Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 198 (2022) 410–416

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Information Sciences and Advanced Technologies : Edge Big Data - AI- IoT (ISAT 2021)
November 1-4, 2021, Leuven, Belgium

Solving Non-Delay Job-Shop Scheduling Problems by a new matrix heuristic

Lotfi NOHAIR*, Abderrahim EL ADRAOUI , Abdelwahed NAMIR

University Hassan II of Casablanca, Morocco

Abstract

In this paper, we address the non-delay job-shop scheduling problem. The objective is to minimize the makespan. The non-delay schedules are active schedules in which we will not keep machine idle when there is at least one job waiting to be executed on that machine. In this contribution, we present a new matrix heuristic to generate non-delay schedules by getting matrices of starting and finishing times. The developed heuristic is computationally fast, simple and easy to implement. Finally, a comparative study between two variants of the developed heuristic highlight that the use of an appropriate Priority Dispatching Rule (PDR) for each machine gives best results comparing with the heuristic which applies the same PDR for all machines.

Keywords: Type your keywords here, separated by semicolons ;

1. Introduction

The deterministic Job Shop Scheduling Problem (JSSP) [1] is a well-known combinatorial optimization problem in operations research, and is omnipresent in many application domains as manufacturing [2], network and supply chains [3, 4]. JSSP consists of a finite set J of n jobs to be processed on a finite set M of m machines. Each job must

* Corresponding author. Tel.: +212696029100.
E-mail address: lotfi.nohair-etu@etu.univh2c.ma

10.1016/j.procs.2021.12.262

be processed on every machine and consists of a chain of m operations which have to be scheduled in a predetermined order. The goal is to optimize a designed objective such as minimizing the time needed to complete all jobs i.e., the makespan. JSSP is an NP-hard problem [5]. Therefore, finding optimal schedules to JSSP is very time consuming and usually impractical. Generally, the practice to find a near-optimal solution is based on heuristics, known as "dispatching rules" [6, 7], or on metaheuristics [8] such as Simulated Annealing [9], Tabu Search [10], ACO [11] and Genetic Algorithms [12].

In this paper, we interest to the non-delay job-shop scheduling problem with minimizing makespan as objective. A schedule is non-delay if we will not keep machine idle when there is at least one job waiting to be processed on that machine [13, 14]. To generate non-delay schedules, Gantt Chart [15] is a tool that can be used manually with priority dispatching rules (PDR) for small size instances. If we increase the number of operations, generating manually non-delay schedules becomes very difficult and time-consuming.

The purpose of this research is to propose a matrix heuristic to generate non-delay schedules using priority dispatching rules in order to find optimal or near optimal schedule. Therefore, the need to choose appropriate Dispatching rules appears. Simulation is used to evaluate the quality of scheduling solution. A comparative study in this paper proves that the use of appropriate PDR for each machine, is very efficient than the use of one rule for all machines. We firstly present a mathematic statement of the job-shop scheduling problem and we discus related works. The proposed methodology especially the developed algorithm for generating non-delay schedules will be described in Section 3. We conclude with computational results to support our main results.

2. BACKGROUND AND RELATED WORKS

2.1. Problem statement

Job-shop scheduling problem consists of a set of n jobs $J = \{J_1, J_2, ..., J_n\}$ and a set of m machines $M = \{M_1, M_2, ..., M_m\}$. Each job $J_j \in J$ must go through all machines in a predetermined order. We use:

$\sigma(i, j)$ to denote the machine on which the ith operation of job j should be processed.

$P_{j,\sigma(i,j)}$ to denote the processing time of the job j on the machine σ(i, j).

$\sigma(i, j)$ to denote the starting time of the ith operation of the job j on the machine $\sigma(i, j)$.

The assumptions considered are [16]:

1) An operation can not be interrupted when it has been started (Non-preemption).
2) A machine can process at most one job at a time (Capacity).
3) An operation can only be scheduled after the preceding ones have ended (Precedence).
4) Machines are never kept idle while work is waiting.

$$\text{Define the makespan : } \quad C_{max} = max_{j\in[1,n]} \{S_{j,\sigma(n,j)} + P_{j,\sigma(n,j)}\} \tag{1}$$

Under the above assumptions, Job-shop scheduling problem can be formulated as follows:

$$\text{Minimize } C_{max} = max_{j\in[1,n]} \{S_{j,\sigma(n,j)} + P_{j,\sigma(n,j)}\} \tag{2}$$

Subject to:

$$S_{j,\sigma(i,j)} + P_{j,\sigma(i,j)} \le S_{j,\sigma(i+1,j)} \qquad j \in [1,n], \quad i \in [1,m-1] \tag{3}$$

$$S_{j,\sigma(i,j)} + P_{j,\sigma(i,j)} \le S_{p,\sigma(k,p)} \quad or \quad S_{p,\sigma(k,p)} + P_{p,\sigma(k,p)} \le S_{j,\sigma(i,j)} \tag{4}$$

$$j, p \in [1,n], i, k \in [1,m] \setminus \sigma(i, j) = \sigma(k, p)$$

$$S_{j,\sigma(1,j)} \ge 0 \qquad j \in [1,n] \tag{5}$$

Equation (1) calculates the makespan C_{max}. Equation (3) allows to respect precedence constraints. Equation (4) ensures capacity or resource constraints. Equation (5) ensures that the starting times of all operations must be positive numbers.

2.2 Related works

In the literature, we find two categories of methods to solve Job Shop Scheduling problems: exact and approximate methods [1]. Exact methods such as Mixed Integer Linear Programming MILP, Branch and Bound, etc. explore the entire solution space to find optimal solution. Thus, these methods are very time consuming to solve JSSP as NP-hard problem. Approximate methods are used to find a near-optimal solution in a short time. In this category, metaheuristics try to improve an initial solution by research in its neighborhood. The improvement-based heuristics include Simulated Annealing [9], Tabu Search [10] and Evolutionary Algorithms such as Genetic Algorithms [12], Particle Swarm Optimization [17], Ant Colonies Optimization [4]... However, the improvement-based heuristics are not practical when the schedule is executing especially in case of a machine breakdown or arrival of new jobs. They have a poor scalability for large problems [18]. This led [18] to use construction based heuristics, known as Priority Dispatching Rules (PDR) which have linear complexity. The majority of priority rules are summarized in [19]. They provide feasible solutions very quickly and they are easily implementable. The goal is to choose the best PDR which optimize a predefined objective. In most of studies which use PDR, authors use only one PDR for all machines [6]. This approach is not efficient since no a PDR outperform the others and JSSP is not analyzed globally. On the other hand, simulation allows to find best PDR for each machine simultaneously. [20] apply LRPT (Longest Remaining Processing Time) rule as the priority rule, and the SPT (Shortest Processing Time) rule as the tie-breaking rule. Also [20] use max-plus algebra as tool to generate feasible schedule for non-delay job-shop scheduling problem. This approach can't lead to find a near optimal or optimal schedule. In our paper we are limited to non-delay job-shop scheduling problems for two main reasons: first, they are not yet studied and there a lack of sample methods to generate non- delay schedules, and secondly, non-delay schedules is small subset of active schedules.

3. PROPOSED METHODOLOGY

3.1 A matrix heuristic for generating a non-delay schedule

Firstly, we generate the initial matrix of release times of operations $R(0)$ of dimension $(n \times m)$. Iteratively, we determine the earliest start time τt and the operation which is scheduled in each stage. If several operations have the same value τ_t, the priority dispatching rule will be applied, and if there is another tie, an additional tie-breaking rule will be applied. τ_t indicates the starting time of the operation to be scheduled in stage t. Finally, our non-delay schedule is completely described by the matrices of starting times S and completion times C. Our heuristic consists of six steps as follows:

Input: the number of tasks n; the number of machines m.

Priority Dispatching Rules $PDR = \{PDR_1; ...; PDR_m\}$, where PDR_i is the priority dispatching rule used for the machine i. $\sigma(i, j)$: the machine used to process the ith operation of job j.

$P_{j,\sigma(i,j)}$: the processing time of the job j on the machine $\sigma(i, j)$.

Step 1: Let t=0. Generate the initial matrix $R(0)$ where:

$$[R(0)]_{j,\sigma(i,j)} = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{k=1}^{i-1} P_{j,\sigma(k,j)} & \text{if not} \end{cases} \quad (6)$$

Repeat

Step 2:

Determine τ_t in stage t, where $\tau_t = min[R(t)]_{j,\sigma(i,j)} \qquad (i, j) \in [1, m] \times [1, n]$ (7)

Step 3: The value of τ_t determines the operation (j^*, m^*) to be scheduled in stage t.

a) If several operations have the same minimum value τ_t and use the same machine, apply the priority dispatching rule of this machine.

b) If there is tie, apply a tie-breaking rule.

Step 4:

Schedule the operation (j^*, m^*). The starting time of the operation (j^*, m^*): $[S(t)]_{(j^*,m^*)} = \tau_t$ (8)

The completion time of the operation (j^*, m^*): $[C(t)]_{(j^*,m^*)} = [S(t)]_{(j^*,m^*)} + [P]_{(j^*,m^*)}$ (9)

Step 5: Update the matrix $R(t+1)$ in stage t+1:

$$[R(t+1)]_{(j^*,m^*)} = +\infty \quad (10)$$

$$[R(t+1)]_{(j,m^*)} = max([R(t)]_{(j,m^*)}; [C(t)]_{(j^*,m^*)}) \quad (11)$$

If $[R(t)]_{(j,m^*)} \neq +\infty$ and $[R(0)]_{j,\sigma(i,j)} > [R(0)]_{(j,m^*)}$ then

$$[R(t+1)]_{j,\sigma(i,j)} = max([R(t)]_{j,\sigma(i,j)}; [R(t+1)]_{(j,m^*)} + ([R(0)]_{j,\sigma(i,j)} - [R(0)]_{(j,m^*)})) \quad (12)$$

End if

Step 6: Increment t by one: $t \leftarrow t+1$

Until all operations have been scheduled $t = m \times n$

3.2 Numerical example:

Table 1: Example of 3 × 3 job shop scheduling problem

J1	M1 (7) M3 (8) M2 (10)
J2	M2 (6) M1 (4) M3 (12)
J3	M1 (8) M2 (8) M3 (7)

Let us apply our heuristic using an example case. Let us call the three jobs as J1, J2 and J3. Let us call the three machines as M1, M2 and M3. Each job has its pre-specified route. So, the job J1 will first visit machine M1, then visit machine M3, and then visit machine M2 and then leaves. The number shown in the brackets are the processing times for the particular job on the particular machine. We use the SPT as priority dispatching rule to solve this example. Firstly, the matrices P and R(0) are generated:

$$P = \begin{pmatrix} 7 & 10 & 8 \\ 4 & 6 & 12 \\ 8 & 8 & 7 \end{pmatrix} \qquad R(0) = \begin{pmatrix} 0 & 15 & 7 \\ 6 & 0 & 10 \\ 0 & 8 & 16 \end{pmatrix} \quad (13)$$

The earliest possible start time in stage $t = 0$: $\tau_0 = min[R(0)]_{j,\sigma(i,j)} = 0$ (14)

At time equal to 0, the operations 11 and 31 can be scheduled on machine M1 and the operation 22 can be scheduled on M2. Without loss of generality, let's start with the machine M1. As there are two operations, we have to choose one of theme for processing. The priority rule must be applied. The operation 11 has a shorter processing time (7 time units) than operation 31 (8 time units). Thus, the operation 11 is scheduled. The starting time and the completion time of the operation 11 are:

$$S_{11} = \tau_0 = 0 \quad ; \qquad C_{11} = S_{11} + P_{11} = 0 + 7 = 7 \quad (15)$$

The updated matrix of release times R(1) equals: $R(1) = \begin{pmatrix} +\infty & 15 & 7 \\ 7 & 0 & 11 \\ 7 & 15 & 23 \end{pmatrix}$ (16)

Eight subsequent stages are carried out in this way. All operations are scheduled. The matrices S and C are generated:

$$S = \begin{pmatrix} 0 & 15 & 7 \\ 7 & 0 & 15 \\ 11 & 25 & 33 \end{pmatrix} \qquad C = \begin{pmatrix} 7 & 25 & 15 \\ 11 & 6 & 27 \\ 19 & 33 & 40 \end{pmatrix} \tag{17}$$

The makespan equals 40 time units. The results obtained with our heuristic match those obtained by Gantt chart with SPT rule (see Figure 1). Bigger-sized problems were fastly and easily solved using our programmed heuristic in MATLAB.

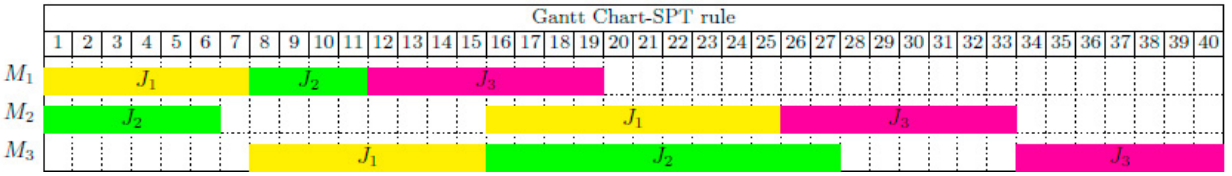

Fig. 1: The Gantt chart of the non-delay schedule from Table 1 using SPT rule

4. COMPUTATIONAL RESULTS

There are two variants of our heuristic depending on whether we use the same priority rule for all machines denoted HR1; or whether we use an appropriate rule for each machine denoted HR2. The two heuristics HR1 and HR2 are applied on the benchmarks of FT (Fisher and Thompson) and LA (Lawrence). We propose comparative studies between the two heuristics HR1 and HR2. Tables 2 and 3 present the makespan obtained for each PDR (Priority Dispatching Rule) by HR1 and HR2 heuristics. These tables also contain the Relative Percentage Deviation (RPD) between the obtained makespan by the heuristic S_h and the optimal makespan S^* for JSSP.

$$RPD = \frac{C_{max}(S_h) - C_{max}(S^*)}{C_{max}(S^*)} \times 100 \tag{18}$$

The Relative Percentage Deviation (RPD) varies between 0% and 26.7% for the heuristic HR1. While the RPD varies between 0% and 12.8% for the heuristic HR2. We can conclude that using appropriate PDR for each machine (HR2) gives best results comparing with HR1 which applies the same PDR for all machines. These heuristics allow us fast admissible solutions, that can be used as initial solutions for others metaheuristics.

Table 2: Experimental results on FT instances for heuristics HR1 & HR2

Instance	Size	S*	HR1	RPD1	HR2	RPD2
FT06	6*6	55	61	10,9	61	10,9
FT10	10*10	930	1074	15,5	1015	9,1
FT20	20*5	1165	1267	8,8	1267	8,8

Table 3: Experimental results on LA instances for heuristics HR1 & HR2

Instance	Size	S*	HR1	RPD1	HR2	RPD2
LA01	10*5	666	735	10,4	696	4,5
LA02	10*5	655	756	15,4	705	7,6
LA03	10*5	597	672	12,6	662	10,9
LA04	10*5	590	675	14,4	644	9,2
LA05	10*5	593	593	0,0	593	0,0
LA06	15*5	926	926	0,0	926	0,0
LA07	15*5	890	974	9,4	890	0,0
LA08	15*5	863	942	9,2	895	3,7
LA09	15*5	951	1015	6,7	951	0,0
LA10	15*5	958	972	1,5	958	0,0
LA11	20*5	1222	1247	2,0	1222	0,0
LA12	20*5	1039	1156	11,3	1057	1,7
LA13	20*5	1150	1157	0,6	1150	0,0
LA14	20*5	1292	1292	0,0	1292	0,0
LA15	20*5	1207	1343	11,3	1237	2,5
LA16	10*10	945	1138	20,4	1021	8,0
LA17	10*10	784	846	7,9	806	2,8
LA18	10*10	848	1004	18,4	923	8,8
LA19	10*10	842	940	11,6	886	5,2
LA20	10*10	902	964	6,9	939	4,1
LA21	15*10	1046	1219	16,5	1120	7,1
LA22	15*10	927	1114	20,2	997	7,6
LA23	15*10	1032	1146	11,0	1060	2,7
LA24	15*10	935	1135	21,4	1035	10,7
LA25	15*10	977	1188	21,6	1087	11,3
LA26	20*10	1218	1413	16,0	1305	7,1
LA27	20*10	1235	1488	20,5	1393	12,8
LA28	20*10	1216	1541	26,7	1353	11,3

5. CONCLUSION, LIMITATIONS AND PERSPECTIVES

In this paper, we address the no-delay job shop scheduling problem with the objective of minimizing makespan. We need to explore the search space by generating no-delay schedules in order to find optimal or near-optimal schedule. The main contribution of this article is new matrix heuristic for generating no-delay schedules. The developed heuristic is computationally fast, simple and easy to implement. The final solution is presented in the form of matrix of starting times S. Thus we can easily obtain the matrix of completion times C and the makespan of schedule. A numerical example was presented to explain our approach. Finally, comparative studies between the two variants of developed heuristic, highlight that using appropriate PDR for each machine gives best results comparing with the heuristic which applies the same PDR for all machines.

Given the simplicity and responsiveness of our heuristic, it can be used easily for complex industries and dynamic environments. Also, it can be used to simulate different scenarios trying to choose the best set of PDR in order to optimize a chosen objective. However, our approach don't usually provide optimal solution and the Relative Percent Deviation RPD reach 12.8%, Which is quite normal, as our heuristic is designed to generate admissible solutions for no-delay job shop scheduling. In the future, we plan to enhance the developed heuristic to solve active job-shop scheduling problem to improve the quality of best solution.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] Jain, Anant Singh, and Sheik Meeran. (1999) "Deterministic job-shop scheduling: Past, present and future." European journal of operational research 113(2): 390-434.
[2] Zhang, Jian, Guofu Ding, Yisheng Zou, Shengfeng Qin, and Jianlin Fu. (2019) "Review of job shop scheduling research and its new perspectives under Industry 4.0." Journal of Intelligent Manufacturing 30 (4): 1809-1830.
[3] Enzo Morosini Frazzon, Andre Albrecht, Matheus Pires, Eduardo Israel, Mirko Kück, and Michael Freitag. (2018) "Hybrid approach for the integrated scheduling of production and transport processes along supply chains." International Journal of Production Research, 56(5): 2019–2035.
[4] Kai Wang, Hao Luo, Feng Liu, and Xiaohang Yue. (2017) "Permutation flow shop scheduling with batch delivery to multiple customers in supply chains." IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48(10): 1826–1837.
[5] Lenstra, Jan Karel, AHG Rinnooy Kan, and Peter Brucker. (1977) "Complexity of machine scheduling problems." In Annals of discrete mathematics, vol. 1, pp. 343-362.
[6] Mohamed Habib Zahmani, Baghdad Atmani, Abdelghani Bekrar, and Nassima Aissani. (2015) "Multiple priority dispatching rules for the job shop scheduling problem." In 2015 3rd International Conference on Control, Engineering & Information Technology (CEIT), pages 1–6. IEEE.
[7] Kaban AK, Othman Z, Rohmah DS. (2012) "Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study." International Journal of Simulation Modelling. 11(3): 129-140.
[8] Pisut Pongchairerks. (2019) "A two-level metaheuristic algorithm for the job-shop scheduling problem." Complexity 2019.
[9] Ponnambalam SG, Jawahar N, Aravindan P. (1999) "A simulated annealing algorithm for job shop scheduling." Production Planning & Control. 10(8):767-777.
[10] Bożejko, Wojciech, Andrzej Gnatowski, Jarosław Pempera, and Mieczysław Wodecki. (2017) "Parallel tabu search for the cyclic job shop scheduling problem." Computers & Industrial Engineering 113: 512-524.
[11] Wu J, Wu GD, Wang JJ. (2017) "Flexible job-shop scheduling problem based on hybrid ACO algorithm." International Journal of Simulation Modelling. 16(3):497-505.
[12] Nisha Bhatt and Nathi Ram Chauhan. (2015) "Genetic algorithm applications on job shop scheduling problem: A review." In 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), pp 7–14. IEEE.
[13] Yuval Cohen, Arik Sadeh, and Ofer Zwikael. (2012) "Finding the shortest non-delay schedule for a resource-constrained project." International Journal of Operations Research and Information Systems (IJORIS), 3(4): 41–58.
[14] Arno Sprecher, Rainer Kolisch, and Andreas Drexl. (1995) "Semi-active, active, and non-delay schedules for the resource- constrained project scheduling problem." European Journal of Operational Research, 80(1): 94–102.
[15] Aleksandar Stanković, Goran Petrović, Žarko Ćojbašić, and Danijel Marković. (2020) "An application of metaheuristic opti- mization algorithms for solving the flexible job-shop scheduling problem." Operational Research in Engineering Sciences: Theory and Applications, 3(3): 13–28.
[16] Carlos Mencía, María R Sierra, and Ramiro Varela. (2013) "Depth-first heuristic search for the job shop scheduling problem." Annals of Operations Research, 206(1): 265–296.
[17] Zhang Z, Guan ZL, Zhang J, Xie X. (2019) "A novel job-shop scheduling strategy based on particle swarm optimization and neural network." Int. J. Simul. Model. 18(4): 699-707.
[18] Chen, Binchao, and Timothy I. Matis. (2013) "A flexible dispatching rule for minimizing tardiness in job shop scheduling." International Journal of Production Economics, 141(1):360–365.
[19] Shrikant S Panwalkar and Wafik Iskander. (1977) "A survey of scheduling rules." Operations research, 25(1): 45–61.
[20] Tena Žužek, Aljoša Peperko, and Janez Kušar. (2019) "A max-plus algebra approach for generating non-delay