International Journal of Production Management and Engineering

doi:10.4995/ijpme.2017.6618

Received 2016-09-20 Accepted: 2016-10-03

A hybrid algorithm for flexible job-shop scheduling problem with setup times

Ameni Azzouz*, Meriem Ennigrou and Lamjed Ben Said

Lab. SOIE. Stratégies d'Optimisation et Informatique IntelligentE, ISG, Institut Supérieur de Gestion, Université de Tunis, Tunisie.

ameni.azzouz@isg.rnu.tn

Abstract: Job-shop scheduling problem is one of the most important fields in manufacturing optimization where a set of n jobs must be processed on a set of m specified machines. Each job consists of a specific set of operations, which have to be processed according to a given order. The Flexible Job Shop problem (FJSP) is a generalization of the above-mentioned problem, where each operation can be processed by a set of resources and has a processing time depending on the resource used. The FJSP problems cover two difficulties, namely, machine assignment problem and operation sequencing problem. This paper addresses the flexible job-shop scheduling problem with sequence-dependent setup times to minimize two kinds of objectives function: makespan and bi-criteria objective function. For that, we propose a hybrid algorithm based on genetic algorithm (GA) and variable neighbourhood search (VNS) to solve this problem. To evaluate the performance of our algorithm, we compare our results with other methods existing in literature. All the results show the superiority of our algorithm against the available ones in terms of solution quality.

Key words: Job-shop scheduling problem, Flexible manufacturing systems, Sequence-dependent setup times, Genetic algorithms, Local search.

1. Introduction

Flexible-job-shop scheduling problem (FJSP) is a well-known NP-hard problem (Garey *et al.,* 1976), which reflect a wide range of scheduling problems encountered in real manufacturing systems. For this reason, FJSP continues to attract the interests of researchers both in academia and industry.

This problem mainly cover two difficulties: the first one is resource assignment problem where each operation can be processed by more than one resource from a set of available resource and has, consequently, a processing time depending on the resource used. The second one is operation sequencing problem with maintaining the feasibility conditions. Recently, many researches have been made to find the near optimal solution of FJSP using a varied range of tools and techniques such as Branch and Bound (Fatahi *et al.,* 2007; Zribi *et al.,* 2007) and Heuristics (Wang and Yu, 2010; Ziaee, 2014). FJSP is known to be strongly NP-hard. Consequently, most of the literature related to the FJSP is based on meta-heuristic methods like genetic algorithms (GAs) (Zhou *et al.,* 2006; Pezzella *et al.,* 2008; Zhang *et al.,* 2011; Zambrano Rey *et al.,* 2014), particle swarm optimization (PSO) (Zhang *et al.,* 2009; Nouiri *et al.,* 2015) simulated annealing (SA) (Najid *et al.,* 2002; Yazdani *et al.,* 2009), tabu search (TS) (Brandimarte, 1993; Fatahi *et al.,* 2007; Vilcot and Billaut, 2011) and beam search (BS) (Wang *et al.,* 2008).

Most job-shop scheduling researches reported in the literature ignore the setup times or consider them as a part of the processing time. However, in many real-life situations such as chemical, printing, pharmaceutical and automobile manufacturing (Kim and Bobrowski, 1994), the setup times are not only often required between jobs but they are also strongly dependent on job itself (sequence independent) and the previous job that ran on the same machine (sequence dependent). Hence, reducing setup times is an important task to improve shop performance. The FJSP has been widely studied. However, few papers have considered this problem with setup times.

Among these, Cheung and Zhou (2001) propose a hybrid algorithm based on genetic algorithm and heuristic rules to solve SDST-JSP with minimizing the makespan. For the same problem, Zhou *et al.* (2006) propose an immune algorithm which certifies the diversity of the antibody. Moghaddas and Houshmand (2008) develop a mathematical and heuristic model based on priority rules. Naderi *et al.* (2009) consider the job shop scheduling with sequence-dependent setup times and preventive maintenance policies using four meta-heuristics based on simulated annealing and genetic algorithms.

Considering the flexibility constraints, flexible job-shop problem presents additional difficulty than the classical JSP and requires more effective algorithms. In recent decades, many attempts have been made to find the near optimal solution of SDST-FJSP using a varied range of tools and techniques. Imanipour (2006) was the first one who investigates the SDST-FJSP. The author modeled the problem as a non linear mixed integer programming model and proposes a tabu search for the same problem. Saidi-Mehrabad and Fattahi (2007) presented a Tabu Search for solving the SDST-FJSP to minimize makespan. They assumed in their research that each operation can be performed by two machine alternatives. They compared their obtained results with the results of the lingo software. Bagheri and Zandieh (2011) propose a variable neighborhood search (VNS) based on integrated approach to minimize an aggregate objective function (AOF) where AOF = αF1+(1-α)F2 and α denote the weight given respectively to makespan (F1) and mean tardiness (F2). To evaluate this model, the authors generate randomly 20 problem instances under four different classes. Using the same AOF, Sadrzadeh (2013) present an artificial immune system algorithm (AIS) and a particle swarm optimization algorithm (PSO) and prove that both algorithms works better than VNS of Bagheri and Zandieh (2011). Mousakhani (2013) formulate the SDST-FJSP as a mixed integer linear programming model to minimize total tardiness and present a meta-heuristic based on iterated local search for the same problem. Oddi *et al.* (2011) considers the SDST-FJSP to minimize the makespan using the iterative flattering search (IFS) and propose a new benchmark which is denoted SDST-HUdata. It consists of 20 instances produced as an extension of the existing well-known benchmarks of FJSP of Hurink *et al.* (1994). Gonzàlez *et al.* (2013) develop memetic algorithm to minimize the makespan which the tabu search was applied to every chromosome generated by the genetic algorithm. In order to evaluate their model, they used the same benchmark as in Oddi *et al.* (2011) and prove that the memetic algorithm has obtained a better result than the IFS. Recently, Rossi (2014) investigate the SDST-FJSP with transportation times using ant-colony algorithm with reinforced pheromone. The most recent comprehensive survey of scheduling problem with setup times is given by Allahverdi (2015).

Nevertheless, most of the above-mentioned research considered only one method optimization to solve SDST-FJSP. However, the literature reviews show that none of these methods are sufficient on their own to solve this NP-hard problem. For that, in this paper, we propose a hybrid genetic algorithm (HGA) based on GA and VNS for the SDST-FJSP. Then, we show that our algorithm can be very effective with respect to the state of the art. The remainder of this paper is organized as follows. In section 2, we formulate the problem and we give an illustrative example. Section 3 presents the proposed algorithm to solve the SDST-FJSP. Section 4 describes the performance of our algorithm on a set of benchmark problems and explains the most interesting results. Conclusions and some future works are presented in section 5.

2. Problem definition

The SDST-FJSP can be defined as follows: this problem consists in performing n jobs on m machines. The set machines is noted M, M={M1,... ,Mk}. Each job i consists of a sequence of ni operations (routing). Each routing has to be performed to complete a job. The execution of each operation j of a job i (noted Oij) requires one machine out of a set of given machines Mi,j (i.e. Mi,j is the set of machines available to execute Oij). The problem is to define a sequence of operations together with assignment of start times and machines for each operation.

Assumptions considered in this paper are the following: (1) jobs are independent of each other; (2) machines are independent of each other; (3) one machine can process at most one operation at a time; (4) no preemption is allowed; (5) all jobs are available at time zero; (6) Setup times are dependent on the sequence of jobs. When one of the operations of a job t is processed before one of those of job i (t≠i) on machine Mk, the sequence dependent setup time is St,i,k>0.

The current SDST-FJSP based on these assumptions is aimed to minimize two kinds of objective functions:

- Minimize the makespan (i.e. the time required to complete all jobs)
- Minimize Aggregate objective function (AOF) where AOF=αF1+(1-α)F2 and α denote the weight given respectively to makespan (F1) and mean tardiness (F2).

FJSP is classified as Total FJSP and Partial FJSP (Kacem *et al.,* 2002). In Total FJSP (T-FJSP), each operation can be processes by all machines. However, in Partial FJSP (P-FJSP), at least one operation may not be processed on all machines. Several researches pointed out that the P-FJSP is more complex as compared to T-FJSP on the same scale. In this paper, we consider the P-FJSP.

To illustrate this problem, we consider an instance with three jobs and three machines. In Table 1, we show the processing time of each operation. The symbol "-" means that the machine can not execute the corresponding operation. Table 2 presents the sequence dependent setup times of each job. For instance, the setup time for job3 after job2 on machine M1 is S2,3,1=2 and the setup time for job2 after job1 on machine M3 is S1,2,3=2.

Table 1. Processing times.

Job	Operation	M1	M2	M3
J1	O11	4	-	5
	O12	-	3	4
	O13	6	5	-
J2	O21	3	-	4
	O22	4	5	-
	O23	-	4	7
J3	O31	5	3	-
	O32	-	4	-
	O33	4	5	3

Table 2. Sequence-dependent setup times.

	Machine1			Machine2			Machine3		
Job	Job1	Job2	Job3	Job1	Job2	Job3	Job1	Job2	Job3
Dummy	3	2	1	2	4	1	4	3	2
Job1	0	1	3	0	2	4	0	2	3
Job2	1	0	2	4	0	3	2	0	4
Job3	1	3	0	2	1	0	3	2	0

Dummy job signifies the starting of a job on each machine. When one of the operations of a jobi is the first operation executed on machine Mk, the dummy job is Di,k. For example, if O3,1 is the first operation executed on machine M2, then, dummy job value would be D3,1=1. In order to explain better this problem, we represent a Gantt Chart of solution on Figure 1.

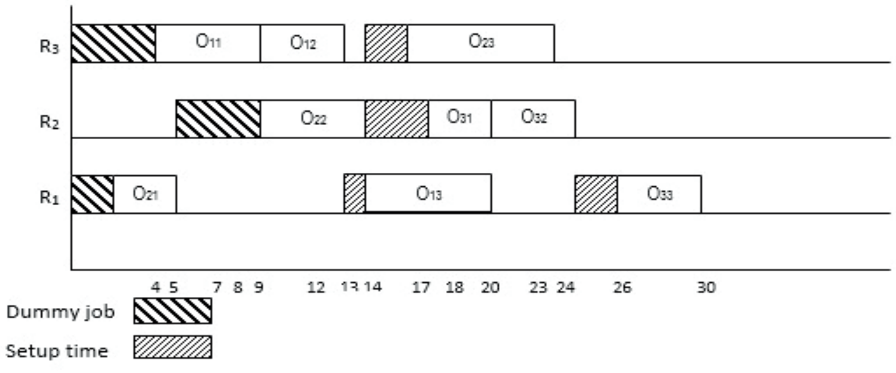

Figure 1. Gantt Chart of solution.

3. Hybrid Genetic algorithm for SDST-FJSP

Since the discovery of the genetic algorithms by Holland (1975), they have been recognized as a powerful methods for solving combinatorial optimization problems such as scheduling problems. GA can find problem solutions by imitating natural selection mechanisms, using choosing, crossing and mutation operations. In this section, we present the step by step algorithm included our adaptation of the different HGA parameters to our problem.

3.1. HGA procedures

Step1 (Initialization Scheme): Generate an initial parent population of N solutions. Here, we propose an improved function of initial population which is based on three traditional dispatching rules as following: 20% using shortest processing time (SPT); 20% using longest processing time (LPT); 20% using heuristic rules based on local search algorithm; the remaining with random solution. We coded the solution as binary matrix where the rows represent Furthermore, the order in which they appear in the matrix describes the sequence of operations present in the solution. The columns correspond to the used machines as described in the Figure 2.

Step2 (selection): Choose (N/2) members from the parent population using tournament selection.

Step3 (Reproduction phase): Regarding to a certain probability, we perform crossover, mutation or VNS algorithm in order to create the offspring population. For that, we use the crossover operator order 1 which consists on selecting randomly two positions XP1 and XP2 in parent1 (which is coded by a binary matrix). Subsequently, the middle part is copied to the offspring1. The rest of this child is filled from the parent2 starting with position XP2+1, and jumping elements that are already presented in offspring1. The same steps are repeated for the second offspring by starting with the parents2. Figure 2 shows an example of this procedure. Till now to the mutation operator, we use the mutation technique proposed by Pezzella *et al.* (2008), in which, we select one operation with the maximum workload (i.e. the amount of work that a machine produces in a specified time period). Then we assign it to the machine with the minimum workload if possible. The VNS algorithm is used as a reproduction operator. In the next section, we will focus our attention on the specific VNS we have used.

Step4 (Environmental selection): Fill the new population using a replacement strategy. The new population is formed with the N best solutions. If the stopping criterion is met then return the best solution; otherwise, return to Step 2.

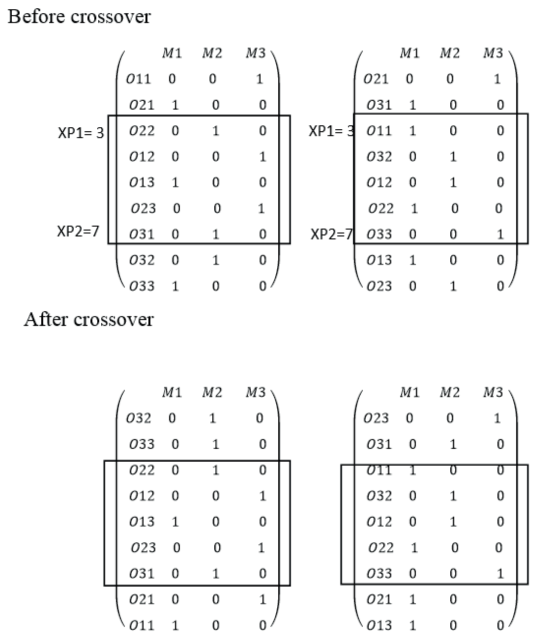

Figure 2. Crossover operator.

3.2. VNS algorithm

Traditionally, the hybridization between GA and local search algorithm is based on simple local search. However, we adapted an advanced local search called variable neighborhood search (VNS) inspired from Bagheri and Zandieh (2011) and Ennigrou and Ghedira (2008) as a local search for our algorithm. More in detail, this VNS based on three kinds of neighborhood structures presented as following:

Step1 (initial solution): we conserve the same solution representation as in GA.

Step2 (Generate neighborhood): In this step, we determine the neighborhood of the current solution. Then, we use three types of moves:

- Two positions p1 and p2 are randomly selected on the solution and then, the operations between them are randomly reordered.

- Operation is chosen randomly and then, changing the assignment of the selected operation to another machine.
- The combination between the two precedent neighborhood structures

Step3 (neighborhood evaluation): After applying each of the neighborhood structures described above, we execute a local search for certain iteration. We use the same local search as in.

Step4 (Final Stage): After a number of iteration of VNS algorithm, the best solution founded is selected to the next population.

4. Experimental study

This section evaluates the performance of our proposed hybrid algorithm for two kinds of objective functions: makespan and Aggregate Objective Function (AOF). For that, we compare our HGA against the available algorithms in the literature including variable neighourbood search (VNS) proposed by Bagheri and Zandieh (2011), an adapted tabu search (TS) proposed by Ennigrou and Ghedira (2008), Artificial Immune System (AIS), Particle swarm Optimization (PSO) from Sadrzadeh (2013) and our GA. Our proposed algorithm has been implemented using JAVA and run on PC with core2Duo, 2,6GHZ and 2GB RAM.

In our experiment, we tested different values for our HGA parameters, and computational experience proves that the following values are more effective for the two problems (i.e. SDST-FJSP with or without learning effects:

- Population size: 150
- Crossover probability: 0.6
- Mutation probability: 0.2
- Local search probability: 0.2
- Number of iteration of HGA (stopping condition): 150 or CPU time limit fixed to n×ni×m×0.1S
- Number of no improvement (stopping condition):20
- Number of iteration of VNS:30.

For the makespan objective function, we consider the same benchmark as in Oddi A. *et al.* (2011) and González, M. *et al.* (2013) which is denoted SDST-HUdata. It consists of 20 instances derived from the first 20 instances of the FJSP benchmark proposed in Hurink *et al.* (1994). Each instance was created by adding to the original instance one setup time matrix St,k for each machine k. The same setup time matrix was added for each machine in all benchmark instances. Each matrix has size n×n, and the value St,i,k. indicates the setup time needed to reconfigure the machine k when switches from job t to job i. These setup times are sequence dependent and they fulfill the triangle inequality. The non-deterministic nature of our algorithm makes it necessary to carry out multiple runs on the same instance in order to obtain meaningful results.

After ten runs of each generated instance by the above-mentioned algorithms, the best solutions obtained for each instance (which is named Solmin) are calculated. We use the relative percentage deviation (RPD) measure to compare the performance of algorithms. RPD is obtained as follows:

RPD = SOLalgo−Solmin/Solmin×100

Where SOLalgo is the makespan of each algorithm.

Table 3 show the performance of the proposed HGA compared with others algorithms. The instance names are listed in the first column, the second column show the size (n×m) of each instance. The third column

Table 3. Summary of results in the SDST-FJSP to minimize the makespan: SDST-HUdata benchmark.

Instance problem	Size n×m	VNS	TS	GA	HGA
La01		2.54	7.36	0.00	**0.00**
La02		1.58	8.53	2.57	**0.33**
La03	10×5	2.66	5.33	**0.13**	0.54
La04		3.50	3.66	4.07	**0.90**
La05		1.11	3.58	**0.16**	0.24
La06		4.45	2.79	2.04	**0.32**
La07		2.92	2.99	1.97	**0.34**
La08	15×5	2.37	2.41	2.86	**0.65**
La09		3.82	3.21	0.41	**0.33**
La10		1.84	4.02	**0.00**	0.62
La11		3.32	2.08	1.61	**0.34**
La12		5.33	2.71	2.38	**0.60**
La13	20×5	3.30	3.14	3.02	**1.02**
La14		1.70	1.76	1.26	**0.93**
La15		2.37	**1.18**	1.97	1.95
La16		2.37	4.04	**0.00**	0.76
La17		4.34	2.39	**0.07**	0.14
La18	10×10	1.86	5.81	0.91	**0.35**
La19		6.37	6.67	**0.11**	0.17
La20		4.84	9.36	2.48	**0.35**
Average		3.12	4.15	1.40	**0.54**

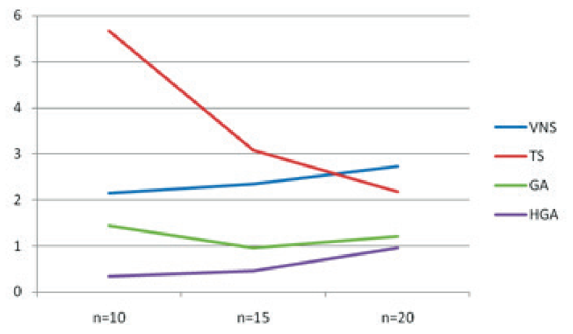

Figure 3. The average RPD of the algorithms versus the number of jobs.

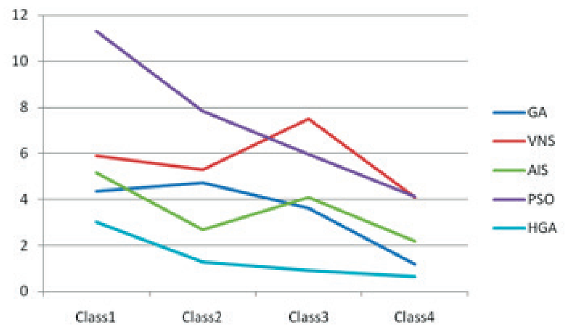

Figure 4. The average RPD of the algorithms of each type of problem class for α=0.25.

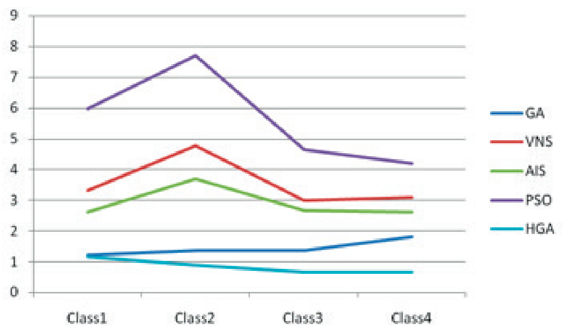

Figure 5. The average RPD of the algorithms of each type of problem class for α=0.5

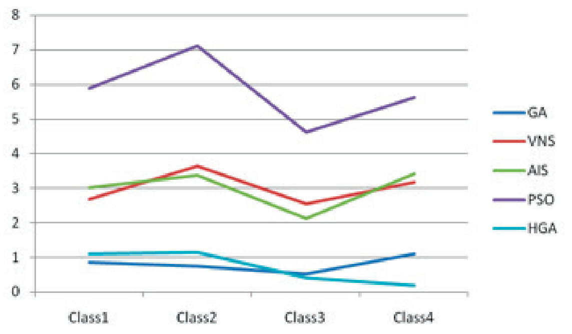

Figure 6. The average RPD of the algorithms of each type of problem class for α=0.75.

represents the flexibility (i.e. the average number of alternative machines for each operation). The third, fourth, fifth and sixth columns report the obtained results of VNS algorithm (Bagheri and Zandieh, 2011), TS algorithm (Ennigrou and Ghedira, 2008), our GA only and HGA.

The obtained results show that the proposed HGA performs better than the others algorithms in 13 instances. Only in instance La03, La05 La10, La16, La17 and La18 GA has gained better results. However, in instance La15, TS obtained the better results. The proposed HGA outperforms the others algorithms with average RPD of 0.54 while the worst performing algorithm is TS with average RPD of 4.15. Moreover, we notice that HGA obtained the best average RPD of 0.35 in the largest number of machine against 4.58, 6.48 and 0.71 for VNS, TS and GA respectively.

To further evaluate the performance of our algorithm, we study the interaction between the performance of the algorithm and the problem size in Figure 3. We remark that our algorithm keeps its robust performance in different problem sizes.

Furthermore, for the AOF, we consider artificial benchmarks according to the function proposed by Bagheri and Zandieh (2011). We propose four classes of instances.These classes are different in number of jobs, n, number ofoperations for each job i, ni, and number of machines, m, that are denoted as (n×ni×m). The generated instances have partial flexibility and the number of available machines for each operation (AMO) is generated randomly according to uniform distribution. Table 4 summarizes the characteristics of the artificial benchmarks used in this paper. In

Table 4. characteristics of the artificial benchmarks

	n×ni×m	AMO	Processing time	SDST	Dummy Jobs
Class1	10×5×5	U(1,5)	U(20,100)	U(20,60)	U(20,40)
Class2	15×5×8	U(1,8)			
Class3	10×10×5	U(1,5)			
Class4	15×10×10	U(1,10)			

order to introduce due dates, we consider the same formula as in (Bagheri & Zandieh, 2011).

Overall, compared to VNS, AIS, PSO and GA, our HGA has a superiority result to minimize the AOF for all α values. Moreover, from the results shown in figures 4, 5 and 6, we remark that HGA is more effective with α =0.25 then α = 0.75. Otherwise, our algorithms have the best results with mean tardiness against makespan objective function.

5. Conclusions

In this paper, we focus on solving the flexible job shop scheduling problem where sequence dependent setup times are also taken into account. We have proposed a hybrid genetic algorithm to minimize two kinds of objective functions: makespan and aggregate objectives function. For that, we tested HGA on two kinds of benchmark. Results showed that the present HGA is better than other algorithms. In future works, it will be interesting to investigate the dynamic scheduling problem to closely reflect the real flexible job shop scheduling environment. For the same reason, we will consider the multi-criteria scheduling problem and the scheduling problems with learning effects considerations.

Acknowledgements

This he authors would like to say thanks to Miguel A. Gonzalez for providing us with the SDST-FJSP instances.

References

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research,* 246(2), 345-378. https://doi.org/10.1016/j.ejor.2015.04.004

Azzouz, A., Ennigrou, M., Jlifi, B. (2015). Diversifying TS using GA in Multi-Agent System for solving Flexible Job Shop Problem. *In Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics*, 94-101. https://doi.org/10.5220/0005511000940101

Azzouz, A., Ennigrou, M., Jlifi, B., Ghedira, K. (2012). Combining Tabu Search and Genetic Algorithm in a Multi-agent System for Solving Flexible Job Shop Problem. *In Artificial Intelligence (MICAI), 2012, 11th Mexican International Conference on, pp. 83-88. IEEE.* https://doi.org/10.1109/micai.2012.12

Bagheri, A., Zandieh, M. (2011). Bi-criteria flexible job-shop scheduling with sequence-dependent setup times Variable neighborhood search approach. *Journal of Manufacturing Systems,* 30(1), 8-15. https://doi.org/10.1016/j.jmsy.2011.02.004

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Journal Annals of Operations Research,* 41(3), 157-183. https://doi.org/10.1007/bf02023073

Cheung, W., Zhou, H. (2001). Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. *Annals of Operations Research,* 107(1), 65-81. https://doi.org/10.1023/A:1014990729837

Fattahi, P., Saidi-Meradbad, M., Jolai, F. (2007). Mathematical Modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing*, 18(3), 331-342. https://doi.org/10.1007/s10845-007-0026-8

González, M. A., Rodriguez Vela, C., Varela, R. (2013). An efficient memetic algorithm for the flexible job shop with setup times. *In Twenty-Third International Conference on Automated*, pp. 91-99.

Hurink, J., Jurisch, B., Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4), 205-215. https://doi.org/10.1007/BF01719451

Imanipour, N. (2006). Modeling solving flexible job shop problem with sequence dependent setup times. *International Conference on Service Systems and Service Management. IEEE*, 2, 1205-1210. https://doi.org/10.1109/icsssm.2006.320680

Kim, S.C., Bobrowski, P.M. (1994). Impact of sequence-dependent setup time on job shop scheduling performance. *The International Journal of Production Research*, 32(7), 1503-1520. https://doi.org/10.1080/00207549408957019

Moghaddas, R., Houshmand, M. (2008). Job-shop scheduling problem with sequence dependent setup times. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2, 978-988.

Mousakhani, M. (2013). Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. *International Journal of Production Research*, 51(12), 3476-3487. https://doi.org/10.1080/00207543.2012.746480

Naderi, B., Zandieh, M., Ghomi, S.F. (2009). Scheduling sequence-dependent setup time job shops with preventive maintenance. *The International Journal of Advanced Manufacturing Technology*, 43, 170-181. https://doi.org/10.1007/s00170-008-1693-0

Najid, N.M., Dauzere-Peres, S., Zaidat, A. (2002). A modified simulated annealing method for flexible job shop scheduling problem. *In proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 5, 6-9. https://doi.org/10.1109/icsmc.2002.1176334

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., Ammari, A. C. (2015). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 1-13. https://doi.org/10.1007/s10845-015-1039-3

Oddi, A., Rasconi, R., Cesta, A., & Smith, S. (2011). Applying iterative flattening search to the job shop scheduling problem with alternative resources and sequence dependent setup times. *In COPLAS 2011 Proceedings of the Workshopon Constraint Satisfaction Techniques for Planning and Scheduling Problems*, pp. 15-22.

Pezzella, F., Morganti, G., Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10), 3202-3212. https://doi.org/10.1016/j.cor.2007.02.014

Rossi, A. (2014). Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *International Journal of Production Economics*, 153, 253-267. https://doi.org/10.1016/j.ijpe.2014.03.006

Sadrzadeh, A. (2013). Development of both the AIS and PSO for solving the flexible job shop scheduling problem. *Arabian Journal for Science and Engineering*, 38(12), 3593-3604. https://doi.org/10.1007/s13369-013-0625-y

Saidi-Mehrabad, M., Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, 32(5), 563-570. https://doi.org/10.1007/s00170-005-0375-4

Vilcot, G., Billaut, J.C. (2011). A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem. *International Journal of Production Research*, 49(23), 6963-6980. https://doi.org/10.1080/00207543.2010.526016

Wang, S.J., Zhou, B.H., Xi, L.F. (2008). A filtered-beam-search-based algorithm for flexible job-shop scheduling problem. *International Journal of Production Research*, 46(11), 3027-3058. https://doi.org/10.1080/00207540600988105

Wang, S., Yu, J. (2010). An effective heuristic for flexible job-shop scheduling problem with maintenance activities. *Computers and Industrial Engineering*, 59(3), 436-447. https://doi.org/10.1016/j.cie.2010.05.016

Yazdani, M., Gholami, M., Zandieh, M., Mousakhani, M. (2009). A Simulated Annealing Algorithm for Flexible Job-Shop Scheduling Problem. *Journal of Applied Sciences*, 9(4), 662-670. https://doi.org/10.3923/jas.2009.662.670

Zambrano Rey, G., Bekrar, A., Prabhu, V., Trentesaux, D. (2014). Coupling a genetic algorithm with the distributed arrival-time control for the JIT dynamic scheduling of flexible job-shops. *International Journal of Production Research*, 52(12), 3688-3709. https://doi.org/10.1080/00207543.2014.881575

Zhang, G., Gao, L., Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications,* 38(4), 3563-3573. https://doi.org/10.1016/j.eswa.2010.08.145

Zhang, G., Shao, X., Li, P., Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering*, 56(4), 1309-1318. https://doi.org/10.1016/j.cie.2008.07.021

Zhou, Y., Beizhi, L., Yang, J. (2006). Study on job shop scheduling with sequence-dependent setup times using biological immune algorithm. *The International Journal of Advanced Manufacturing Technology*, 30(1), 105-111. https://doi.org/10.1007/s00170-005-0022-0

Ziaee, M. (2014). A heuristic algorithm for solving flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 71(1), 519-528 https://doi.org/10.1007/s00170-013-5510-z

Zribi, N., Kacem, I., El Kamel, A., Borne, P. (2007). Assignment and scheduling in flexible job-shops by hierarchical optimization. *In Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE* (4), 652-661. https://doi.org/10.1109/TSMCC.2007.897494