Job-shop Scheduling using Genetic Algorithm

Wu Ying *, *Li Bin* **

* Pattern Recognition & Artificial Intelligence Laboratory Automation Dept. , Tsinghua Univ.

E_Mail address : ys4wy@info.au.tsinghua.edu.cn

** CIMS Center Room 609 , Automation Dept. ,Tsinghua Univ. 100084 Beijing. P.R.China

Abstract: Job-shop scheduling , a typical NP-complete problem, is an important step in planning and manufacturing control of CIMS environment . Researches on job-shop scheduling focus on knowledge-based approach and heuristic searching which are useful except the difficulty of getting knowledge[3]. Genetic algorithms are optimization methods which use the ideas of the evolution of the nature. Simple as genetic algorithms are, they are efficient[1][2]. Three novel genetic algorithms model, such as decimal idle time coding genetic algorithm(DITCGA)[5], binary idle time coding genetic algorithm(BITCGA), and adaptive idle time coding genetic algorithm(AITCGA), are presented to design job-shop scheduling algorithm in this paper. Using the idle processing time to code this problem, we efficiently reduce the solution space. In our approaches, adaptive learning mechanism is applied to guide the searching or evolution process. The simulation results show the efficiency of these approaches.

Keyword: FMS, job-shop , genetic algorithm .

I. INTRODUCTION

Job-shop scheduling, a typical NP-complete problem, is an important step in planning and manufacturing control of CIMS environment. In many cases, scheduling is so difficult that even a mediocre problem is painstaking . Optimum solution can be found only for certain problem. However, the larger the dimension is, the more difficultly the optimum solution can be found. So, we expect to find a feasible solution to solve the problem. Realistically, we are fairly satisfied to find a feasible solution that is nearly optimum. When we consider job-shop scheduling, we encounter a relatively complex problem, because there are many parts and machines to arrange, and the constraint is somehow harsh.

Because job-shop scheduling affect the efficiency of manufacture, many researches are in progress and many optimum approaches are proposed to get a satisfied solution of job-shop scheduling[3]. Knowledge based methods and heuristic searching are fairly successful. But there is still a question : how to acquire the knowledge needed?

Genetic algorithm (GA) is a relatively new approach of optimum searching[4]. GA inherits its ideas from evolution of the nature. Even a simple GA algorithm appears to be robust, and the complexity of algorithm and result of GA is irrelevant to the length of genetic string and the original state of population. Genetic algorithm is so simple that it only involves some selection, crossover and mutation operations, but it is so efficient that it can find a nearly optimum solution even for a large scale problem[4]. For an instance, travel salesman problem (TSP), a traditional NP-completed problem, is impossible for even the fastest computer to solve when the city number is greater than 100. Surprisingly enough, GA can give a satisfied solution.

Essentially, however, GAs are searching approaches. So the searching space or the solution space will affect the efficiency and the convergence of GAs. GA can get nearly optimum solution when the solution space is simplistic, while it can not even convergence to a feasible solution when the solution space is rather complex.

Whether can we consider to combine the knowledge-based methods and the GAs? It is reasonable that the knowledge can lead the searching and reduce the searching space and enhance the searching efficiency.

In this paper, we propose three methods based on genetic algorithm to find a nearly optimum solution to job-shop scheduling problem. First, we describe the job-shop scheduling problem. Second, we introduce the principle of genetic algorithms. Third, our approaches, includes genetic coding of the problem, description of fitness function and designation of some important operator such as selection, crossover and mutation. Three methods such as decimal idle time coding genetic algorithm(DITCGA), binary idle time coding genetic algorithm(BITCGA) and adaptive idle time coding genetic algorithm(AITCGA) are presented. And next, a series of simulations are showed to illustrate our methods and ideas. Next, the three methods are compared. Finally,

0-7803-3280-6/96/$5.00

some important conclusions are made .

II. THE DESCRIPTION OF JOB SHOP SCHEDULING

There are a number of parts to be processed by several machines. The working process of each part is determined by the technical requirement. Job-shop scheduling is to determine the processing sequence of the parts on these machines , s.t.

1. m---the number of machine , n---the number of part ;
2. the i part is composed of a fixed work process ;
3. If we use i to represent the ith part ,and k means the kth step. Let m(i,k) mean the machine to process the kth step of the ith part;
4. one machine can only process one part at one time ;
5. process (i , k) must be processed after process (i ,k-1);
6. $t_{i,k}$ is the processing time of process (i,k), mi,k is the machine number of process (i,k) ;
7. a step can not be separated .

(i,k,m(i,k)) means that the kth step of the ith part is processed by the m(i,k)th machine ,and ki means the last step of the ith part. Above are determined by technical requirement. We can see that job-shop scheduling problem is a typical linear programming. If let $S_{i,k}$ mean the start processing time of the kth step of the ith part, we can write the constraint below:

$$S_{i.k}-S_{i,k-1}+t_{i,k}\le 0, \qquad 1\le i\le n; 1\le k\le ki \qquad (1)$$

$$S_{i,1}\ge 0, \qquad 1\le i\le n \qquad (2)$$

$$S_{i,k}-S_{i,p}+t_{i,k}\le 0 \text{ or } S_{j,p}-S_{i,k}+t_{j,p}\le 0, \qquad 1\le i,j\le n; 1\le k,p\le ki \qquad (3)$$

In these constraints, (1) means that process (i , k) must be processed after process (i ,k-1), and (2) means that the start processing time must be no less than zero, and (3) means that a certain machine only can process one part at one time ,which can avoiding conflict of two parts.

The objective of scheduling is to minimize the end processing time of all parts, that is to say that let all parts be completed as early as possible. The objective function is $\max(S_{i,ki})$, and the optimizing process is $\min(\max(S_{i,ik}))$.

III. GENETIC ALGORITHM

In the magic nature, creatures are developing form elementary micro creatures to advanced and complex human beings. Undergoing a relatively long time, the structure of creatures are changing to adapt to the changing environment. In the evolutionary process, gene play a key role from which descendants get many beneficial materials from older generation. According to Darwin 's theory, natural selection discards inferior genes and inferior individuals which are not adapt to the environment. In genetics, crossover and mutation of gene are the main changes which lead to the difference between the two generations .

When we use Genetic algorithms, a exceptional optimization method, we must code the given problem firstly, and determine the fitness function according to the objective of optimization. Sure, different genetic descriptions or coding strategies lead to different results, such as algorithmic efficiency and the quality of the solution. Essentially, genetic algorithms are a searching process. If we do not make appropriate gene description, the searching space is rather lager than we anticipate GA can find a optimum solution, because there are innumerable local minimums in the solution space. On the other hand, If we code the problem properly, we can narrow the searching range to find a nearly optimum solution or even optimum solution rapidly.

Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics. GAs use probabilistic transition, not deterministic rules. GAs search from a population of points instead of a single point. In GAs, a solution of a problem is coded to a string or a chromosome. By using operations such as reproduction, crossover and mutation, the string which is fitter than the others get better chance to reproduce .

A simple GA is composed of three operators: reproduction, crossover and mutation. Reproduction is a process in which individual strings are copied according to their objective function values (or fitness function). Copying strings according to their fitness values means that strings with higher value have a higher probability of contributing one or more offspring in the next generation. The implementation of reproduction can be done by roulette wheel. After reproduction, crossover may proceed in two steps : First, members of the newly reproduced strings in the mating pool are mated at random; second, each pair undergoes crossing over. For example, consider string A1, A2 that A1 =0 1 1 0 | 1 ,

A2 =1 1 0 0 | 1

choosing a random number such as 4 , then we get the result

A1 =0 1 1 0 | 0

A2 =1 1 0 0 | 0 .

The mechanics of reproduction and crossover are surprisingly simple, involving random number generation, string copies and some partial string exchange. However, this algorithm is efficient in optimization .

Following is the typical genetic algorithm:

step 1 code the given problem use gene string;

step 2 initialize a population pop(n) and let n=0;

step 3 evaluate the fitness function value fit(n) of each

string in the population pop(n).From the fit(n), we can see which string representing a solution is what we want. If we find a satisfied solution ,we may cease the process, else let n=n+1 and continue;

step 4 select descendant pop(n) from the older generation pop(n-1);

step 5 crossover and mutate pop(n) according to certain strategy;

step 6 go to step 3

IV. JOB SHOP SCHEDULING BASED ON GA

1. Decimal Idle Time Coding Genetic Algorithm (DITCGA) In this section, we code the job-shop scheduling which is a NP-complete problem, and give the representation of the fitness function. Next we construct crossover and mutation operator.

First, we construct the description of the chromosome. A solution of the scheduling is code to a large string which is made up by several sub strings each of which stands by a machine. That is, for the kth machine, the sub string is $\{PRT_{k1}, PRT_{k2}, ...PRT_{ki}, ...PRT_{kn}, IDTM_{k1}, IDTM_{k2}, ...IDTM_{ki}, ...IDTM_{kn}\}$,
where PRT_{ki} means a step of a part is processed by the kth machine, and $IDTM_{ki}$ within the upper limit of idle time means the waiting time (or idle time) of the kth machine to process a part. So if there are m machines, the whole string is composed by m sub strings which is made up of 2*n genes where n means the number of steps to be processed by a machine.

For an instance, a sub string for the kth machine which processes five parts can be written like {2 5 4 1 3 0 1 0 3 0 }. In this sub string, the first five genes (2 5 4 1 3) means that the 2nd, 5th, 4th, 1st, 3rd parts have a step to be processed by the kth machine with the fixed order. The latter five genes (0 1 0 3 0) mean that machine process the 2nd, 4th and 3rd parts at once when completing a previous part, and wait 1 unit of time before processing the 5th part and 3unit of time before 1st part.

If we only use the fist half of this kind of sub string, we have to use knowledge based methods to get a completed determined solution because the exact arrangement can not be determined only when knowing the order of each part. So, the advantage of GA is not used completely. We have to add some genes. We face two choices: using idle time of each machine or using start time of each part. In our coding method, idle time of machines are used other than start time of each part to sharply narrow the solution space .

It is not guaranteed that this kind of string represents a feasible solution, because it may not satisfy the constrain in the formulas (1), although (2) and (3) are naturally satisfied. This problem can be successfully solved by using a punishment factor in fitness function which is discussed next .

Second, we construct the fitness function. The objective of job-shop scheduling is $\min(\max(S_{i,ik}))$ with the objective function $obj=\max(S_{i,ik})$. But the selection operation of genetic algorithm requires positive fitness function value and GAs maximize the fitness function which is different from the objective of job-shop scheduling $\min(\max(S_{i,ik}))$. So, we change the objective function to fitness function using fitness=C-obj , where C is a given number to guarantee the fitness to get positive value. Further, we use a punishment factor $PNSH(S_{i,k}-S_{i,k-1}+t_{i,k})$. Finally, the fitness function is

$$fitness = C - A \times \max(S_{i,ik}) - B \times \sum_i \sum_k PNSH(S_{i,k} - S_{i,k-1} + t_{i,k})$$

where A and B are weights.

Third, we construct mutation and crossover operators. When we select two sub strings, say, A1 and A2 as parents according to fitness value using roulette, they undergo mutations which are independent two operations to two halves : the first half of parts and the other half of idle time. An integer position l is selected uniformly at random between 1 and half length of sub string. For the first half, exchange the two parts of the position l and l-1. If l=1, the exchange l and l+1. For the other half, after select a random position l', just replace the genes of l' by a new random value ranging from 0 to the upper limit of idle time. For example A1= 2 4 1 5 3 0 1 2 0 1, and l=2, l'=3 so the result of mutation is A1'=4 2 1 5 3 0 1 1 0 1. Each sub strings undergoes mutation independently.

After selection and mutation, crossover may proceed in two steps. The first is for the first half. After selecting a position t, genes before t are kept for the two children. AR1 is the remnant after eliminating genes before position t of A1 in A2 and AR2 is the remnant after eliminating genes before position t of A2 in A1. Child A1' is produced by replaced the genes after t by AR1 and Child A2' is produced by replaced the genes after t by AR2.The second step is exchanging the genes after a random position t' of the other half to form two children. For example,

A1=2 4 |1 5 3 0 1 2 | 0 1

A2=3 1 |4 2 5 1 0 0 | 0 3 and t=2, t'=3. AR1=3 1 5, AR2=2 4 5, the two children are

A1'=2 4 3 1 5 0 1 2 0 3

A2'=3 1 2 4 5 1 0 0 0 1 .

This approach gets a satisfied result in job-shop scheduling. The results of software simulation will be showed below to illustrate the efficiency of this coding strategy.

2.Binary Idle Time Coding Genetic Algorithm (BITCGA)

The structure of chromosome in BITCGA is the same as DICCGA and the coding strategy is almost the same as the DITCGA only with the difference of idle time coding. In this algorithm, idle time is coded into binary number of certain bits. For an instance, 101 represents the decimal number 5. The binary coding is used to replace the decimal coding in DITCGA. For example, a sub string can be {2 4 3 1 5 000 001 010 000 111}.

The importance of the bits allocated to the idle time in BITCGA is what the upper limit of idle time to DITCGA. When we allocate more bits to idle time, the searching space is more complex and enormous.

The DITCGA uses float numbers to code the idle time. it is easy to understand but it may be not fit the genetic. When using binary coding, we can see that the idle time is changing when crossover happens. This method bring about more chances for the idle times to change so as to seek a new solution.

3.Adaptive Idle Time Coding Genetic Algorithm (AITCGA)

The algorithm is described below:

step 1 initialize a population pop(n) and let n=0;

step 2 evaluate the fitness function value fit(n) of each string in the population pop(n). If we find a satisfied solution ,we may cease the process , else let n=n+1 and continue;

step 3 get the best individual (decoded to be the best solution) from the fitness value. So, the end processing time can be obtain. If the end processing time of scheduling does not change even undergoing a relatively long time, we reduce the upper limit of idle time to bring forth a new population. This procedure can reduce the searching space. After that, save the population before the great fluctuation;

step 4 if the upper limit of idle time change, then a new population is produced to goto step 2;

step 5 if after evolution of a given times, the end processing time is still larger than the end processing time before the great change. It can be said that there is no optimum solution in the reduced searching space. So, we will restore the population before the great change to come back to a larger space to search what we want. And goto step 2.

step 6 select descendant pop(n) from the older generation pop(n-1).

step 7 crossover and mutate pop(n) according to certain strategy.

step 8 go to step 2.

We consider the effect of the parameter of upper limit of idle time. First, it can affect the search space. The complexity of the searching space is a important factor for every optimization approach. Even the search space is relatively complex and huge, the GA can often give a feasible solution. However, the efficiency can be improved if we reduce the search space to a more bounded range. In job shop scheduling, we can see that the essence of this problem is to search a solution in a $\mathbf{R}^n$ space which is nearly infinitive. In the knowledge based approaches, the heuristic method leads us to reach the feasible solution in spite that it can leads to a local minim in many cases. Essentially, GA is a probability approach rather than a determined approach. But it doesn't exclude a heuristic idea. We can take advantage both to contract the search space.

Second, the end processing time is hard to reduce if the upper limit of idle time is inappropriately selected, because the search space may be too complex and tremendous. There is a dilemma between the probability of find a solution and searching efficiency. when we use unchanging upper limit of idle time, the GA algorithm can find a fairly well solution, but not the best solution. For example, when we adopt upper limit of idle time 7, the algorithm repeated 300 times to get a solution of end processing time 42. But when we use upper limit of idle time 3, the end processing time of first found feasible solution is 42, and reached 35 which can be considered a nearly optimum solution after 300 times repetition.

The basic idea of this method is to self-adjust the upper limit of idle time. When the algorithm has come to its stable point with a upper limit of idle time, the upper limit of idle time is reduced and the population are adjusted to proper style.

4. Comparison of the Three Algorithms

AITCGA is more complicated than the other two algorithms. It combines the probabilistic transition and deterministic mechanism. The searching procedure changes greatly when the upper limit of idle time is changing. DITCGA and BITCGA are of the same complexity. However, BITCGA includes greater probability of mutation than DITCGA.

DITCGA and BITCGA are of almost the same searching efficiency. If the upper limit of idle time or the bits allocated to idle time is fixed, the searching space is consequently determined. If the space is small, it may be easy to find a optimum solution but with a risk that a solution may be not included in this space. On the other hand, when the searching space is large, it can be guaranteed that GAs can find the optimum solution but it needs relatively larger searching time. AITCGA is the tradeoff between the probability of reaching a solution and the searching speed.

V. SIMULATIONS

Simulation programs are written in MATLAB language. The technical requirement is[3]:

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

The weights: A=1,B=100 . The probability of crossover PC=0.7 . The probability of mutation PM=0.2. The number of population is 50. Fig1 and Fig2 are the results of DITCGA. Fig3 and Fig4 are the results of BITCGA. And Fig5 and Fig6 are the results of AITCGA.

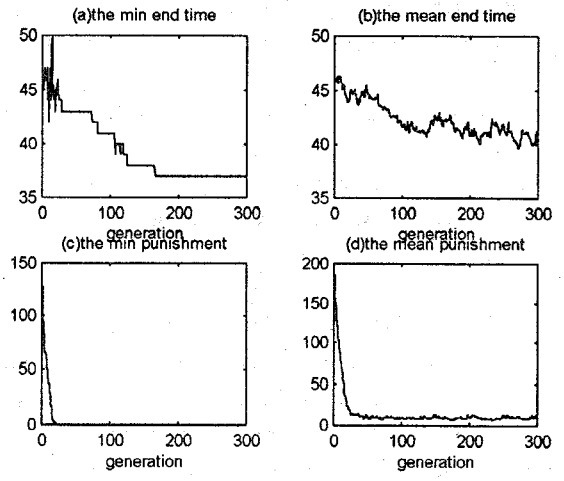

Fig 1. Result of DITCGA (a)the min end processing time, (b)the mean end processing time, (c)the min punishment,(d) the mean punishment.

The coded solution of DITCGA described is:

solution =

3 1 2 8 4 5 7 6 0 0 0 0 2 0 0 0
6 7 5 4 2 8 1 3 1 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 2

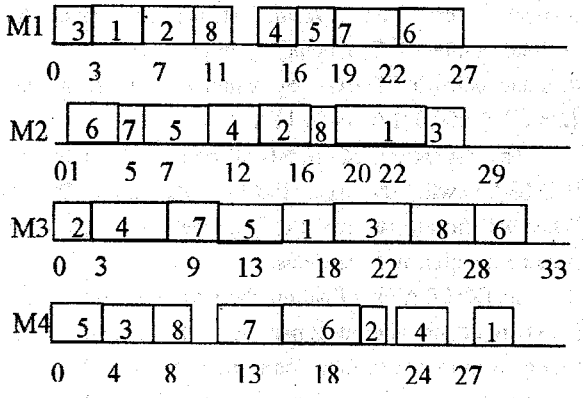

Fig 2. scheduling result of DITCGA. The number in rectangles are parts and the number under the lines are the start processing time of each part.

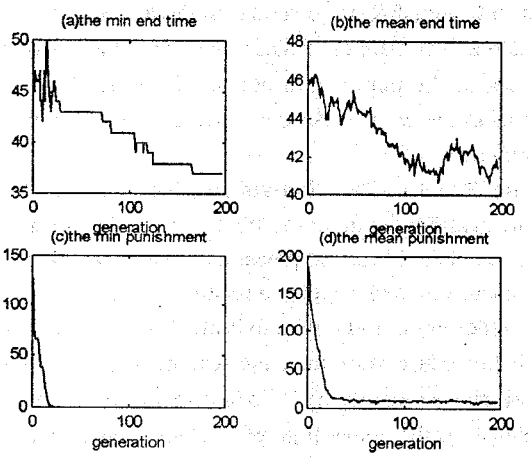

Fig 3. Result of BITCGA (a)the min end processing time, (b)the mean end processing time, (c)the min punishment,(d) the mean punishment.

The coded solution of BITCGA is:

solution =

3 2 1 8 4 7 5 6 000 000 000 000 010 001 000 000
6 7 5 4 2 8 1 3 001 000 000 000 000 000 001 000
2 4 7 5 1 3 8 6 000 000 000 000 000 000 000 000
5 3 8 7 6 2 4 1 000 000 000 010 000 001 000 000

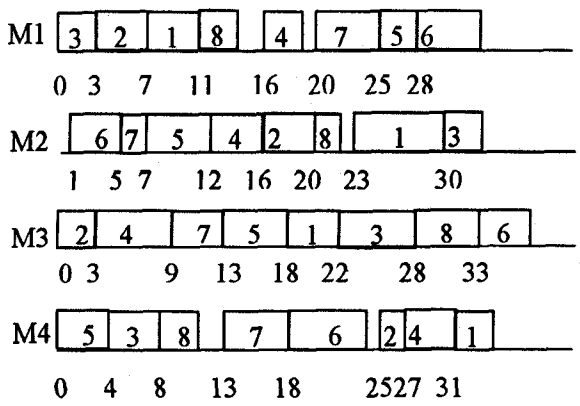

Fig 4. scheduling result of BITCGA. The number in rectangles are parts and the number under the lines are the start processing time of each part.

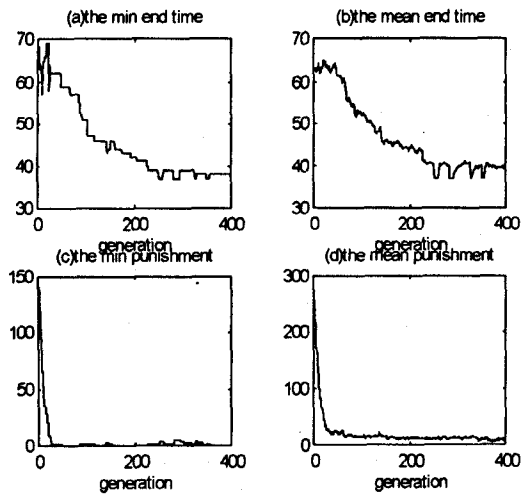

Fig 5. Result of AITCGA. (a)the min end processing time, (b)the mean end processing time, (c)the min punishment,(d) the mean punishment.

The coded solution of AITCGA is:

solution =

1 3 2 8 7 4 6 5 0 0 0 1 0 0 0 0
7 6 5 2 4 1 8 3 0 0 1 0 0 1 0 0
2 7 4 1 5 3 6 8 0 0 0 0 1 0 0 0
5 8 7 6 3 4 2 1 0 0 0 0 0 1 0 0

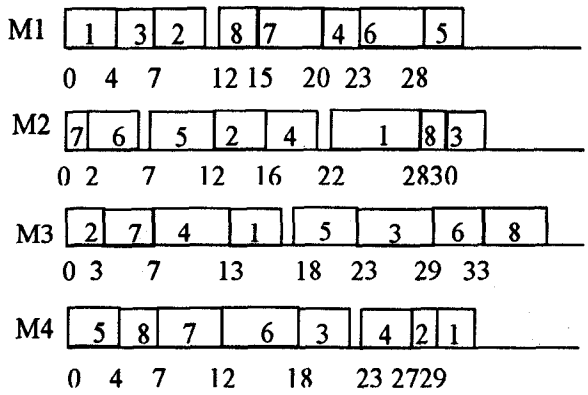

Fig 6. scheduling result of AITCGA. The number in rectangles are parts and the number under the lines are the start processing time of each part.

VI. CONCLUSIONS

Three GA based methods, DITCGA, BITCGA and AITCGA, are submitted in this paper. Idle time coding of job-shop is appropriate to narrow the solution space. The introduction of punishment factor simplifies the genetic processing. This approach is efficient with its shorter searching time. Form the simulation, we can see that, the upper limit of idle time is significant to the searching procedure. By using the combination of probabilistic transition and deterministic mechanism, AITCGA can select the upper limit of idle time automatically.These algorithms can satisfy the rapid response requirement in scheduling and has great application value .

REFERENCE:

[1].David E.Goldberg "Genetic Algorithms in Search Optimization & Machine Learning" Addison-Wesley Publishing Company, INC. 1989

[2].Zhang Changshui et.al. "A Genetic Algorithm of Solving Job-Shop scheduling Problem" Acta Electronica Sinica Vol.23 No.7 July 1995 1-5

[3].Shen Gang et.al "A Job-Shop Scheduling Solution with Neural Network" Acta Electronica Sinica Vol.23 NO.8 Aug.1995 48-51

[4].David B.Fogel."An Introduction to Simulated Evolutionary Optimization" IEEE Trans.Neural Networks, 1994, Jan.5(1)

[5].Wu Ying ,Li Bin. "Jobshop Scheduling Using Genetic Algorithms" the 3rd International Ccnference on Signal Processing . ICSP'96. Beijing