VOL. 1, NO. 2, Oct 2010 E-ISSN 2218-6301
Journal of Emerging Trends in Computing and Information Sciences
©2009-2010 CIS Journal. All rights reserved.
http://www.cisjournal.org

Solving Fuzzy based Job Shop Scheduling Problems using Ga and Aco

Surekha P[1], S.Sumathi[2]

1 Research Scholar, EEE, PSG College of Technology, Coimbatore

2 Asst. Professor, EEE, PSG College of Technology, Coimbatore

Email: surekha_3000@yahoo.com

ABSTRACT

In this paper, we present a genetic algorithm and ant colony optimization algorithm for solving the Job-shop Scheduling Problem (JSSP). The genetic algorithm comprises of different stages like generating the initial population, selecting the individuals for reproduction and reproducing new individuals. Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ants, which is also used to solve this combinatorial optimization problem. In JSSP ants move from one machine (nest) to another machine (food source) depending upon the job flow, thereby optimizing the sequence of jobs. The sequence of jobs is scheduled using Fuzzy logic and optimizes using GA and ACO. The makespan, completion time, makespan efficiency, algorithmic efficiency and the elapsed time for the genetic algorithm and the ant colony algorithm are evaluated and compared. The improvement in the performance of the algorithms based on the computed parameters is also discussed in this paper. Computational results of these optimization algorithms are compared by analyzing the JSSP benchmark instances, FT10 and the ABZ10 problems.

Keywords : *Fuzzy Logic, Planning, Scheduling, Makespan, Genetic Algorithm, Ant Colony Optimization.*

1. INTRODUCTION

Job Shop Scheduling Problem (JSSP) is one of the most difficult combinatorial optimization problem, which is used in complex equipment manufacturing system to validate the performance of heuristic algorithms. The research on JSSP not only promotes the development of relative algorithms in the field of artificial intelligence, but also provides means of solutions and applications for complex JSSP. JSSP can be thought of as the allocation of resources over a specified time to perform a predetermined collection of tasks. Job shop scheduling has received a large amount of attention, because it has the potential to dramatically decrease costs and increase throughput, thereby, increasing the profits in automation industries. Job-shop is a system that processes *n* number of tasks on *m* number of machines. The total ordering defines a set of precedence constraints, meaning that no activity can begin execution until the activity that immediately precedes it in the complete ordering has finished execution. Each of the *m* activities in a single job requires exclusive use of the resources defined in the problem [4]. No activities that require the same resource can overlap in their execution and once an activity is started it must be executed for its entire duration. Usually, these orders differ in terms of processing requirements, materials needed, processing time, processing sequence and setup times. Job-shop problems are widely known as a NP-hard (Non-Polynomial Deterministic) problem and commonly defined as a set of jobs whose operations are to be processed in an uninterrupted manner on a given machine for a specified length of time.

GA is a local search algorithm that follows the evolution paradigm. Starting from an initial population, the algorithm applies genetic operators in order to produce offsprings, which are presumably more fit than their ancestors [7]. Every individual in the population represents a solution at the end of every generation. The major strength of GA when compared with other local search algorithms lies in the fact that in a GA framework more strategies can be adopted together to find individuals to add to the mating pool, both in the initial population phase and in the dynamic generation phase. Such an adaptation allows the search space to be explored at every algorithm step.

The Ant Colony System (ACS) algorithm is a distributed algorithm which is extensively used to solve NP-hard combinatorial optimization problems. Its original model is based on the foraging behavior of real ants who find an approximately shortest way to the food by detecting the density of pheromone deposited on the route [5]. In real ants, the term pheromone denotes the chemical substance deposited by ants as they move, but in artificial ants it acts like an attraction for the other ants to follow. In general, ants create pheromone paths from their nests to the available food sources and the shortest path is the one with highest pheromone concentration between the source and the destination.

In JSSP, the time required for all operations to complete their processes is called makespan. In this paper, the objective is to minimize this makespan value by applying the genetic algorithm and the ACO algorithm. When the makespan is minimized, at least one of the optimal solutions is active so that no operation can be processed without violating the technological constraints. For this reason, every time when makespan is optimized, a schedule can be described by the processing orders of operations on the machines.

Giffler and Thompson [9] proposed algorithms using priority dispatching heuristics to solve production scheduling problems for single and multiple schedules. They demonstrated that every optimal schedule is the same as an active optimal schedule. Brooks and White

©2009-2010 CIS Journal. All rights reserved.

[10] developed a procedure for finding optimal solutions for the production scheduling problem incorporating a wide variety of possible criteria and constraints, using the algorithm developed by Giffler and Thompson. Blackstone [11] provides a comparison of several dispatching rules. Their work illustrated several measurements criteria that were used to evaluate dispatching rules and identified several rules that exhibit good overall performance. Cheng [12] was the first to provide a tutorial survey of recent works on solving classical JSSP using Genetic Algorithms (GA) and addressed the key issue of encoding a GA for solving the JSSP. Carlos A. Coello Coello et al. [1] proposed an ant based system to solve the JSSP and experiments show that the proposed approach can reduce the number of evaluations performed without a degradation of performance. The performance evaluation of optimization techniques are based on computational intelligence paradigms for solving job shop scheduling problem to obtain optimized completion time, to decrease the makespan of job sequence and in turn to increase the feasibility of scheduling process.

The organization of the paper is as follows: Section 2 gives a brief introduction about job shop scheduling problem, and Section 3 describes the planning and scheduling of jobs using fuzzy logic. Section 4 explains the algorithm of JSSP using GA and section 5 delineates the application algorithm of ACO for JSSP. The simulation results are analysed in section 6 and section 7 concludes this paper.

2. JOB SHOP SCHEDULING PROBLEM

Recently, researchers have been focusing on investigating machine scheduling problems in manufacturing and service environments where jobs represent activities and machines represent resources, and each machine can process one job at a time.

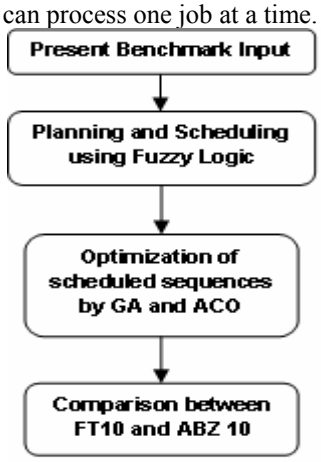

Fig 1 Stages in Solving JSSP

Fig 1 illustrates the stages involved in solving the job shop scheduling process in this paper. Planning of machines and scheduling of jobs to each machine with respect to levels is performed for better optimization results. Fuzzy logic is used to implement the planning and scheduling of jobs. The scheduled job sequences are then optimized using GA and ACO and the best algorithm is determined based on parameters like makespan, mean completion time, mean waiting time, least completion time and algorithmic efficiency.

Job-shop scheduling is an activity that comprises of a set of jobs to be processed on a set of machines. The job-shop scheduling problem can be defined as the allocation of machines over time to perform a collection of minimizing or maximizing a specific performance measure while satisfying the operation precedence constraints, machine capacity constraint, processing time and ready time requirements. The resource used in an industrial environment is known as machines and the basic task modules which performs the operation is called job [14]. Each job may be comprised of several elementary tasks called operations, which are interrelated by precedence constraints. The processing of an operation requires the use of a particular machine for an uninterrupted duration, called the processing time of the operation. Each machine can process only one operation at a time. The routing, processing times and precedence constraints are specified by a process plan. The main distinction between the classic flow-shop and a job-shop is that, in the former case each job passes the machines in the same order whereas in the latter case the machine order may vary per job. Since workflow in a job-shop is not unidirectional, scheduling becomes quite tedious. For a particular job-shop process plan, several feasible schedules can be generated and measured. The processing order on each machine that minimizes the corresponding cost is desired by the objectives such as minimization of process cost, makespan and flow time or maximization of throughput, systems/resource utilization and production rate. The job-shop scheduling problem (JSSP) can be described as a set of n jobs denoted by J_j where j =1, 2... n. It has to be processed on a set of m machines denoted by M_k where k =1, 2… m. Operation of j th job on the k th machine will be denoted by O_{jk} with the processing time p_{jk}. Once a machine starts to process a job, no interruption is allowed. Each job is composed of a sequence of operations and the operation order on the machines is pre-specified. Each operation is characterized by the required machine and the fixed processing time. The process plan specifies the routing, processing times and precedence constraints among operations of each job. There are several constraints on jobs and machines:

- The same machine does not encounter more than one job
- No priority is defined for the operations defined in the job
- No operation can be preempted
- A machine can process only one job at a time

3. PLANNING AND SCHEDULING

The fuzzy parameters used for assigning jobs to machine with respect to their processing time are tabulated in Table 2. Here numbers of input parameters are jobs and levels and output parameter is processing time. Ten jobs and ten machines is taken as test samples and processed for planning and scheduling from the benchmark instances. Based on the processing time fuzzy logic is used to group the crisp values interms of linguistic variables

©2009-2010 CIS Journal. All rights reserved.

like short processing time (SPT), medium processing time (MPT) and long processing time (LPT). A large computation effort may be actually saved by avoiding failures with problems whose lack of feasibility is due to insignificant constraint violations [3]. The computation effort can be actually saved by performing a proper planning strategy.

The procedure for planning using fuzzy logic is shown below:

Step 1*: Assign jobs and levels as the input variables*
Step 2: *Assign the processing time as the output variable*
Step 3: *Formulate the rules based on triangular membership function with the linguistic parameters such as SPT, MPT and LPT.*
Step 4: *Sequence the jobs depending upon the rule base.*
Step 5: *Repeat step 3 and 4 for each level.*
Step 6: *Stop the planning phase*

Table.1. Scheduled Sequences of Jobs

a. Fisher and Thompson 10x10 instance

LEVEL	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
1	J1 J2 J9	J3 J4 J7 J10	J5 J6 J8							
2	J3 J5 J7 J8 J10	J1 J6 J9	J2 J4							
3	J4	J5 J8	J1 J10	J3 J7 J9	J2	J6				
4			J3 J7	J1 J6	J4	J5 J8 J9	J10			J2
5			J9	J2 J5	J1 J8		J4 J7		J3 J6 J10	
6		J2			J5	J1 J3 J7	J8		J4	J6 J9 J10
7	J6					J10	J1 J2 J9	J3 J4	J5 J8	J7
8				J4 J10		J2	J3 J6	J1 J5 J9	J7	J8
9					J6 J9 J10			J2 J7 J8	J1	J3 J4 J5
10				J8	J3 J7	J4	J5	J6 J10	J2 J9	J1

The scheduling of the jobs with respect to machine and level is shown for both benchmark FT10 and ABZ10 in Table 1.a and 1.b. The job is aligned in the order for machines with the data available in the benchmark instances. The time taken by a job to wait in a queue is calculated as waiting time. The jobs aligned to each machines with respect to levels are tabulated to show the job sequence ordering according to the given benchmark instance. The job sequence after scheduling is optimized to rearrange the job sequence so that less waiting time and minimum makespan are obtained. Once the priority rules are evaluated and the jobs are scheduled then the system is prepared for optimizing the job sequence.

b. Adams, Balas, and Zawack 10x10 instance

LEVEL	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
1	J9	J6		J5 J10	J1 J8	J2		J4 J7		J3
2	J10	J7	J4	J2	J5 J6		J8 J9		J1 J3	
3	J3	J4 J10		J8 J9	J7	J6	J1 J2			J5
4		J3	J8	J7	J2 J4	J1	J6	J9	J5 J10	
5	J5 J7	J1	J2 J8 J9	J4			J3	J10	J6	
6			J1 J5 J6 J9			J3	J4 J8	J10	J2 J7	
7		J2	J7 J9			J4	J5 J10	J3 J6 J8		J1
8	J4 J8	J2			J3 J10	J5 J7 J9		J1		J6
9	J1		J3	J6		J10	J7	J2 J5	J8 J9	J4
10	J6	J5	J10	J1 J3					J4	J2 J7 J8 J9

4. GENETIC ALGORITHM FOR JSSP

Job-shop scheduling is one of the most interesting and challenging applications for GA's. The major difference among different genetic algorithm applications is the chromosome representation. A very important issue in building a GA for the job-shop problem is to devise an appropriate representation of solutions together with problem-specific genetic operations. This enables all the chromosomes to be generated, either in the initial phase or during the evolutionary process to produce feasible schedules [14]. GA have been used for a wide variety of problems such as machine learning, cellular manufacturing, and combinatorial optimization, inventory control, traveling salesman and game playing. The application of GA's in scheduling was introduced by Davis and Liepins to investigate the simplest scheduling problem of a static queue of jobs with specified due dates and run times without precedence constraints. Gupta studied a single machine model with an objective to minimize flow time variance. Lee and Kim examine the performance of a parallel GA for a model in which earliness and tardiness penalties are allowed to be arbitrary and jobs share a common due date. Cheng consider a model of identical parallel machines where the objective is to minimize the maximum weighted absolute lateness about an unrestricted common due date [1]. The makespan of the operation is used as fitness value. Makespan is denoted as *C*max is the time when the last operation leaves the workplace which is given by the equation (1).

$$C_{\max} = \max\left(C_1, C_2, \ldots, C_n\right) \quad \text{Where,}$$

$$(C_j) = \sum_{k-1}^{n} (W_{jk} + \rho_{jm(k)}) \qquad (1)$$

C_j is the completion time of job *j,* W_{jk} is the waiting time of job *j* at sequence *k* and *p jm*(*k*) is the processing time needed by job *j* on machine *m* at sequence *k* .The GA consists of four main stages: evaluation, selection,

©2009-2010 CIS Journal. All rights reserved.

crossover and mutation. The evaluation procedure measures the fitness of each individual solution in the population and assigns it a relative value based on the defining optimization (or search) criteria. The algorithm of GA used for optimizing Job Shop Scheduling problem is summarized as follows:

Step 1: Initialization - The chromosomes consist of genes which define the assignment of operations to the machines. Every chromosome in the population represents a solution for the problem.

Step2: Initial population - Based on the scheduling phase the initial population is formed considering the processing time and the operations assigned on the machines. The size of the population depends upon the number of jobs and levels in the benchmark instance.

Step 3: Fitness Evaluation – For each chromosome the fitness function is evaluated according to Eq. (1), the objective to minimize the makespan.

Step 4: Selection – The best chromosomes are chosen for reproduction using the method of tournament selection.

Step 5: Genetic operators – Crossover and mutation are applied to the selected chromosomes and new individuals are generated until a maximum is reached.

Step 6: Termination –If maximum number of generations is reached then stop else continue from Step 3.

5. ANT COLONY OPTIMIZATION BASED JSSP

Ant colonies exhibit very interesting behaviours, though one specific ant has limited capabilities, the behaviour of a whole ant colony is highly structured. They are capable of finding the shortest path from their nest to a food source, without using visual cues but by exploiting pheromone information [5]. While walking, ants can deposit some pheromone on the path. The probability that the ants coming later choose the path is proportional to the amount of pheromone on the path, previously deposited by other ants. This theory was the basis for forming the Ant Colony Optimization (ACO) algorithm using artificial ants. The artificial ants are designed based on the behavior of real ants. They lay pheromone trails on the graph edges and choose their path with respect to probabilities that depend on pheromone trails and these pheromone trails decrease progressively by evaporation.

At the end of each generation, each ant present in the population spawns a complete tour traversing all the nodes based on a probabilistic state transition rule. The nodes are chosen by the ants based on the order in which they appear in the permutation process. The node selection process involves a heuristic factor as well as a pheromone factor used by the ants. The heuristic factor, denoted by η_{ij}, and the pheromone factor, denoted by τ_{ij}, are indicators of how good it seems to have node j at node i of the permutation. The heuristic value is generated by some problem dependent heuristics whereas the pheromone factor stems from former ants that have found good solution. The next node is chosen by an ant according to the following rule that has been called pseudo random proportional action choice rule [6]. With probability q_0, where $0 \leq q_0 < I$ is a parameter of the algorithm, the ant chooses a node from the set of nodes (s) that have not been selected so for which maximizes $(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}$, where $\alpha \geq 0$ and $\beta \geq 0$ are constants that determine the relative influence of the pheromone values and the heuristic values on the decision of the ant. The probability of choosing the next node is chosen from the set S according to the probability distribution given by:

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{h \in S}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

This probability also known as the transition probability is a trade-off between the pheromone factor and the heuristic factor. The heuristic factor is computed as $\eta_{ij} = 1/F(X_j), j \in S$, where F(X) represents the cost function of X. While constructing its tour, an ant will modify the amount of pheromone on the passed edges by applying the local updating rule $\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\tau_0$, where $\tau_{ij}(t)$ is the amount of pheromone on the edge (i, j) at time t; ρ is a parameter governing pheromone decay such that $0 < \rho < 1$; and τ_0 is the initial value of pheromone on all edges. Once all ants have arrived at their destination, the amount of pheromone on the edge is modified again by applying the global updating rule $\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)$, where $\Delta\tau_{ij}(t) = L^{-1}$, if this is the best tour, otherwise $\Delta\tau_{ij}(t) = 0$, and L indicates the length of the globally best tour. The pheromone updating rule was meant to simulate the change in the amount of pheromone due to both the addition of new pheromone deposited by ants on the visited edges and to pheromone evaporation [2]. The algorithm stops iterating either when an ant found a solution or when a maximum number of generations have been performed. The ACO procedure for the JSSP is shown below:

Step 1: Initialization
Let M_{ij} denote the machine index
Assign no. of ants = no. of jobs
Each job is a set of operations denoted by {$O_{11}, O_{12}, \ldots, O_{JOBS,1}$}
Initialise parameters α, β, ρ and φ
Record the current position of ants
Step 2: Building ant tour
If ant k is the first operation then
Assign the first operation O_{k1} of job 1 to ant k's first visited operation
Record the visited operation of ant k in its memory
Else

©2009-2010 CIS Journal. All rights reserved.

Choose the next operation O_{k2} based on constraints of the job
If next operation is not the last then
Assign M_{ij} after next operation
Determine the transition probability based on

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{h \in S}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Step 3: Apply the local pheromone updating rule

$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\tau_0$, *where* $\tau_{ij}(t)$ *is the amount of pheromone on the edge (i, j) at time t;* ρ *is a parameter governing pheromone decay such that* $0 < \rho < 1$*; and* τ_0 *is the initial value of pheromone on all edges.*

Step 4: Apply the global updating rule

$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t)$, *where* $\Delta\tau_{ij}(t) = L^{-1}$, *if this is the best tour, otherwise* $\Delta\tau_{ij}(t) = 0$, *and L indicates the length of the globally best tour.*

Determine the time taken by the ant k and compute the shortest time

Step 5: If stopping condition is true then stop else continue from Step 2.

6. EXPERIMENTAL RESULTS

The Fisher and Thompson 10x10 instance (FT10) and Adams, Balas, and Zawack 10x10 instance (ABZ10) benchmark problems were used to compare and analyze the performance of computational intelligence paradigms for solving job shop scheduling problem. The procedures of GA and ACO were simulated using MATLAB R2008b on Intel core 2 Duo (1.73GHz), 1GB RAM PC.

A. Scheduling

The job-shop is concerned with the simultaneous and synchronized ordering of operations on several machines. Based on each level, sequence of the jobs will vary, and hence scheduling has to be processed before optimizing the jobs. For the FT10 and ABZ10 benchmark, during scheduling, the machines will process different jobs in different levels irrespective of the processing time. The Waiting Time (WT), Completion Time (CT) and its Priority Level (PL) of corresponding job are computed after the job passes through all machines so that all operations are completed. The waiting time, completion time are graphed for comparison and priority level corresponding to their completion time is ranked and plotted in graph which is shown in Fig 2 and Fig 3 respectively. For the FT10, the makespan value is calculated as 1356 sec and for ABZ10 the makespan is 1908 sec. Thus scheduling of the both benchmark instance are computed and the makespan value is obtained. This makespan value is kept as reference for optimization using computational intelligence techniques.

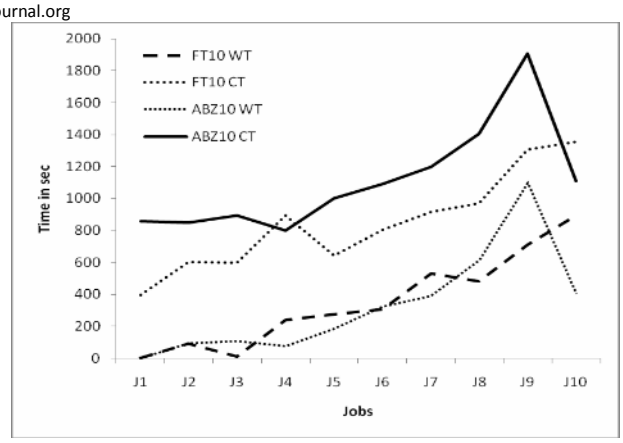

Fig 2 Graphical representation of CT and WT

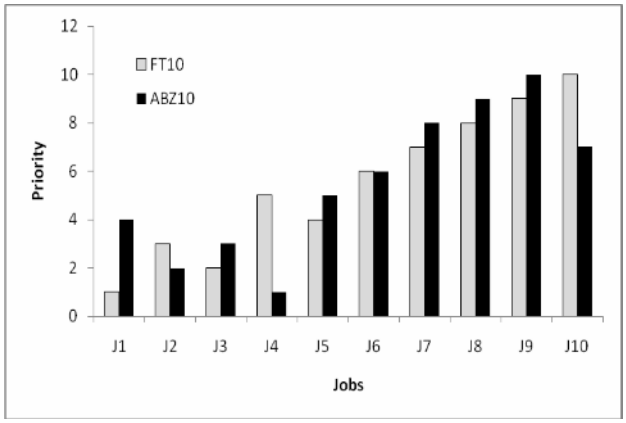

Fig 3 Priority Level Vs Jobs for FT10 and ABZ10

B. Genetic Algorithm

Genetic algorithms are applied to solve a problem using the principle of evolution. In the search process it will generate a new solution using genetic operator such as selection, crossover and mutation. The search procedure will stop once there is no improvement in next iteration. Here each chromosome is represented by a list of job order, and the machine sequence taken from the FT10 and ABZ10 benchmark problems. The algorithmic parameters used for simulating JSSP using GA are:

Population size : based on level and machine (for most of the instances it was 10)
Crossover probability : 0.6
Mutation probability : 0.0333
Crossover : Two point crossover
Mutation : Bit flip
Number of generations : 100

The Waiting Time (WT), Completion Time (CT) and the Priority Level (PL) of the corresponding job are evaluated during the GA generational run for several iterations. The graphical representation of the computed waiting time, completion time and the priority levels are shown in Figs 4 and 5 for both FT10 and ABZ10 problems.

VOL. 1, NO. 2, Oct 2010 E-ISSN 2218-6301
Journal of Emerging Trends in Computing and Information Sciences
©2009-2010 CIS Journal. All rights reserved.
http://www.cisjournal.org

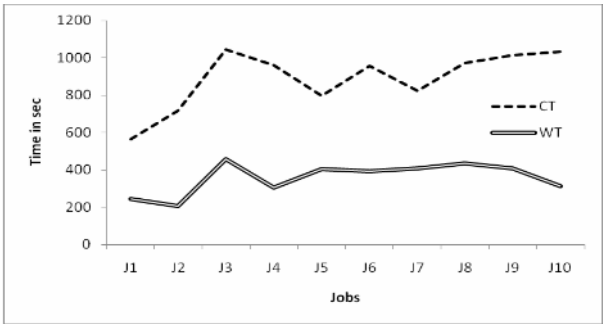

Fig 4 Graphical representation of CT and WT for FT10 using GA

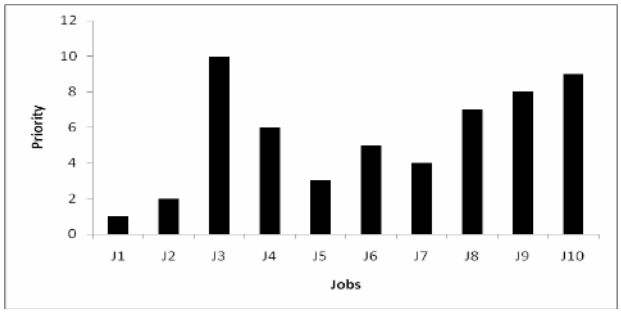

Fig 5 Priority Level Vs Jobs for FT10 using GA

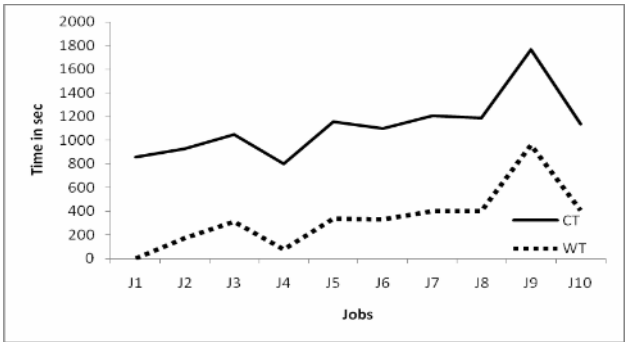

Fig 6 Graphical representation of CT and WT for ABZ10 using GA

The makespan value is computed from these evaluations as 1046 sec for the FT10 problem and it is found that the makespan value is decreased while optimizing the problem using GA when compared to the scheduling phase. Similarly the ABZ10 problem was also simulated and the results are shown in Figures 6 and 7 respectively. Using GA for the ABZ10 problem the makespan value is found to be 1769sec, which is a decreased value when compared with the makespan obtained in the scheduling phase.

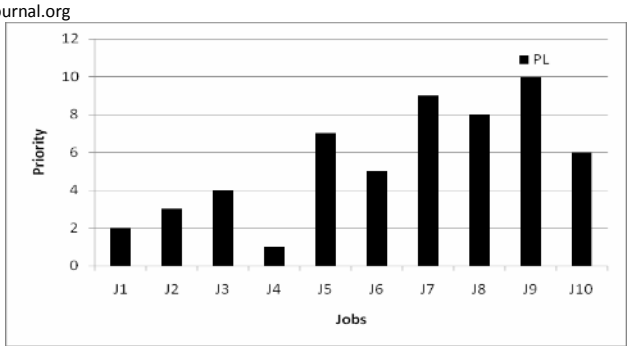

Fig 7 Priority Level Vs Jobs for ABZ10 using GA

C. Ant Colony Optimization

In JSSP, the ants construct a sequence by first choosing a job for the first position, then a job for the second position, and so on until all jobs are scheduled. The parameters of the ant system for solving the FT10 and ABZ10 problems are divided into two groups: those that influence the state transition (a and b) and those that determine the pheromone update (the evaporation constant r and the number of ants *m*). From simulation it appears that the parameters Q (pheromone allocation per unit of distance) and t0 (initial pheromone level) are of very little importance to the algorithm's performance. The parameters and their values are used for running the ACO algorithm is shown below:

Number of ants = based on level and machine
Weight of pheromone trail α = 1
Weight of heuristic information β = 1
Pheromone evaporation parameter ρ = 0.7
Constant for pheromone updating Q = 10

The ACO algorithm for the JSSP problem was tested on both FT10 and ABZ10 problems and the makespan was computed as 1052 sec and 1502 sec respectively. The performance characteristics of the problems based on waiting time and completion time are shown in Fig 8. The priority level of the jobs is shown in Fig 9.

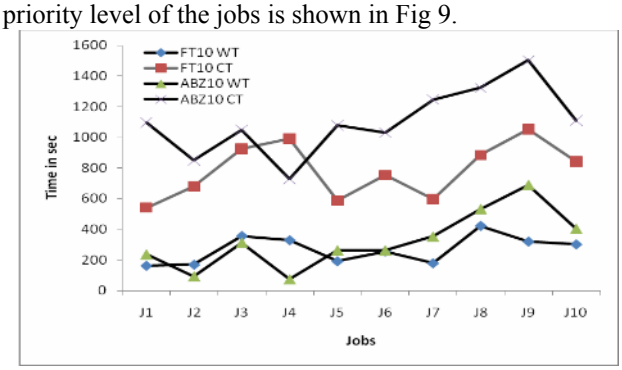

Fig 8 Graphical representation of CT and WT using ACO

©2009-2010 CIS Journal. All rights reserved.

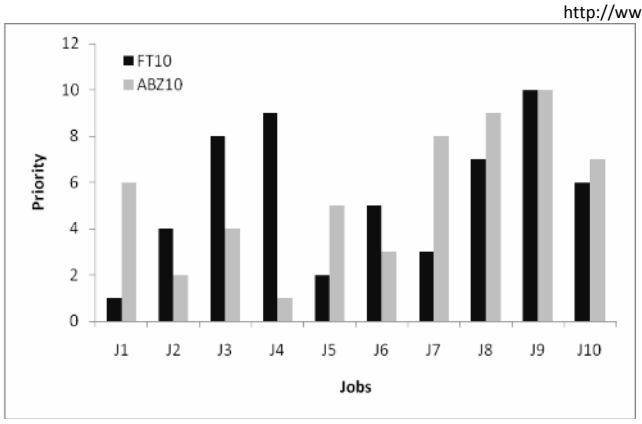

Fig 9 Priority Level Vs Jobs using ACO

D. Comparative Analysis

The waiting time, completion time and the priority levels for FT10 and ABZ10 are tabulated for the Scheduling phase, GA and ACO in Table 2. The makespan value for the FT10 and ABZ10 problem has decreased for both GA as well as ACO when compared with the value obtained in the scheduling phase. For the FT10 problem, GA computed the makespan as 1046 sec and ACO as 1052 sec, which indicates that performance of the algorithm depends upon the problem defined. While considering the ABZ10 problem, it can be seen that ACO performs better compared to GA, since the makespan for ACO is 1502 sec and this is a reduced value than 1769 sec, the makespan of GA.

Table 2 Comparison of Benchmark Instances

JOB	WT SCH	WT GA	WT ACO	CT SCH	CT GA	CT ACO	PL SCH	PL GA	PL ACO
FT10 Benchmark Instance									
J1	0	245	164	395	564	542	1	1	1
J2	91	207	171	601	717	681	3	2	4
J3	11	459	357	598	**1046**	925	2	10	8
J4	238	305	330	893	960	991	5	6	9
J5	277	404	193	644	797	586	4	3	2
J6	307	393	257	803	956	753	6	5	5
J7	530	407	180	917	823	596	7	4	3
J8	481	433	423	972	972	884	8	7	7
J9	708	409	321	1304	1014	**1052**	9	8	10
J10	892	313	303	**1356**	1032	843	10	9	6
ABZ10 Benchmark Instance									
J1	0	0	237	859	859	1096	4	2	6
J2	94	173	94	851	930	851	2	3	2
J3	106	314	314	895	1050	1057	3	4	4
J4	75	75	75	798	798	728	1	1	1
J5	183	340	264	999	1156	1080	5	7	5
J6	322	329	263	1090	1097	1031	6	5	3
J7	390	401	354	1199	1210	1247	8	9	8
J8	614	399	533	1404	1189	1323	9	8	9
J9	1097	958	691	**1908**	**1769**	**1502**	10	10	10
J10	405	405	405	1109	1139	1109	7	6	7

In Table 3, the various parameters such as Makespan (MS), Mean Completion Time (MCT), Least Completion Time (LCT), Mean Waiting Time (MWT), Least Waiting Time (LWT), Elapse Time (ET), MS Reduction Efficiency and Algorithm Efficiency for the optimization techniques with the corresponding benchmark is compared and analyzed. Applying ACO for FT10, the Mean Completion Time is reduced to 785.3 sec while the other techniques GA and the scheduling obtained higher MCT. For ABZ10, the mean completion time is reduced to 1101.7 sec using Ant Colony Optimization.

Table 3 Comparative Analysis of GA and ACO

Parameters		Scheduling	GA	ACO
MS (sec)	FT10	1356	1046	1052
	ABZ10	1908	1769	1502
MCT (sec)	FT10	848.3	888.8	785.3
	ABZ10	1111.2	1119.7	1101.7
LCT (sec)	FT10	395	564	542
	ABZ10	798	798	728
MWT (sec)	FT10	353.5	357.5	269.9
	ABZ10	328.6	339.4	323
LWT (sec)	FT10	0	207	171
	ABZ10	0	0	94
Elapse time (sec)	FT10	-	0.0780	1.4679
	ABZ10	-	0.624	1.4976
MS reduction efficiency %	FT10	-	22.87	22.42
	ABZ10	-	7.29	21.88
Algorithm efficiency%	FT10	-	77.13	77.58
	ABZ10	-	92.71	78.72

The Least Completion Time is also less for ACO, 542 sec for FT10 and 728 sec for ABZ10. Comparing mean waiting time, the ant colony optimization performs better with least MWT for FT10 with 269.9 sec and for ABZ10 with 323 sec. The time taken for the algorithms to perform is known as the elapsed time. The elapsed time for the performance of ant colony optimization is more when compared with GA. The elapse time, MS reduction efficiency and algorithm efficiency for scheduling cannot be computed, since the makespan value for scheduling is manipulated manually. With the overall performance, it can be concluded that ACO is superior than genetic algorithm for solving JSSP.

7. CONCLUSION

This paper presents a novel knowledge-based approach for the job shop scheduling problem (JSSP) by utilizing the various constituents of the computational intelligence techniques such as Genetic Algorithm (GA), and Ant Colony Optimization (ACO). The well known Fisher and Thompson 10x10 instance (FT10) and Adams, Balas, and Zawack 10x10 instance (ABZ10) problem is selected as the experimental benchmark problem and simulated using MATLAB R2008b. This research focused primarily on discovering new approaches that can match the computational intelligence techniques in solving Job Shop Scheduling problems. Significant improvements can be made by modifying the goals of this paper and adopting techniques to extend the knowledge of job shop scheduling problems. The research dealt specifically with the classical 10x10 job shop scheduling problem with the objective of minimizing the makespan. The research can

©2009-2010 CIS Journal. All rights reserved.

be extended to a larger size problem to analyse the performances. Dynamic and stochastic job-shop problems which incorporates attributes such as non-zero ready times, inter – travel time, multiple scheduling time, multiple job routes simultaneously, assign weights to jobs and uncertain processing times can also be solved using the computational intelligence techniques. These features increase the search space of solutions and make the task of scheduling very complicated. This provides a wide area for research giving rise to potential areas in the scheduling problems to be solved.

REFERENCES

[1] Coelle, Carlos A. [2007] "An Ant System with steps counter for the Job Shop Scheduling Problem." IEEE trans. Evolutionary Computation, pp. 477-484.

[2] Dorigo, Marco. [1997] "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem." IEEE Transactions on Evolutionary Computation, vol 1.

[3] Dubois, Didier. [1993] "Fuzzy constraints in job-shop scheduling." IJCAI'93 Workshop on Knowledge-Based Scheduling.

[4] J. Christopher Beck, Patrick Prosser and Evgeny Selensky. [2002] "Graph Transformations for the Vehicle Routing and Job Shop Scheduling Problems." In Graph Transformation, Germany: Springer-Verlag, pp. 60-74.

[5] Jun Zhang, Xiaomin Hu, X. Tan, J.H. Zhong and Q. Huang. [2006] "Implementation of an Ant Colony Optimization for job shop scheduling problem." Transactions of the Institute of Measurement and Control, 93-108.

[6] Merkle, Daniel. [2000] "An Ant Algorithm with a new pheromone evaluation rule for total tardiness problem." Real World Applications of Evolutionary Computation, Springer, Gremany.

[7] Pezzella F, et al. [2007] "A genetic algorithm for the Flexible Job-shop Scheduling Problem." Computers and Operations.

[8] Fevrier Valdez, Patricia Melin ,Olivia Mendoza, [2008] "A new evolutionary method with fuzzy logic for combining particle swarm optimization and Genetic algorithm: the case of neural networks optimization", IEEE, Vol. 7, pp 45-49.

[9] Giffler, B. and Thompson,G.L. [1960] "Algorithms for solving production scheduling problems. Operations Research." Operations Research, vol. 8, pp. 487-503

[10] Brooks, G.H. and White,C.R. [1965] "An algorithm for finding optimal or near optimal solutions to the production scheduling problem." Journal of Industrial Engineering, vol. 1, pp. 34-40

[11] Blackstone Jr., J.H.,Philips,D.T., and Hogg,G.L. [1982] "A state-of-the-art survey of dispatching rules for manufacturing job-shop operations." International Journal of Production Research, vol. 20, pp. 27-45

[12] Cheng, R.,Gen,M. and Tsujimura,Y. [1995] "A tutorial survey of job-shop scheduling problems using genetic algorithms." Computers and Industrial Engineering, vol 37, pp. 51-55

[13] Xiaoyu Song ,Limei Sun ,Qiuhong Meng [2008] "Deadlocks solving strategies in hybrid PSO algorithm for job shop scheduling" IEEE, Vol. 4, pp71-75.

[14] Innani, Alok D. [2004] 'Applying Data Mining to Job Shop Scheduling Problems using Regression Analysis' Ohio University, Japan.

[15] Jain.A.S, Meeran.S, [1999] "Deterministic Job-Shop Scheduling:Past,Present and Future", European Journal of Operational Research,113(2), pp.390-434.

[16] Li Xiaoping, [1999] "Solving Job Shop Scheduling Problem using Demarcation-Genetic algorithm", Electric Machines and control,3(2), pp. 93-96.

[17] http://www.swarmintelligence.org

[18] http://www.aco-metaheuristic.org

[19] http://people.brunel.ac.uk/