VNU Journal of Science, Mathematics - Physics 26 (2010) 213-221

Active schedules and a new hybrid genetic algorithm for the job shop scheduling problem

Nguyen Huu Mui*, Vu Dinh Hoa

Department of Information Technology, Hanoi University of Education, 136 Xuan Thuy, Hanoi Vietnam

Received 10 October 2010

Abstract. Active schedules and a new genetic algorithm for solving job shop scheduling problem are presented in this paper. In the proposed method, a chromosome representation of the problem is natural numbers, the GT algorithm is used to generate a set of active solutions, the mutation is implemented on the all machines concurrently. Especially, *we propose a new crossover operator that combines the uniform crossover operator with GT algorithm and is implemented on 3 parents.* The approach was tested on a set of benchmark programs and compared with other approaches. The computation results validated the effectiveness of the proposed algorithm.

Keywords: Jobshop, Scheduling, Schedule, Genetic Algorithm.

1. Introduction

Research in scheduling theory has evolved over the past fifty years and has been the subject of much significant literature with techniques ranging from unrefined dispatching rules to highly sophisticated parallel branch and bound algorithms and bottleneck based heuristics. Not surprisingly, approaches have been formulated from a diverse spectrum of researchers ranging from management scientists to production workers. However with the advent of new methodologies, such as neural networks and evolutionary computation, researchers from fields such as biology, genetics and neurophysiology have also become regular contributors to scheduling theory emphasising the multidisciplinary nature of this field.

The job shop scheduling problem - JSP is well known as one of the most difficult NP-hard ordering problems. In fact, we only know of few polynomially solvable cases of the JSP. Most JSP are NP-hard [1]. There have been some approaches for solving the JSP. For example, the *branch and bound* approach the *shifting bottle-neck* approach, the *simulated annealing* approach, *Genetic Algorithm-GA* methods etc. Recently, many researchers have used *hybrid* methods to solve JSP such as Chaoyong Zhang et al. [2], Lee Hui Peng et al. [3], M. Chandrasekaran [4], F. Guerriero [5], Rui Zhang et al. [6], etc. If JSP has been interested so much, it was because the importance of JSP both in theory and practice. This paper propose a new genetic algorithm for solving job shop scheduling problem.

* Corresponding author: E-mail: muithu@yahoo.com

2. Problem description

The JSP can be described as following:

A set of n jobs $\{J_i\}_{1\le i\le n}$ which is to be on a set of machines $\{M_j\}_{1\le j\le m}$. The problem can be characterized as:

1. Each job must be processed on each machine in the order given in a pre-defined technological sequence of machines.
2. Each machine can process only one job at a time.
3. The operation of job J_i is processed on machine M_j is called the operation O_{ij}.
4. The processing time of O_{ij} is denoted p_{ij}.
5. The starting time and completion time of an operation O_{ij} is denoted as s_{ij} and c_{ij}.
6. The time required to complete all the jobs is called the *makespan*, which is denoted as C_{max}. By definition, $C_{max} = \max\{c_{ij}\}_{1\le i\le n, 1\le j\le m}$.

3. Active schedules [5] and GT algorithm [7]

Schedules can be classified into one of following three types of schedules:

- Semi-active schedule: These feasible schedules are obtained by sequencing operations as early as possible. In a semi-active schedule, no operation can be started earlier without altering the processing sequences.
- Active schedule: These feasible schedules are schedules in which no operation could be started earlier without delaying some other operation or breaking a precedence constraint. Active schedules are also semi-active schedules. An optimal schedule is always active, so the search space can be safely limited to the set of all active schedules.
- Non-delay schedule: These feasible schedules are schedules in which no machine is kept idle when it could start processing some operation. Non-delay schedules are necessarily active and hence also necessarily semi-active.

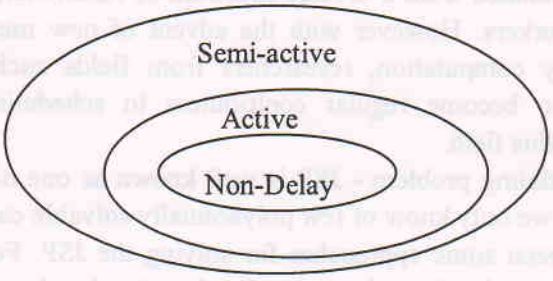

Fig. 1. Types of Schedules.

The JSP is given by the job sequence matrix $\{T_{ik}\}$ and processing time matrix $\{p_{ik}\}$. An active schedule can be generated by using the GT algorithm proposed by Giffler & Thompson [7]. The algorithm is described as following:

1. Initialize G as a set of operations that are first in the technological sequence (the first column of the matrix $\{T_{ik}\}$, i.e. $G = \{O_{1T_{11}}, O_{2T_{21}}, ..., O_{nT_{n1}}\}$. For each operation $O \in G$, $ES(O) := 0$ and $EC(O) := p(O)$.

2. Find the earliest complete operation $O_{*j} \in G$. A subset of G that consists of operations processed on machine M_j is denoted G_j (by 2.1).

3. Calculate the conflict set $C[M_j, k] \subset G_j$, where k -1 is the number operations already scheduled on M_j (by 2.2).

4. Select one of the operations in $C[M_j, k]$ randomly. Let the selected operation be O_{i*j}.

5. Schedule O_{i*j} as k^{th} operation on M_j; i.e. $S_{jk} := i*$, with its starting and completion times equal to $ES(O_{i*j})$ & $EC(O_{i*j})$: $s(O_{i*j}) = ES(O_{i*j})$; $c(O_{i*j}) = EC(O_{i*j})$.

6. For all operations $O_{ij} \in G_j \setminus \{O_{i*j}\}$:

- Update $ES(O_{ij})$ as: $ES(O_{ij}) := \max\{ES(O_{ij}), EC(O_{i*j})\}$.
- Update $EC(O_{ij})$ as: $EC(O_{ij}) := ES(O_{ij}) + p(O_{ij})$.

7. Remove O_{i*j} from G, and add an operation O_{is} that is the next to O_{i*j} in the technological sequence of J_i to G if such O_{is} exits. i.e., if $j = T_{ik}$ and $k < m$, then $s := T_{ik+1}$ and $G := (G\setminus\{O_{i*j}\} \cup \{O_{is}\}$. Calculate $ES(O_{is})$ and $EC(O_{is})$ as:

- $ES(O_{is}) := \max\{EC(O_{i*j}), EC(PM(O_{is}))\}$.
- $EC(O_{is}) := ES(O_{is}) + p(O_{is})$.

8. Repeat from step 1 to step 7 until all operations are scheduled.

9. Output the solution matrix $\{S_{jk}\}$as the active schedule obtained with set of starting and completion times $\{s(O_{ij})\}$ and $\{c(O_{ij})\}$, where $i = S_{jk}$.

4. The proposed genetic algorithm

4.1 Solution coding

We suppose there are n jobs are given to be processed on m machines. The number of operations of job J_i is denoted $job[i]$ ($job[i] \le m$, with every i). The sum of operations has to process of the all jobs are $L = \sum_{i=1}^{n} job[i]$. We code operations of J_1 from 1 to $job[1]$, of J_2 from $job[1] + 1$ to $job[1] + job[2]$,..., of J_n from $job[1] + job[2] + ... + job[n-1] + 1$ to L. Thus, one solution is one certain permutation of the natural numeral chain {1, 2, 3, ..., L} satisfy of the problem constraints.

By definition, $C_{max} = \max\{c_{ij}\}_{1 \le i \le n, 1 \le j \le m}$.

Table 1. The JSP with 3 jobs and 3 machines

Job	Machine (processing time)		
1	1 (3)	2 (3)	3 (3)
2	1 (2)	3 (3)	2 (4)
3	2 (3)	1 (2)	3 (1)

The problem is equivalently represented by the job sequence matrix $\{T_{ik}\}$ and processing time matrix $\{p_{ik}\}$as following:

$$\{T_{ik}\} = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{vmatrix} \qquad \{p_{ik}\} = \begin{vmatrix} 3 & 3 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \end{vmatrix}$$

For example, the problem with 3 jobs and 3 machines given in Table 1. Operations are coded by natural numbers as in Table 2.

Table 2. The operations are coded by natural numbers

Jobs	Operation coding		
J_1	1	2	3
J_2	4	5	6
J_3	7	8	9

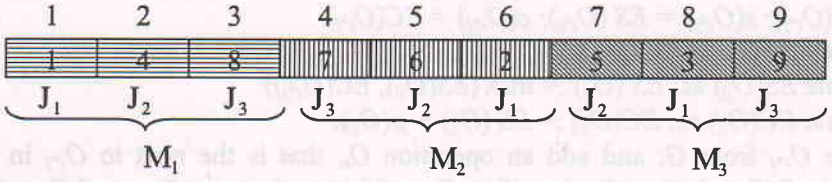

Fig. 2. A valid solution for the problem job shop 3 × 3.

Explanation: According to $\{T_{jk}\}$, operations 1, 4, 8 are processed on machine 1. Thus, codes on the M_1 are one certain permutation of the chain {1, 4, 8}. Similarly, codes on the M_2 are one certain permutation of the chain {2, 6, 7}, codes on the M_3 are one certain permutation of the chain {3, 5, 9}. A valid solution for the problem may be as Figure 2. This solution can be showed by a solution matrix S_{jk}. Where, $S_{jk} = i$, means that k^{th} operation on machine M_j is job J_i.

$$\{S_{jk}\} = \begin{vmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{vmatrix}$$

4. 2 Generating a set initial solutions

JSP represented by the technological sequence matrix $\{T_{ik}\}$, and the processing time matrix $\{p_{ik}\}$. A initial active schedule for the problem can be generated by using the GT algorithm [6] presented in the section 3.

4.3 Construct fitting function

The fitting function is represented as following:

Fitness = $M - C_{max}$, there C_{max} is makespan of the solution, M is parameter given for change min problem to max problem (because genetic algorithm only applies immediately for max problem).

4.4 Genetic operators

a) Selection operator

According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods to select the best chromosomes. In this paper, we use the " Roulette Wheel Selection". Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have.

b) Mutation operator

- Random selection of one operation ($ope1$) of parent. Define the machine (M_{ope1}) which performs the operation that has just been selected and define the position ($pos1$) of this operation in the parent.

- Random selection one operation (*ope*2) of parent. Define the machine (M_{ope2}) which performs the operation has just been selected and define the position (*pos*2) of this operation.
- If $M_{ope1} = M_{ope2}$ then implement mutation.The getting result is child chromosome. On the other hand, the parent chromosome is kept intact.
- For all operations of the child chromosome, update its starting and completion times.

Example, the parent is selected for mutation as following:

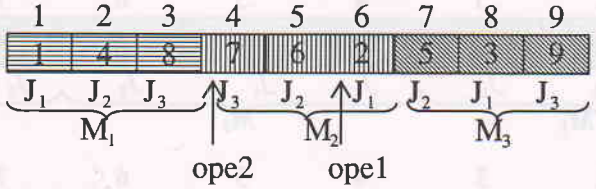

Fig. 3. The parent chromosome for insert mutation.

+ For example: *ope*1 = 2 ---> M_{ope1} = 2 and *pos*1 = 6.
+ *ope*2 = 7 ---> M_{ope2} = 2 and *pos*2 = 4.
+ $M_{ope1} = M_{ope2}$ ---> insert operation 2 in to position 4. We have child:

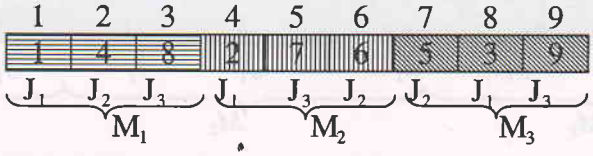

Fig. 4. The child chromosome after insert mutation.

The mutation operator is implemented on the all machines, then the random selection is implememted m times.

c) Crossover operator

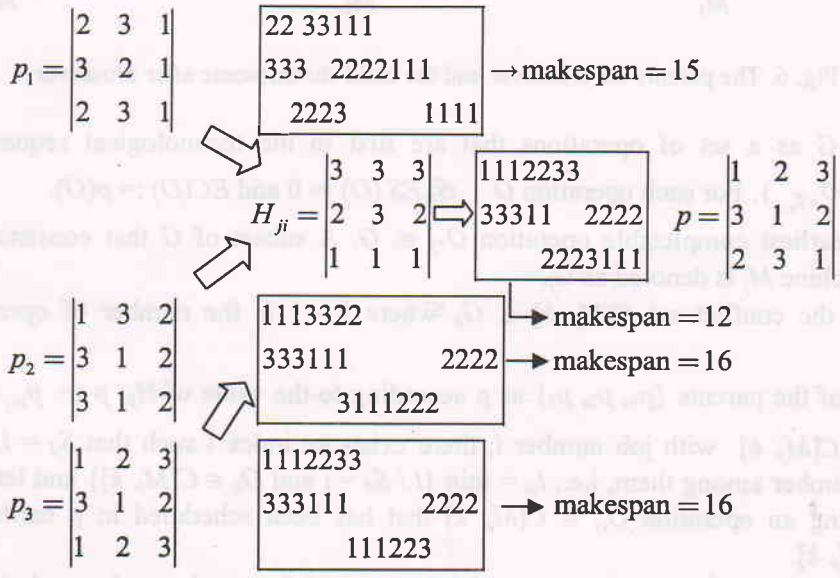

Fig. 5. The uniform crossover use GT algoriths with 3 parents.

A scheduling problem represented by $\{T_{ik}\}$, the technological sequence matrix, and $\{p_{ik}\}$, the processing time matrix. The crossover operator combines the uniform crossover with GT algorithm and is implemented on 3 parents p_1, p_2 and p_3. This parents are showed by correlative solution matrixs $S^1 = \{S^1_{jk}\}$, $S^2 = \{S^2_{jk}\}$ and $S^3 = \{S^3_{jk}\}$. Genes of offspring $p = \{S_{jk}\}$ are ones from each parent. The crossover operator can be described as:

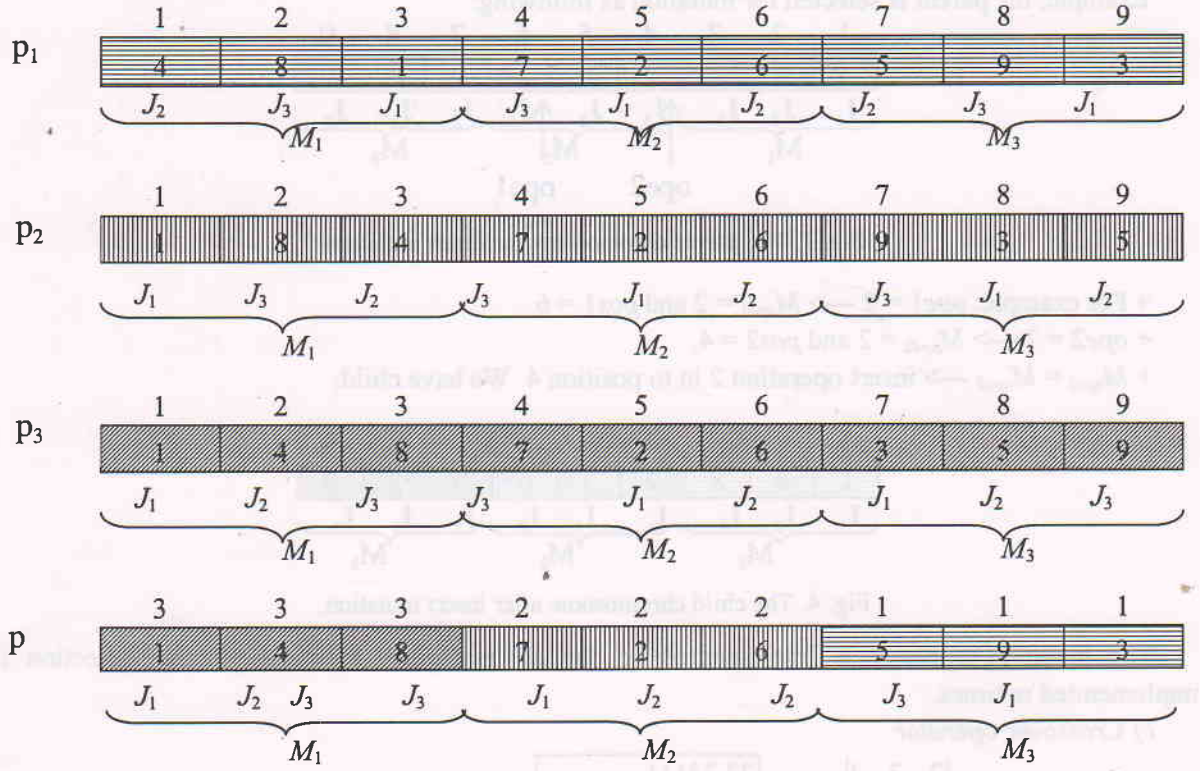

Fig. 6. The parents for crossover and the child chromosome after crossover.

1. Initialize G as a set of operations that are first in the technological sequence, i.e., $G = \{O_{1T_{11}}, O_{2T_{21}}, ..., O_{nT_{n1}}\}$. For each operation O $\in G$, $ES(O) := 0$ and $EC(O) := p(O)$.

2. Find the earliest completable operation $O_{*j} \in G$. A subset of G that consists of operations processed on machine M_j is denoted as G_j.

3. Calculate the conflict set $C[M_j, k] \subset G_j$, where $k - 1$ is the number of operations already scheduled on M_j.

4. Select on of the parents $\{p_1, p_2, p_3\}$ as p according to the value of H_{ji}, $p := p_{H_{ij}}$ and $S^p = s^{H_{ij}}$. For earch $O_{ij} \in C[M_j, k]$ with job number i, there exists an index l such that $S_{jl} = i$. Let l_m be the smallest index number among them, i.e., $l_m = \min \{l / S_{jl} = i \text{ and } O_{ij} \in C[M_j, k]\}$ and let $r := S_{jlm}$. This results in selecting an operation $O_{rj} \in C[M_j, k]$ that has been scheduled in p earliest among the members of $C[M_j, k]$.

5. Schedule O_{rj} as the k^{th} operation on M_j; i.e. $S_{jk} := r$, with its starting and completion times equal to $ES(O_{rj})$ and $EC(O_{rj})$ respectively: $s(O_{rj}) = ES(O_{rj})$; $c(O_{rj}) = EC(O_{rj})$.

6. For all $O_{ij} \in G_j \setminus \{O_{rj}\}$ update:

- $ES(O_{ij})$ as: $ES(O_{ij}) := \max\{ES(O_{ij}), EC(O_{rj})\}$.
- $EC(O_{ij})$ as: $EC(O_{ij}) := ES(O_{ij}) + p(O_{ij})$.

7. Remove O_{rj} from G (and therefore from G_j), and add operation O_{rs} that is the next to O_{rj} in the technological sequence to G if such O_{rs} exists; i.e., if $j = T_{ik}$ and $k < m$, then $s := T_{i,k+1}$ and $G := (G\setminus\{O_{rj}\} \cup\{O_{rs}\}$.

Calculate $ES(O_{rs})$ and $EC(O_{rs})$ as:

- $ES(O_{rs}) := \max\{EC(O_{rj}), EC(PM(O_{rs}))\}$.
- $EC(O_{rs}) := ES(O_{rs}) + p(O_{rs})$.

8. Repeat from step 1 to step 7 until all operations are scheduled.

9. Output the solution matrix $\{S_{jk}\}$ as the active schedule obtained with the set of starting and completion times $\{s(O_{ij})\}$ and $\{c(O_{ij})\}$ respectively, where $i = S_{jk}$. Figure 8 shows an example of the uniform crossover operator with GT algorithm and is implemented on 3 parents p_1, p_2 and p_3 with an inheritance matrix H_{ji}.

The parents for crossover and the child chromosome after crossover are showed by picture as Figure 6.

4.5 Evolutional algorithm

The evolutional algorithm for the job shop problem is described as following:

```
Procedure GA_JSP
Begin
        t = 0
        Initialize P(t)    {using GT algorithm}
        Evaluate P(t)
        While not termination condition do
        Begin
           Build intermediate solution set P'(t):
                - Perform the mutation operator for P(t), we get P1(t)
                - Perform the crossover operator for P(t), we get P2(t)
                - P'(t) = P(t) ∪ P1(t) ∪ P2(t)
            Evaluate P'(t)
            t = t + 1
            Select P(t) from P'(t-1)
            Evaluate P(t)
          If Eval(P(t - 1)) ≥ Eval(P(t)) then t = t - 1
        End
End
```

5. Experimental results

Based on the proposed method, we implemented a program to find an optimal schedule for the JSP. The program ran on the PC with Core 2 Dual CPU and is applied to Muth and Thompson's benchmark problems. The results as in Table 3. Among the research works applied GA to the JSP,

Methods proposed by Yamada are counted as one of the best methods until now. Thus, for the sake of comparison we is chosen the GA and GT/GA methods of Yamada [8] in Table 4 and Table 5.

Table 3. Experimental results of the proposed method

Problems	Population Size	Generation number	Crossover rate (pc)	Mutation rate (p_m)	result	Real optimal result
mt06 (6 × 6)	50	200	0.5	0.05	55	55
mt10 (10 × 10)	500	200	0.8	0.05	930	930
mt20 (20 × 5)	500	200	0.8	0.05	1185	1165

Table 4. Experimental results of the SGA method of Yamada

Problems	Population size	Generation number	Crossover rate (pc)	Mutation rate (pm)	result	Real optimal result
mt06	100	200	0.9	0.01	55	55
mt10	1000	200	0.9	0.01	965	930
mt20	2000	200	0.9	0.01	1215	1165

Table 5. Experimental results of the GA/GT method of Yamada

Problems	Population size	Generation number	Crossover rate (pc)	Mutation rate (pm)	result	Real optimal result
mt066	100	200	0.9	0.01	55	55
mt10	1000	200	0.9	0.01	930	930
mt20	2000	200	0.9	0.01	1184	1165

6. Conclusion

In this paper we presented of active schedules and proposed a new genetic algorithm for the JSP. In the proposed method, we sensibly combined of the results of the predecessors with our new propositions. We proposed a genetic algorithm for the JSP with a new crossover operator. Base on proposed method, we implemented a program to find an optimal schedule for the JSP. The program ran with inputs are the mt benchmark problems and provided good results. The program gave the real optimal solution with small and medium-sized testing problems. However, for the large-sized testing problems as mt20, the proposed method has not found an optimal solution. In the future, we will try to find new genetic operators more suitable for the JSP.

References

[1] P. Lopez, F. Roubellat, *Production Scheduling*, ISTE Ltd, 2008.

[2] Chaoyong Zhang, Yunqing Rao, Peigen Li, ZaiLin Guan, An very fast TS/SA algorithm for the job shop scheduling problem, *Computers and Operations Research* 35(1) (2007) 282.

[3] Lee Hui Peng, Sutinah Salim, *A Modified Giffer and Thompson Genetic Algorithm on the job shop scheduling problem*, MATEMATIKA (University Teknology Malaysia) 22(2) (2006) 91.

[4] M. Chandrasekaran, P. Asokan, S. Kumanan, T. Balamurugan, S. Nickolas, Solving job shop scheduling problems using artificial immune system, *International Journal Advanced Manufacturing Technology,* 31 (2006) 580.
[5] F. Guerriero, Hybrid Rollout Approaches for the Job Shop Scheduling Problem, *Jounal Optimization Theory and Applications,* 139 (2008) 419.
[6] Rui Zhang, Cheng Wu, A hybrid approach to large-scale job shop scheduling, *Application Intelligence*, 32 (2010) 47.
[7] B. Giffler, G.L. Thompson, Algorithms for Solving Production Scheduling Problems, *Operations Research*, 8(4) (1960) 487.
[8] T. Yamada, *Studies on Metaheuristics for Jobshop and Flowshop scheduling problems*, Kyoto University, Kyoto - Japan (2003).