Management and Production Engineering Review
Volume 14 • Number 3 • September 2023 • pp. 148–155
DOI: 10.24425/mper.2023.147196

Optimizing the Job Shop Scheduling Problem with a no Wait Constraint by Using the Jaya Algorithm Approach

Aimade Eddine BOUGLOULA*

University Batna2, Department of Industrial Engineering, Algeria

Received:25 May 2023
Accepted: 10 June 2023

Abstract
This work is interested to optimize the job shop scheduling problem with a no wait constraint. This constraint occurs when two consecutive operations in a job must be processed without any waiting time either on or between machines. The no wait job shop scheduling problem is a combinatorial optimization problem. Therefore, the study presented here is focused on solving this problem by proposing strategy for making Jaya algorithm applicable for handling optimization of this type of problems and to find processing sequence that minimizes the makespan ($C_{\max}$). Several benchmarks are used to analyze the performance of this algorithm compared to the best-known solutions.

Keywords
Scheduling; Optimization; Scheduling problem; Job shop; No-wait problem; Jaya algorithm.

Introduction

Job shop scheduling problem (JSSP) is one of the most popular combinatorial optimization problems. Over the past decades, JSSPs have attracted considerable attention and extensive techniques have been developed to solve static JSSPs (Chaudhry, 2015). It can effectively help manufacturers improve production efficiency and reduce production costs (Dao et al., 2018; Gong & al, 2019). The classical JSSP is such a problem, which deals with the sequencing operation of a set of jobs on a set of machines with the objective to optimize some criterion or criteria. Generally, the main criterion is to search minimum value of makespan (Mahapatra, 2012). Most scheduling problems belong to the class of NP-hard problems. This class of problems is distinguished by rapid growth in the number of potential solutions with modest growth in the number of jobs to be scheduled. The growth is so quick that even the fastest computer could not search through every potential solution to large-scale problems in a reasonable amount of time (Pongchairerks & Kachitvichyanukul, 2009).

In this paper, the author is interested to optimize the job shop scheduling problem with a no wait constraint which occurs when two consecutive operations in a job must be processed without any waiting time either on or between machines. Therefore, the purpose of the study presented here is to solve this problem by proposing strategy for making Jaya algorithm applicable for handling optimization of this type of problems and to find processing sequence that minimizes the makespan ($C_{\max}$).

The remainder of this paper is organized as follows: Section 2 gives review of relevant literature, Section 3 introduces the NWJSSP and gives a description of the problem. Next, Section 4 illustrates the framework of the Jaya algorithm and describes the use of this algorithm on a benchmark instance ft06. Section 5 illustrates the experimental results of application of the Jaya algorithm to the set of la01-la20 benchmarks and makes comparison to the performance results of the best-known solution (BKS) for each instance. Finally, the last Section presents the concluding remarks and future research directions.

Literature review

Up to date, many different mathematical methods and techniques such as mixed integer programming (Gong et al., 2019), disjunctive graph (Gröflin & Klinkert, 2007), priority dispatch rules and neural networks (Weckma, 2008) are developed to optimize

***Corresponding author**: Aimade Eddine Bougloula – University Batna2, Department of Industrial Engineering, 53 Constantine Road, Fesdis, Batna 05078, Algeria, phone: +(213) 697 672 405, e-mail: a.Bougloula@univ-batna2.dz*

JSSPs. Recently (Hadri et al., 2023) are interested in another variant of (JSSP). It is about the job shop scheduling problem with resources availability constraints. To be processed in the system, each job needs a number of consumable resources that are available in a limited quantity. They suggest two different methods to solve this problem. First, they proposed a set of four heuristics based on priority rules. Then, they called genetic algorithm. Using a real job shop manufacturing system data and a large-scale experiment they tested the performance of the proposed methods.

Algorithms based on Nature behaviour have contributed and made great progress in solving optimization problems in the latest years (Mokhtari, 2004; Schuster, 2006). These Meta-heuristics have characteristics such as parallelism, diversity, robustness and good compatibility (He et al., 2021). They have been widely used to trait the JSSPs. These meta-heuristics include the genetic algorithm (GA) (Kurdi, 2016), simulated annealing (SA) (Aydin et al., 2004), Tabu search (TS) (Song et al., 2008), ant colony optimization (ACO) (Chaouch et al., 2017), particle swarm optimization (PSO) (Lin et al., 2010), bee colony optimization (Asadzadeh, 2016).

In many types of industry as steel-making industry (Pinedo, 2016; Tang et al., 2000), concrete manufacturing (Grabowski & Pempera, 2000; Deng et al., 2015), chemical and pharmaceutical industries (Engin & Güçlü, 2018), food industry (Naderi & Zandieh, 2014), etc. A variant of JSSPs that called the No Wait Job Shop Scheduling Problem is met. In this case, a no-wait constraint in scheduling occurs when two consecutive operations of a job must be processed without any interruptions. The main reason for the occurrence of no-wait in process is the technology requirement in this kind of manufacturing environments. In such a manufacturing environment, an operation should immediately follow the previous operation because of the temperature or other characteristics of the material (Allahverdi, 2016). The research interest on this kind of scheduling has increased recently. In (Pranzo & Pacciarelli, 2016) showed that Mascis & Pacciarelli formulate the problem by means of an alternative graph and investigate some effective ideas used to develop a branch and bound algorithms to solve this problem.

Furthermore, (Framinan & Schuster; 2006) proposed a metaheuristic, called complete local search with memory, by integrating a new timetabling procedure for the problem. In addition, (Zhu & Li, 2011) presented a modified complete local search memory algorithm where an efficient Shift Penalty-Based Timetabling method is proposed. Moreover, (Bansal et al., 2005) used approximation algorithms that run in polynomial time. (Vermeulen, 2011) develop an exact solution method using ILP (Integer Linear Programming) and a combination of binary search with CP (Constraint Programming) to find an optimal makespan. Deng et al. (2019) formulate the no-wait job shop problem with a total flow time criterion based on time difference and decomposes the problem into timetabling and sequencing sub-problems. They proposed by adopting favorable features of the iterated greedy algorithm, the population-based iterated greedy (PBIG) algorithm for the sequencing sub-problem. Metaheuristics have been widely used by researchers as approaches to solve the NWJSP. Mokhtari (2014) proposed a new optimization method that is based on a combination of an enhanced variable neighbourhood search and an artificial neural network. Aitzai (2016) proposed an exact method based on the branch-and-bound algorithm by defining a new technique of branching and a swarm optimization (PSO) algorithm with an efficient approach to move a particle to the new position. On the other hand, many researchers proposed evolutionary algorithms to solve this problem. As Bozejko & Makuchowski (2011) used for NWJSSP a methodology of automatic genetic algorithm parameters adjustment with a makespan criterion. Sachchida et al. (2018) developed an evolutionary algorithm with guided mutation (EA/G) based hyper-heuristic which employs an evolutionary algorithm to explore the search space and a generic guided mutation, multi-insert points and multi-swap. Bürgy & Gröflin (2012) proposed a highly efficient algorithm based on a compact formulation of the NWJS problem and a characterization of all feasible insertions as the stable sets in a derived comparability graph. Sundar et al. (2017) used a hybrid artificial bee colony (ABC) algorithm with determination of a neighbouring solution with the local search.

Rao (2016) introduced a simple yet powerful optimization algorithm called Jaya algorithm, for solving the constrained and unconstrained optimization problems. This algorithm is based on the concept that the solution obtained for a given problem should move towards the best solution and should avoid the worst solution. It is showed that it has a strong potential to solve the constrained and unconstrained optimization problems. Gao et al. (2016) used this algorithm to solve the Flexible Job Shop Problem with new job insertion and a discrete version of Jaya is proposed. More further Chaithanya et al. (2017) in their work, a Jaya algorithm is proposed for simultaneous scheduling of jobs and tools neglecting tool transfer times between machines it makespan as an objective. Li et al. (2020) studied a Flexible Job Shop Schedul-

ing considered simultaneously constraints including setup time and transportation time. Moreover, the energy consumptions during the machine processing and staying at the idle time was taken into account for green production.

Until writing this paper, to the best of the author knowledge, there is no researcher interested by using Jaya algorithm to trait the no wait job shop scheduling problem. Based on the efficiency of Jaya algorithm, the complexity of the NWJSSP that is a non-deterministic polynomial time NP-hard problem to a considerable degree. This paper is interested to take advantages of this algorithm and optimize the makespan on a No Wait Job Shop Scheduling Problem.

Problem description

The no wait job shop scheduling problem can be described as follows: n jobs need to be processed on m machines in the workshop. Consider a set of n jobs $J = [J_1, \ldots, J_n]$ and a set of m machines $M = [M_1, \ldots, M_m]$. Each job follows a different machine sequence formed by a subset of M. When a job ends its processing on a machine, it is not allowed to experience any delay and immediately it has to start its processing on the next machine (no-wait constraints). Any job cannot be processed on more than one machine simultaneously. Each machine can only process one operation at a time. Preemption not allowed once a job starts to be processed on a machine. Setup time is not considered in this paper. In the three-field notation, the NWJSS problem with makespan as the objective is denoted by $J/\text{no-wait}/C_{\max}$. The problem is to determine the schedule of the jobs J_i on the machines M_j that minimizes the maximum completion time, i.e. the makespan $C_{\max}$ as in (Pranzo & Pacciarelli, 2016).

$$\min(C_{\max}) \tag{1}$$

The main constraints in this problem are the precedence constraint and the no-wait constraint between the operations of a same job. They can be represented as follow.

$$t_{ij} + p_{ij} \leq t_{ij+1} \quad \forall\, i = 1, \ldots, n \;\text{ and }\; \forall\, j = 1, \ldots, m \tag{2}$$

$$t_{ij+1} - p_{ij} \leq t_{ij} \quad \forall\, i = 1, \ldots, n \;\text{ and }\; \forall\, j = 1, \ldots, m \tag{3}$$

As explained in (Schuster, 2006), to solve the no-wait job shop problem it can be decomposed into two sub problems that can informally be described as follows:

- Sequencing problem: given an instance of the no-wait job shop problem, find a processing sequence of an optimal schedule.
- Timetabling problem: given a processing sequence from step 1, find a feasible set of starting times for the jobs that realizes the minimum makespan within the given processing sequence.

Jaya algorithm

Jaya means victory in Sanskrit. The algorithm always tries to get victorious by reaching the best solution and moving away from the worst solution. Hence, it is named Jaya. Figure 1 shows the flowchart of standard Jaya algorithm (Rao, 2016).

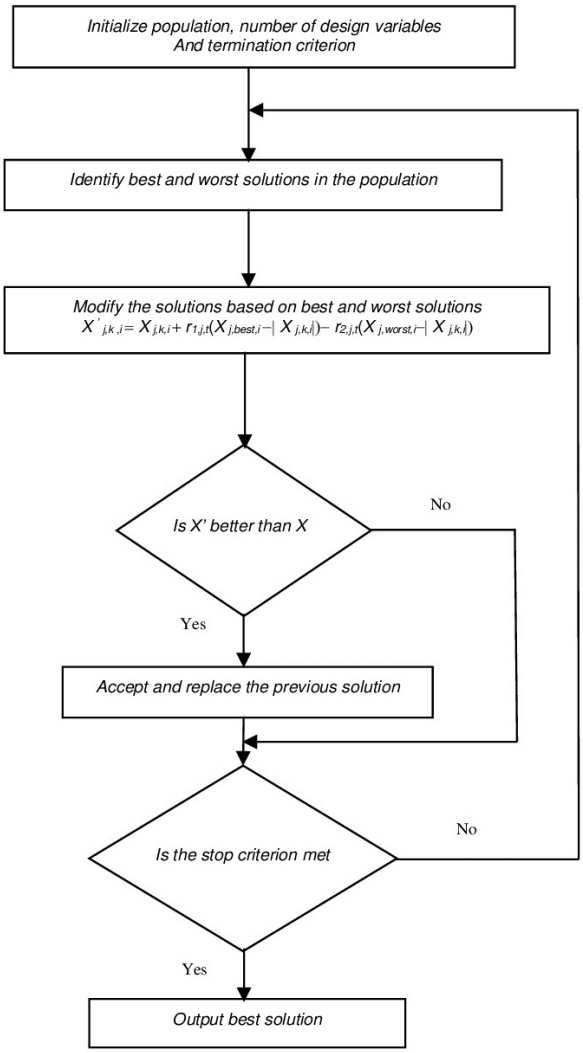


Fig. 1. Flowchart of standard Jaya algorithm

In Jaya algorithm, the first step is to set population size, the termination criterion, initialize the population and design variables. At any iteration, the best and worst solutions are identified from population. If $X_{j,k,i}$ is the value of the j-th variable for the k-th candidate during the i-th iteration, a new candidate solution $X'_{j,k,i}$ can be generated by the following equation:

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,t}(X_{j,\text{best},i} - |X_{j,k,i}|) - r_{2,j,t}(X_{j,\text{worst},i} - |X_{j,k,i}|) \quad (4)$$

$X'_{j,k,i}$ is the updated value of $X_{j,k,i}$, r_1 and r_2 are two uniform random numbers in range $[0, 1]$. If the objective value of $X'_{j,k,i}$ is better than that of $X_{j,k,i}$, the $X'_{j,k,i}$ is accepted and replaces $X_{j,k,i}$. All the accepted values at the end of each iteration are maintained and these values become the input for the next iteration.

The Jaya algorithm initially has been proposed for solving continuous optimization problems (real values). However, the NWJSSP is a combinatorial optimization problem. Therefore, a proposed strategy for making Jaya algorithm applicable for handling discreet optimization problems is described as follow.

To initialize a group of solutions, this study utilizes a simple and efficient way. A scheduling vector is defined as a permutation of jobs. The sequence of jobs makes sure that jobs will be processed in a given order as in Ozolinz (2018). For example, if we have $n = 4$ jobs, the elements of the scheduling vector or the processing sequence will be $\Pi = (4, 2, 1, 3)$. That means the system will start processing from job 4 then job 2 and so on and then we have only to find a feasible set of starting times (or completion times) for the jobs. To use Jaya algorithm we associate to every element of the scheduling vector a real number in the range $[0, 1]$. Those real numbers constitute a real vector $\boldsymbol{X}$ that will be sorted in descending order and the scheduling vector will be rearranged as the real numbers associated to them. We use this encoding scheme to obtain discrete individuals for the Jaya algorithm. The population initialization mechanism is presented below:

Step 1: N individuals are generated randomly first to construct an initialization population. For each individual X in the population, each dimension variable of X is a uniform random number in the range $[0, 1]$ and the numbers of variables of X present the jobs to be scheduled.

Step 2: All the variables of X are rearranged in descending order. By analogy, the positions of variables of X are rearranged in the same way. Then we obtain an individual scheduling vector.

Following this, a population consisting of N encoded feasible individuals scheduling vector is always obtained.

To illustrate this encoding mechanism, a benchmark instance **ft06** is used as an example. Basic parameters of **ft06** are depicted in Table 1. Each cell contains a pair (p_{ij}, l) where p_{ij} is the processing time of job J_i on machine M_j while l means that the l-th operation of job J_i is processed on machine M_j.

Table 1
Basic parameters of ft06

Jobs	Machines					
	M1	M2	M3	M4	M5	M6
J_1	(1, 3)	(3, 1)	(6, 2)	(7, 4)	(3, 6)	(6, 5)
J_2	(8, 2)	(5, 3)	(10,5)	(10, 6)	(10, 1)	(4, 4)
J_3	(5, 3)	(4, 4)	(8, 6)	(9, 1)	(1, 2)	(7, 5)
J_4	(5, 2)	(5, 1)	(5, 3)	(3, 4)	(8, 5)	(9, 6)
J_5	(9, 3)	(3, 2)	(5, 5)	(4, 6)	(3, 1)	(1, 4)
J_6	(3, 2)	(3, 4)	(9, 6)	(10, 1)	(4, 5)	(1, 3)

Figure 2 depicts the encoding steps for the six-jobs and six-machines **ft06** JSP example. Through the encoding steps, an encoded individual will be generated. According to parameters on Table 1, the encoded individual will be decoded to obtain an active scheduling scheme, as shown in Fig. 3.

Step 1: Initial individual.

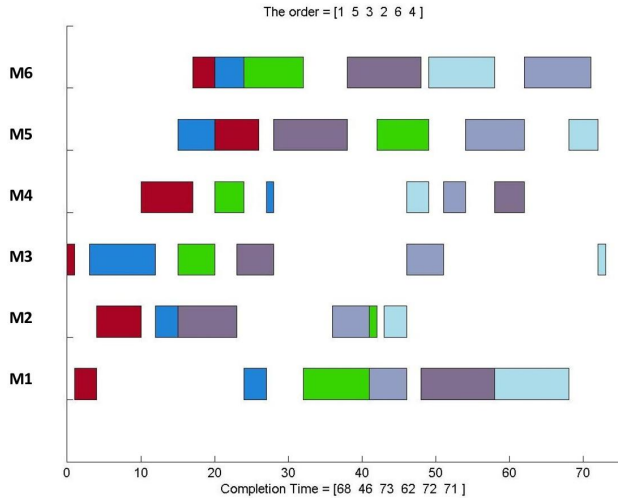
X	0,12	0,85	0,72	0,94	0,50	0,23
Jobs	1	2	3	4	5	6

Step 2: Encoded individual.

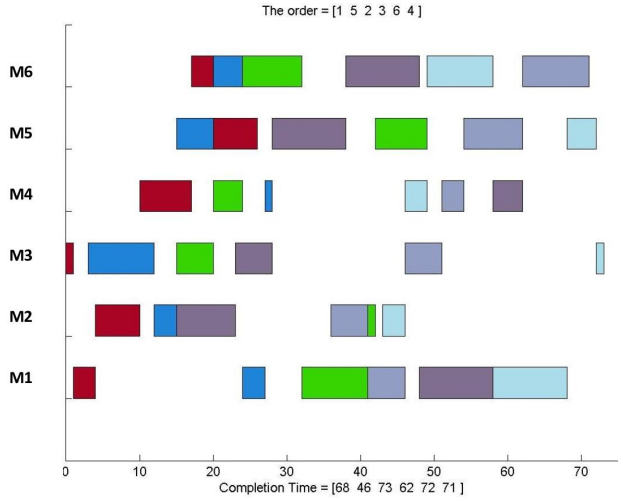
X	0,12	0,23	0,50	0,72	0,85	0,94
Jobs	1	6	5	3	2	4

Fig. 2. Encoding steps

The solution found by applying the Jaya algorithm and illustrated by Fig. 3 shows tow processing sequences that give the same value of the optimal scheduling. Figure 3a gives the first processing sequence $\Pi = (1, 5, 3, 2, 6, 4)$ with timetable (completion times) $T_\Pi = (46, 62, 68, 71, 72, 73)$ and Fig. 3b gives the second processing sequence $\Pi = (1, 5, 2, 3, 6, 4)$ with $T_\Pi = (46, 62, 68, 71, 72, 73)$.



(a)



(b)

Fig. 3. Processing sequence and timetable of the benchmark instance ft06

As it can be seen, the position of Job 2 does not make difference if it is processed before the Job 3.

Experimentation, results and interpretation

This section is dedicated to the synthesis of the results obtained by the application of the Jaya algorithm to the set of (Lawrence, 1984) benchmarks of no wait job shop problem. Note that the objective is to minimize the completion time (Makespan) of all the jobs and, for each case, the experimentations are performed in the same technical circumstances.

Experimental parameters

The algorithm was coded in MATLAB R2013a. Based on the existing literature and our experimental observation, the population size was set as 150 and the maximum number of generations was set as 100. Each instance was independently run 20 times on the computer.

Results and interpretation

The results of application of the Jaya algorithm to the set of la01–la20 benchmarks of no wait job shop problem are presented in Table 2, in six columns. The first (instances) represents the name of instance. The second (Size) represents the size of instance (number of jobs x number of machines). The third column (BKS) stands for the best-known solution for each instance (Ozolins, 2018). The fourth column (PSO)

Table 2
Obtained results for Jaya algorithm

Instance	Size	BKS	PSO	JAYA	PRD
la01	10 × 5	971	1115	975	0.41
la02	10 × 5	937	1069	963	2.77
la03	10 × 5	820	1003	820	**0.00**
la04	10 × 5	887	1012	887	**0.00**
la05	10 × 5	777	1000	781	0.51
la06	15 × 5	1248	1536	1348	8.01
la07	15 × 5	1172	2240	1244	6.14
la08	15 × 5	1244	1672	1336	7.40
la09	15 × 5	1358	1680	1403	3.31
La10	15 × 5	1287	1534	1357	5.44
la11	20 × 5	1671	1786	1848	10.59
la12	20 × 5	1452	1808	1630	12.26
la13	20 × 5	1624	1915	1790	10.22
la14	20 × 5	1691	1897	1823	7.81
la15	20 × 5	1694	2066	1910	12.75
la16	10 × 10	1575	1790	1575	**0.00**
la17	10 × 10	1371	1492	1384	0.95
la18	10 × 10	1417	1684	1417	**0.00**
la19	10 × 10	1482	1750	1482	**0.00**
La20	10 × 10	1526	1884	1526	**0.00**

stands for the solutions given by the particle swarm optimization (Aitzai et al., 2014).

The fifth gives the $C_{\max}$ value returned by the Jaya algorithm. The sixth column gives the average percentage relative deviation (PRD). The PRD is calculated by the formula as follow:

$$\text{PRD} = \frac{C_{\text{Alg}} - C_{\text{Ref}}}{C_{\text{Ref}}} \times 100 \tag{5}$$

where C_{Alg} is the objective value found by Jaya Algorithm, and C_{Ref} is the (BKS) best-known solution found for the instance.

Considering those results obtained, it can be notice that the non-optimality of this algorithm has failed in some cases to find the optimal sequence like the best-known solutions., especially for the 20×5 type benchmark instances. May be it needs local search solution procedures. Despite this, it remains better than PSO proposed by (Aitzai et al., 2014). In other cases, the optimal sequence found by Jaya algorithm is so close to the best-known solution or even identical to them, specifically for the 10×10 type benchmark instances.

However, the comparison of the results obtained by Jaya algorithm with those obtained by PSO shows a notable difference with an average of more than 17%. Moreover, it shows that Jaya algorithm is best.

In addition, in the last column, it can be seen the mean of PRD of all instances is about 4.4% and 70% of instances have less than 10% percentage relative deviation from the best-known solutions.

The objective of the different experimentations done over a set of test data is to explore the performance of this algorithm and to validate the implementation of this method which is adapted to the resolution of job shop problems with no-wait constraint.

Conclusions

This paper presented the job shop scheduling problem with no-wait constraint, which is very important in a lot of applications in industry. Mainly in food industry and pharmaceutical industry, where human health is threatened. An adapted Jaya algorithm is proposed for the resolution of this problem. Good results have been obtained compared to those in literature, especially for the 10×10 type benchmark instances, which promises to get more practical computational results, in the future, for the modified Jaya algorithm using local search operator.

Generally, compared with most of the existing methods to solve the no-wait job shop scheduling problem, the Jaya Algorithm has the advantages of few parameters, simple structure, easy implementation and relatively an excellent performance. In this way, future work will mainly focus on developing local search operators to perform the exploitation tasks and on testing an enough size of the samples to establish a formal analysis about this approach.

References

AitZai, A., Benmedjdoub, B., & Boudhar, M. (2014). Branch-and-bound and PSO algorithms for no-wait job shop scheduling. *Journal of Intelligent Manufacturing*, 27, 679–688

Allahverdi, A. (2016). A survey of scheduling problems with no wait in process. *European Journal of Operational Research*, 255, 665–686.

Asadzadeh, L. (2016). A parallel artificial bee colony algorithm for the job shop-scheduling problem with a dynamic migration strategy. *Computer & Industrial Engineering*, 102, 359–367.

Aydin, M.E., & Fogarty, T.C. (2004). A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems. *Journal of Heuristics*, 10, 269–292.

Bansal, N., Mahdian, M., & Sviridenko, M. (2005). Minimizing makespan in no-wait job shops. *Mathematics of Operations Research*, 30(4), 817–831.

Bozejko, W., & Makuchowski, M. (2011). Solving the no-wait job-shop problem by using genetic algorithm with automatic adjustment. *International Journal of Advanced Manufacturing Technology*, 57, 735–752.

Bürgy, R., & Gröflin, H. (2012). Optimal job insertion in the no-wait job shop. *Journal of Combinatorial Optimization*, 26(2), 345–371.

Chaithanya, M., Reddy, N.S.R., & Reddy, P.R. (2017). Sequencing and Scheduling of Jobs and Tools in a Flexible Manufacturing System using Jaya Algorithm. *International Journal for Research in Applied Science & Engineering Technology* (IJRASET), 5(11).

Chaouch, O.B., Driss, K., & Ghedira, A. (2017). Modified ant colony optimization algorithm for the distributed job shop scheduling problem. *Procedia Computer. Science*, 112, 296–305.

Chaudhry, I.A., & Khan, A.A. (2015). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3), 551–591.

Dao, T.K., Pan, T.S., Nguyen, T.-T., & Pan, J.S. (2018). Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *Journal of Intelligent Manufacturing*, 29, 245–462.

Deng, G., Zhang, Z., Jiang, T., & Zhang, S. (2019). Total flow time minimization in no-wait job shop using a hybrid discrete group search optimizer. *Applied Soft Computing*, 81.

Deng, G., Wei, M., Su, Q., & Zhao, M. (2015). An effective co-evolutionary quantum genetic algorithm for the no-wait flow shop scheduling problem. *Advances in Mechanical Engineering*, 7(12), 1–10.

Engin, O., & Güçlü, A., (2018). A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Applied Soft Computing*, 72, 166–176.

Framinan, J., & Schuster, C. (2006). An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. *Computer & Operations. Research*, 33(5), 1200–1213.

Gao, K., Sadollah, A., Zhang, Y., Su, R., & Li, K.G.J. (2016). *Discrete Jaya algorithm for flexible job shop scheduling problem with new job insertion.* 14th international conference on control, automation, robotics and vision (ICARCV), 1–5.

Gong, G.L., Deng, Q.W., Chiong, R., Gong, X., Huang, H. (2019). An effective memetic algorithm for multi-objective job-shop scheduling. Knowl-Based System, 182.

Grabowski, J., & Pempera, J. (2000). Sequencing of jobs in some production system. *European Journal. Operation Research*, 125, 535–550.

Gröflin, H., & Klinkert, A. (2007). Feasible insertions in job shop scheduling, short cycles and stable sets. *European Journal of Operational Research*, 177(2), 763–785.

Hadri, A., Bougloula, A.E., Belkaid, F., & Smadi, H. (2023). An efficient approach for solving a job shop scheduling problem with resources constraints: a case study iCIM 3000'. *Int. J. Operational Research*, 46(1), 73–92.

He, L., Li, W., Chiong, R., Abedi, M., Cao, Y., & Zhang, Y. (2021). Optimising the job-shop scheduling problem using a multi-objective Jaya algorithm. *Applied Soft Computing*, 111.

Kurdi, M. (2016). An effective new island model genetic algorithm for job shop scheduling problem. *Computer Operation Research*, 67, 132–142.

Lawrence, S. (1984). *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Li, J.; Deng, J., Li, C., Han, Y., Tian, J., Zhang, B., & Wang, C. (2020). An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. *Knowledge-Based Systems*, 200, 2020.

Lin, T.L., Horng, S.J., & Kao, T.W. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37(3), 2629–2636.

Mahapatra, D. (2012). *Job Shop Scheduling Using Artificial Immune System.* BSc, National Institute of Technology, Rourkela, India.

Mokhtari, H. (2014). A two-stage no-wait job shop scheduling problem by using a neuro-evolutionary variable neighborhood search. *The International Journal of Advanced Manufacturing Technology*, 74, 1595–1610.

Naderi, B., & Zandieh, M., (2014). Modeling and scheduling no-wait open shop problems. *International Journal of Production Economics*, 158, 256–266.

Ozolinz, A. (2016). A new exact algorithm for no wait job shop problem to minimize makespan. *Operational Research*, 20(4) 2333–2363

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems.* 5th ed. Berlin Heidelberg.

Pongchairerks, P., & Kachitvichyanukul, V. (2009). A particle swarm optimization algorithm on job-shop scheduling problems with multi-purpose machines. *Asia Pacific Journal of Operational Research*, 26(02), 161–184

Pranzo, M., & Pacciarelli, D. (2016). An iterated greedy metaheuristic for the blocking job shop scheduling problem. *Journal of Heuristics*, 22, 587–611.

Rao, R. (2016). Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 7(1), 19–34.

Sachchida, N. C., Shyam, S., Donghwi, J., & Ho, M.L. (2018). *An Evolutionary Algorithm Based Hyper-heuristic for the Job-Shop Scheduling Problem with No-Wait Constraint: Theory and Applications.* ICHSA, 249–257.

Schuster, C.J. (2006). No-wait Job Shop Scheduling: Tabu Search and Complexity of Subproblems. *Mathematical Methods of Operations Research*, 63(3), 473–491

Song, X.Y., Meng, Q.H., & Yang, C. (2008). Improved taboo search algorithm for job shop scheduling problems. *Syst. Eng. Electron*, 30, 93–96.

Sundar, S., Suganthan, P.N., Jin, C.T., Xiang, C.T., & Soon, C.C. (2000). A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. *Soft Computer*, 21(5), 1193–1202.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operation Research*, 120, 423–435.

Vermeulen, H., Hoogeveen, H., & VandenAkker, J.M. (2011). *Solving the no-wait job shop problem: An ILP and CP approach*. The 8th International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) techniques in Constraint Programming, 2011, Berlin, Germany.

Weckma, G.R., Ganduri, C.V., & Koonce, D. (2008). A neural network job-shop scheduler. *Journal of Intelligent Manufacturing*, 19(2), 191–201.

Zhu, J., & Li, X., (2011). An Effective Meta-Heuristic for No-Wait Job Shops to Minimize Makespan. *IEEE Transactions on Automation Science and Engineering*, 9(1), 189–198.