A Fast TS Algorithm for Solving the Flexible Job-shop Scheduling Problem

Jun-qing Li
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
lijunqing@lcu.edu.cn

Quan-ke Pan
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
panquanke@lcu.edu.cn

Sheng-xian Xie
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
xsx@lcu.edu.cn

Yu-ting Wang
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
wangyuting@lcu.edu.cn

ABSTRACT

Flexible job-sop scheduling problem (FJSP) is harder than the classical JSP (Job-shop Scheduling Problem). In this paper, a novel and fast tabu search (TS) algorithm are proposed for solving the FJSP, the objective is to make the complete time minimum, i.e., to get the best makespan. In the new fast algorithm, TS algorithm was used to produce initial solutions, and a new public critical block structure was proposed to find a better solution around the solutions. The computational results have proved that the proposed hybrid algorithm is efficient and effective for solving FJSP, especially for the problems with large scale.

Key Words: Flexible Job-shop Scheduling Problem; Tabu Search; public critical block structure.

1. Introduction

The classical job-shop problem (JSP) [1, 2] schedules *n* jobs on *m* machines with some given constraints and to minimize some given criterions. Flexible job-shop problem (FJSP) is harder than the classical job-shop problem. In FJSP, one operation can be operated on a set of machines. Therefore, there are two schedule parts in the FJSP: first, assign a proper machine from a set of machines to operation each operation; second, sequence each operation on every given machine. The former problem can be seen as a parallel machine problem, which is also a NP-hard problem. The latter is equal to a classical job-shop problem.

The FJSP recently captured the interests of many researchers. The first paper about FJSP was proposed by Brucker and Schlie (Brucker & Schlie, 1990), which discuss a simple FJSP model with two jobs and operations performed on any machines with the same processing time. To solve more genera FJSP problems with more than two jobs and machines, many researchers proposed hierarchical approaches, i.e., decomposing the problem into two stages: machine assignment sub-problem and job shop sub-problem. The first author to use the hierarchical idea was Brandimarte (Brandimarte, 1993), who solved the first stage with some existing dispatching rules and the second stage with tabu search heuristic algorithms. Mati (Mati Rezg & Xie, 2001) proposed a greedy heuristic for simultaneously dealing with the two stages. Kacem (Kacem, Hammadi & Borne, 2002) solved the two stage problems with the GA. Zhang (zhang & Gen, 2005) proposed a multistage-based GA to solve multi-objective FJSP with *k* stages and *m* state. In recent years, the tabu search algorithm has been verified by many researchers adapt to solve the FJSP problems, and has put up efficiency in solving the NP-hard combinatorial optimization problems. Noureddine(Noureddine, 2007) illustrated a combined ant system optimization with local search methods, including tabu search for solving the FJSP

problems. However, the authors have not tested the algorithm on large scale problems. Brandimarte (Brandimarte, 199) proposed a routing and scheduling method for the FJSP problems with the tabu search algorithm. Saidi-mehrabad (Saidi-mehrabad, 2007) gave a detailed solution for solving FJSP with tabu search method.

In this paper, we give a new fast TS algorithm for the FJSP solutions, and propose a public critical structure for the sequence part. In each generation, we use the TS algorithm to solve the machine assignment problem, and then, we use the public critical block structure to solve the operation sequence problem. After a detailed experiment, the result verifies that our novel method can get better solutions in very short period.

2. Problem formulation

The FJSP can be an extension of the classical JSP; therefore, we can formulate the FJSP based on JSP. Consider a set of n jobs, noted $J = \{J_1, J_2, ...J_n\}$, every job in the set J has a given number operations, and should be operated on a given machine from a machine set named $M = \{M_1, M_2, ...M_m\}$. So, there are n jobs and m machines. In the classical JSP problem, with n jobs and m machines, there are $n*m$ operations. However, in FJSP problems, the operation number can vary with the problem assumption. There are two kinds of FJSP, i.e., T-FJSP and P-FJSP. For the T-FJSP, each job can be operated on every machine from the set M; for the P-FJSP, there is a problem constraint for the operating process, in table 1, we can see that one operation of a job must be processed by a set of machines in $M' \subseteq M$. In the sequencing stage for the FJSP, we must consider the candidate machine set size for every operation waiting for processed. The detailed definition of the FJSP as follows:

- A set of J independent jobs.
- Each job J_i can be operated on a given set of machines M_i.
- The $O_{i,j}$ represents the j^{th} operation of J_i. The machines set waiting for processing the $O_{i,j}$ noted by $M_k \subseteq M$.
- We use $p_{i,j,k}$ to represent the processing time of $O_{i,j}$ operated on the k^{th} machine.
- There have two assumptions: a started operation can not be interrupted; each machine only can process one operation at the same time.
- The objective in our paper is to find the minimum time of the whole operations.

Table 1 Processing time table

	M_1	M_2	M_3	M_4
O_{11}	7	6	4	5
O_{12}	4	8	5	6
O_{13}	9	5	4	7
O_{21}	2	5	1	3
O_{22}	4	6	8	4
O_{23}	9	7	2	2
O_{31}	8	6	3	5
O_{32}	3	5	8	3

3. Encoding of Solutions

The FJSP problems involve two decision stages, i.e., machine assignment stage and operation sequence stage. Therefore, a solution consists of two parts of vectors, A_1 (machine assignment vector) and A_2 (operation sequence vector). A_1 represents the corresponding selected machine for every operation; A_2 indicates the operation sequence on every machine.

The length of vector A_1 equals the total number of operations. Each element in A_1 represents the selected machine number for processing the operation in that position. For example, in Fig.1, the 5^{th} element in A_1 is 2, which means machine 2 is selected to process the operation O_{21}.

During the initialization phase, every solution in the population will get a storage space for the string A_1 with the length equals l. For each position in A_1, the most suitable machine will be selected from a set of alternative machines. So, the computational complexity of the initialization of A_1 will be determined by the total number of operations, and the average number of alternative machines for every operation, and it can be noted as $O(\sum_{i=1}^{l} al_i)$.

For the vector A_2, Gen et al. proposed a relative nice solution. They name all operations for a job with the same symbol. For example, in Fig.2, we place the symbol 2 in both position 1 and 4. The figure means that the first operation selected to be processed is the first operation of job 2, i.e., O_{21}, and the 4^{th} operation is O_{22}. Each job i will be placed in A_2 exactly n_i times, here, n_i represents the operation number of job i. so, the length of A_2 also equals l. The computational complexity of initialization of A_2 is $O(l)$.

position	1	2	3	4	5	6	7	8
operation	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}
machine	4	3	2	1	2	1	2	3

Fig.1. Machine assignment vector example

position	1	2	3	4	5	6	7	8
operation	2	1	3	2	1	2	1	3

Fig.2. Operation sequence vector example

4. The fast TS Algorithm

4.1. The Tabu Search Algorithm

TSA is proposed by Glober in 1986, which is a famous local search algorithm to solve combined optimize problem [9]. TSA has two main features: (1) the capability to avoid local optimization. TSA uses a tabu table to memory the better local neighbors which have been searched and will be neglected; (2) the capability to find better resolution. TSA uses an aspiration rule to exploit a prohibited resolution. During a situation that all the resolution in the tabu table is prohibited, the aspiration can make the whole search processing continue. The basic flow of the TS algorithm was shown in Fig3.

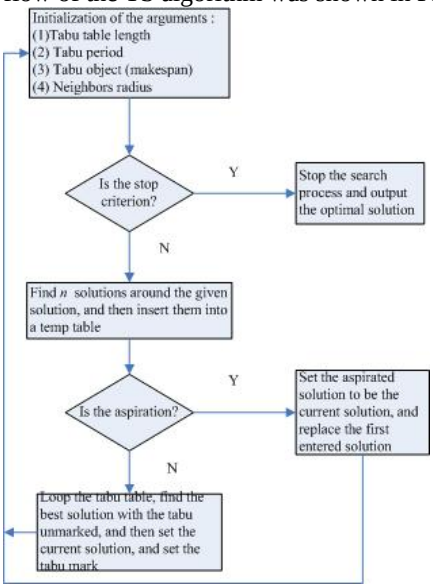

Fig.3. the flow of TS algorithm

4.2. Public Critical Block Structure

The critical problem of local search is how to define the effective neighborhood around the given solution. The promising neighborhood is based on the concept of critical path, which was firstly proposed by Adams (Adams, Balas, & Zawack, 1988) in solving JSP problems. Many researches have verified that block structure neighborhood will decrease the search space deeply. Block structure is based on the critical path. The critical path is composed by many critical operations which must be operated on the same machine.

We propose a public critical block structure based on the basic critical block theory. In order to describe the new algorithm easily, we give several definitions, JP_i, JS_i, MP_i, MS_i indicates the immediate job predecessor, job successor, machine predecessor and machine successor of the operation respectively.

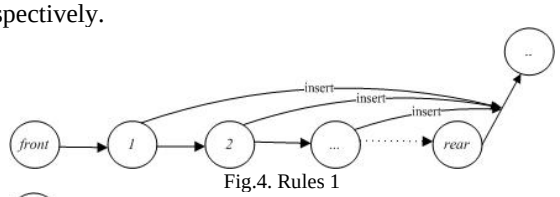

Fig.4. Rules 1

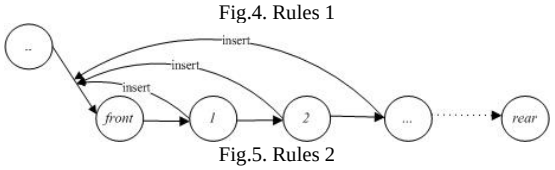

Fig.5. Rules 2

The process of the public critical block was illustrated in Fig.6.

procedure : getPublicCriticalOperations
input: a set named M_c including all critical operations
output: all public critical operations
begin
for every critical operation CO_i in M_c
get the start time s_i and the end time e_i of the operation
for every other critical operatins in M_c
get the start time s_j and the end time e_j
if occurs one of the case as followings:
1) $e_i > e_j$ && $s_j < s_i$ && $s_i < e_j$
2) $s_j < e_i$ && $e_i < e_j$ && $s_j > s_i$
3) $s_j > s_i$ && $e_j < e_i$
4) $s_i > s_j$ && $e_i < e_j$
then mark the operation CO_i as a non-critical operation
end for
end for
output the unmarked operations in M_c.
end

Fig.6. the process of Public critical block

We give six rules as following:

Rules 1: As shown in Fig4. If a public critical path block containing *u* and *v* also contains JS_v, that is, *v* is the block rear, and then inserts *u* right after *v*.

Rules 2: If a public critical path block containing *u* and *v* also contains JS_v, that is, *v* is the block rear, and then inserts *v* right before the first successive internal operations.

Rules 3: If a public critical path block containing *u* and *v* also contains JS_v, that is, *v* is the block rear, and then inserts each internal operation just after *v*.

Rules 4: If a public critical path block containing *u* and *v* also contains JP_u, that is, *u* is the block head, and then moves *u* right after the first successive internal operations.

Rules 5: As shown in Fig5. If a public critical path block containing *u* and *v* also contains JP_u, that is, *u* is the block head, and then inserts each internal operation exactly before *u*.

5. Computational result

To test our novel algorithm, we realized our algorithm with VC6.0. The hardware environment of our testament is Pentium4 2GHZ. In our experiment, we tested different values for a list of algorithm parameters, and the detailed parameters as follows:

- tabu table length: n*m/2;
- tabu period: n*m/4;
- neighbors radius: 300

The detailed makespan we have found were illustrated in table 2. From the computational results, we can see that our novel method seems better in almost all MK case, especially the hard cases such as MK09 and MK10. The GA [5] seems better than our method in several cases such as MK02, MK07. The reason is that the GA algorithm takes longer computational time than our method. With our novel and fast algorithm, we can find the optimal solution in very short time.

Table 2 Computational result

Name	n	m	[11]	GA[5]	our method
MK01	10	6	40	40	40
MK02	10	6	29	26	28
MK03	15	8	-	204	204
MK04	15	8	67	60	60
MK05	15	4	176	173	173
MK06	10	15	67	63	62
MK07	20	5	147	139	142
MK08	20	10	523	523	523
MK09	20	10	320	311	308
MK10	20	15	229	212	219

6. Conclusion

In this paper, we give a novel fast algorithm with TS and public critical block structure. In the sequencing stage, we use public critical block structure to find the best solution, and in the machine assignment stage, we use TS algorithm to find the near optimize solution around the given solution. The computational result shows that our algorithm can get better result than the GA algorithm. The next work should focus on how to decrease the computation time and make the algorithm more robust.

7. Acknowledgement

This research was partially supported by grants 2006AA01Z455 from the National High-Tech Research and Development Plan of China, 60874075, 70871065 from National Science Found from China, 2004ZX17 and 2004ZX14 from the Natural Science Foundation of Shandong Province, J08LJ20 from Science Research and Development of Provincial Department of Public Education of Shandong and X061015 from the Science Foundation of Liaocheng University.

References

[1] Sonmez, A. I. & Baykasoglu, A. (1998). A new dynamic programming formulation of (n *m) flowshop sequencing problems with due dates. *International Journal of Production Research*, 36, 2269-2283.

[2] M.R. Garey, D.S. Johnson, R. Sethi, The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*. 1(2) (1996), 117-129.

[3] Zhang, H. and Gen, M., Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Journal of Complexity International*, 11, 223--232, 2005

[4] Kacem, I., Hammadi, S. and Borne, P. (2002), Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems, *IEEE Transactions on Systems, Man and Cybernetics*, Part C, 32(1):408-419.

[5] F. Pezzella, G. Morganti, G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers & Operations Research, 2008, 35:3202-3212.

[6] Jie Gao, Mitsuo Gen, Linyan Sun: A hybrid of genetic algorithm and bottleneck shifting for flexible job shop scheduling problem. *GECCO* 2006: 1157-1164

[7] Wang Ling. Shop Scheduling with Genetic Algorithms [M]. Beijing: Tsinghua University Press, 2003.

[8] Xing Wen-xun, Xie Jin-xing. Modern optimization algorithm [M]. Beijing: Tsinghua University Press, 2005.

[9] Nowicki E, Smutnicki C. A fast taboo search algorithm for the job-shop problem, Management Science, 1996, 42:797-813.

[10] Wang L, Zheng DZ. An effective optimization strategy for job-shop scheduling problems. Computer and Operations Research, 2001, 28:585-596.

[11] Nhu Binh Ho, Joc Cing Tay, Edmund M.-K. Lai. An effective architecture for learning and evolving flexible job-shop schedules. European Journal of Operational Research, 179(2007):316-333.

[12] Van Laarhoven P J M, Aarts E H L, Lenstra J K. Job shop scheduling by simulated annealing. Operations Research, 1992,40:113-125

[13] Glover F., "Tabu Search: A Tutorial", Interfaces, 1990,20(4):74-94.

[14] Wang Ling. Intelligent Optimization Algorithms with Applications [M]. Beijing: Tsinghua University Press, 2001.