DE GRUYTER OPEN

Print ISSN: 1312-2622; Online ISSN: 2367-5357
DOI: 10.1515/itc-2017-0029

An Algorithm for Generating a Dispersed Population of Feasible Schedules for Flexible Job Shop Problems

L. Kirilov, V. Guliashki

Key Words: *Flexible job shop problems (FJSP); scheduling; heuristics; optimization.*

Abstract. *The flexible job shop problems (FJSP) are an important class of scheduling problems and they have a significant practical value. Unfortunately it is not easy to solve job shop problems and in particular FJSPs because they are NP-hard problems. In this paper we propose a method for generating a set of feasible schedules for a given FJSP.*

Introduction

The job shop scheduling problem is well-known from operations research and computer science and is of high practical value with appli¬cations in many real-life situations [1,10]. While first approaches in this area consider optimality of schedules for a single objective function, multi-objective formulations of the problem have become gradually of increasing importance [4]. A theory of multi-criteria scheduling is presented in [11]. A survey of methods for job shop scheduling using multi-criteria decision making is presented in [9]. In this paper the multi-criteria flexible job shop scheduling problem is considered as an extension of the popular multi-criteria job shop scheduling problem. During the last decades many researchers have devoted considerable efforts to developing evolutionary multi-criteria algorithms.

The problem of scheduling arises when planning and controlling the decision-making process of manufacturing and service industries. It can be schematized as follows: There is a number of N jobs to be executed. Each job consists of a given sequence of operations which needs to be performed using a number of M machines. All operations for each job must be performed in the order given by the sequence. Each operation demands the use of a particular machine for a given time. Each machine can process only one operation at a time. The goal is to find a schedule optimizing the above problem according to the given objective function (cost function, make-span, tardiness, maximal workload etc.). Scheduling consists of assigning each operation of each job a start time and a completion time on a time scale of the machine with the preference relations.

The most used in practice is the job shop scheduling problem. It is a difficult computational problem. Optimal solutions for job shop scheduling can be found in polynomial time if the number of jobs is 2, or if the number of machines is 2 and all jobs have 1 or 2 operations, or if the number of machines is 2 and all operations have duration 1. In all cases the problem obtained by incrementing the number of machines, jobs, operations or durations by 1, is NP-hard [5, 7]. We have no intention to go in details in the theory of computational complexity. For our purposes is enough to say that NP-hard means Non-deterministic Polynomial-time hard problems.

Below are presented the basic formulations of the classical job shop problem (JSP) and of the flexible job shop problem (FJSP).

Job Shop Problem (JSP)

The JSP is formulated as follows: There is given a set of n jobs: $J_1, ..., J_n$, which have to be performed on m machines $M_1, ... , M_m$.

For each job there is given the operative consequence of the jobs composing this job. Namely:

$J_i = (O_i, 1,... , O_{i, j(i)})$, $j(i)$ is the number of operations for the corresponding job, i=1,...,n.

It is well-known which operation on which machine should be executed. Therefore another formulation of this model is:

$J_i = (M_i, 1,…, Mi, _{j(i)})$.

The processing times for each possible operation on each machine are known: $p_{i,k}$, i=1,...,n; k = 1, ... , j(i).

The optimal schedule according preliminary given criterion (criteria) has to be found. For example one criterion could be the minimization of make-span (time window) – C_{max}.

This is the most often used and chronologically the earliest developed model – see for example [5,6].

Flexible Job Shop Problem (FJSP)

This model represents an extension of the above job shop problem. Here each operation can be executed not only on one machine, but on a given subset of machines. This subset is naturally different for each operation. In other words, it is not a priori known which operation on which machine should be performed.

This model is closer to real life production situations and could be applied, when some or all machines are multi-functional (multitasking) – i.e. they could perform more than one operation (not at the same time) with corresponding different processing times. Among the first researchers suggesting this model are Bruker and Schlie – [2].

As noted in [3] the FJSP is a problem of high complexity and practical value, and it has been widely investigated for the last two decades. Researches on its multiobjective version started about ten years ago, but most studies focused on searching for the single optimal solution with respect to a certain aggregated objective. Research works aiming at obtaining the set of Pareto optimal solutions appeared during the recent three years.

There are two variants of FJSP – [8]:

First, when each operation of each job can be executed on any, no matter which, machine. This case is relevant to the total / global flexibility (total flexible job shop problem – T-FJSP).

Second, even not each (but at least one) operation can be performed on any machine. This case refers to the partial flexibility (partial flexible job shop problem – P-FJSP).

Sometimes it is necessary to generate a number of feasible schedules. They can be used as a population in evolutionary or genetic algorithms for further calculations.

In this paper we present a method to generate a large number of feasible schedules for FJSPs.

The Proposed Approach

A specific schedule is defined when for each operation from each job it is known the machine to be performed on, the starting time (or ending time). One way of schedule representation is the Gant' diagram. But for our purposes we need more effective way of schedule representation in order to perform different operations with it.

Coding

We will use the two-vector coding as in [14].

A schedule is presented in the forms of vector A and vector B, see *figure 1* and *figure 2*.

The two strings have equal length and it is just the sum of all operations for all jobs.

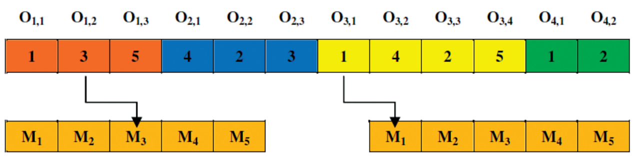
Let we have a number of four jobs to be performed on a set of five machines – M_1, M_2, M_3, M_4, M_5;

4 jobs - $J_1(O_1, O_2, O_3)$, $J_2(O_1, O_2, O_3)$, $J_3(O_1, O_2, O_3, O_4)$, $J_4(O_2, O_4)$.

Then in this case the strings' length is 12.

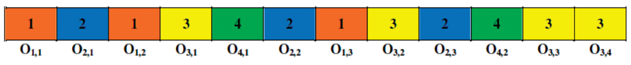
The number in each box of string A denotes on which machine the corresponding operation to be performed – see *figure 1*.

The operations and precedence relations between job's operations are given in string B – see *figure 2*. If we take for example job $J_1(O_1, O_2, O_3)$ we see three boxes with value "1". The first one found from left to right means $O_{1,1}$, the second one found from left to right means $O_{1,2}$ and etc.



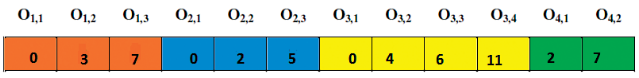
$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{4,1}$ $O_{4,2}$
1 3 5 4 2 3 1 4 2 5 1 2
M_1 M_2 M_3 M_4 M_5
M_1 M_2 M_3 M_4 M_5

Figure 1. Representation of A-string



1 2 1 3 4 2 1 3 2 4 3 3
$O_{1,1}$ $O_{2,1}$ $O_{1,2}$ $O_{3,1}$ $O_{4,1}$ $O_{2,2}$ $O_{1,3}$ $O_{3,2}$ $O_{2,3}$ $O_{4,2}$ $O_{3,3}$ $O_{3,4}$

Figure 2. Representation of B-string



$O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{3,1}$ $O_{3,2}$ $O_{3,3}$ $O_{3,4}$ $O_{4,1}$ $O_{4,2}$
0 3 7 0 2 5 0 4 6 11 2 7

Figure 3. Representation of C-string

One schedule is fully described with the operations, the corresponding machines to be performed and the starting times (or ending times). Note that it is possible that a schedule to be described by the A- and B-strings. But for our purposes we need an additional string. Therefore we introduce array C with the same length $J = J_1 + ... + J_n$ and with the same construction as an array A – see *figure 3*. But each cell of an array C contains the starting time for the corresponding machine from the same cell in the array A.

In this way we code each possible schedule with the help of three arrays: A, B and C. Actually the arrays A and C are sufficient to form the Gant' diagram.

An Algorithmic Scheme

The scheme consists of two basic steps – generating a set of B-strings and generating a corresponding set of A-string and C-string so the triples A-B-C to form feasible schedules of the considered FJSP problem.

Algorithm:

1) Input – parameters of JSP

N – the number of jobs;

J_i – length of the job i, $i = 1, ... ,n$;

M – number of machines.

Table P(J,m) with processor times for each operation and machine.

Here $J = J_1 + ... + J_n$. Note that some cells could be empty if the operation cannot be performed on some machines and vice versa. This could be the case for flexible job shop problems or extended flexible job shop problems [15,12].

L – integer, indicating the number of schedules to be generated.

2) Step – generating a set of B-strings

Randomly generate a set of L "B-strin"s, such that for each string:

"1" appears exactly J_1 times;

"2" appears exactly J_2 times;

....

"n" appears exactly J_n times.

Let us denote the set of "B-string"s as $\{B\} = \{B^1, ..., B^L\}$

3) Step – generating a set of A-strings and C-strings

```
For j = 1, L
do
    For i = 1, n
    do
            FLAG(i) = 0        " flags "
    end
```

```
For i = 1, m
    TM(i) = 0        "initialize summary machine times"
end i
For i = 1, J
    do
    IF (B(j,i) is equal to one of the {1,2, ... , n}
        FLAG(i) = FLAG(i)+1
```

Assign the operation O(B(j,i), FLAG(i)) to the **available** machine Ms with total processor time TM(s) = min TM(g), g = 1,..., m. If more than one machine exists, then select the machine Ms according to the second objective P (f,s) = min {P(f, g), g = 1,..., m}, where f = J_1 + ... + $J_{B(j,i)-1}$ + B(j,i) + FLAG(i) (see the row J_1 + ... + $J_{B(j,i)-1}$ + B(j,i) + FLAG(i) in the table P with processor times).

Again if more than one machine exists then select the machine with smaller index.

Note that in the array TM(s) we take into account both the real processor time and idle time with accumulating.

Note, if TM(s) + P(f,s) = TM(s(O(B(j,i), FLAG(i-1)))) then work with s = s(O(B(j,i), FLAG(i-1))). In other words, we select the same machine on which the previous operation was performed.

Define cell A(f) = s

"Define starting time for operation O(B(j,i), FLAG(i)"

Define cell C(f) = max(TM(s), TM(s(O(B(j,i), FLAG(i-1))),

"Here TM(s(O(B(j,i), FLAG(i-1)) is the ending time for the operation O(B(j,i), FLAG(i-1)) on the machine M(s(O(B(j,i), FLAG(i-1)) with exception for the "first" job's operation – the case "i = 1"

"Update"

TM (s) = max(TM(s), TM(s(O(B(j,i), FLAG(i-1))) + P (f,s)

```
    end i
end j
```

4) Result

The result is a set of a number of L array triples A-B-C containing the precedence relations between operations, the machines to operate and the starting times for each operation. In this way the schedules are fully described.

Discussion

The proposed algorithm is designed for flexible job shop problems.

It can be applied also for the model of extended flexible job shop problems without any essential difficulties – see [12,13]. It can also be applied for the basic job shop problem. Remember that for job shop problems it is known in advance which operation on which machine has to be performed. Therefore the construction of an array A is even easier in this case.

The basic idea for selecting the appropriate machine for operation to be performed is to choose the most "unemployed" machine according to the objective "total machine processing time". In the above algorithmic scheme it is defined as M(j), j= 1, ,... ,m.

The algorithm works as follows. First, a family of arrays B is constructed. After that the arrays A and C are defined taking into account the precedence relations between operations in each job.

The generated schedules are randomly generated in the space of feasible solutions (schedules). Their properties additionally depend by the properties of the probability distribution.

The above algorithm could be applied in methods where a number of schedules have to be processed simultaneously. For example, these can be evolutionary or genetic or parallel methods for solving job shop problems.

An Illustrative Example

Consider the following FJSSP with three machines and three jobs and total of nine operations.

	O(i,j)	M1	M2	M3
J1	O(1,1)	3	5	5
	O(1,2)	4	-	3
J2	O(2,1)	6	5	-
	O(2,2)	-	4	5
	O(2,3)	3	4	5
J3	O(3,1)	6	5	3
	O(3,2)	4	5	6
	O(3,3)	4	4	4
	O(3,4)	2	3	4

We will define two schedules according the above scheme – L = 2.

Let the result from step 2 is {B} = {B^1, B^2} as follows:

B^1 = (3, 2, 1, 1, 2, 3, 3, 2, 3)

B^2 = (2, 3, 1, 3, 2, 1, 3, 2, 3)

Then the result from step3 is:

A^1 = (1, 3, 2, 1, 1, 3, 2, 3, 2)

C^1 = (0, 3, 0, 4, 7, 0, 4, 9, 13)

A^2 = (1, 3, 2, 2, 3, 3, 1, 1, 1)

C^2 = (0, 3, 0, 5, 7, 0, 3, 7, 11)

Conclusion

Populations of solutions are used in all evolutionary and genetic algorithms for solving different optimization problems. In particular, scheduling problems are solved very successfully by means of different heuristic population-based strategies because they are NP-hard problems.

The proposed method generates a set of feasible schedules for FJSP with an arbitrary size. It is a two stage heuristic. The generated solutions could be used for further calculations in different optimization methods to find the optimal solution of FJSP.

Acknowledgement

This study is partially supported by the project No. BG161PO003-1.1.06-0083 to the EU operative program

„Development of Bulgarian economy competitiveness” entitled: “Scientific research for the purposes of development of software tool for generating efficient schedules by an innovative method for multiple objective optimization in discrete manufacturing within the scope of small and medium enterprises”.

References

1. Blazewicz J., K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. Scheduling Computer and Manufacturing Processes. Berlin, Heidelberg, New York, Springer Verlag, 2. Edition, 2001.
2. Bruker, P., R. Schlie. Job Shop with Multi-purpose Machine. – *Computing*, 45, 1990, 369-375.
3. Chiang, Tsung-Che, Lin, Hsiao-Jou. A Simple and Effective Evolutionary Algorithm for Multiobjective Flexible Job Shop Scheduling. – *Int. J. Production Economics*, 141, 2013, 87-98.
4. Daniels, R. Incorporating Preference Information into Multi-Objective Scheduling. – *European Journal of Operational Research*, 77, 1994, 272-286.
5. Garey, M., D. Johnson, and R. Sethi. The Complexity of Flowshop and Jobshop Scheduling. – *Mathematics of Operations Research*, 1, 1976, 2, 117–129.
6. Lawler, E., J. Lenstra, A. Rinnooy Kan, and D. Shmoys. (1993) Sequencing and Scheduling: Algorithms and Complexity. A.H.G. Rinnooy Kan S.C Graves and P.H. Zipkin, Editors. Logistics of Production and Inventory, Elsevier, 4, 1993, 9, 445–522.
7. Lenstra, J. K., A. R. Kan, P. Brucker. Complexity of Machine Scheduling Problems. – *Annals of Discrete Mathematics*, 1, 1977, 343-362.
8. Motaghedi-Iarijani, A., K. Sabri-Iaghaie, M. Heydari. Solving Job Shop Scheduling with Multi Objective Approach. – *Int. J. of Industrial Engineering and Production Research*, 21, 2010, No 4, 197-209.
9. Parveen, S. and H. Ullah. Review on Job-Shop and Flow-Shop Scheduling Using Multi Criteria Decision Making. – *Journal of Mechanical Engineering*, ME 41, December 2010, No. 2, Transaction of the Mech. Eng. Div., The Institution of Engineers, Bangladesh, 130-146.
10. Pinedo, M. Planning and Scheduling in Manufacturing and Services. Berlin, Heidelberg, New York, Springer Verlag, 2005.
11. T'kindt, V., J.-C. Billaut. Multicriteria Scheduling: Theory, Models and Algorithms. Berlin, Heidelberg, New York, Springer Verlag, 2002, 2006.
12. Kirilov, L., V. Guliashki. (2014) An Extension of Flexible Job Shop Problem (FJSP) and Method for Solving. Proceedings of CompSysTech '14 Proceedings of the 15th International Conference on Computer Systems and Technologies, ACM New York, NY, USA ©2014, ACM International Conference Proceeding Series, 883, 2014, 210-217.
13. Kirilov, L., V. Guliashki, K. Genova. Shifting Bottleneck (SB) Approach for Solving Job Shop Scheduling Problems (JSSP’s). Proceedings of Int. Conference Robotics, Automation and Mechatronics’14 RAM 2014, 5-7 November 2014, Sofia, 22-25.
14. Zhang, G., X. Shao, P. Li, L. Gao. An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-objective Flexible Job Shop Scheduling Problem. – *Computers & Industrial Engineering*, 56, 2008, 4, 1309-1318.
15. Birgin, E., P. Feofiloff, C. Fernandes, E. de Melo, M. Oshiro and D. Ronconi. A MILP Model for an Extended Version of the Flexible Job Shop Problem. – *Optimization Letters, 8*, 2013, No. 4, 1417-1431.

Manuscript received on 05.12.2016

__Leoneed Kirilov__ works at the Institute of Information and Communication Technologies – Bulgarian Academy of Sciences, Bulgaria. His research interests are Multiple Criteria Decision Making, Decision Support Systems, Modelling, Optimization Methods and Applications (Multiple Objective Optimization, Decision Analysis, Decision Making, Hydrologic Modelling, Schedule Optimization, Monte Carlo Simulation, Materials Science, Nanotechnology). Up to now his research activity is published in about 100 papers and two monographs (almost all in English). He is a member of International Society on MCDM, Union of Scientists in Bulgaria.

Contacts:
Institute of Information and Communication Technologies (IICT)
Bulgarian Academy of Sciences, Sofia, Bulgaria
www.iict.bas.bg, lkirilov@iinf.bas.bg, l_kirilov_8@abv.bg

__Vassil Guliashki__ is currently an Associate Professor at the Institute of Information and Communication Technologies – Bulgarian Academy of Sciences (IICT – BAS), Department “Information Processes and Decision Support Systems”. He earned his Master of Science (Automation and System-Technique) at Technical University – Sofia, Bulgaria, in 1988 and his PhD in Technical Cybernetics from the Institute of Information Technologies – BAS (IIT – BAS since 1992) in 1994, with scientific advisors Prof. Vassil Vassilev and Prof. Vassil Sgurev. In his thesis work entitled “Algorithms for Solving Convex Nonlinear Integer Programming Problems” he created a Tabu Search heuristic algorithm for single objective problems and an interactive reference direction algorithm for multiple objective problems. After completion of his PhD, Vassil Guliashki accepted a position as Research Associate II and I degree resp. at the IIT – BAS in the period 1995–1997. In 2009, V. Guliashki accepted an Associate Professor position at the IIT – BAS. The main research areas of Vassil Guliashki are Discrete Optimization, Meta heuristic strategies, Evolutionary algorithms, Multiple Objective Programming, Decision Support Systems, Linear Discriminant Analysis, Combinatorial optimization. He participates in solving many practical problems by means of single and MCDM approaches. Vassil Guliashki has more than 100 refereed publications, one monograph book. He takes participation in more than 30 research and applied projects up to now.

Contacts:
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences, Sofia, Bulgaria
e-mails: vggul@yahoo.com, v_guliashki@bas.bg