U.P.B. Sci. Bull., Series D, Vol. 82, Iss. 3, 2020 ISSN 1454-2358

SOLVING THE FLEXIBLE JOB SHOP SCHEDULING USING A NEW MACHINE NEIGHBORHOOD STRUCTURE

Guohui ZHANG[1]*, Jinghe SUN[1]

As an important part of the actual production, transportation time directly affects the product quality and production efficiency. In this paper, a flexible job shop problem with transportation time is studied. And an improved genetic algorithm with a new machine neighborhood structure is proposed to balance exploration and exploitation. The machine neighborhood structure is designed according to the characteristic of flexible job shop scheduling problem, which is one operation could be processed on multiple machines. And the neighborhood solutions can be searched by changing the processing machine of the operations on the critical path. By testing the benchmark and comparing with the other three algorithms, the experimental results show that the proposed algorithm is effective in solving this problem.

Keywords: Flexible job shop scheduling problem, Transportation time, Genetic algorithm, Machine neighborhood structure

1. Introduction

Flexible job shop scheduling problem (FJSP) is an extension of job shop scheduling problem (JSP), which is a well-known NP-hard combinatorial optimization problem [1]. The transportation time of semi-finished products accounts for a large part of the production cycle of the discrete manufacturing shop. The reasonable arrangement of transportation time will make the production planning more practical, then make the production and operation more effective and stable. Therefore, the FJSP with transportation time is studied in this paper.

FJSP was first proposed in 1990 and solved by a polynomial graphical algorithm [2]. After that, many scholars participated in the study of this problem, and FJSP also has a lot of development, such as the FJSP problem considering the setup time and transportation time constraints or the actual scheduling problem [3-6]. However, exact mathematical method are not effective for solving FJSP with large instances [7], intelligent optimization algorithms are more effective for this kind of NP-hard problem and gradually widely used [8]. Intelligent optimization algorithms for solving FJSP are mainly divided into two categories: bio-inspired

[1] School of Management Engineering, Zhengzhou University of Aeronautics, China
* Prof., School of Management Engineering, Zhengzhou University of Aeronautics, China, e-mail: zgh09@zua.edu.cn

swarm algorithm and local search algorithm. The former is to search the optimal solution in the global solution space, such as genetic algorithm (GA) [9], particle swarm optimization algorithm (PSO) [10], ant colony optimization algorithm [11], etc. Local search algorithm is a neighborhood search for a feasible solution, such as variable neighborhood search [12], tabu search algorithm [13]. A large number of experimental studies show that these algorithms have some disadvantages, where some hybrid algorithms are proposed to amend them [14-15].

The FJSP with transportation time is more in line with the actual discrete manufacturing systems, which is more complex and flexible, and it is difficult to find a high-quality solution. Therefore, designing an efficient algorithm is very important. In this paper, an improved GA with a new machine neighborhood structure (GAMNS) is proposed to improve the quality of solutions of the FJSP with transportation time. The machine neighborhood structure (MNS) is designed according to the characteristic of the problem.

The remainder of this paper is organized as follows: problem description is given in Section 2. Section 3 describes the GAMNS algorithm. The experimental results and discussion is shown in Section 4. Finally, Section 5 refers to conclusions and future research directions.

2. Problem description

The description of FJSP with transportation time and symbol definition are as follows: n jobs $\{J_1, J_2, J_3, \ldots, J_i, \ldots, J_n\}$ are processed in m machines $\{M_1, M_2, M_3, \ldots, M_k, \ldots, M_m\}$. Each job has h processing operations, the operations of the same job are processed in sequence. Define that $O_{i,h}$ is the h-th operation of job J_i, $O_{i,(h-1)}$ is the pre-operation of $O_{i,h}$ in the job J_i, $O_{i',h'}$ is the pre-operation of $O_{i,h}$ on the same machine. Each operation processing machine M_k may not be unique, and the processing time $p_{i,h,k}$ on each machine may be different. Each operation has its own start time $S_{i,h,k}$ and finish time $F_{i,h,k}$, the processing periods of these operations affects each other. When multiple operations of a job are processed on different machines, the job need to be the transported and takes corresponding transportation time: when operation $O_{i,(h-1)}$ of a job J_i is completed, it is necessary to determine whether the job needs to be transferred to another machine. If necessary, the transportation time $t_{l,k}$ between the two machines M_l and M_k should be considered, otherwise the transportation time is neglected. Notice that the transportation time $t_{l,k}$ and processing time $p_{i,h,k}$ of the job J_i are in series, and the starting time of the operation $O_{i,h}$ is the larger of the arrival time of the J_i ($F_{i,(h-1),k}+t_{l,k}$) and the allowable starting time ($F_{i',h',k}$) of the machine M_k. The optimization objective is to minimize the makespan when the processing time and transportation time of each operation are considered simultaneously. In the FJSP with transportation time, several constraints and assumptions are made as follows:

- Jobs and machines are independent and available at time zero.
- Operation preemption of the same job is not allowed, and each machine can handle only one operation at a time. Different operations of one job cannot be processed simultaneously.
- Processing time of each operation corresponds to its processing machine, and the processing time of all operations is known.
- Once the operation is completed, the job will immediately be transported to the processing machine of the next operation and take corresponding transportation time.
- Transportation time is only related to the selected processing machine and direction, and the transportation time between different machines is constant and known.
- Setup time of each operation is included in the processing time.

In order to describe the problem more easily, we give an illustrative instance of the FJSP with transportation time. In Table 1, there are 2 jobs and 5 machines. "-" indicates that the operation cannot be processed on the corresponding machine. When an operation is completed on the machine, the job is immediately sent to perform the next operation, and the transportation time varies according to the transportation route. The transportation time of the instance is shown in Table 2, which shows the time taken to transport from the corresponding machine on the left to the corresponding machine on the upper side.

Table 1

The processing time of the instance

Job	Processing Time (min)					
	Operation	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	-	6	4	-	5
	$O_{1,2}$	3	5	6	4	7
J_2	$O_{2,1}$	3	-	6	-	5
	$O_{2,2}$	4	6	5	-	7
	$O_{2,3}$	8	7	9	5	8

Table 2

The transportation time of the instance

Machine	Transportation Time (min)				
	M_1	M_2	M_3	M_4	M_5
M_1	0.0	2.2	4.4	2.5	3.0
M_2	2.8	0.0	7.3	3.4	2.0
M_3	3.9	2.1	0.0	1.3	2.8
M_4	3.7	1.6	2.4	0.0	1.3
M_5	1.8	2.9	1.9	4.0	0.0

3. The GAMNS algorithm

The advantage of GA with respect to other algorithms is due to the fact that more strategies could be adopted together to find good individuals to keep the balance between the diversification and the intensification during the search process.

In this paper, the proposed GAMNS adopts general GA framework and is mainly divided into three parts: initial population, genetic evolution, local search based on the MNS. The GAMNS framework is shown in the Fig.1. The initial population, genetic evolution and the MNS are discussed in the following sub-sections.

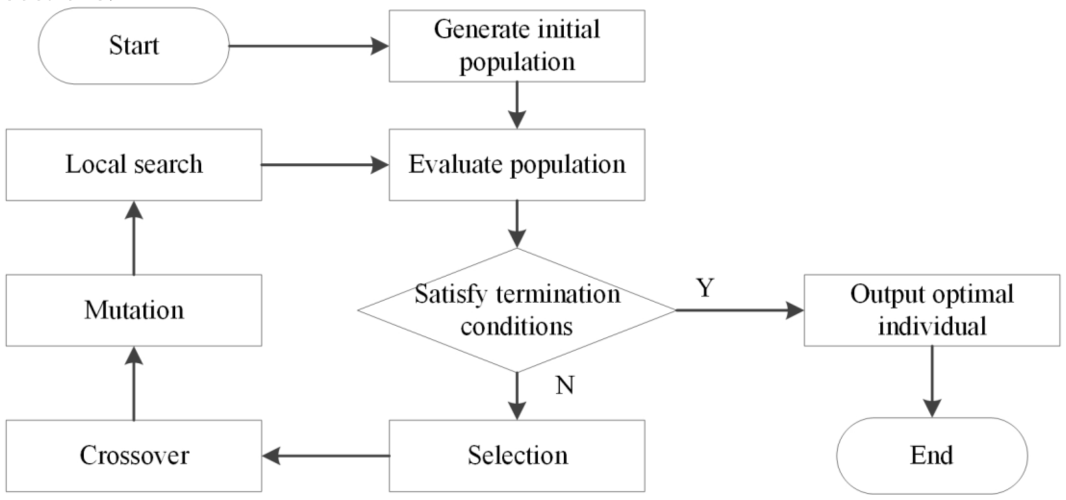

Start
Generate initial population
Local search
Evaluate population
Mutation
Satisfy termination conditions
Y
Output optimal individual
N
Crossover
Selection
End

Fig. 1. The framework of GAMNS

3.1 Coding and decoding

Coding and decoding is the method of transforming the feasible solutions of a given problem from the solution space to a search space that the algorithm can handle. The problem in this paper includes two sub-problems: machine selection and operation sequencing. Therefore, the two-stage integer coding and insertion decoding based on the characteristics of the problem are adopted. The detailed description can be seen in the literature [5].

3.2 Initial population

Initial population could be obtained by various methods, however, initial population method has different effects on the quality of the final solution under different iterations. A good initialization method can improve the quality of the final solution in a finite number of iterations. Therefore, we use a combined initialization method to obtain the initial population, which covers the whole solution space and increases the initial search direction. The detailed steps could see the [9].

3.3 Genetic evolution strategy

Selection: selection operation is to make excellent individuals survive with greater probability, avoid damaging good genes through cross and mutation operations, and combine good genes to produce better individuals. Tournament selection method is adopted as the selection method for the GAMNS. The makespan of the individual is smaller, then the probability for selecting the individual is higher. Meanwhile, in order to avoid the damage of good genes by genetic operation, the next generation is selected from the combined parent and offspring population.

Crossover: excellent crossover operation can preserve the good genes in the paternal generation by exchanging information among the paternal individuals, so as to produce good new individuals. It can also reduce blind search and achieve simple and efficient search. We use the multi-points crossover for machine selection part, and the jobs integral crossover for the operation sequencing part. The detailed steps can be seen in the literature [12].

Mutation: mutation can increase population diversity and avoid premature convergence. The popular mutation methods are all very common. In this paper, the method of replacing the processing machine with probability is adopted for the machine selection part. The probability is generated by the processing time, that is, the shorter the processing time, the greater the probability. In the operation sequencing part, the method of random operation forward insertion is adopted.

3.4 The machine neighborhood structure

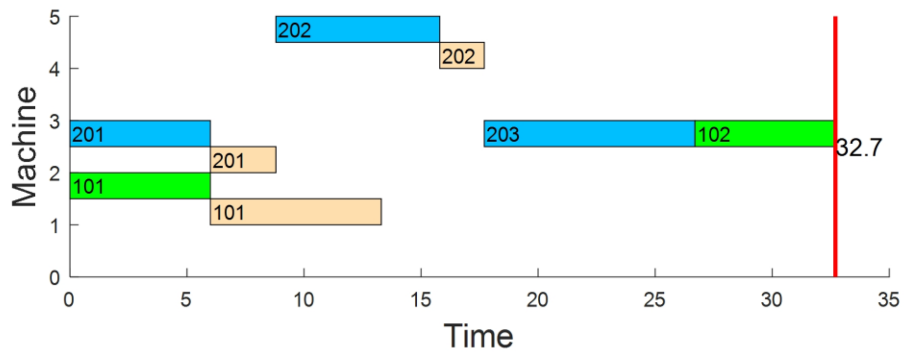
Local search is to find the local optimal solution by searching neighborhood solutions of one feasible solution. Local search can effectively improve the quality of global optimal solution, because exploitation of local search is deeper. Neighborhood structure is the key to local search, which directly affects the performance of local search.

In this paper, a new MNS is proposed to improve the quality of solutions. The MNS is designed according to the characteristic of the FJSP, which one operation can be processed on several machines. First of all, we define one operation sequence as the same niche. No matter how to change the processing machine, the solution belongs to the niche. Then we only need to search the selection method of the processing machine under the condition that the operation sequence remains unchanged. Therefore, we change the processing machine of the operation on the critical path, and the operation is selected from back to front. Search speed can be adjusted by the maximum number of iterations. If the makespan cannot be reduced by replacing the machines of all operations on the critical path, the solution is considered as the neighborhood optimal solution. The pseudocode is shown in Algorithm 1.

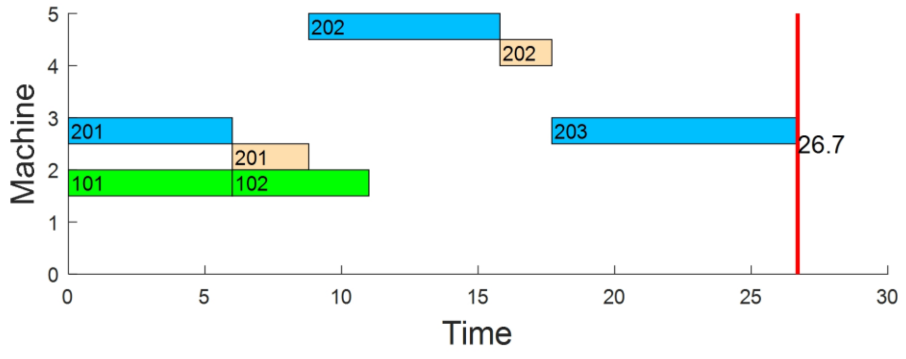
Algorithm 1 Search the neighborhood solution by MNS

```
Input: a feasible solution $S$ and maximum number of iterations $k$
Output: the neighborhood optimal solution $S_{opt}$
1 $Iter = 0$
2 while $Iter < k$
3     Find out the critical path and get its length $L$
4     $g = 0$
5     for i = 1 : $L$
6         Select the last $i$ operation on the critical path
7         Replace the machine with one that makes the operation to be completed earliest, get solution $s'$
8         if $f(s) > f(s')$
9             break
10        else
11            $g = g + 1$
12        end
13    end
14    $S_{opt} = s'$
14    if $g == L$
15        break
16    end
17    $Iter = Iter + 1$
18 end
```

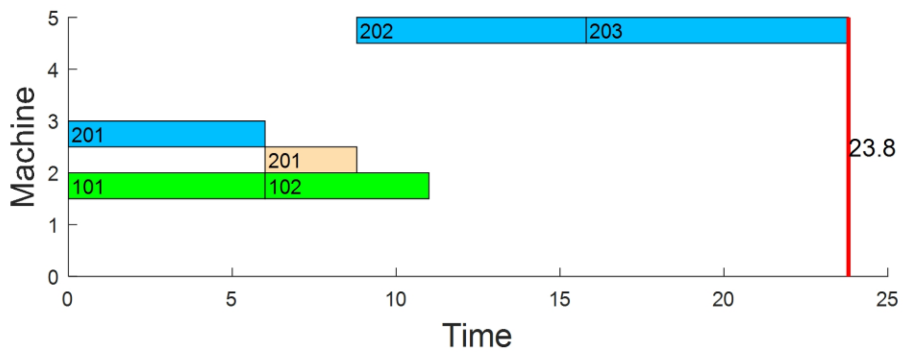
Taking the instance in Table 1 and Table 2 as an example, the updating process of a feasible solution using MNS is shown in Fig. 2. These colored rectangles are the processing time of the operations, and these beige rectangles at the bottom right of processing time rectangles represent the transportation time of the jobs. In Fig. 2(a), the operation $O_{1,2}$ is the last operation on the critical path, it's processed on the machine M_2, which the makespan of this operation can be minimized, as shown in Fig. 2(b). The same, in Fig. 2(b), the operation $O_{2,3}$ can be processed on the machine M_5, according the alternative machines set. The final result is shown in Fig. 2(c).



(a) A feasible solution



(b) Updated by MNS



(c) Updated by MNS again

Fig. 2. Updating process of a feasible solution using MNS

4. Experimental results and comparison

All algorithms are performed in MATLAB R2017a on a PC with Inter(R) Core(TM) i7-4785T CPU @ 2.2GHz and 8Gb of RAM. The experimental content is to test the performance of the proposed GAMNS. The experimental data sets were in two groups, one from Kacem etc. [16] (Kdata), which includes 4 test instances, and the other from Brandimarte's data sets [17] (Bdata). The transportation time between machines are generated randomly, as shown in Table 3. Through the analysis of main effect, the parameters of GAMNS are determined to minimize the mean value of makespan, as shown in Table 4. The termination condition is that the number of iterations is equal to the maximum number of iterations or the optimal solution is not updated for 30 generations.

Table 3

The transportation time between the different machines

Machine	Transportation Time (min)														
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
M_1	0.0	2.2	4.4	2.5	3.0	1.9	4.1	2.2	3.9	1.9	0.5	0.7	2.5	3.9	0.5
M_2	2.8	0.0	1.3	3.4	2.0	4.2	2.9	1.6	4.9	3.4	2.9	2.7	2.4	4.6	2.1
M_3	3.9	2.1	0.0	1.3	2.8	2.6	1.9	2.2	1.6	1.7	4.9	4.1	4.1	1.4	1.6
M_4	3.7	1.6	2.4	0.0	1.3	1.0	3.3	3.2	1.9	1.5	0.4	4.0	1.7	1.5	5.0
M_5	1.8	2.9	1.9	4.0	0.0	1.5	4.0	3.4	1.7	3.7	3.7	4.5	4.0	4.9	2.8
M_6	2.2	1.2	2.2	2.9	2.5	0.0	1.8	3.9	2.0	3.5	3.1	2.9	1.3	4.6	4.4
M_7	4.8	3.3	3.7	2.2	2.1	3.0	0.0	4.1	2.1	2.1	3.2	3.2	4.9	2.8	3.3
M_8	1.8	1.8	1.4	1.1	2.4	2.4	1.2	0.0	3.6	3.8	3.7	1.5	1.8	3.3	1.3
M_9	2.3	1.7	3.4	3.2	3.9	1.5	1.3	3.0	0.0	1.3	1.9	4.7	2.3	4.6	0.6
M_{10}	2.4	1.2	1.5	1.4	3.4	4.7	1.5	3.6	2.2	0.0	5.0	3.6	1.3	4.3	1.3
M_{11}	1.9	4.7	0.6	4.4	2.0	2.4	4.9	2.0	3.6	4.9	0.0	2.2	2.8	4.3	4.8
M_{12}	2.2	4.5	1.9	4.2	1.7	3.5	1.7	2.8	1.7	5.1	2.4	0.0	1.9	4.8	4.9
M_{13}	2.6	4.7	2.1	3.4	4.2	1.6	0.2	2.7	3.7	3.2	0.3	2.8	0.0	2.2	3.5
M_{14}	3.0	4.7	0.8	3.4	0.7	1.8	4.0	1.5	4.2	1.6	3.3	2.1	1.6	0.0	4.9
M_{15}	0.2	2.9	4.6	1.8	2.6	4.9	4.3	2.3	0.9	2.8	4.7	1.2	4.9	3.9	0.0

Table 4

Parameters settings of the GAMNS

Parameters	Value
population size	100
maximum iterations of global search	200
crossover probability	0.8
mutation probability	0.2
maximum iterations of local search	10

The proposed GAMNS is compared with three algorithms reported in literature which are as follows: PSO [10], IGA [5], MOHPIOSA [19]. In order to evaluate the effectiveness, all algorithms were run 10 times consecutively, and get the optimal value of each experimental data sets. The results of the two groups are shown in Table 5 and Table 6.

Table 5

Experimental results 1

Problem $n \times m$	PSO	IGA	MOHPIOSA	GAMNS
4×5	12.3	12.3	12.3	12.3*
8×8	27.5	24.4	23.2	19.3*
10×10	25.4	20.6	15.6	15.2*
15×10	38.8	35.9	32.8	26.9*

Table 6

Experimental results 2

Problem	PSO	IGA	MOHPIOSA	GAMNS
Mk01	54.9	53.3	48.4	43.9*
Mk02	52.2	49.4	42.5	35.9*
Mk03	249.9	237.9	221.2*	222.0
Mk04	91.8	85.1	77.3	77.0*
Mk05	207.0	199.1	188.9	186.4*
Mk06	142.7	136.5	127.5	106.7*
Mk07	211.7	194.7	186.4	165.0*
Mk08	547.9	530.4	528.6	523.0*
Mk09	415.5	410.3	376.1	366.0*
Mk10	373.0	354.4	324.3	307.0*

As shown in the Table 5, "Problem $n \times m$" denotes that there are n jobs and m machines about the each instance. The computational results of the GAMNS are better than the other algorithm. The 4 instances of the Kdata are characterized by the fact that each procedure can be processed on all machines, so the flexibility of jobs is very high. It can be seen that the search effect of the GAMNS is significantly better than other algorithms, which indicates that it is feasible and effective to search the neighborhood solution by changing the machine of the operations on the critical path. Taking the solutions of the same process sequence as one same niche, and then only changing the processing machine can avoid repeated search, and the search effect is better.

From table 6, it can be seen that GAMNS has better results than other algorithms in 9 out of 10 instances. Moreover, the larger the problem scale and the higher the flexibility, the more obvious the advantages of GAMNS are. Therefore, the MNS is more suitable for solving FJSP with greater flexibility in limited iterations. Take the instance Mk01 for example, the convergence curve of Mk01 is shown in Fig. 3. The Gantt chart of Mk01 by GAMNS is shown in Fig. 4. As shown in Fig. 3, the convergence of PSO, IGA and MOHPIOSA are earlier than that of GAMNS, and the quality of solutions are worse. Because these four algorithms set the same termination criteria, it shows that GAMNS has better ability to jump out of the local optimal solution and exploitation. As shown in Fig.4, there are two critical paths. The operations on the first critical path are $O_{9,6}$-

$O_{2,4}$-$O_{5,5}$-$O_{5,4}$-$O_{7,3}$-$O_{7,2}$-$O_{9,2}$-$O_{6,2}$-$O_{6,1}$-$O_{10,1}$. The operations on the second critical path are $O_{3,5}$-$O_{3,4}$-$O_{3,3}$-$O_{3,2}$-$O_{6,3}$-$O_{10,4}$-$O_{10,3}$-$O_{10,2}$-$O_{10,1}$. Without changing the processing sequence, it is impossible to reduce the makespan simply by changing the processing machines of these operations. Therefore, this solution is the neighborhood optimal solution defined in this paper.

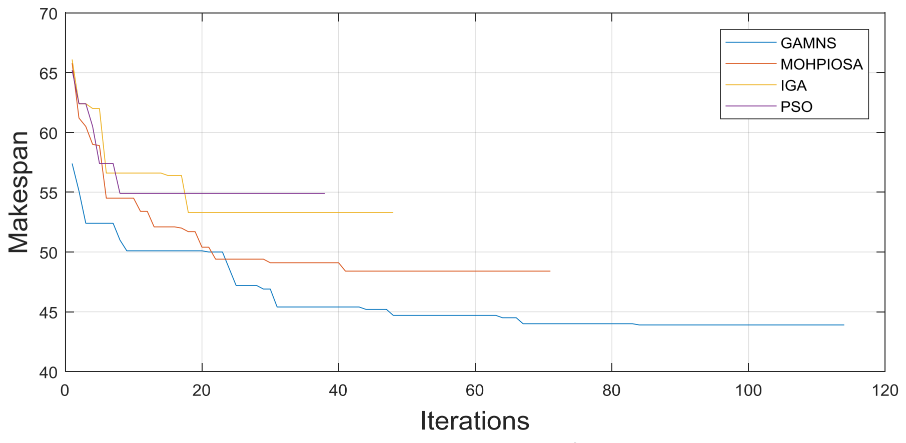


Fig. 3. Convergence curve of Mk01

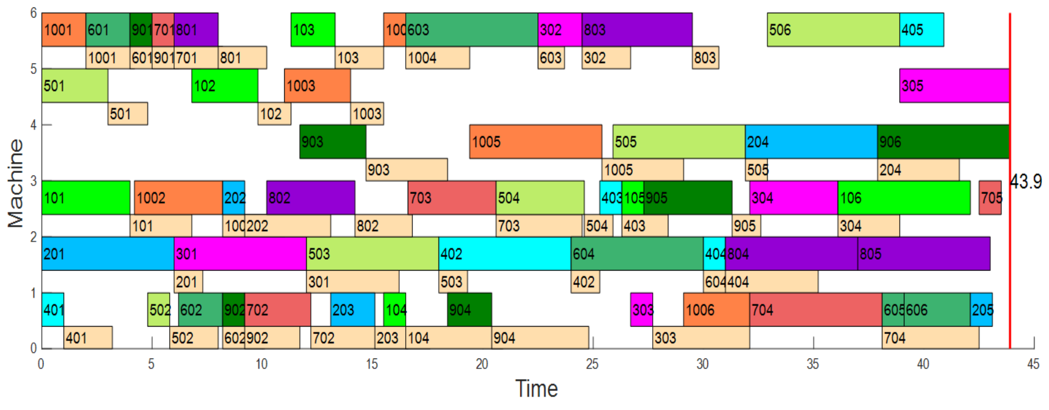


Fig. 4. The Gantt chart of the solution of Mk01

6. Conclusions

There are many studies on FJSP, but the studies on transportation time constraints are far from enough, and there is no good way to find the optimal solution for the single objective of minimizing the makespan. In this paper, an improved genetic algorithm with the machine neighborhood structure is proposed to solve FJSP with transportation time. Through the comparison of the results by the four algorithms, it can be shown that the GAMNS can effectively improve the quality of the final solution by local search using the MNS. Therefore, the GAMNS is effective in solving this problem.

In the future, as the production planning becomes more accurate, it is very important to further study the FJSP with transportation time and other extension problems. The research on the algorithm structure designed according to the characteristics of problems is more practical than the general algorithm research.

Acknowledgements

This paper presents work funded by the National Natural Science Foundation of China (Nos. U1904167, 51705472, 71871204), Humanities and Social Sciences of Ministry of Education Planning Fund (No. 18YJAZH125), and Postgraduate Education Innovation Program Fund of Zhengzhou University of Aeronautics (No. 2019CX17).

R E F E R E N C E S

[1]. *M. R. Garey, D. S. Johnson, and R. Sethi*, "The complexity of flowshop and jobshop scheduling", Mathematics of Operations Research, **vol. 1**, no. 2, May. 1976, pp. 117-129.
[2]. *P. Brucker and R. Schile*, "Job-shop scheduling with multi-purpose machines", Computing, **vol. 45**, Dec. 1990, pp. 369-375.
[3]. *A. Rossi*, "Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships", International Journal of Production Economics, **vol. 153**, July. 2014, pp. 253-267.
[4]. *S. Karimi, Z. Ardalan, B. Naderi, and M. Mohammadi*, "Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm", Applied Mathematical Modelling, **vol. 41**, Jan. 2017, pp. 667-682.
[5]. *G. H. Zhang, J. H. Sun, X. Liu, G. D. Wang, and Y. Y. Yang*, "Solving flexible job shop scheduling problems with transportation time based on improved genetic algorithm", Mathematical Biosciences and Engineering, **vol. 16**, no. 3, Feb. 2019, pp. 1334-1347.
[6]. *Q. K. Pan, L. Gao, and L. Wang*, "A multi-objective hot-rolling scheduling problem in the compact strip production", Applied Mathematical Modelling, **vol. 73**, Sept. 2019, pp. 327-348.
[7]. *F. Pezzella, G. Morganti and G. Ciaschetti*, "A genetic algorithm for the flexible job-shop scheduling problem", Computers & Operations Research, **vol. 35**, Oct. 2008, pp. 3202-3212.
[8]. *X. L. Wu, X. L. Shen, and C, B. Li*, "The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously", Computers & Industrial Engineering, **vol. 135**, Sept. 2019, pp. 1004-1024.
[9]. *G. H. Zhang, L. Gao and Y. Shi*, "An effective genetic algorithm for the flexible job-shop scheduling problem", Expert Systems with Applications, **vol. 38**, no. 4, Apr. 2011, pp. 3563-3573.
[10]. *G. H. Zhang, X. Y. Shao, P. G. Li and L. Gao*, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem", Computer & Industrial Engineering, **vol. 56**, May. 2009, pp. 1309-1318.
[11]. *C. Blum, and M. Sampels*, "An ant colony optimization algorithm for shop scheduling problems", Journal of Mathematical Modelling & Algorithms, **vol. 3**, no. 3, Sept. 2004, pp. 285-308.

[12]. *G. H. Zhang, L. J. Zhang, X. H. Song, Y. C. Wang, and C. Zhou*, “A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem”, Cluster Computing, no. 22, Feb. 2019, pp. 11561–11572.

[13]. *J. Q. Li, K. Q. Pan, P. N. Suganthan, and T. J. Chua*, “A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem”, International Journal of Advanced Manufacturing Technology, **vol. 52**, no. 5, Feb. 2011, pp. 683-697.

[14]. *X. Y. Li, and L. Gao,* “An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem”, International Journal of Production Economics, **vol. 174**, Apr. 2016, pp. 93-110.

[15]. *X. Y. Li, L. Gao, Q. K. Pan, L. Wang, and K. M. Chao*, “An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop”, IEEE Transactions on Systems, Man and Cybernetics: Systems, Dec. 2018, pp. 1-13.

[16]. *I. Kacem, S. Hammadi, and P. Borne*, “Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic”, Mathematics and Computers in Simulation, **vol. 60**, Sept. 2002, pp. 245-276.

[17]. *P. Brandimarte*, “Routing and scheduling in a flexible job shop by tabu search”, Annals of Operations Research, **vol. 41**, no. 3, Sept. 1993, pp. 157-183.

[18]. *X. L. Wu, X. L. Shen, and C. B. Li,* “The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously”, Computers & Industrial Engineering, **vol. 135**, Sept. 2019, pp. 1004-1024.