



Dual Hybrid Algorithm for Job Shop Scheduling Problem

Do Tuan Hanh

Hanoi National University of Education
136 Xuân Thủy Cầu Giấy Hanoi Vietnam

Vu Dinh Hoa

Hanoi National University of Education
136 Xuân Thủy Cầu Giấy Hanoi Vietnam

Nguyen Huu Mui

Hanoi National University of Education
136 Xuân Thủy Cầu Giấy Hanoi Vietnam

Abstract

This paper presents a dual hybrid method to solve job shop scheduling problem (JSP). We use a genetic algorithm (GA) combined with the dual hybrid approach to obtain a new genetic algorithm (GTD-GA). The algorithm proposed new hybrid approach called dual hybrid, and new data organization with the purpose of maximizing the GT algorithm. They're building program for the experimental schedule finding maximum job shop problem. In order to prove the effectiveness of the algorithm, we ran on the standard given by Muth and Thompson and compare with the results of Yamada used GT-GA for JSP. Moreover, we also compare them with the local search genetic algorithm of M. Kebabla, L. H. Mouss for the job shop problem.

Keywords: Jobshop Scheduling, Schedule, Genetic Algorithm.

1. INTRODUCTION

Job shop scheduling problem (JSP) is one of the well-known classic problems. There are many approaches to solve it and the first work on this field is perhaps the paper of Fiedman and Akers [3].



JSP general statements for a set n jobs $\{J_j\}_{1 \leq j \leq n}$ and a set of m machines $\{M_i\}_{1 \leq i \leq m}$ must satisfy the following conditions:

1. At the same time each machine can process only a single job.
2. The set of n jobs $\{J_j\}_{1 \leq j \leq n}$ has processed on all m machines $\{M_i\}_{1 \leq i \leq m}$.
3. Each job must be processed on each machine in a given order of operations. The sequence of operations performed each task in turn on the machine called sequential technology.
4. The handling of a job J_j at the machine M_i is called the operation O_{ij} .
5. Each operation O_{ij} executed must be uninterrupted on the given machine M_i .
6. The beginning processing time and the finishing time of operation O_{ij} are denoted respectively by $T_{S_{ij}}$ and by $T_{F_{ij}}$. The processing time operation O_{ij} is denoted by $Time_job_{ij}$.
7. The time to complete the processing of all jobs is called makespan and is denoted by $Time_F_{max}$.

The requirement of JSP problem is to determine a schedule (order processing jobs on each machine) with the possible smallest $Time_F_{max}$.

Example: A JSP for 4 machines and 4 jobs including sequential technology of machines for each job with the processing time of each job on each machine (in parentheses) is given in Table 1.

Table 1. JSP 4 jobs, 4 machines

Jobs	Machine (processor time)			
1	1(1)	2(4)	3(3)	4(15)
2	2(3)	3(5)	4(12)	1(7)
3	4(8)	1(9)	2(16)	3(10)
4	2(11)	3(6)	1(2)	4(13)



Job shop scheduling problem is a NP-hard problem and is one of the most difficult combinatorial optimization problems. JSP problem can be solved with a polynomial algorithm only in some special cases. For $n = 2$, Akers [2] proved that the problem of finding JSP can be solve with a polynomial time algorithm using shortest path algorithm. Using a method of Kravchenko and Sotskov [11], Brucker [5] give a polynomial time algorithm for the JSP with $m = 2$ machines and $n = k$ (k is a constant) jobs. Except for some special cases mentioned above, the remaining cases of JSP are NP-hard. This is the result of a research of Shaklevich Sotskov [15]. Table 2 summarizes the complexity of JSP.

Table 2. The computational complexity of JSP

		Number of m machines			
		2	3	Constant	Depending on the
The number n jobs	2	P	P	P	P
	3	P	P	P	NP-hard
	Constant	P	P	NP-hard	NP-hard
	Depending on the	P	NP-hard	NP-hard	NP-hard

JSP was published first time in 1955. A statement for this problem can be found in a paper of Akers and Friedman. Followed them are the studies of Bowman (1959) and Wagner (1959) in which there was only one job that can be handled on maximal 3 computers. The year 1963 marked an important milestone in JSP when Fisher and Thompson [12, 7] studied a JSP with 10 jobs and 10 machines. This test was considered to be one of the most intractable of JSP because the optimal solution was found ever after more than 20 years later. They proposed GT algorithm as active schedules. The result of H. Fisher and GLThompson [7] can be found in the book "Industrial Scheduling", edited by JF-Muth and GL- Thompson [12] in 1963.

The method of solving the problem as 'branching method approach' introduced by GH Brooks and CR White [4] who pioneered solving the job shop. Finally, in 1985, E. Carlier and Pinson [6] succeeded in solving optimization problems Mt10 using algorithms branch access. Finding optimal solution for JSP with unsightly test is a major challenge because it is a NP-hard problem. Thus, approximate methods have been developed. One of the approximate methods considered to be most effective is genetic algorithm.



John Holland [9] is considered as the founder of genetic algorithm. GA (Genetic Algorithms) is characterized by a search strategy based on population and by the genetic operators: selection, mutation and crossover. Nakano and Yamada [13] developed the first classical hybrids method and obtained quite good results. Yamada [16] developed GT-GA algorithm with many significant achievements. In 2012, M. Kebabla, LH Mouss [10] offered “A local search genetic algorithm for the job shop scheduling problem”. Recently, Nguyen Huu Mui and Vu Dinh Hoa [14] proposed a new hybrid method coded calendars by natural numbers and a new crossover operator combined 3 individual parent to an individual children.

After a period of researching methods to solve the problem, we find a new hybrid method with new mutations which may be more efficiently. Our innovations are:

1. Reorganize database to reduce leverage calculations suitable grafting and new mutations.
2. Use the GT algorithm taking good jealous of mutant to a better generation, with the aim to get earlier convergence than the GT-GA.
3. Apply new crossover method: from 2 individual parents P1, P2 hybridized according to GT algorithm to F1 and F2 offspring, then proceed hybridized form F1 and F2 to FF1 and FF2. With this new crossover approach, this starting personal space can be expanded to diverse selection closer to a optimal experience.

2. CONTENT

In what follows, we present a method of organizing new data processing allows new mutations and new crossover.

2.1. New data encryption

Suppose there are n jobs to be processed on m machines. T matrix is called matrix sequentially technology. We will encode matrix C where C_{ij} is 1 if the job j is done on machine i at some time, and C_{ij} is 0 if the job j is not done on machine i . L_T is the matrix where L_T_{ij} is the next machine which performs the job i , according the sequential technology, after this job was performed on the j machine. F_T is the matrix where F_T_{ij} is the machine which performed the job i just before it has to be performed on the j machine according the sequentially technology.



For example is the problem of 4 jobs, 4 machines given in Table 1. T matrix and transformation matrix C, L_T, F_T are given in Figure 1, respectively:

$$\begin{array}{l} \{T_{ij}\} = \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \end{array} \end{array} \quad \begin{array}{l} \{C_{ij}\} = \begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \end{array}$$

$$\begin{array}{l} \{L_T_{ij}\} = \begin{array}{cccc} 2 & 3 & 4 & 0 \\ 0 & 3 & 4 & 1 \\ 2 & 3 & 0 & 1 \\ 4 & 3 & 1 & 0 \end{array} \end{array} \quad \begin{array}{l} \{F_T_{ij}\} = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ 4 & 0 & 2 & 3 \\ 4 & 1 & 2 & 0 \\ 3 & 0 & 2 & 1 \end{array} \end{array}$$

Figure 1. The matrix T, C, L_T, F_T

2.2. Mutation operator

Operator mutations on individual parent P were carried out by following these steps:

1. Choose a random matrix H where H_{ij} is 0 or 1, apply the GT algorithm to generate mutant P_C using the matrix H.
2. Create G, the set of the first stage in the sequence of all these technology jobs, using the matrix C (only if C_{ij} is 1) so that if $O_{ij} \in G$ then $T_S_{ij}=0$; and $VT[i] = 1$.
3. Construct the set $V_G \subset G$ satisfying that if $O_{ij} \in V_G$, then T_S_{ij} is the possible minimum value.
4. Construct the set $E_G \subset V_G$ satisfying if $O_{ij} \in E_G$, then $H_{iVT[i]}=1$ and $j = P_{iVT[i]}$.
5. If $E_G \neq \emptyset$, then choose randomly $O_{ij} \in E_G$. If $E_G = \emptyset$, choose $O_{ij} \in V_G$ randomly.
6. $P_C_{iVT[i]} = j$; $VT[i] = VT[i]+1$; Remove O_{ij} from the set G; if $k \neq 0$ (with $k = L_T_{ij}$), then add O_{ik} in G.
7. Update to T_S_{ij} if $O_{ij} \in G$ using the following formula:
$$r = P_C_{iVT[i]-1}$$

$$k = F_T_{ij}$$

$$T_S_{ij} = \text{Max}(T_S_{ir} + \text{Time_job}_{ir}, T_S_{kj} + \text{Time_job}_{kj})$$
8. Go to step 3 until all stages are scheduled in mutant P_C.



If mutant P_C is less adaptive than individuals P, then we repeat these steps to create mutant P or stop once reaching the maximum number of mutations (through variable input level).

For example, individual parent P and H matrix generated mutant P_C in Figure 2:

$$\{P_{ij}\} = \begin{matrix} 1 & 3 & 4 & 2 \\ 4 & 1 & 2 & 3 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \end{matrix} \quad \{H_{ij}\} = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{matrix} \quad \{P_{C_{ij}}\} = \begin{matrix} 1 & 3 & 2 & 4 \\ 2 & 1 & 4 & 3 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \end{matrix}$$

Figure 2. The P parent and mutant P_C matrix H

Time_F_{max} Individual's parent P is: 66

Time_F_{max} P_C of mutants is: 48

2.3. GTD Crossover

Operator 2 hybrid created on individual P1 and P2 parents was conducted according to the following steps:

1. Choose a random matrix H which H_{ij} the value 0 or 1. Using genetic algorithms get better GT parents P1 and P2 through the matrix H to generate F1 and F2 offspring. From the F1 and F2 be the application of the GT get good genes of their parents through the matrix H to generate FF1 and FF2 offspring.

2. Create G is the set of the first stage in the sequence of all these technologies jobs through the matrix C ($C_{ij} = 1$ will be included in G), if $O_{ij} \in G$ is $T_{S_{ij}}=0$; and $VT[i] = 1$.

3. Construct set $V_G \subset G$ satisfying if $O_{ij} \in V_G$, then $T_{S_{ij}}$ is the minimum value that can be obtained.

4. Construct set $E_G_F \subset V_G$ satisfying if $O_{ij} \in E_G$, then $H_{iVT[i]j} = 1$ and $j = P1_{iVT[i]}$.

5. If $E_G_F \neq \emptyset$, then choose randomly $O_{ij} \in E_G$, then trip to step 8.

6. If $E_G_F = \emptyset$, Set $E_G_M \subset V_G$ satisfied if $O_{ij} \in E_G$, then $H_{iVT[i]j} = 0$ and $j = P2_{iVT[i]}$.

7. If $E_G_M \neq \emptyset$, then choose randomly $O_{ij} \in E_G_M$. Conversely, choose $O_{ij} \in V_G$ randomly.

8. $F1_{iVT[i]j} = j$; $VT[i] = VT[i]+1$; O_{ij} removed from the set G; if $k \neq 0$ (with $k =$



$L_{T_{ij}}$) then add O_{ik} in G .

9. Update to $T_{S_{ij}}$ if $O_{ij} \in G$ using the following formula:

$$r = P_{C_{iVT[i]-1}}$$

$$k = F_{T_{ij}}$$

$$T_{S_{ij}} = \text{Max}(T_{S_{ir}} + \text{Time_job}_{ir}, T_{S_{kj}} + \text{Time_job}_{kj})$$

10. Go to step 3 until all phases are scheduled in individual F1 crossovers.

This methodology improves our differences. We produce F2 simultaneously in step 4 with additional conditions: $j_1 = P2_{iVT[i]}$, with additional conditions in Step 6: $j_1 = P1_{iVT[i]}$ and in step 8 with $F2_{iVT[i]} = j_1$. Using F1 and F2 as the parents of FF1 and FF2 to produce these steps as well as adopted hybrid matrix H. From two individual P1 and P2 parents we are offering first creation of four new individual at F1, F2, FF1, FF2 with the same operators number as the creation of our first offspring using GT_GA algorithm. For example, we give two individual crossover parents P1 and P2 through the matrix H as shown in Figure 3:


	1 3 4 2		1 3 4 2		1 0 1 0
	2 4 1 3		4 1 2 3		0 1 0 0
$\{P1_{ij}\} :$	2 4 1 3	$\{P2_{ij}\} :$	4 1 2 3	$\{H_{ij}\}$	0 0 1 1
	3 2 4 1		3 4 1 2		1 0 1 0

Figure 3. Parents P1, P2 and matrix H

	1 3 4 2		1 3 4 2
	4 2 1 3		2 1 4 3
$\{F2_{ij}\} =$	4 2 1 3	$\{F1_{ij}\} =$	2 1 4 3
	3 4 1 2		3 2 1 4
	1 3 4 2		1 3 4 2
	2 4 1 3		4 2 1 3
$\{FF1_{ij}\} =$	2 4 1 3	$\{FF2_{ij}\} =$	4 2 1 3
	3 2 4 1		3 4 1 2

Figure 4. The generated in children F1, F2, FF1, FF2



Table 4. Time_Fmax of individual values P1, P2, F1, F2, FF1, FF2

Individual	P1	P2	F1	F2	FF1	FF2
Time_F _{max}	66	52	48	66	66	50

2.4. Operator selection

Operators for selectively choosing individual n_gen generation $t + 1$ are described as follows:

1. Construct intermediate solution set $P'(t)$ applying mutation and hybridization:

- + Apply mutation operator for $P(t)$ is $P_1(t)$.
- + Apply crossover operator for $P(t)$ is $P_2(t)$.
- + $P'(t) = P(t) \cup P_1(t) \cup P_2(t)$.

2. Choose one individual with the best adapted from the first generation to generation t , called P_{min} , in the following way:

Choose individual which may be the most adaptable in the set $P'(t)$, called P'_{min} , if $t = 1$ then $P_{min} = P'_{min}$. If $t > 1$ and if P'_{min} have better adaptability than P_{min} , then replace P_{min} with P'_{min} .

3. Get $n_{gen} - 1$ random individuals remaining in $P'(t)$ according to the principles of the wheel.

2.5. Genetic Algorithms for JSP

Void GA_JSP ()

```
{
    t ← 0 ;                // t is the number of generations of evolution
    Initialize P (t);      // GT algorithm
    while ( t < n_g ) do    //n_g is generations
    {
        Construct intermediate solution set P '(t) by applying mutation and
        hybridization:
        + Apply mutation operator for P (t) to obtain P1 (t) in Section 2.2
        + Apply crossover operator for P (t) to obtain P2 (t) in Section 2.3
        +  $P'(t) = P(t) \cup P_1(t) \cup P_2(t)$ 
        Selective P (t) from P '(t) in Section 2.4
        t = t+1 ;
    }
}
```



The correctness of the algorithm is confirmed by the following characteristics:

1. When initializing n_{gen} individual's initial population we use GT algorithm to generate a schedule, so each individual children is born with an active schedule [13] [16].
2. During mutation, mutant individuals are modified by taking a number of individual genes fathers through the matrix H applying GT algorithm, so it remains an active schedule.
3. Allow crossovers from two parent individuals P1 and P2 take individual genes through the parent matrix H applying GT algorithm to generate children F1 and F2 schedule, so each individual children is born by this calendar will be an active one. From F1 and F2 and through the matrix H obtained simultaneously applied genetic algorithms GT with calendar children FF1 and FF2 is a schedule which actively.
4. The algorithm will stop when it runs through all generations.
5. Since the algorithm maintains the best solution in the population, before or after the selection, so the algorithm is converged to the global optimum which is not reduced by the properties of the next generation. These results are discussed with schema theorem and are proved in [8].

3. RESULT AND CONCLUSIONS

Based on the method proposed in this paper, we have designed a program in C++ language to run tests on a PC with a processor Core 2 Dual-speed 2 4GHz. The performance of the algorithm was analyzed on a set of benchmarks on the job-shop scheduling problem instances from literature. The size of the benchmark instances varied from 10 to 20 jobs and from 5 to 20 machines. We consider (FFT6, FT10, FT20) proposal by Fisher and Thompson [12]; three problems (ABZ5, ABZ6, ABZ7) generated by Adams, Balas & Zawack [1]. Table 5 presents the result compared our proposed GTD-GA algorithm with the GT-GA by Yamada [8].

Table 5. Comparison of GTD-GA and GTD-GA

Instance(size)	Best generated solution with GT-GA	Best generated solution with GTD-GA	Best known solution
FT06 (6 × 6)	55	55	55
FT10 (10 × 10)	930	930	930
FT20 (20 × 5)	1185	1182	1165



FT06 proposal for optimal results also FT10 for optimal results are obtained only after 80 trials. GT-GA algorithm by Yamada [13] [16] proposed FT06 gives optimal results after 2 trials, FT10 gives optimal results after 100 trials. Table 6 compares the results of GTD-GA algorithm with the results of GLS algorithm proposed by M. Kebabla, LH Mouss [10].

Table 6. Comparison of GT-GA and GLS

Instance(size)	Best generated solution with GT-GA	Best generated solution with GA in 5 attempts	Best generated solution with GLS in 5 attempts	Best known solution
ABZ5 (10 × 10)	1234	1238	1234	1234
ABZ6 (10 × 10)	943	944	943	943
ABZ7 (15 × 20)	665	680	667	656

In the tests, ABZ7 GTD-GA algorithm gives better results than GLS algorithm [10]. The comparing test results may indicate that the method we propose has the following advantages:

1. The organization using the 2 matrices L_T and F_T reduces the numbers of search operations.
2. When we use a mutation associated with GT algorithms, it searches nearby so quickly and find a better value mutation father and minimize the number of calculations.
3. Our first children are 4 instead of getting only one child by the same parents and with the same number of operators.

Our designed program is tested on a standard data of Muth and Thompson and gives better results than the latest algorithm GA. However, for problems with large-size test ABZ7 (15 × 20), the program has to find the optimal schedule. In the future we will continue to develop out new methods with higher efficiency.

REFERENCES

- [1] Adams, J., Balas, E. and Zawack, D., 1988. *The Shifting Bottleneck Procedure for Job Shop Scheduling*, Management Science, Vol. 34, No. 3, pp. 391–401.



- [2] Akers, S.B., 1956. *A graphical approach to production scheduling problems*, Operations Research, Vol. 4, No. 2, pp. 244-245.
- [3] Akers, S.B. and Friedman, J., 1955. *A non-numerical approach to production scheduling problems*, Operations Research, Vol. 3, No. 4, pp. 429-442.
- [4] Brooks, G. H. and White, C. R., 1969. *An algorithm for finding optimal or near optimal solutions to the production scheduling problem*, The Journal of Industrial Engineering, Vol. 16, No. 1, pp. 34-40.
- [5] Brucker, P., 1994. *A polynomial time algorithm for the two machines Job Shop scheduling problem with a fixed number of jobs*, OR Spektrum, Vol. 16, No. 1, pp. 5-7.
- [6] Carlier, J. and Pinson, E., 1989. *An algorithm for solving the job-shop problem*, Management Science, Vol. 35, No. 2, pp. 164-176.
- [7] Fischer, C. and Thompson, G. L., 1963. *Probabilistic learning combinations of local job-shop scheduling rules*, In Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, pp. 225-251.
- [8] Günter Rudolph, 1994. *Convergence Analysis of Canonical Genetic Algorithms*, IEEE Transactions on Neural Networks, special issue on evolutionary computation, Vol. 5, No. 1, pp. 96-101.
- [9] Holland, J. H., 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.
- [10] Kebabla, M. and Mouss, L. H., 2012. *A local search genetic algorithm for the job shop scheduling problem*, Revue des Sciences et de la Technologie – RST, Vol. 3, No. 1, pp. 61-68.
- [11] Kravchenko, S. A. and Sotskov, Y. N., 1996. *Optimal makespan schedule for three jobs on two machines*, ZOR - Mathematical Methods of Operations Research, Vol. 43, pp. 233-238.
- [12] Muth, J. F. and Thompson, G. L., 1963. *Solving Production Scheduling Problems*, In Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, Ch. 3.
- [13] Nakano, R. and Yamada, T., 1991. *Conventional genetic algorithm for job shop problems*, In Proceedings of International Conference on Genetic Algorithms (ICGA '91), pp. 474-479.
- [14] Nguyễn Hữu Mùi and Vũ Đình Hoà, 2011. *Một thuật toán di truyền lai mới cho bài toán lập lịch công việc*, Kỷ yếu hội nghị khoa học công nghệ quốc gia lần thứ V, pp. 239-249.
- [15] Sotskov, Y.N. and Shaklevich, N. V., 1995. *NP-hardness of shop-scheduling problems with three jobs*, Discrete Applied Mathematics, Vol. 59, No. 3, pp. 237-266.
- [16] Yamada, T., 2003. *Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems*, Japan: Kyoto University, pp. 42-70.

This paper may be cited:

Hanh, D. T., Hoa, V. D. and Mui, N. H. 2014. Dual Hybrid Algorithm for Job Shop Scheduling Problem. *International Journal of Computer Science and Business Informatics*, Vol. 14, No. 3, pp. 14-24.