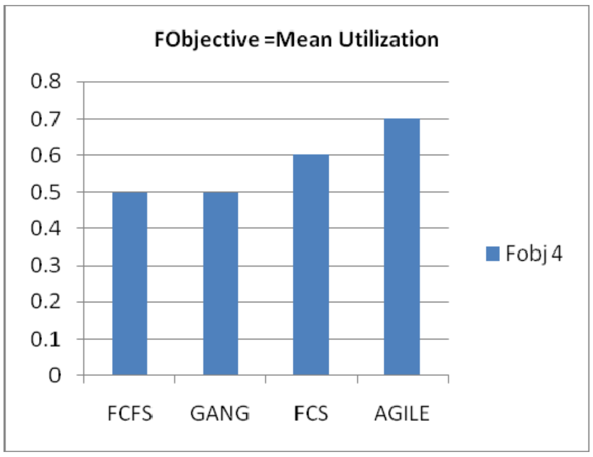

Fig 15.: Optimized Results for Medium Grain Workloads

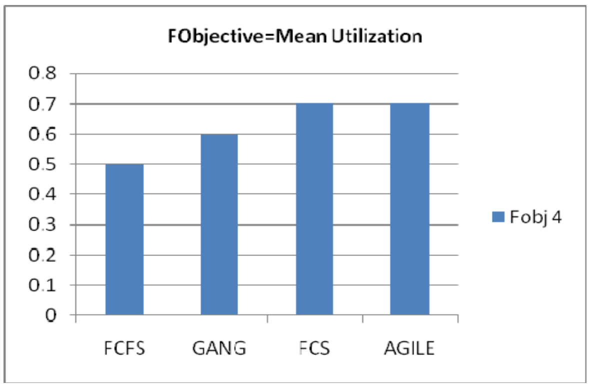

Fig 16. Optimized Results for Coarse Grain Workloads

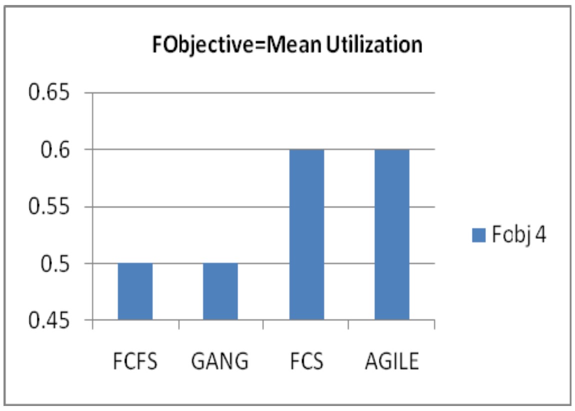

Fig 17. Optimized Results for Independent Workloads

VI. Conclusion

In this paper ,we have evaluated genetic optimization learning paradigms. The genetic fuzzy classification infers fuzzy rules whereby each rule has its own definition of membership functions. It is obseved that Genetic Fuzzy Systems significantly improve the achieved quality of the schedule.

References

[1] Carsten Franke, Joachim Lepping, and Uwe chwiegelshohn, Development of scheduling strategies with Genetic Fuzzy Systems,Volume 8 ,Issue 1 ,pages 706 -721 ,2008
[2] T. B¨ack and H.-P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation, 1(1):1–23, 1993.
[3] H.-G. Beyer and H.-P. Schwefel. Evolution Strategies – A Comprehensive Introduction.Natural Computing, 1(1):3–52, 2002.
[4] A. Bonarini. Evolutionary Learning of Fuzzy rules: competition and cooperation In W. Pedrycz, editor, Fuzzy Modelling: Paradigms and Practice, pages 265–284.Kluwer Academic Press, 1996
[5] . R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. Special Issue onGrid Computing, Proceedings of the IEEE, 93(3):698–714, 2005.
[6] 5. S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger,
U. Schwiegelshohn, W. Smith, and D. Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In D. G. Feitelson and L. Rudolph, editors,Proceedings of the 5th Job Scheduling Strategies for Parallel Processing, volume 1659 of Lecture Notes in Computer Science (LNCS), pages 67–90. Springer, 1999.
[7] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. OnAdvantages of Grid Computing for Parallel Job Scheduling. In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID2002), pages 39–46, Berlin, 2002. IEEE Computer Society Press.
[8] C. Ernemann, V. Hamscher, and R. Yahyapour. Economic Scheduling in Grid Computing. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, Pro-ceedings of the 8th Job Scheduling Strategies for Parallel Processing, volume 2537of Lecture Notes in Computer Science (LNCS), pages 128–152. Springer, 2002.
[9] C. Ernemann, U. Schwiegelshohn, N. Beume, M. Emmerich, and L. Sch¨onemann.Scheduling Algorithm Development based on Complex Owner Defined Objectives.Technical Report CI-190/05, Dortmund University, Germany, 2005.
[10] http://sfbci.uni-dortmund.de/Publications/Reference/Downloads/1900 5.pdf.C. Ernemann, B. Song, and R. Yahyapour. Scaling of Workload Traces. In D. G.
[11] Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, Proceedings of the 9th Job Scheduling Strategies for Parallel Processing, volume 2862 of Lecture Notes in Computer Science (LNCS), pages 166–183. Springer, 2003

TABLE 1:Scheduling Results of Fine Grain Workloads with FCFS,GANG,FCS and Agile Algorithm

Metric /Algorithm	FCFS	GANG	FCS	AGILE
AWT (Time in sec)	2936	2110	2110	2110
TAT (Time in sec)	31660	16810	16810	16810
MRT (Time in sec)	5752	3350	3350	3350
MRET (Time in sec)	4120	3380	3380	3380
MSL (a Value)	71.105	49.2	49.2	49.2
MU (a value)	0.5	0.6	0.6	0.6

TABLE 2:SCHEDULING RESULTS OF MEDIUM GRAIN WORKLOADS WITH FCFS,GANG,FCS AND AGILE ALGORITHM

Metric/ Algorithm	FCFS	GANG	FCS	AGILE
AWT (Time in sec)	45163	11291	9033	6452
TAT (Time in sec)	25894	6138	3600	93
MRT (Time in sec)	45960	10866	9228	6570
MRET (Time in sec)	45120	11280	9033	6452
MSL (a Value)	3750	282	144	13
MU (a value)	0.5	0.5	0.6	0.7

TABLE3: RESULTS OF COARSE GRAIN WORKLOADS WITH FCSFS, GANG, FCS AND AGILE ALGORITHM

Metric/ Algorithm	FCFS	GANG	FCS	AGILE
AWT (Time in sec)	53002	13251	663	221
TAT (Time in sec)	34018381	8504595	425229	141743
MRT (Time in sec)	53794	13448	672	224
MRET (Time in sec)	53002	13251	663	221
MSL (a Value)	0.057	0.007	0.002	0.0002
MU (a value)	0.5	0.6	0.7	0.7

TABLE 4: SCHEDULING RESULTS OF INDEPENDENT WORKLOADS WITH FCSFS, GANG, FCS AND AGILE ALGORITHM

Metric/ Algorithm	FCFS	GANG	FCS	AGILE
AWT (Time in sec)	0	0	0	0
TAT (Time in sec)	84192617	1202752	616043	205347
MRT (Time in sec)	2087	29.821	15.27	5.091
MRET (Time in sec)	314.966	4.499	2.3046	0.77
MSL (a Value)	49.89	0.713	0.365	0.121
MU (a value)	0.5	0.5	0.6	0.6

TABLE 5: OPTIMIZATION RESULTS OF A FINE GRAIN WORKLOADS

Metric /Algorithm	AWT	TAT	MRT	MRET	MSL	MU	Fobj1	Fobj2	Fobj3	Fobj 4
FCFS	45163	25894	45960	45120	3750	0.5	71055	91000	3748	0.5
GANG	11291	6138	10866	11280	282	0.5	17422	22144	280	0.5
FCS	9033	3600	9228	9033	144	0.6	12630	18258	142	0.6
AGILE	6452	93	6570	6452	13	0.7	6543	13026	13.5	0.7

TABLE 6: OPTIMIZATION RESULTS OF A MEDIUM GRAIN WORKLOADS

Metric /Algorithm	AWT	TAT	MRT	MRET	MSL	MU	Fobj1	Fobj2	Fobj3	Fobj 4
FCFS	53002	34018381	53794	53002	0.057	0.5	34071381	106794	0.057	0.5
GANG	13251	8504595	13448	13251	0.007	0.6	8517843	26693	0.007	0.6
FCS	663	425229	672	663	0.002	0.7	425888	1330	0.002	0.7

AGILE	221	141743	224	221	0.0002	0.7	141963	445	0.0002	0.7

TABLE 7 :OPTIMIZATION RESULTS OF A COARSE GRAIN WORKLOADS

Metric /Algorithm	AWT	TAT	MRT	MRET	MSL	MU	Fobj1	Fobj2	Fobj3	Fobj4
FCFS	0	84192617	2087	314.966	49.89	0.5	84192617	2401.8	49.89	0.5
GANG	0	1202752	29.821	4.499	0.713	0.5	1202752	33.921	0.713	0.5
FCS	0	616043	15.27	2.3046	0.365	0.6	616043	17.5704	0.365	0.6
AGILE	0	205347	5.091	0.77	0.121	0.6	205347	5.861	0.121	0.6

TABLE 8:OPTIMIZATION RESULTS OF A INDEPENDENT WORKLOADS.

Metric /Algorithm	AWT	TAT	MRT	MRET	MSL	MU	Fobj1	Fobj2	Fobj3	Fobj 4
FCFS	2936	31660	5752	4120	71.105	0.5	34593	9869	70.1	0.5
GANG	2110	16810	3350	3380	49.2	0.6	18918	6725	49	0.6
FCS	2110	16810	3350	3380	49.2	0.6	18918	6725	49	0.6
AGILE	2110	16810	3350	3380	49.2	0.6	18918	6715	48	0.6