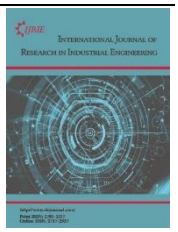
E-ISSN: 2717-2937 | P-ISSN: 2783-1337

International Journal of Research in Industrial Engineering

www.riejournal.com

Int. J. Res. Ind. Eng. Vol. 12, No. 4 (2023) 431–449.



Paper Type: Research Paper

A Vibration Damping Optimization Algorithm to Solve Flexible Job Shop Scheduling Problems with Reverse Flows

Esmaeil Mehdizadeh[1,*], Fatemeh Soleimaninia[1]

[1] Department of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran; emehdi@qiau.ac.ir; fatemesoleimaninia@gmail.com.

Citation:

Mehdizadeh, E., & Soleimaninia, F. (2023). A vibration damping optimization algorithm to solve flexible job shop scheduling problems with reverse flows. *International journal of research in industrial engineering, 12*(4), 431-449.

Received: 18/01/2023 Reviewed: 20/02/2023 Revised: 19/03/2023 Accepted: 08/05/2023

Abstract

The Flexible Job shop Scheduling Problem (FJSP), as a Production Scheduling Problem (PSP), is generally an extension of the Job shop Scheduling Problem (JSP). In this paper, the FJSP with reverse flow consisting of two flows of jobs (direct and reverse) at each stage is studied; the first flow initiates in Stage 1 and goes to Stage C (the last stage), and the second flow starts with Stage c and ends up in Stage 1. The aim is to minimize the makespan of the jobs (the maximum completion time). A Mixed Integer Programming (MIP) is presented to model the problem and the Branch and Bound (B&B) method is used to solve the problem. A numerical small-size problem is presented to demonstrate the applicability, for which the Lingo16 software is employed for a solution. Due to the NP-hardness of the problem, a meta-heuristic, namely the Vibration Damping Optimization (VDO) algorithm with tuned parameters using the Taguchi method, is utilized to solve large-scale problems. To validate the results obtained using the proposed solution algorithm in terms of the solution quality and the required computational time, they are compared with those obtained by the Lingo 16 software for small-size problems. Finally, the performance of the proposed algorithm is compared with a Genetic Algorithm (GA) by solving some randomly generated larger-size test problems, based on which the results are analyzed statistically. Computational results confirm the efficiency and effectiveness of the proposed algorithm and show that the VDO algorithm performs well.

Keywords: Vibration damping optimization, Scheduling, Flexible job shop, Reverse flow, Mathematical programming, Genetic algorithm.

1 | Introduction

Production scheduling is an important challenge in the field of production and operation planning and attracted the attention of many researchers. It can be defined as the determination of a sequence to assign tasks that utilize resources to produce products [1]. Being classified as an NP-hard problem, Job shop Scheduling (JSP) is the most known and complex scheduling problem [2]. Besides, the Flexible Job shop Scheduling Problem (FJSP) is one of the extensions of the JSP with more complication and Mati and Xia [3] proved that it belongs to the class of NP-hard problems as well.

One of the most important issues in the field of FJSP is the so-called reverse flows involved within an assembly/disassembly network, in which non-conforming and used assembled products are returned by the customers. To have a sustainable production line, these products are then disassembled to use their good components in recovery activities. Over the past few years, most of the research on reverse logistics dealt with the return of products in a supply chain [4].

Corresponding Author: emehdi@qiau.ac.ir

https://doi.org/10.22105/riej.2023.383451.1363

The first crucial step involved in product recovery operation is disassembly. Disassembly scheduling, as one of the important operational problems, can be generally defined as the problem of determining the quantity and the timing of the end-of-use/life products while satisfying the demand for their parts over a planning horizon [5]. Some disassembly operations are almost always needed in remanufacturing, recycling, and disposal. Several questions arise which increase the uncertainty in disassembly yield: 1) is the component missing? 2) is the component functional? 3) What is the version of the component? If these questions can be answered before the disassembly of the component, unnecessary disassembly of a non-functional or unneeded component can be avoided [6]. In disassembly, the flow process is divergent and there is a high degree of uncertainty in the quality of the returned products. The reusability of parts creates a demand and availability of the reusable parts is significantly less predictable than what is found in the assembly process [7].

As an operational (short-term) decision, the current study seeks to determine proper scheduling for the FJSP of an assembly/disassembly network that involves reverse flows. In what comes in the next section, some relevant works in this area are surveyed in chronological order.

2 | Literature Review

In this section, a few relevant and recent researches on the JSP are first reviewed. Then, some studies on the reverse flow and disassembly problems are surveyed.

Gao et al. [8] addressed the FJSP with two constraints, namely fuzzy processing time and new job insertion. A Two-stage Artificial Bee Colony (TABC) algorithm with several improvements was proposed to solve FJSP with fuzzy processing time and new job insertion constraints.

A JSP was considered in Giglio et al. [9], where the authors proposed a design to integrate an energy-efficient JSP and lot sizing for manufactured raw materials and remanufactured return products. Stating that their Mixed-Integer Linear Programming (MILP) formulation was NP-Hard, they proposed a relax-and-fix heuristic to solve their problem. Wu and Sun [10] formulated a FJSP, in which turn-on/off machines and speed levels were taken into account to save energy. They solved their problem using a Non-dominated Sorted Genetic Algorithm (NSGA-II).

A Flexible Job-shop Rescheduling Problem (FJRP) for new job insertion was addressed by Gao et al. [11]. Their paper deal with bi-objective FJRPs to minimize: 1) instability and 2) one of the following indices: a) makespan; b) total flow time; c) machine workload; and d) total machine workload. A metaheuristic algorithm, named DJaya presented to solve FJRP.

Gong et al. [12] introduced a Double Flexible Job shop Scheduling Problem (DFJSP) model that not only takes into account the processing time but also considers green production measures including environmental protection and human factor indicators. They solved their problem by utilizing a Newly proposed Hybrid Genetic Algorithm (NHGA).

A FJSP to minimize the total energy consumption of the shop was investigated by Meng et al. [13]. The authors solved their proposed models using the CPLEX SOLVER, based on which the effectiveness of their approach was assessed. They showed that their MILP models outperform the existing model that minimizes idle time. Gong et al. [14] developed a many-objective integrated energy- and labor-aware FJS model. The objectives involved in their model included makespan, total labor cost, total energy cost, total workload, and maximal workload. They used an NSGA-III meta-heuristic to solve their problem.

On the subjects of reverse flow and disassembly, we refer interested readers to Dondo and Mendez [15], Osmani and Zhang [16], and Giri et al. [17] for the latest research in reverse flows conducted in supply chains. Here some works related to heuristic and meta-heuristic approaches to solve the reverse flow problem in assembly and disassembly environments are discussed. McGovern and Gupta [7] developed

a Genetic Algorithm (GA) for disassembly line balancing problems. Adenso-Diaz et al. [18] presented a path-relinking-based heuristic that seeks the optimal disassembly sequence plan. Duta et al. [19] provided a GA for finding the disassembly sequence with the best financial income. Kim et al. [20] provided a Branch and Bound (B&B) algorithm to meet the demand for end-of-use product components by determining the quality and time of this component's disassembly. Ilgin and Gupta [6] studied the sensors implanted in products on the performance of a washing machine disassembly line. Wan & Gonnuru [21] proposed a GA that provides a disassembly sequence to maximize the benefits by taking into account the recovery value and the disassembly cost. Abdeljaouad et al. [22] proposed a hybrid heuristic algorithm for job-shop scheduling problems with reverse flows for disassembly. Tanimizu et al. [23] proposed a scheduling method for open-shop scheduling problems containing both disassembly and post-processing operations. Finally, Ren et al. [24] proposed a model for the asynchronous Parallel Disassembly Planning (i.e. aPDP) problem. They adopted an efficient meta-heuristic based on GA to identify the disassembly sequence and manipulator allocation. Aghighi et al. [25] investigated open shop scheduling with the reverse flow. A MILP model was developed to solve the problem. The proposed model includes an objective to minimize the maximum completion time of all jobs (makespan). A numerical example is presented and solved by using the GAMS software to validate the proposed mathematical model. Seven meta-heuristic algorithms (VDO, SA, ICA, BAT, HSA, ACO, and CS) were used to solve larger-size examples.

Given the above-mentioned literature, in this paper, a FJSP with reverse flow is studied. A Mixed Integer Programming (MIP) model is presented and the B&B method is used to solve small-size problems. Due to the NP-hardness of the problem, a meta-heuristic algorithm, namely a Vibration Damping Optimization (VDO) algorithm is used to solve large-scale problems. The Taguchi method is used to calibrate the parameters of the algorithm, in an attempt to find better solutions. Finally, the proposed algorithm is compared with the GA by solving some randomly generated test problems.

The remainder of the paper is organized as follows; in Section 3 the problem is defined, the assumptions are made, the notations are defined, and the problem is formulated. The proposed solution approach comes in Section 4. Some problems are solved in Section 5 to validate the proposed algorithm as well as to assess its performance when compared to the one of a GA. Finally, the conclusion and future research recommendations are in Section 6.

3 | The Problem and its Model

In this section, the problem is defined explicitly. Then, notations are defined, based on which the mathematical formulation of the problem is developed.

3.1 | Problem Definition

In this paper, there are n jobs that must be scheduled at c stages on m_k machines in the kth workshop. There are two operating ranges: the direct jobs cover the stages in the order $1, 2, \dots, c$, and the reverse jobs cover the stage in the order $c, c-1, \dots, 2, 1$. The set of E_1 shows the set of direct jobs and E_2 shows the set of reverse jobs. *Fig. 1* shows the direct and reverse flows involved in this job shop.

Note that for any flexible job-shop scheduling problem with reverse flows, where there are two flows of jobs at each stage in opposite directions, optimal solutions are using the following policies [22]:

- *At the first stage, all of the direct jobs should be processed before the reverse jobs (on any machine).*
- *At the last stage, all of the reverse jobs should be processed before the direct jobs (on any machine).*

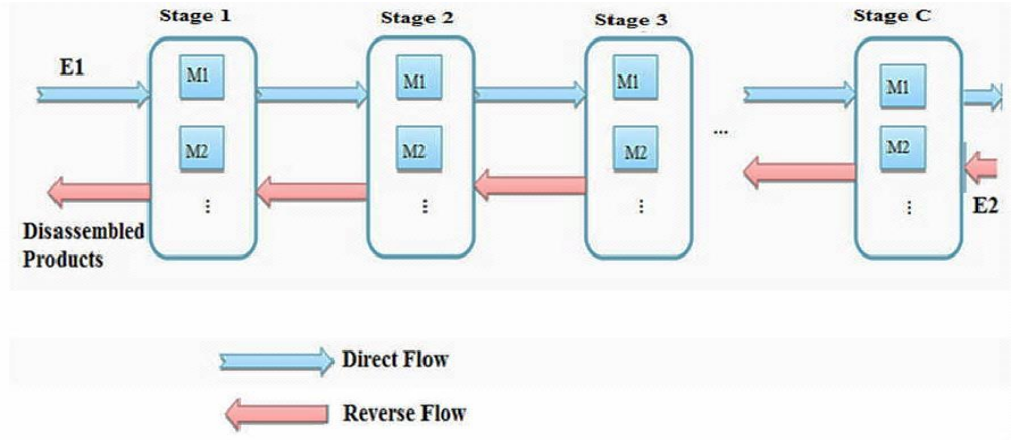


Fig. 1. An overview of the flows.

3.2 | Assumptions

There are two types of jobs (direct and reverse).

Direct and reverse flows enter and exit each stage.

All jobs are available at zero time and are independent.

There are m_k jobs in stage $k; k = 1,2, \dots, c$.

All machines are independent of each other.

All machines are continuously available.

At a time, each machine can process at most one job, and each job can be processed only on one machine.

Each machine can process only one operation.

The machines in each stage are different from each other.

The machines of a direct and a reverse path are the same.

Preemption is not allowed. In other words, the processing of a given job on a machine cannot be interrupted once started.

The processing time can be different in various machines.

The machine setup time is ignored.

3.3 | Notations

The indices, the sets, the parameters, and the decision variables defined to formulate the problem are as follows.

3.3.1 | Indices

j: An index used for a job, $j = 1,2, \dots, n$.

k: An index used for a stage, $k = 1,2, \dots, c$.

s: An index used for a machine in stage k, $s = 1,2, \dots, m_k$.

i: The position index, $i = 1,2, \dots, n$.

h: The previous position index, $h = 1,2, \dots, i$.

S_1: Number of jobs in the direct flow.

$n - S_1$: Number of jobs in the reverse flow.

3.3.2 | The sets

$j = \{j_1, j_2, \dots, j_n\}$: The set of all jobs.

$E_1 = \{j_1, \dots, j_{S_1}\}$: The set of direct jobs.

$E_2 = \{j_{S_1+1}, \dots, j_n\}$: The set of reverse jobs.

3.3.3 | The parameters

n: The number of all jobs.

m: The number of all machines.

m_k: The number of jobs in stage k.

n: The number of positions.

O_{kjs}: The operation of job j on machine s at stage k.

P_{kjs}: The time required to process job j on machine s at stage k.

M: A large positive number.

3.3.4 | Decision variables

t_{kj} : The starting time of job j in stage k.

P'_{kj}: The real processing time (real duration) of job $j$at stage k.

$$X_{jki}^{S} = \begin{cases} 1, & if\ job\ j\ is\ assigned\ to\ position\ i\ at\ stage\ k, \\ 0, & Otherwise. \end{cases}$$

3.4 | Mathematical Modeling

The proposed mathematical model of the problem follows:

$$\text{Min } Z = C_{max}, \tag{1}$$

s.t.

$$\sum_{s=1}^{m_k}\sum_{i=1}^{n} X_{jki}^{S} = 1 \quad \text{for all } j = 1,\dots,n \;;\; k = 1,\dots,c, \tag{2}$$

$$\sum_{j=1}^{n} X_{jki}^{S} \le 1 \quad \text{for all } i = 1,\dots,n \;;\; k = 1,\dots,c \;;\; S\in m_k, \tag{3}$$

$$t_{1j} \le M\left(1 - \sum_{s=1}^{m_1} X_{j11}^{S}\right) \quad \text{for all } j = 1,\dots,S_1, \tag{4}$$

$$t_{cj} \le M\left(1 - \sum_{s=1}^{m_c} X_{jc1}^{S}\right) \quad \text{for all } j = S_1+1,\dots,n, \tag{5}$$

$$P'_{kj} = \sum_{i=1}^{n}\sum_{s=1}^{m_k}\left(P_{kjs}\right)X_{jki}^{S} \quad \text{for all } j = 1,\dots,n;\; k = 1,\dots,c, \tag{6}$$

$$t_{kj} + P'_{kj} - t_{k+1)j} \le 0 \quad \text{for all } k = 1,\dots,c-1;\; j = 1,\dots,S_1, \tag{7}$$

$$t_{kj} + P'_{kj} - t_{k-1)j} \le 0 \quad \text{for all } k = 2,\dots,c;\; j = S_1+1,\dots,n, \tag{8}$$

$$t_{cj} + P'_{cj} \le C_{max} + M\left(1 - \sum_{s=1}^{m_c} X_{jci}^{S}\right) \quad \text{for all } j = 1,\dots,S_1 \quad \text{for all } i = 1,\dots,n, \tag{9}$$

$$t_{1j} + P'_{1j} \le C_{max} + M\left(1 - \sum_{s=1}^{m_1} X_{j1i}^{S}\right) \quad \text{for all } j = S_1+1,\dots,n \quad \text{for all } i = 1,\dots,n, \tag{10}$$

$$t_{kq} + M\left(1 - X_{qk\,i+1)}^{S}\right) \ge t_{kj} + P'_{kj} - M\left(1 - X_{jkh}^{S}\right) \quad \text{for all } j,q = 1,\dots,n$$
$$\text{for all } k = 1,\dots,c \quad \text{for all } i = 1,\dots,n-1 \quad \text{for all } h = 1,\dots i \quad \text{for all } s = 1,\dots,m_k, \tag{11}$$

$$X_{jki}^{S} \in \{0,1\}. \tag{12}$$

The mathematical *Eq. (1)* represents the objective function minimizing the completion time (C_{max}) of the last job on the last machine. *Constraint (2)* defines that each job at each stage can be processed only in one position and only on one machine. *Constraint (3)* ensures that only one job can be processed at a position of one stage on a machine. *Constraint (4)* assures that if a direct job is processed at the first position of the first stage (on each machine), its starting time of operation at the first stage is zero. *Constraint (5)* defines that if a reverse job is processed at the first position of the last stage (on each machine), its starting time of operation at the last stage is zero. *Constraint (6)* shows the real processing time of job *j* at the k^{th} stage. *Constraint (7)* implies that the starting time of a direct job at each stage is equal to its starting time at the previous stage plus its processing time (a pre-required direct job constraint). *Constraint (8)* indicates that the starting time of a reverse job at each stage is obtained by adding its starting time at the next stage with its processing time (pre-required reverse job constraint). *Constraint (7)* and *Constraint (8)* are initial precedence constraints on the operations of the jobs. *Constraint (9)* and *Constraint (10)* determine the finishing times (C_{max}) of the direct and reverse jobs, respectively, obtained by the finishing times of the last job. *Constraint (11)* ensures that if job *q* at stage *k* is processed after job *j*, then the starting time of job *q* must be equal to the starting time of job *j* plus its processing time. *Constraint (12)* defines the type of the variable being used.

3.5 | An Illustrative Example

A small example is solved in this section using the Lingo 16 software to illustrate the applicability of the model developed in Section 3.4. In this example, there are 3 direct jobs (1, 2, and 3) and 3 reverse jobs (4, 5, and 6) to be scheduled at 5 stages. While there are 2 machines in Stages 1 and 2, in the other three stages, 3 machines are available. *Table 1* shows the duration of each job at each stage on each machine and *Fig. 2* displays the optimal solution.

Table 1. The duration of each job at each stage on each machine [p(k, j, s)=P_{kjs}].

p(1,1,1)= 4	p(2,1,1)=2	p(3,1,1)=3	p(4,1,1)=1	p(5,1,1)=1
p(1,1,2)=5	p(2,1,2)=5	p(3,1,2)=4	p(4,1,2)=1	p(5,1,2)=1
p(1,1,3)=3	p(2,1,3)=3	p(3,1,3)=1	p(4,1,3)=2	p(5,1,3)=1
p(1,2,1)=4	p(2,2,1)=5	p(3,2,1)=3	p(4,2,1)=2	p(5,2,1)= 1
p(1,2,2)=5	p(2,2,2)=2	p(3,2,2)=5	p(4,2,2)=1	p(5,2,2)=1
P(1,2,3)=1	p(2,2,3)=4	p(3,2,3)=4	p(4,2,3)=1	p(5,2,3)=1
p(1,3,1)=1	p(2,3,1)=4	p(3,3,1)=4	p(4,3,1)=1	p(5,3,1)=1
p(1,3,2)=1	p(2,3,2)=1	p(3,3,2)=4	p(4,3,2)=2	p(5,3,2)=1
p(1,3,3)=5	p(2,3,3)=1	p(3,3,3)=3	p(4,3,3)=3	p(5,3,3)=1
p(1,4,1)=3	p(2,4,1)=5	p(3,4,1)=3	p(4,4,1)=3	p(5,4,1)=1
p(1,4,2)=1	p(2,4,2)=1	p(3,4,2)=2	p(4,4,2)=1	p(5,4,2)=1
p(1,4,3)=3	p(2,4,3)=2	p(3,4,3)=3	p(4,4,3)=2	p(5,4,3)=1
p(1,5,1)=1	p(2,5,1)=1	p(3,5,1)=3	p(4,5,1)=1	p(5,5,1)=1
p(1,5,2)=2	p(2,5,2)=4	p(3,5,2)=5	p(4,5,2)=2	p(5,5,2)=1
p(1,5,3)=1	p(2,5,3)=4	p(3,5,3)=5	p(4,5,3)=2	p(5,5,3)=1
p(1,6,1)=3	p(2,6,1)=2	p(3,6,1)=2	p(4,6,1)=2	p(5,6,1)=1
p(1,6,2)=5	p(2,6,2)=5	p(3,6,2)=4	p(4,6,2)=1	p(5,6,2)=1
p(1,6,3)=1	p(2,6,3)=3	p(3,6,3)=5	p(4,6,3)=1	p(5,6,3)=1

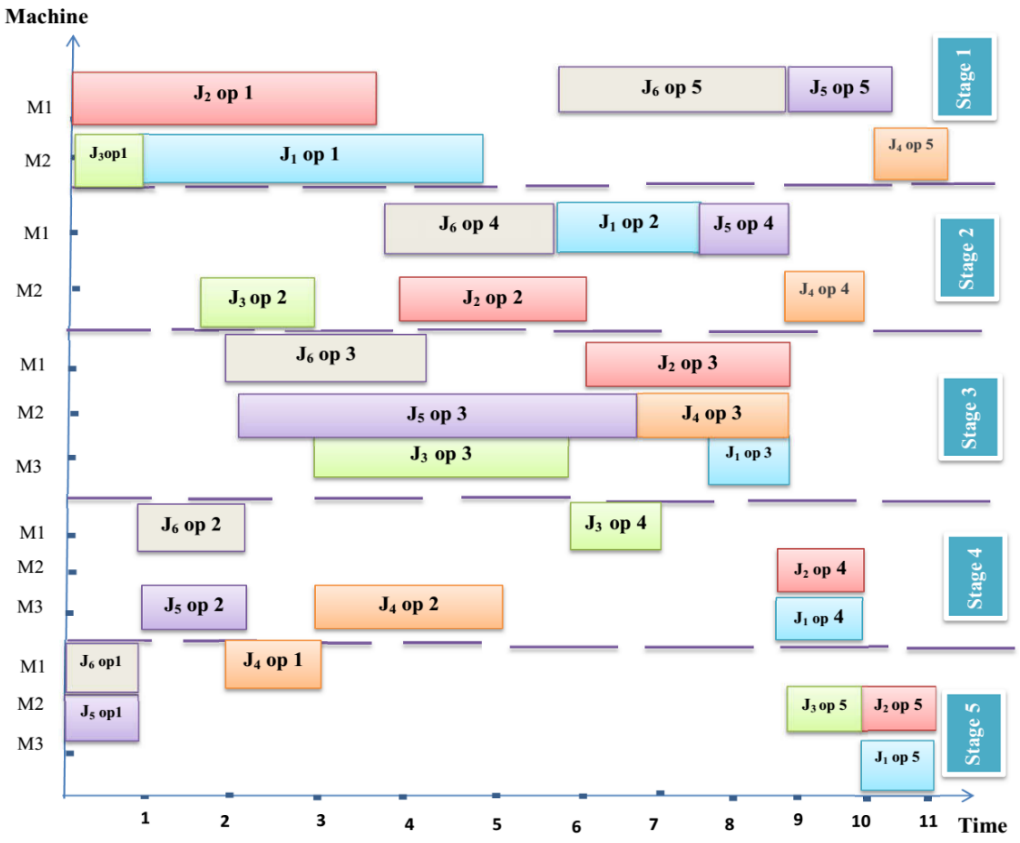


Fig. 2. The optimal solution obtained by Lingo 16.

4 | The Solution Approach

Due to the NP-hardness of the problem [22], a VDO algorithm is utilized in this section to solve large-scale problems. VDO, was first proposed by Mehdizadeh et al. [26] and has been used to solve many optimization problems especially Production Scheduling Problems (PSPs) so far. Aliabadi et al. [27] studied the scheduling and sequence of multi-product flow-shops problem with sequence-dependent setup times. Three meta-heuristics including a hybrid Particle Swarm Optimization (PSO), a hybrid VDO, and a hybrid GA were used to solve this problem. Mehdizadeh et al. [26] used a VDO algorithm for the identical parallel machine scheduling problem with sequence-independent family setup times to minimize the total weighted completion time and computationally compare the results obtained by the proposed VDO with the results of the GA and branch-and-bound method. Yazdani et al. [28] employed two meta-heuristics, namely SA and VDO to minimize the makespan of a Dual-Resource Constrained Flexible Job-shop Scheduling Problem (DRCFJSP). Alaghebandha et al. [29] considered an efficient Hybrid Vibration Damping Optimization (HVDO) with an Imperialist Competitive Algorithm and Simulated Annealing to solve the economic lot sizing and scheduling problem in distributed permutation flow shop problem with several non-identical factories and machines and used GA and VDO for comparison. Rashidi Komijan et al. [30] applied a VDO algorithm to solve an integrated airline feet assignment and crew scheduling problem. To evaluate the performance of the proposed VDO algorithm, the obtained results were compared with PSO and optimal solutions. Aghighi et al. [25] used the VDO algorithm and six other algorithms including SA, ICA, BAT, HSA, ACO, and CS) to solve larger-size examples of open shop scheduling with the reverse flow. Soofi et al. [31] used the VDO algorithm to solve the Dual-Resource Constrained Flexible Job-shop Scheduling (DRCFJSS) problem under machine breakdown and operational uncertainty.

4.1 | Vibration Damping Optimization Algorithm

There is a useful connection between vibration damping and combinatorial optimization. VDO algorithm is a powerful meta-heuristic algorithm with a stochastic search method based on the concept of vibration damping in mechanical vibration [32], [33]. All bodies possessing mass and elasticity are capable of vibration. Vibrating systems are all subject to damping to some degree because energy is dissipated by friction and other resistances. This process is called vibration damping. At a given amplitude, the probability distribution of the oscillatory system is determined by the Rayleigh probability. At high amplitude, the probability converges to 1 for all energy states. It can also be seen that there exists a small probability that the system might have high energy at low amplitudes. Therefore, the statistical distribution of energies allows the system to escape from a local energy minimum. *Table 2* indicates an analogy between optimization problems and vibration damping.

Table 2. The analogy between optimization problems and VDO algorithm [26].

VDO	Optimization Problems
System states	Feasible solution
Energy	Objective function
Change of state	Neighbouring solution
Amplitude	Control parameter
Vibration damping	Heuristic solution
Degrees of freedom	Number of decision variables

The algorithm consists of a sequence of iterations. Each iteration consists of randomly changing the current solution to create a new solution in the neighborhood of the current solution. The neighborhood is defined by the choice of the generation mechanism. Once a new solution is created the corresponding change in the cost function is computed to decide whether the newly produced solution can be accepted as the current solution. If the change in the cost function is negative the newly produced solution is directly taken as the current solution. Otherwise, it is accepted according to Raleigh's probability. If the difference between the cost function value of the current and the newly produced solutions is equal to

or larger than zero, a random number r in [0, 1] is generated from a uniform distribution, and if $r \leq p(A)$, then the newly produced solution is accepted as the current solution. If not, the current solution is unchanged. The pseudo-code of the vibration damping algorithm for single objective function problems is shown in *Algorithm 1*.

Algorithm 1. The pseudo-code of the VDO algorithm [26].

Step 1. Generating a feasible initial solution.
Step 2. Initializing the algorithm parameters, which consist of: initial amplitude (A_0), maximum iteration at each amplitude (L), damping coefficient (γ), and standard deviation (σ). Finally, parameter t is set in one (t=1)
Step 3. Calculating the objective value U_0 for the initial solution.
Step 4. Initializing the internal loop. In this step, the internal loop is carried out for l =1 and repeat while l < L.
Step 5. Neighbourhood generation.
Step 6. Accepting the new solution
Set $\Delta = U - U_0$ Now, if Δ < 0, accept the new solution, else if Δ > 0 generate a random number r between [0, 1];
If $r < 1 - exp\left(\frac{-A_t^2}{2\sigma^2}\right)$, then accept a new solution; otherwise, reject the new solution and accept the previous solution.
If $l > L$, then $t+1 \rightarrow t$ and go to Step 7; otherwise, $l+1 \rightarrow l$ and go back to Step 5.
Step 7. Adjusting the amplitude. In this step, $A_t = A_0 \, exp(\frac{-\gamma t}{2})$ is used for reducing amplitude at each iteration of the outer cycle of the algorithm. If $A_t = 0$ return to Step 8; otherwise, go back to Step 4.
Step 8. Stopping criteria. In this step, the proposed algorithm will be stopped after pre-specified minimum amplitude A_{min} is achieved. At the end, the best solution is obtained.

4.1.1 | Solution representation

A solution is designed and represented in two steps. In Step 1, direct jobs are placed before the reverse jobs, and in Step 2, the reverse jobs are placed before the direct jobs. Note that the direct and the reverse jobs are selected randomly. In addition, a solution is represented by two C × N matrices called *S* and *m*, where *C* is the number of stages and *N* is the permutation of all jobs (direct and reverse). Depending on the rows of the *S* matrix, the order the jobs are placed in each stage is determined. In addition, the *m* matrix shows how the machines are assigned to the jobs placed in each stage. *Fig. 3* depicts a solution structure with 10 jobs, 4 stages, and up to 4 machines in each stage. In this figure, the number 3 in the first row of the *S* matrix and the number 1 in the same row of the m matrix indicate that the third job (direct) at the first stage is processed on Machine 1.

S

1	2	4	5	**3**	6	9	8	7	10
1	5	2	3	4	9	8	7	10	6
9	6	8	7	10	1	4	5	2	3
8	7	10	9	6	1	2	3	4	5

m

4	3	4	4	**1**	4	1	3	2	4
1	1	2	3	3	2	2	3	2	1
2	3	3	1	1	3	2	2	1	2
1	2	1	4	3	4	4	1	4	3

Fig. 3. Proposed solution representation.

4.1.2 | Initial solution

Choosing the proper start solution plays an essential role in the quality of the near-optimum solution and the time required. Various methods such as a related priority dispatching rule or a random method can be

applied to yield an initial solution. In this research, a random method is employed to select initial solutions for the VDO algorithm.

4.1.3 | Neighborhood structure

Three operators (swap mutation, insertion, and reversion) are used in this paper to construct a neighborhood for the solution representation *S*. In the swap mutation shown in *Fig. 4*, two cells in each row of the *S* matrix corresponding to the job orders in different stages are chosen randomly and then their locations are exchanged.

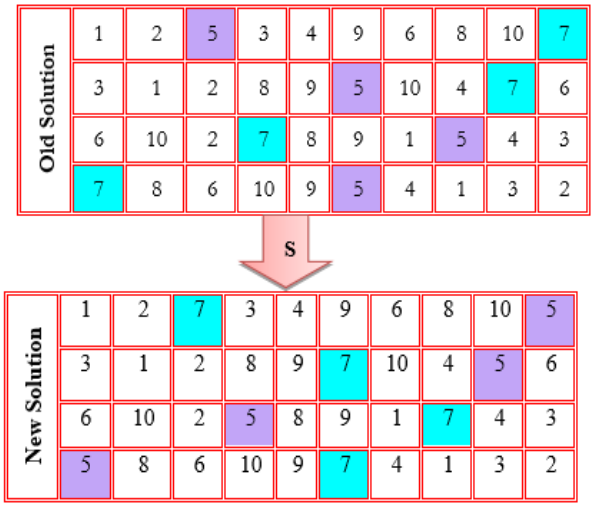


Fig. 4. The considered swap operator.

Next, the fitness of the solution before and after the swap is compared. If the fitness is not improved, then the solution remains unchanged. Similarly, the insertion and reversion operations are employed as shown in *Figs. 5* and *6*, respectively. In addition, the uniform operator is used for the solution representation m (the m matrix), based on which 50% of the cells are selected randomly and then reproduced. *Fig. 7* depicts this operation.

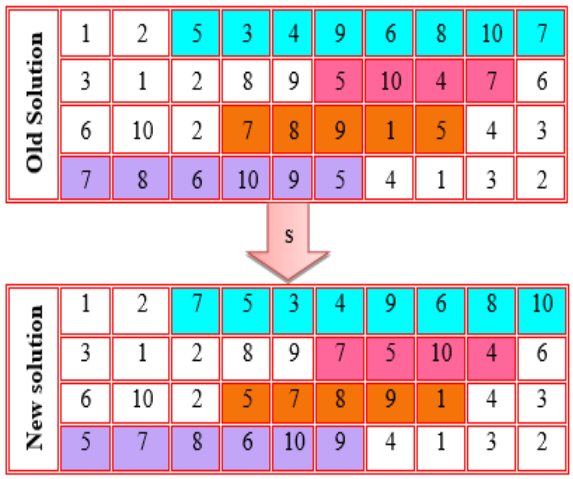


Fig. 5. The insertion operator.

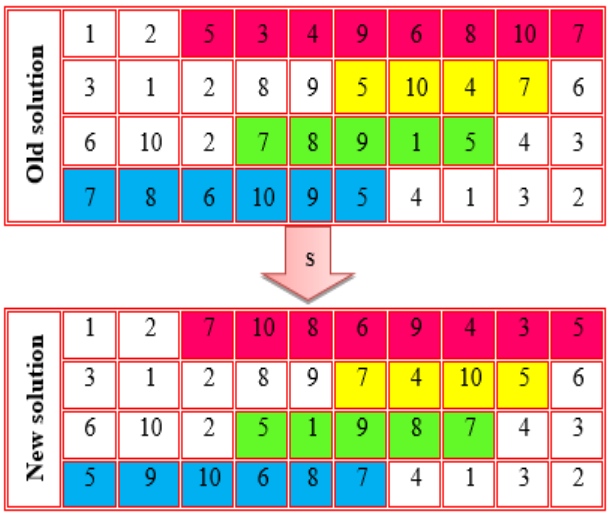


Fig. 6. The reversion operator.

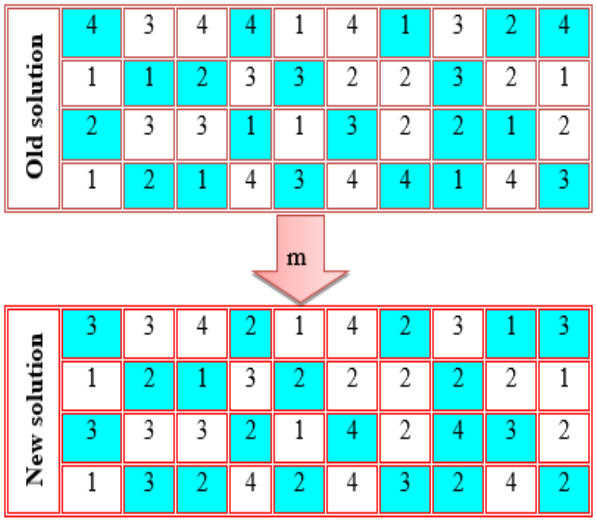


Fig. 7. The uniform operator.

For a new solution to remain feasible, when the S operator is applied, it is necessary to make sure that each job in each row is replaced only once. Moreover, in calculating the makespan (fitness function), the first operation of each direct job should be scheduled at the first stage and the first operation of each reverse job should be scheduled at the last stage. When the m operator is applied, the new solution remains feasible anyway.

4.1.4 | Reducing amplitude

After each iteration in the main loop of the VDO algorithm, the amplitude is reduced using *Eq. (13)*. This can reduce the probability of accepting neighbors generated to avoid increasing the number of iterations and hence to increase the speed [32].

$$A_t = A_0 e^{-\gamma t/2)}. \quad (13)$$

4.1.5 | Neighborhood acceptance/rejection mechanism

After each generation, a new solution is created based on which the corresponding change in the objective function is computed to decide whether the newly produced solution can be accepted as the current solution [26]. If the new member overcomes the current member, the new member replaces it. Otherwise,

the probability in *Eq. (14)* is calculated after that a uniform random number *r* is generated within [0, 1]. If *r* is less than the probability obtained by *Eq. (14)*, the new solution will replace the current solution.

$$P = 1 - e^{-\frac{A^2}{2\sigma^2}}. \quad (14)$$

4.1.6 | Repeating the number of inner loops

Another important parameter for finding better solutions is the number of neighborhood searches for each amplitude *L* repeated for each member of the population. The amplitude is used to reduce them in each iteration [34]. The number of repetitions of the inner loop depends on the success rate of the forced vibration. The number of these repetitions is directly related to the algorithm time, meaning that a large number of the inner loop leads to an increase in the time of the algorithm while not improving the quality of the solutions.

4.1.7 | Stop criterion

While there are different criteria such as the maximum number of iterations available in the literature to stop a meta-heuristic, in this paper the VDO algorithm and other mentioned algorithms are stopped when it achieves a good convergence behavior.

5 | Computational Results

To demonstrate the applicability of the proposed methodology alongside validating the results obtained in terms of the solution quality and computational time, the solutions obtained using the proposed VDO algorithm are compared with those obtained by the Lingo 16 software, when both solve some small-size problems. In addition, a GA [35] is also utilized to analyze and compare the results found by the VDO algorithm when they solve some randomly generated test problems. Before doing this, the parameters of both solution algorithms are first calibrated in the next subsection using the Taguchi method [36] to find better quality solutions in a reasonable time.

5.1 | Parameter Tuning

In this paper, the Taguchi method is implemented to tune the parameters of the VDO algorithm and GA. In this method, a statistical measure, called the Signal-to-Noise (S/N) ratio, is calculated to evaluate the performance of the algorithm based on a specified combination level of its parameters. Here, the term "signal" denotes the mean objective function value which is desirable, and "noise" denotes the standard deviation of the objective function values which is undesirable. The aim is to maximize the signal-to-noise ratio [31], [32].

The parameter levels of the algorithms to solve a typical problem are presented in *Tables 3* and *4*. Besides, the L25 design of the Taguchi method is employed for the proposed algorithms. The algorithms are coded in MATLABR2013a, Version 8.1.0.604, and are executed on a PC with 6 GB RAM and a 2.30 GHz CPU. For each algorithm, the related S/N ratio is obtained using the Minitab 17 software as illustrated in *Figs. 8* and *9*. In these figures, the best level of each parameter is selected to be the one with the highest S/N ratio. As a result, the proper values of the parameters are presented in *Tables 5* and *6*.

Table 3. Parameters and their levels for the GA.

Parameters	Npop	P_c	P_m
Presented levels	25	0.6	0.1
	50	0.7	0.15
	100	0.8	0.20
	150	0.9	0.30
	200	0.95	0.40

Table 4. Parameters and their levels for the VDO algorithm.

Parameters	A_0	γ	L
Presented levels	100	0.001	1
	300	0.005	2
	500	0.01	3
	700	0.05	4
	1000	0.1	5

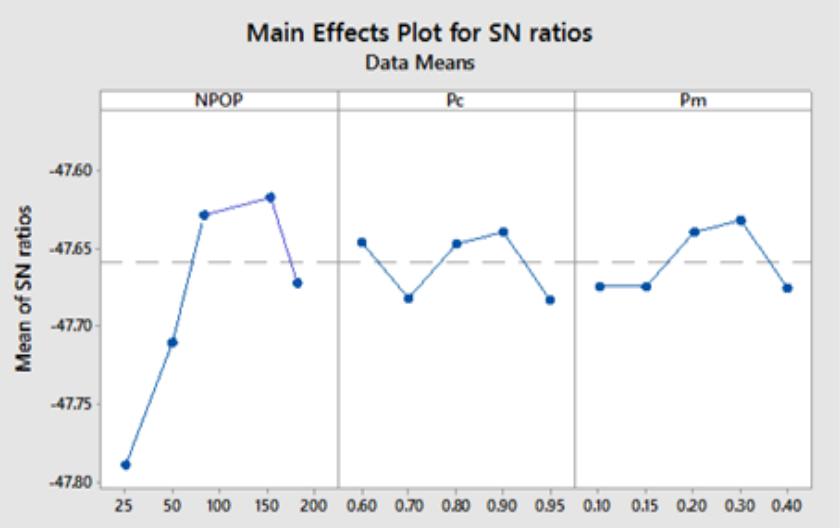


Fig. 8. The mean S/N ratio plot at each level of the parameters for the objective function values found by GA (Larger mean of S/N is better).

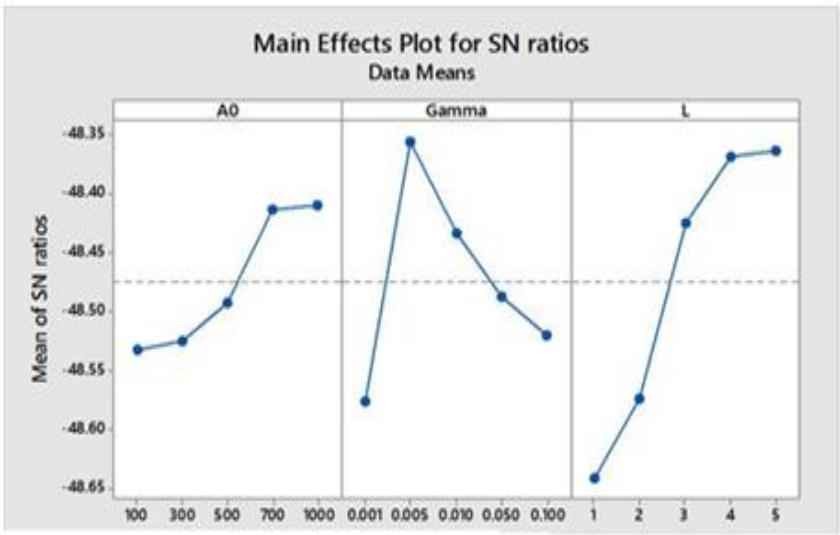


Fig. 9. The mean S/N ratio plot at each level of the parameters for objective function values found by the VDO algorithm (A larger mean of S/N is better).

Table 5. Optimal levels of the GA parameters.

Parameters	Npop	P_c	P_m
Selected levels	Forth level	Forth level	Forth level
The amount of levels	150	0.90	0.30

Table 6. Optimal levels of the VDO algorithm parameters.

Parameters	A_0	γ	L
Selected levels	Forth level	Second level	Fifth level
The amount of levels	700	0.005	5

5.2 | Numerical Results and Validation

To validate the proposed algorithm in terms of the solution quality (makespan) and computational time, the results found by the proposed VDO algorithm are compared with those obtained by the Lingo 16 software and GA.

5.2.1 | Validation results

At first, the results found by the proposed VDO algorithm are compared with those obtained by Lingo 16 software 14 problems. In the first 10 small-size problems, the numbers of direct and reverse jobs are both 2 and 3, while the number of stages is chosen in the range 2-7. Besides, the maximum number of available machines in each stage is 3 and the processing times are selected randomly from a uniform distribution in [1, 99]. The VDO algorithm stops when it converges to a solution.

The optimal makespans of 14 test problems obtained by Lingo software are shown in the 6th column of *Table 7*. The 8th and the 9th columns of this table present the best and the average solutions found by the VDO algorithm when it solves each problem 5 times. A review of the results in *Table 7* shows that the VDO algorithm and the Lingo software 16 are capable of attaining the optimal solutions for test problems 1-10. In the last four test problems, however, the VDO algorithm works better, whereas Lingo is unable to solve these problems in reasonable computational times.

In addition, the best solutions found by the VDO algorithm for all small-size problems are identical to the optimal solutions obtained by Lingo while the computational time required by Lingo is larger than the one needed in the VDO algorithm to solve each problem. The required computational times for the proposed algorithm are significantly smaller than the Lingo software 16 as shown in *Table 7*. Furthermore, the average solutions found by the VDO algorithm are not far from the best solutions. This somehow validates the proposed VDO algorithm.

5.2.2 | Experimental results

As Lingo is unable to solve medium and large-size problems in a reasonable computational time, in this section, the proposed solution algorithm is compared with (GA) when both solve 30 randomly generated test problems. In these problems, the numbers of direct and reverse jobs range are in 6-20, the number of stages is in the range of 2-3, the maximum number of machines in each stage is 3, and again the processing times are selected randomly from a uniform distribution in [1, 99]. The algorithms are stopped when they converge. In addition, each of the problems is solved by the two algorithms five times. The last six columns of *Table 8* contain the best solution, the average solution, and the computational time of the two algorithms.

Table 7. The result obtained by the proposed VDO and LINGO.

Problem					LINGO Results		VDO Results			
Number	Number of Direct Jobs	Number of Reverse Jobs	Stages	Maximum Number of Machines in Each Stage	Lingo Results (the Best Solution Found)	Lingo Time (s)	Best VDO	Solution Average	Average of CPU-Times (s)	GAP between Lingo and Best VDO
1	2	2	2	3	7	14	7	7.2	3	0
2	2	2	3	3	16	22	16	16.1	6.1	0
3	2	2	4	3	14	35	14	14	14	0
4	2	2	5	3	23	20	23	23.4	16	0
5	2	2	6	3	24	27	24	24.2	23	0
6	2	2	7	3	28	40	28	28	34.1	0
7	3	3	2	3	6	113	6	6.2	50.2	0
8	3	3	3	3	25	150	25	25.3	100.2	0
9	3	3	4	3	23	270	23	23.6	80.6	0
10	3	3	5	3	27	1703	27	27.4	120.2	0
11	4	4	2	3	N/A	3740	10	10.7	214.72	-
12	4	4	3	3	N/A	3867	16	16.6	222.94	-
13	4	4	4	3	N/A	3998	14	14	235.19	-
14	5	5	2	3	N/A	4000	18	18.2	216.77	-

Table 8. Results obtained by the proposed VDO and GA.

Problem					VDO Results		GA Results			
Number	Number of Direct Jobs	Number of Reverse Jobs	Stages	Maximum Number of Machines in Each Stage	Best Solution	Average Solution	Average CPU-Time (s)	Best Solution	Average Solution	Average CPU-Times (s)
1	6	6	2	3	27	27.2	27.42	28.40	28.56	42.88
2	6	6	3	3	37.70	38.08	50.31	48	48.96	84.53
3	7	7	2	3	29.11	29.65	99.65	51	51.68	117.70
4	7	7	3	3	31.20	31.82	24.30	83.20	83.78	53.66
5	8	8	2	3	29.90	32.09	84.51	68	68.54	105.12
6	8	8	3	3	52.10	52.77	80.40	135.31	136.77	182.326
7	9	9	2	3	36	36.17	55.32	46.8	47.05	95.49
8	9	9	3	3	52	52.50	87.13	170	170.27	123.54
9	10	10	2	3	36.10	36.72	96.23	72	72.89	147.17
10	10	10	3	3	29.1	30.19	114.21	214	214.88	164.103
11	11	11	2	3	39	39.44	214.72	118.10	118.59	259.63
12	11	11	3	3	63.11	63.92	222.94	99	99.28	261.03
13	12	12	2	3	47	47.6	235.19	56.11	56.58	299.8
14	12	12	3	3	83.91	84.04	216.77	84	84.47	238.33
15	13	13	2	3	49	49.23	225.39	279	279.61	247.56
16	13	13	3	3	59.89	61.04	239.51	303.99	304.91	269.22
17	14	14	2	3	55	55.48	248.23	305.12	305.456	302.31
18	14	14	3	3	55.97	56.03	264.23	274	274.17	332.6
19	15	15	2	3	63	63.41	285.33	77.11	77.52	354.6
20	15	15	3	3	92	92.46	304.23	121	121.58	399.02
21	16	16	2	3	76	76.43	684.22	230	230.92	768.3
22	16	16	3	3	82.99	83.50	680.42	229.11	229.57	798.4
23	17	17	2	3	84	84.40	699.54	263	263.57	814.3
24	17	17	3	3	115.96	116.14	704.26	289	289.40	825.4
25	18	18	2	3	96	96.7	725.33	135.2	135.73	840.3
26	18	18	3	3	128	128.17	739.22	202	202.14	869.69
27	19	19	2	3	153	153.68	804.20	206.96	207.26	902.33
28	19	19	3	3	155.10	155.57	822.25	218	218.42	934.5
29	20	20	2	3	188	188.21	840.50	294	294.57	955.4
30	20	20	3	3	200	201.13	891.23	314	314.12	999.12

The results in *Table 8* show that the VDO algorithm has a better performance in terms of solution quality and computational time compared to GA. To further compare the two solution algorithms, another measure, namely the Relative Percentage Deviations (RPD) is defined in *Eq. (15)*.

$$RPD = \left|\frac{Alg_{sol} - Min_{sol}}{Min_{sol}}\right| * 10. \quad (15)$$

In *Eq. (13)*, Alg_{sol} is a solution found for a problem by either of the two solution algorithms and Min_{sol} is the smallest solution when both algorithms solve the problem 5 times. The average RPDs of the VDO algorithm and GA for each of the 30 test problems are reported in *Table 9* and the graphical comparison of the relative deviation index is shown in *Fig. 10*. The results in *Table 9* as well as in *Fig. 10* indicate that VDO algorithm is the better algorithm in terms of RPD.

Table 9. Averag RPD of the VDO algorithm and GA.

Problem Number	Algorithm	
	GA	VDO
1	0.05	0.00
2	3.71	0.40
3	0.90	0.09
4	2.08	0.17
5	1.52	0.18
6	7.03	0.94
7	0.73	0.33
8	5.26	0.93
9	1.68	0.35
10	6.90	0.11
11	3.36	0.45
12	2.65	1.35
13	9.28	0.75
14	10.21	2.09
15	1.08	0.81
16	2.11	1.24
17	9.08	1.04
18	10.23	1.06
19	1.85	1.33
20	3.47	2.40
21	8.49	1.81
22	7.44	2.07
23	8.69	2.09
24	9.64	3.27
25	3.99	2.59
26	6.43	3.71
27	6.62	4.65
28	7.03	4.72
29	9.83	5.91
30	10.54	6.39
Average	5.40	1.77

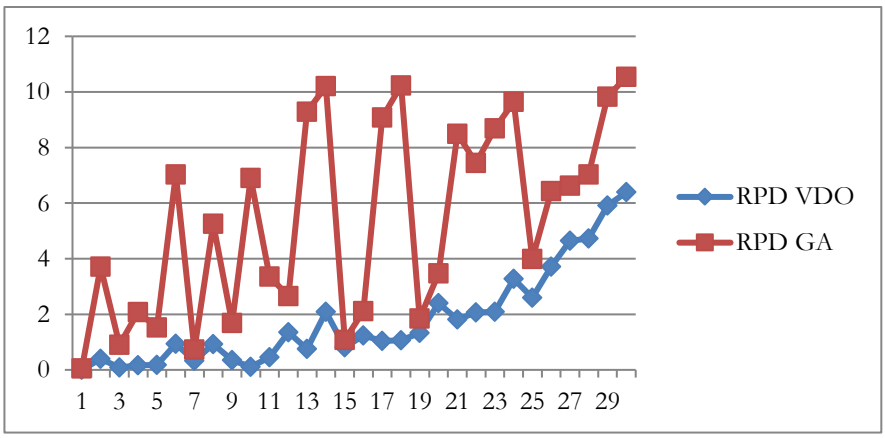


Fig. 10. Graphical comparisons of the RPD index.

To verify the validity of the algorithms in terms of average RPD statistically, their means are compared using a two-sample Student's t-test. The samples have been accomplished using the MINITAB 17.0 software and are summarized in *Table 10*, where the p-value of the test statistic in comparing the two means becomes 0.000. It means that the two RPD means are significantly different at almost 100% confidence level. In other words, the VDO is the better algorithm in terms of the average *RPD* for sure. In addition, the mean plot and the LSD intervals of the two algorithms in *Fig. 11* reveal this fact better.

Table 10. Statistical comparison of VDO and GA in terms of average RPD.

Factor	N	Mean	St.Dev	Se Mean	P-Value	T-Value	Result
VDO	30	1.77	1.76	0.32	0.000	5.12	H_0 is rejected
GA	30	5.40	3.46	0.63			

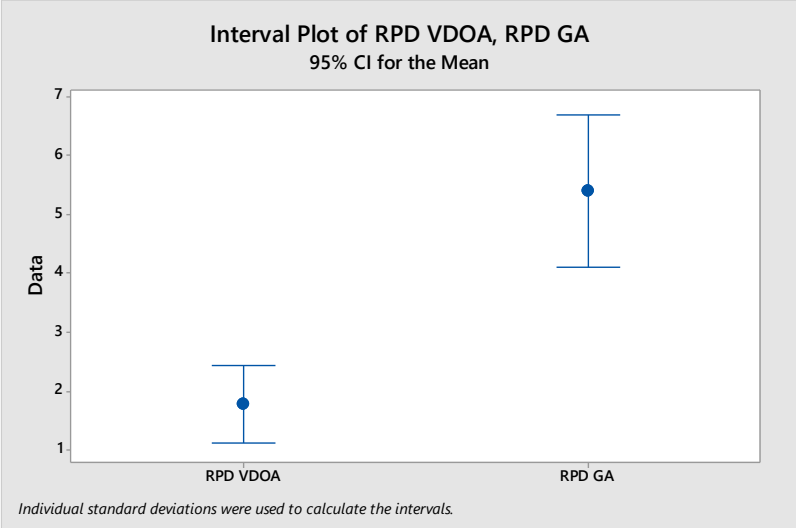


Fig. 11. Mean plot and LSD intervals of RPD for the proposed meta-heuristic algorithms.

6 | Conclusions and Future Research

In this paper, the FJSP with two flows of jobs (direct and reverse) at each stage was investigated. A mathematical programming model was presented and a numerical example was solved by Lingo 16 software to demonstrate the applicability. A small example was solved by using the Lingo 16 software to illustrate the applicability of the developed mathematical. Then the results found by the proposed VDO algorithm were compared with those obtained by Lingo 16 software on 14 problems. A review of the results showed that the VDO algorithm and the Lingo software 16 were capable of attaining the optimal solutions for small test problems. In the last test problems, however, the VDO algorithm worked better, whereas Lingo was unable to solve these problems in reasonable computational times. In addition, the best solutions found by the VDO algorithm for all small-size problems were identical to the optimal solutions obtained by Lingo while the computational time required by Lingo was larger than the one needed in the VDO algorithm to solve each problem. Furthermore, the average solutions found by the VDO algorithm are not far from the best solutions. Due to the NP-hardness of the problem, a metaheuristic algorithm namely the VDO algorithm was applied to solve large-scale problems. To validate the results obtained by the VDO algorithm in terms of solution quality and time, the results were compared with those obtained from Lingo 16 software for small-size problems. Finally, the proposed algorithm was compared with a GA by solving some randomly generated large test problems, based on which the results were analyzed statistically. Computational results showed that the VDO algorithm had better performance than GA. Developing a multi-objective mathematical model for the FJSP with reverse flows, taking into account real conditions such as sequence-dependent setups-times, blocking, etc., and considering the reverse flow concepts in other scheduling environments can be considered as future studies.

References

[1] Pinedo, M. (1995). *Scheduling: theory, algorithms and applications*. Prentice-Hall.

[2] Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, *1*(2), 117–129.

[3] Mati, Y., & Xie, X. (2004). The complexity of two-job shop problems with multi-purpose unrelated machines. *European journal of operational research*, *152*(1), 159–169. DOI:10.1016/S0377-2217(02)00675-6

[4] Salema, M. I. G., Barbosa-Povoa, A. P., & Novais, A. Q. (2010). Simultaneous design and planning of supply chains with reverse flows: A generic modelling framework. *European journal of operational research*, *203*(2), 336–349.

[5] Kim, H. J., Lee, D. H., & Xirouchakis, P. (2007). Disassembly scheduling: Literature review and future research directions. *International journal of production research*, *45*(18–19), 4465–4484.

[6] Ilgin, M. A., & Gupta, S. M. (2011). Recovery of sensor embedded washing machines using a multi-kanban controlled disassembly line. *Robotics and computer-integrated manufacturing*, *27*(2), 318–334.

[7] McGovern, S. M., & Gupta, S. M. (2007). A balancing method and genetic algorithm for disassembly line balancing. *European journal of operational research*, *179*(3), 692–708. DOI:10.1016/j.ejor.2005.03.055

[8] Gao, K. Z., Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F., & Sadollah, A. (2016). Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-based systems*, *109*, 1–16.

[9] Giglio, D., Paolucci, M., & Roshani, A. (2017). Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems. *Journal of cleaner production*, *148*, 624–641.

[10] Wu, X., & Sun, Y. (2018). A green scheduling algorithm for flexible job shop with energy-saving measures. *Journal of cleaner production*, *172*, 3249–3264.

[11] Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2018). Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm. *IEEE transactions on cybernetics*, *49*(5), 1944–1955.

[12] Gong, G., Deng, Q., Gong, X., Liu, W., & Ren, Q. (2018). A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators. *Journal of cleaner production*, *174*(c), 560–576. DOI:10.1016/j.jclepro.2017.10.188

[13] Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019). MILP models for energy-aware flexible job shop scheduling problem. *Journal of cleaner production*, *210*, 710–723. DOI:10.1016/j.jclepro.2018.11.021

[14] Gong, X., De Pessemier, T., Martens, L., & Joseph, W. (2019). Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation. *Journal of cleaner production*, *209*, 1078–1094. DOI:10.1016/j.jclepro.2018.10.289

[15] Dondo, R. G., & Méndez, C. A. (2016). Operational planning of forward and reverse logistic activities on multi-echelon supply-chain networks. *Computers & chemical engineering*, *88*, 170–184.

[16] Osmani, A., & Zhang, J. (2017). Multi-period stochastic optimization of a sustainable multi-feedstock second generation bioethanol supply chain − A logistic case study in Midwestern United States. *Land use policy*, *61*, 420–450. DOI:10.1016/j.landusepol.2016.10.028

[17] Giri, B. C., Chakraborty, A., & Maiti, T. (2017). Pricing and return product collection decisions in a closed-loop supply chain with dual-channel in both forward and reverse logistics. *Journal of manufacturing systems*, *42*, 104–123. https://doi.org/10.1016/j.jmsy.2016.11.007

[18] Adenso-Díaz, B., García-Carbajal, S., & Gupta, S. M. (2008). A path-relinking approach for a bi-criteria disassembly sequencing problem. *Computers and operations research*, *35*(12), 3989–3997.

[19] Duta, L., Filip, F. G., & Popescu, C. (2008). Evolutionary programming in disassembly decision making. *International journal of computers, communications & control*, *3*(3), 282–286.

[20] Kim, H. J., Lee, D. H., Xirouchakis, P., & Kwon, O. K. (2009). A branch and bound algorithm for disassembly scheduling with assembly product structure. *Journal of the operational research society*, *60*(3), 419–430. DOI:10.1057/palgrave.jors.2602568

[21] Wan, H. Da, & Gonnuru, V. K. (2013). Disassembly planning and sequencing for end-of-life products with RFID enriched information. *Robotics and computer-integrated manufacturing*, *29*(3), 112–118. DOI:10.1016/j.rcim.2012.05.001

[22] Abdeljaouad, M. A., Bahroun, Z., Omrane, A., & Fondrevelle, J. (2015). Job-shop production scheduling with reverse flows. *European journal of operational research*, *244*(1), 117–128. DOI:10.1016/j.ejor.2015.01.013

[23] Tanimizu, Y., Sakamoto, M., & Nonomiya, H. (2017). A co-evolutionary algorithm for open-shop scheduling with disassembly operations. *Procedia cirp*, *63*, 289–294. DOI:10.1016/j.procir.2017.03.138

[24] Ren, Y., Zhang, C., Zhao, F., Xiao, H., & Tian, G. (2018). An asynchronous parallel disassembly planning based on genetic algorithm. *European journal of operational research*, *269*(2), 647–660.

[25] Aghighi, S., Niaki, S. T. A., Mehdizadeh, E., & Najafi, A. A. (2021). Open-shop production scheduling with reverse flows. *Computers and industrial engineering*, *153*, 107077. DOI:10.1016/j.cie.2020.107077

[26] Mehdizadeh, E., Tavakkoli-Moghaddam, R., & Yazdani, M. (2015). A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times. *Applied mathematical modelling*, *39*(22), 6845–6859. DOI:10.1016/j.apm.2015.02.027

[27] Aliabadi, M., Jolai, F., Mehdizadeh, E., & Jenabi, M. (2011). A flow shop production planning problem with basic period policy and sequence dependent set up times. *Journal of industrial and systems engineering*, *5*(1), 1–19.

[28] Yazdani, M., Zandieh, M., Tavakkoli-Moghaddam, R., & Jolai, F. (2015). Two meta-heuristic algorithms for the dual-resource constrained flexible job-shop scheduling problem. *Scientia iranica*, *22*(3), 1242–1257.

[29] Alaghebandha, M., Naderi, B., & Mohammadi, M. (2018). Modeling of Scheduling and Economic Lot Sizing In Distributed Permutation Flow Shops with Non-Identical Multi Factory. *Modern research in decision making*, *3*(3), 129–155.

[30] Rashidi Komijan, A., Tavakkoli-Moghaddam, R., & Dalil, S. A. (2021). A mathematical model for an integrated airline eet assignment and crew scheduling problem solved by vibration damping optimization. *Scientia iranica*, *28*(2E), 970–984. DOI:10.24200/sci.2019.51516.2230

[31] Soofi, P., Yazdani, M., Amiri, M., & Adibi, M. A. (2021). Robust fuzzy-stochastic programming model and meta-heuristic algorithms for dual-resource constrained flexible job-shop scheduling problem under machine breakdown. *IEEE access*, *9*, 155740–155762. DOI:10.1109/ACCESS.2021.3126820

[32] Mehdizadeh, E., & Tavakkoli-Moghaddam, R. (2009). *Vibration damping optimization algorithm for an identical parallel machine scheduling problem* [presentation]. Proceeding of the 2nd international conference of iranian operations research society, babolsar, iran (pp. 20–22).

[33] Mehdizadeh, E., & Nezhad Dadgar, S. (2014). Using vibration damping optimization algorithm for resource constraint project scheduling problem with weighted earliness-tardiness penalties and interval due dates. *Economic computation and economic cybernetics studies and research*, *48*(1). https://www.researchgate.net/profile/Esmaeil-Mehdizadeh/publication/282032450_Using_vibration_damping_optimization_algorithm_for_resource_constraint_project_scheduling_problem_with_weighted_earliness-tardiness_penalties_and_interval_due_dates/links/5601cad408aeb30ba735566e/Using-vibration-damping-optimization-algorithm-for-resource-constraint-project-scheduling-problem-with-weighted-earliness-tardiness-penalties-and-interval-due-dates.pdf

[34] Fattahi, P., Hajipour, V., & Nobari, A. (2015). A bi-objective continuous review inventory control model: Pareto-based meta-heuristic algorithms. *Applied soft computing*, *32*, 211–223.

[35] Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. The MIT Press.

[36] Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley & Sons.

[37] Molla-Alizadeh-Zavardehi, S., Hajiaghaei-Keshteli, M., & Tavakkoli-Moghaddam, R. (2011). Solving a capacitated fixed-charge transportation problem by artificial immune and genetic algorithms with a Prüfer number representation. *Expert systems with applications*, *38*(8), 10462–10474. DOI:10.1016/j.eswa.2011.02.093

[38] Molla-Alizadeh-Zavardehi, S., Sadi Nezhad, S., Tavakkoli-Moghaddam, R., & Yazdani, M. (2013). Solving a fuzzy fixed charge solid transportation problem by metaheuristics. *Mathematical and computer modelling*, *57*(5–6), 1543–1558. DOI:10.1016/j.mcm.2012.12.031