Proceedings of the 5th WSEAS Int. Conf. on Signal Processing, Robotics and Automation, Madrid, Spain, February 15-17, 2006 (pp345-350)

Extending MATLAB and GA to Solve Job Shop Manufacturing Scheduling Problems

Hamidullah Khan Niazi[1], Sun Hou-Fang[2], Zhang Fa-Ping[3], Riaz Ahmed[4]
([1, 4] National University of Sciences and Technology (NUST), CAE, Pakistan
([2, 3] Department of Manufacturing and Automation, Beijing Institute of Technology 100081, China)

Abstract: Job shop scheduling problem has been always a hardest task in the combinatorial research. Keeping in view the strong computational power of MATrix LABoratory (MATLAB) and robustness of GA, we have used a different novel approach for solving difficult shop floor scheduling problems. In this paper, parallel genetic algorithm based solution methodology has been presented and the algorithm is implemented using powerful MATrix LABoratory (MATLAB) environment to solve practical problems of job shop. The special coded mutation and crossover operators were designed to avoid any infeasible formulation of children. The solution result reveals that this methodology can be used to solve the complex optimization problems. In this work, job shop scheduling problem has been formulated and subsequently solved with the parallel genetic algorithm approach. The robustness and flexibility of GA offers a lot to tackle the stochastic solution of nondeterministic polynomial (NP) hard problems. The work is supported by the experimental and simulation results. The make span minimisation performance criteria were chosen in the experimental analysis.

Key words: Job shop, MATLAB, parallel genetic algorithm, optimisation

1 Introduction

Job shop scheduling problems (JSSP) are the most frequently encountered problems in practical manufacturing environment. Due to the exponential growing search space in the combination of goals and resources, the problem is NP-complete [1,2]. Many researchers have tried to tackle the practical job shop problems with different approaches and many of them found the results with reasonable degree of confidence. The complexity of the problem has, however, been so dynamic that the no single all-round strategy can be applied to tackle all the real situations at frequently changing shop floor environment. The traditional approaches to the solution of scheduling problems depend heavily on dispatching rules and knowledge-based systems. The disadvantage with such dispatching rules is that there is no single rule that will be effective for all production conditions, and manual selection or updating is required where using rules [3,4]. Also the traditionally operational methods such as dynamic programming, branch & bound method, and integer programming can give an optimal solution only for a reasonably sized problem [5]. The use of knowledge base expert system also demonstrate inability to solve the complex job shop scheduling due to lack of limited and updated knowledge or expertise required from human experience.

In this paper, we propose novel parallel genetic algorithm (PGA) approach using MATrix LABoratory (MATLAB) [6] GA toolbox to solve the JSSP in a parallel machine environment. The experimental problems have been attempted using MATLAB computing environment. Due to the advantages of retarding premature convergence problem, we have focused on the parallelization strategy in GA. The goal of this research is to propose an efficient PGA-based scheduling methodology using MATLAB computing environment to address the JSSP. The work is supported by the experimental results and conclusion.

2 Problem Formulation for Job Shop Scheduling

The concept of genetic algorithm (GA) has been successfully applied to many useful areas like artificial intelligence, pattern recognition and control problems. However, there is not enough reported work on combinatorial problems. The application of GA can be useful with reasonable

time due to recent advancement in computing area. Due to the stochastic nature of the GA, it can be usefully applied to the NP hard problems with useful results. In this paper, the PGA approach using MATLAB environment has been used. We exploited the flexibility and easy tradeoff in the GA parameters to avoid any premature convergence and have introduced a diversity in children selection. We start with a generic problem of Pm | $|C_{max}$ [7] for m machines in parallel. All jobs are released simultaneously where the objective is to minimize the makespan with no preemption allowed. This approach regards determination of an optimal job-shop schedule as a linear programming problem with integer adjustments, and then employs genetic algorithms to optimise a solution. The problem formulation Let C_{ik} denote the completion time of job i on machine k (i.e. the completion time of the particular operation of job i that needs machine k), $m_{(ijk)}$ be the machine for operation(i,j,k) and $t_{ijk,}$ the processing time for operation (i,j,k). The values of C_{ik} will off course help to determine the schedule. Most of the excellent work and problem formulation already done by J G Qi and et al [8] was carried forward and new GA design parameters were introduced in the latest MATLAB release 14 environment to avoid the problem of premature convergence with improved and some new results. Such problems and issues have been reported by many researchers. This goal of minimum premature convergence with improved results was main focus in mind while solving the JSSP. We tried the parallel GA approach and premature convergence problem of GA with the intelligent combination and mix of crossover and mutation operators was reduced to minimum. Here job-shop operation is described by a triplet (i,j,k) i.e. operation j of job i is to be executed on machine k. If the operation (i,j-1,h) requires machine h, then operation (i,j,k) would require machine k, then used inequalities precedence constraints.

$$C_{ik} - C_{ih} - t_{ijk} \geq 0, \quad 1 \leq j \leq m, \quad 1 \leq i \leq n \qquad (1)$$

Where m and n are the number of machines and number of jobs, respectively.

It is necessary to ensure that no two operations are processed simultaneously by the same machine. For example, if job i precedes job *p* on machine *k* (i.e. operation (*i*,*j*,*k*) is completed before operation (*p*,*q*,*k*)), this is equivalent to a constraint of:

$$C_{pk} - C_{ik} - t_{pqk} \geq 0 \qquad (2)$$

On the other hand, if job p precedes job i on machine k, then it is also necessary to ensure that:

$$C_{ik} - C_{pk} - t_{ijk} \geq 0 \qquad (3)$$

In order to adjust these constraints in the formulation, a variable y_{ipk} is applied to specify the operation sequence, i.e. $y_{ipk} = 1$ if job *i* precedes job *p* on machine *k*, and $y_{ipk} = 0$ otherwise, the above constraints equation (2) and (3) become:

$$C_{pk} - C_{ik} + N((1 - y_{ipk}) - t_{pqk} \geq 0 \qquad (4)$$

$$C_{ik} - C_{pk} + Ny_{jpk} - t_{ijk} \geq 0 \qquad (5)$$

Where constant *N* represents an arbitrary very large positive number. Therefore, the entire job-shop problem can be formulated as following keeping in view the makespan criteria:

$$\text{Minimize } (\max \sum_{i=1}^{n} C_{iki}) \qquad (6)$$

Subject to

$$\begin{aligned} &C_{ik} - C_{ih} - t_{ijk} \geq 0 && 1 \leq j \leq m,\ 1 \leq i \leq n \\ &C_{pk} - C_{ik} + N(1 - y_{ipk}) - t_{pqk} \geq 0 && i \geq 1,\ p \leq n,\ 1 \leq k \leq m \\ &C_{ik} - C_{pk} + Ny_{ipk} - t_{ijk} \geq 0 && i \geq 1,\ p \leq n,\ 1 \leq k \leq m \\ &C_{ik} \geq 0 && y_{ipk} = 0 \text{ or } 1 \end{aligned} \qquad (7)$$

$y_{ipk} = 1$ if job *i* precedes job *p* on machine *k*, and $y_{ipk} = 0$ otherwise, k_i denotes the machine at which the last operation of job *i* is scheduled.

Generally, makespan is important in parallel machines for scheduler to get a good balance [7]. The completion time of the last job would be the makespan of the schedule. In JSSP, the makespan represents also a good performance measure. The schedule with the minimal makespan often implies a high utilization of machines.

3 A Proposed Parallel Genetic Algorithm on MATLAB Environment

The proposed structure of PGA with a Matlab GA toolbox structure is given in Figure 1. The overall structure of this PGA and its working is described next.

In this work, the initial population is created randomly. The fitness of these solutions is evaluated. If the fitness criteria is not met in the beginning, then new population are created using selection, crossover and mutation processes.

The objective of the selection method practiced in genetic algorithms is to give polynomial increasing trials to the fittest individuals. In this paper, different selection techniques like stochastic

uniform, roulette and tournament selection have been tried to pick the fittest initial solutions in the current scenario. Elitism strategy picks the fittest individual copying without mutation and crossover. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution to date. This strategy has given favorable results in our computational results.

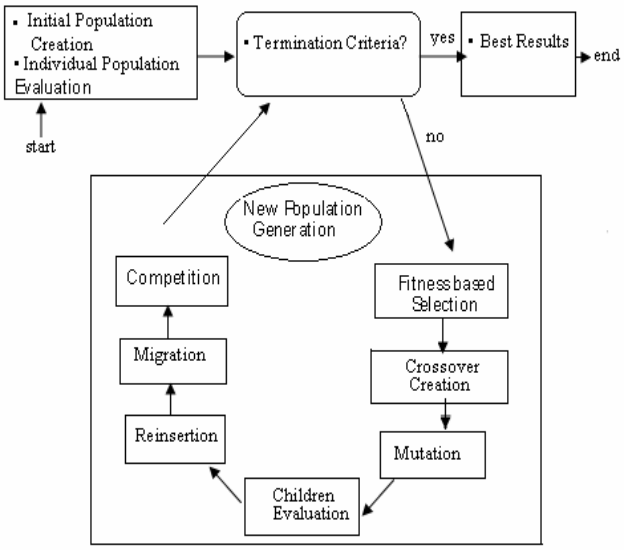

Fig 1: A structure of parallel GA

In a genetic algorithm, crossover recombines the genetic material in two parent's chromosomes to make two children, in order to generate a better solution. In this paper, four kinds of cross over operators; namely two point, intermediate, scattered and a custom crossover operators were tried. However, the special custom crossover SCX operator was found to be the most suitable. The SCX operator is actually a combination of order based crossover and multipoint crossover operators. This operator always creates the valid and feasible genes. The order-based crossover operator applied to the first half of the substring is often used in sequencing problems because it maintains precedence constraints of operations from the parents to the offspring. It works by incorporating a string sequence from two different parents into two new strings. The mechanism of this crossover is addressed by first determining two cut points at two random positions on the gene strings, then passing the portion between the two cut points to the two offspring. The operator then begins to construct the remaining left-hand and right-hand sides. of the first offspring by going over the second parent and eliminating the same numbers with the passed portion of the first parent, and filling up the missing left and right sides according to an order existing on the second parent. A genetic algorithm will do this operation pair-wise. Figure 2 (a) illustrates the SCX operator with an example given in Figure 2(b).

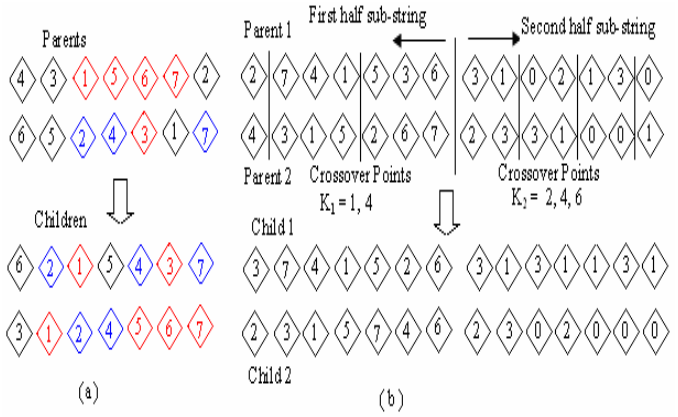

Fig 2:(a) SCX operator(b) An operator's example [8]

In this paper, we introduced a custom mutation instead of using Gaussian or uniform mutation from the tool box which sometime gives infeasible children during the swapping process. The custom mutation works in two steps: firstly it allows mutating the two genes randomly, based on position followed by another mutation which is dictated by an order based swapping. This process is explained in Figure 3 (a and b).

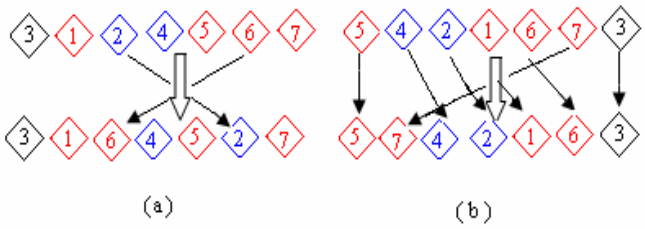

Fig 3: (a) position based swapping (b) An order based swapping

The objective of scheduling is to minimise the total completion time of all jobs, i.e. all jobs should be completed as early as possible. The suitable objective functions are max (C_{iki}), and the optimising process is min(max(C_{iki})). When formulating constraints penalty approach suggested by Goldberg [9] has been followed in this work as mentioned in equations (10) and (11) in section 5. This approach transforms a constrained problem into an unconstrained problem by associating a cost, or penalty, with all constraint violations for the objective function.

The migration model divides the population in multiple subpopulations, also called island populations. Migration options specify how individuals move between subpopulations. When migration occurs, the best individuals from one subpopulation replace the worst individuals in another subpopulation depending upon the competition. Individuals that migrate from one subpopulation to another are copied. They are not removed from the source subpopulation.

The important control parameters in the island migration model are: topology, direction, interval, fraction (rate) etc. In our approach, we have focused mainly on three migration parameters i.e. *direction, interval* and its *fraction*. The number of exchanged individuals, the selection method of the individuals for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulations and the exchange of information between subpopulations. For example for a *interval* of 10 generations, if individuals migrate from a subpopulation of 60 individuals into a subpopulation of 100 individuals and you set *fraction* to 0.1, then the number of individuals that would migrate is 0.1 * 60 = 6 after every 10 generation in the set *direction* (which could be *forward or both*). In this paper, we used the forward migration model with fraction of .25 and migration interval of 25 generations.

The genetic processes of crossover, selection and replacement continue unless interrupted under certain termination criteria. The termination criteria used in this paper was as to be 200 maximum number of desired generation and stall generation as to be 60.

4 Fitness Function and Chromosome and Constraint Representations

The objective of scheduling is to minimize the maximum completion time i.e minimize makespan where objective function is,

Obj = max (C_{iki}), so optimization function is, min(max(C_{iki})). Now using Goldberg [9] approach for handling constraint in GA problem by transforming constrained problem into unconstrained problem by employing penalty function for the constraint violation.

Minimize z(x) (8)

Subject to $g_i(x) \geq 0$ (I = 1, 2, …n) (9)

where 'x' is an m vector. Thus the transformed unconstrained problem becomes:

$$\text{minimize} \quad z(x) + \mathbf{r} \sum_{i=1}^{n} \phi[g_i(x)] \qquad (10)$$

Where ϕ - is a penalty function and **r** is a penalty coefficient. A number of alternatives exist for handling the penalty function ϕ. In this paper, we have squared ϕ $[g_i(x)] = g_i^2(x)$ for all violated constraints transformation. The chromosome was constructed on the basis of both job orders and waiting times [10]. The representation of the chromosome is composed of several substrings, each of which is referred to a machine as a resource.

For the *k*th machine, the substring is represented as $\{J_{k1}, J_{k2}, J_{k3,\ldots\ldots}J_{kn}, W_{k1}, Wk_2,\ldots\ldots,W_{kn}\}$ where J_{ki} represents an operation of a job processed by the *k*th machine, and *Wki* is within the upper limit of the waiting time of the *k*th machine to process a job. Thus, if there are *m* machines, the whole string is made up of *m* substrings which are then composed of 2**n* genes where *n* denotes the number of jobs to be processed by the machine[8].

For instance, a substring of k_{th} machine which processes 5 parts can be represented like {2 4 5 3 1 0 1 0 2 3}. In this substring, first five genes (2 4 5 3 1) means 2nd, 4th, 5th, 3rd and 1st parts have a step to be processed by the k_{th} machine in a fixed order. The later 5 genes (0 1 0 2 3) means that 2nd and 5th parts needs to be processed at once when completing a previous part, and wait 1 unit if time before processing 4th part, wait 2 units of time before processing 3rd part and wait 3 units of time before processing 1st part. The use of this kind of first half substring leaves us with only two choices i.e. either to use idle time or start time. We have used idle time in our coding to narrow down the solution space. This type of string description naturally satisfies the constraints of Eqs (2) and (3) or Eqs (4) and (5), but it may not satisfy the constraint Eq. (1). To obtain a feasible solution, a penalty approach is adopted in this paper; therefore the fitness function is finally constructed as

$$\text{Fitness} = \min[\max(C_{iki})] + A \sum_{ij} (C_{ik} - C_{ih} - t_{ijk})^2 \qquad (11)$$

where 'A' is a weight

5 Simulation Results and Discussion

The simulation programs were written in MATLAB and computed with MATLAB GA toolbox. The machines required for each operation and the operation-processing times are deterministic without any pre-emption on any machine. The optimal solution determined by a genetic algorithm provides a schedule for each machine. Example data of 8 jobs with workstations of 4 machines [11] is given in Table 1.

Table 1: Job shop Scheduling data

Jobs	Machine Sequence				Processing Times				Jobs Due date
1	M1	M3	M2	M4	4	4	7	3	32
2	M3	M1	M2	M4	3	4	4	2	30
3	M1	M4	M3	M2	3	4	6	3	33
4	M3	M2	M1	M4	6	4	3	4	29
5	M4	M2	M3	M1	4	5	5	3	28
6	M2	M4	M1	M3	4	6	5	4	34
7	M2	M3	M4	M1	2	4	5	5	30
8	M4	M1	M2	M3	3	3	3	5	36

So the machine sequence matrix (with a workstation of 4 machines) and processing time's matrices are:

$$m(i,k) = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 3 & 1 & 2 & 4 \\ 1 & 4 & 3 & 2 \\ 3 & 2 & 1 & 4 \\ 4 & 2 & 3 & 1 \\ 2 & 4 & 1 & 3 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \end{bmatrix} \quad t_{i,k} = \begin{bmatrix} 4 & 4 & 7 & 3 \\ 3 & 4 & 4 & 2 \\ 3 & 4 & 6 & 3 \\ 6 & 4 & 3 & 4 \\ 4 & 5 & 5 & 3 \\ 4 & 6 & 5 & 4 \\ 2 & 4 & 5 & 5 \\ 3 & 3 & 2 & 5 \end{bmatrix}$$

Following parameters were used: weight A = 100, probability of crossover (P_c) = 0.7 and the probability of mutation (P_m) = 0.2

The evaluation parameters used in the optimization process are given in Table 2. The simulation result is given in Figure 4.

Table 2: genetic algorithm parameters

Elite count	2
Number of initial population	50
The length of chromosomes	64
Crossover rate	0.75
Mutation rate	0.20
Maximum number of generations	200
Stall generation	60
Insertion rate	0.85
Number of subpopulations	7.5
Migration rate	0.25
Number of generations between migration	25
Number of individuals per subpopulation	10
Time limit (seconds)	30
Stall Time Limit (seconds)	30

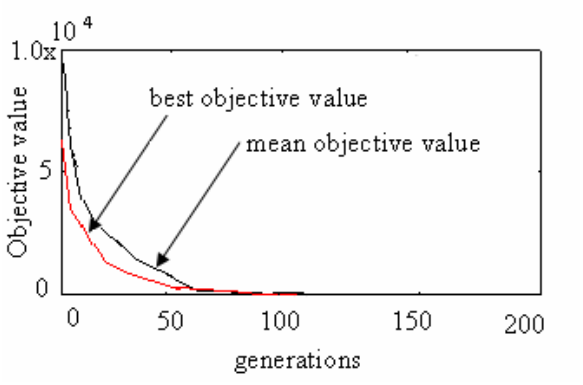

Fig 4: best and mean value plot for objective function

The optimal solution found is as follows:

Job Sequence								Waiting Time							
3	1	2	8	4	5	7	6	0	0	0	0	2	0	0	0
6	7	5	4	2	8	1	3	0	0	0	0	0	0	0	0
2	4	7	5	1	3	8	6	0	0	0	0	0	0	0	0
5	3	8	7	6	2	4	1	0	0	0	2	0	0	1	1

The optimal solution using Gantt chart based on the PGA is plotted in Figure 5.

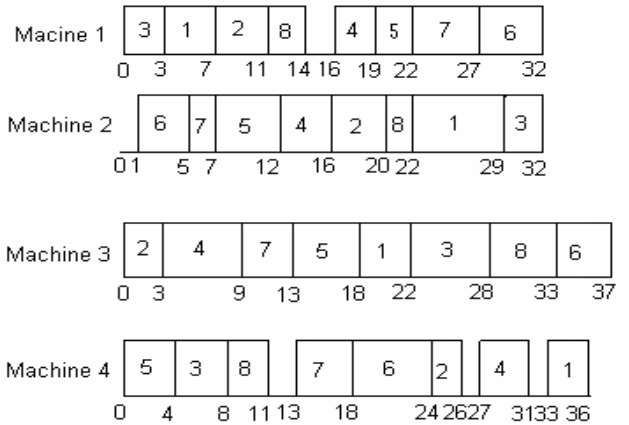

Fig 5: Gantt chart showing scheduling result.

In this paper, we also conducted some experiment on randomly generated problem consisting of 20 jobs and 5 machines. The primary objective was to see the affect of population on performance. We first investigated how the population size influences the performance by proposed PGA. First we fixed the maximum population size to be 200 and P_m and P_c both to be both 0.15. Under such lower ratios of crossover and mutation, the population size becomes the dominant factor in determining the performance of the proposed GA. Later on the population size was varied from 10 to 90. The results of experiment for the best, average and the worst values are plotted in Figure 6 for over 90 random runs

for each parameter setting. It is found that the algorithm performance does not change significantly beyond the population size of 60.

In another performance test, the algorithm was tested for fixing the population size to be 100 and maximum number of generation to be 200. From the performance graph in Figure 7, it is verified that mutation plays an important role in the genetic performance of the algorithm.

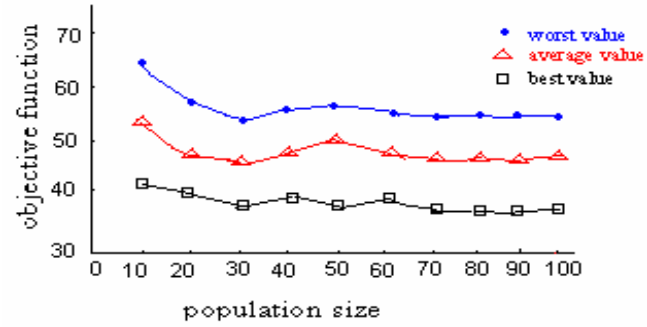

Figure 6: Affect of population on performance.

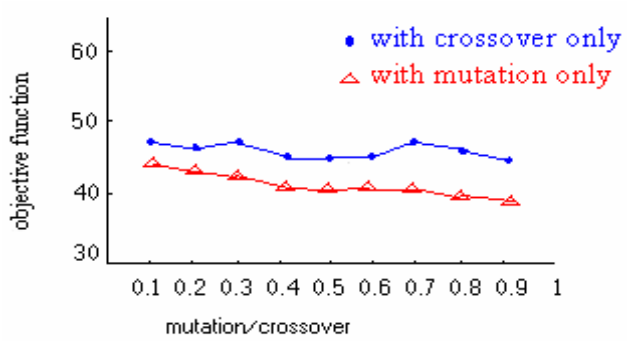

Fig 7: Average performance comparison for various P_m and P_c.

The crossover, however, depends on the partitioning structure of the algorithm.

6 Conclusion

In this paper, we have tested the parallel genetic algorithms approach using MATLAB GA toolbox with a suitable designing and selection of genetic operators. The algorithm was implemented in calculating the makespan of JSSP. The performance of the algorithm has been tested for various JSSP with favorable results. It is verified that the proposed algorithm approach using MATLAB toolbox can be used to solve the other parallel machine job shop scheduling problems with vast computing capabilities of MATLAB

References

[1] Sethi R. On the complexity of mean flow time scheduling. Mathematics of Operations Research 1977; 2(4):320–30.

[2] Garey MR, Johnson DS. Computer and intractability: a guide to the theory of NP-completeness, San Francisco: W H Freeman, 1979.

[3] J. H. Blackstone Jr, D. T. Phillips and G. L. Hogg, "A state-of the- art survey of dispatching rules for manufacturing job shop operation", International Journal of Production Research, 20(1), pp. 27–45, 1982.

[4]. R. Haupt, "A survey of priority rule-based scheduling", OR Spektrum, 11, pp. 3–16, 1989.

[5] Potts CN. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. Discrete Applied Mathematics 1983; 10(2):155–64.

[6] MATLAB 7.0 GA Algorithm and Direct Search Toolbox R 14
http://www.mathworks.com/products/gads/

[7] Pinedo, Michael, Scheduling, Theory, Algorithms, and Systems, Prentice Hall, New Jersey, 07632

[8] J G Qi et al, **"**The Application of Parallel Multi population Genetic Algorithms to Dynamic Job-Shop Scheduling" Int J Adv Manuf Technol (20 00) 16:609–615.

[9] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[10] R. W. Cheng, M. S. Gen and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms-I. Representation", Computers and Industrial Engineering, 30(4), pp. 983–997, September 1996.

[11] Shen Gang et al. "A job shop Scheduling Solution with Neural Network" Acta Electronica Sinica Vol 23 No.8 Aug.1995 48-51