European J. of Industrial Engineering

ISSN online: 1751-5262 - ISSN print: 1751-5254
https://www.inderscience.com/ejie

A bi-objective MILP model for an open-shop scheduling problem with reverse flows and sequence-dependent setup times

Saba Aghighi, Esmaeil Mehdizadeh, Seyed Taghi Akhavan Niaki, Amir Abbas Najafi

DOI: 10.1504/EJIE.2024.10059187

Article History:

Received:	03 August 2021
Last revised:	29 March 2023
Accepted:	02 April 2023
Published online:	30 April 2024

A bi-objective MILP model for an open-shop scheduling problem with reverse flows and sequence-dependent setup times

Saba Aghighi and Esmaeil Mehdizadeh

Department of Industrial Engineering,
Faculty of Industrial and Mechanical Engineering,
Qazvin Branch,
Islamic Azad University,
Qazvin, Iran
Email: saba.aghighi1@gmail.com
Email: emehdi@qiau.ac.ir

Seyed Taghi Akhavan Niaki*

Department of Industrial Engineering
Sharif University of Technology,
Azadi Ave., Tehran, Iran
Email: niaki@sharif.edu
*Corresponding author

Amir Abbas Najafi

Faculty of Industrial Engineering,
K. N. Toosi University of Technology,
Tehran, Iran
Email: aanajafi@kntu.ac.ir

Abstract: In this research, the scheduling problem of open-shop scheduling problem (OSSP) with sequence-dependent setup time (SDST) is investigated considering the reverse flow (assemble/disassemble flow on the same machines). The problem is formulated as a bi-objective mixed-integer linear programming (MILP) model. It involves reverse flows to minimise the completion time (C_{max}) and total tardiness. Since the OSSP is an NP-hard problem, a vibration damping-based multi-objective optimisation algorithm (MOVDO) is employed to solve large test problems in a reasonable runtime. Analysing the results of this algorithm was compared to an Epsilon-constrained method, which produced similar results in small problem sizes. Additionally, this algorithm is compared to other multi-objective algorithms, such as MOACO, MO-Cuckoo search, and NSGA-II, in terms of its performance. Based on the performance of these algorithms, we show that the proposed MOVDO algorithm performs better than the other algorithms to solve this problem. Eventually, a case study is investigated to validate the mathematical model and demonstrate the application. Comparing the proposed model to the results in the real world, the proposed model shows an improvement.
[Received: 3 August 2021; Accepted: 2 April 2023]

Keywords: open-shop scheduling problem; OSSP; reverse flows; sequence-dependent setup times; SDST; vibration damping-based optimisation algorithm; MOVDO.

Reference to this paper should be made as follows: Aghighi, S., Mehdizadeh, E., Niaki, S.T.A. and Najafi, A.A. (2024) 'A bi-objective MILP model for an open-shop scheduling problem with reverse flows and sequence-dependent setup times', *European J. Industrial Engineering*, Vol. 18, No. 3, pp.433–468.

Biographical notes: Saba Aghighi received her BSc, Master's, and PhD degrees in Industrial Engineering from Islamic Azad University in 2011, 2014 and 2021. Her research interests are in the areas of operation research, production planning, production scheduling, and meta-heuristic algorithms.

Esmaeil Mehdizadeh is currently an Associate Professor of Industrial Engineering at Islamic Azad University, Qazvin Branch, Iran. He received his PhD degree from Islamic Azad University, Science and Research Branch, Tehran in 2009 in Industrial Engineering. His research interests are in the areas of operation research, production planning, production scheduling, fuzzy sets, and meta-heuristic algorithms.

Seyed Taghi Akhavan Niaki is a Distinguished Professor of Industrial Engineering at the Sharif University of Technology. His research interests are in the areas of simulation modelling and analysis, applied statistics, multivariate quality control, and operations research. Before joining the Sharif University of Technology, he worked as a Systems Engineer and Quality Control Manager for the Iranian Electric Meters Company. He received his Bachelor of Science degree in Industrial Engineering from the Sharif University of Technology, and his Master's, and PhD degrees both in Industrial Engineering from West Virginia University. He is a member of the board of editors in several international journals. He is also a member of Alpha-Pi-Mu.

Amir Abbas Najafi is a Professor of Industrial Engineering at K.N. Toosi University of Technology. He received his BS. degree in Industrial Engineering from Isfahan University of Technology in 1996, and his MS and PhD degrees in Industrial Engineering from Sharif University of Technology in 1998 and 2005, respectively. His research interests include applied operations research, project scheduling and management, financial machine learning, and portfolio selection models.

1 Introduction

The purpose of resource allocation scheduling in a planning horizon is to optimise the use of available resources. A schedule that uses production capacity efficiently can increase profitability for production units. Time is always the most valuable resource. As a result, resource scheduling increases efficiency, maximises production capacity, ultimately increases profitability for an organisation, and reduces the time required to accomplish tasks (Baker and Trietsch, 2009).

The scheduling of shop environments such as flow shops and job shops applies to many industrial and service processes. A job-shop production system has a predetermined constant processing route for each job. However, sometimes the decision is made by someone responsible for scheduling. This model is known as an open shop when there is no restriction on the processing route. In this type of shop, the processing routes of different jobs can differ from one another, and each job may have its processing route (Tavakkoli-Moghaddam and Seraj, 2009). Since open-shop scheduling problem (OSSP) are common in real-world environments, providing a suitable model will be very useful to managers and practitioners. Medical clinics, car repair (battery making, refinishing, engine repair, etc.), and administrative processes in universities are just some examples of real open-shop scheduling problems.

The above specifications of an open-shop scheduling problem state that the jobs do not have a specific order, so the problem-solving space is considerably larger than that of other shop scheduling problems. Upon reviewing the literature, it appears that researchers paid less attention to the open-shop problem. Due to their similarity to real-world conditions, modelling scheduling problems, including open-shop scheduling problems, include a variety of processing constraints, including precedence constraints, breakdowns, machine eligibility restrictions, setup times, etc. This research aims to determine the limitation of setup times based on sequence (sequence-dependent setup times – SDST). A setup time is considered in the scheduling problem as the time taken to set up machines between jobs (Noori-Darvish et al., 2012). Typically, setup time is part of the processing time. However, this assumption leads to scheduling difficulties since by taking setup time into account, the completion time can be drastically reduced (Allahverdi, 2015). Moreover, in many manufacturing industries such as printing, automobiles, pharmaceuticals, chemistry, and so on, the processing time is an influential character that depends on the previous processes on the same machine (Shen et al., 2018; Allahverdi et al., 2008). The setup operations, such as cleaning or changing tools, are not only necessary between jobs but also highly dependent on the previous process on the same machine (Naderi et al., 2011). During SDST, a machine's startup time is influenced by the previous job that was processed on the same machine.

Further, this study considers the reverse flow of jobs as assembly/disassembly operations in an open-shop environment. It is the nature of jobs, not their processing route, that determines what is considered direct and reverse in this study. Research has focused on the management of reverse flows or return flows in industrial production processes. Effective management of reverse product flows is vital to the survival of most businesses. Technology advancements have resulted in shorter product life cycles than before. Due to strict environmental laws and the need to respond to customers promptly, defective, outdated, or unsold products have become valuable. Thus, scheduled product returns are vital (Eydi et al., 2020). Generally, reverse logistics is more important for industries whose products have a high value or a large return rate. Reverse flow includes repair and replacement, product modernisation, remanufacturing, recycling, and the sale or reuse of disassembled parts (Tibben-Lembke and Rogers, 2002). It is now possible to reproduce computers, mobile phones, and copy and print machines. In addition to reducing production costs, reverse logistics can improve customer service and manufacturers' competitiveness (Dolgui et al., 2006).

In the production cycle, one-way straight or reverse planning has a low optimisation level in terms of cost and service level, so it is better to consider both direct and reverse flow simultaneously (Amin and Zhang, 2013). Despite its application in the real world (automotive industry, electronics, weapons systems, etc.), less research has been conducted on this category of reverse flow. In part, this is due to the complexity of moving to more advanced systems, such as open-shop environments. Due to the lack of traditional, analytical, and accurate methods, finding suitable and optimal scheduling for these systems is a time-consuming and uneconomical process. Consequently, metaheuristic algorithms were used to solve the proposed problem.

Section 2 reviews some of the existing articles on open-shop scheduling problems. A bi-objective integer programming approach is presented in Section 3 to model this problem when the setup time is sequence-dependent, and reverse flow is considered. In Section 4, the problem-solving method is discussed using a multi-objective vibration damping-based optimisation (MOVDO), NSGA-II (Deb et al., 2002), MOACO (Cheng et al., 2012), and MO-cuckoo search (Yang and Deb, 2013) algorithms. Section 5 includes a numerical example, the analysis of the computational results, and the implementation of the algorithms in a case study. Section 6 concludes with some conclusions and suggestions for the future.

2 Literature review

In recent years, many researchers have examined the OSSP. Despite this, open-shop problems have a minimal share of the literature. In most cases, researchers used heuristic and meta-heuristic algorithms to solve these problems. This section discusses some of their studies.

Gonzalez and Sahni (1976) introduced OSSP in 1976. Since then, the OSSP has drawn the attention of researchers around the world. Khuri and Miryala (1999) examined the complexity of the open-shop scheduling problem and proved that it is NP-hard. Dror (1992) solved the OSSP to minimise both the makespan and the average floating time according to the machine's time-dependent processing characteristic with a precise algorithm. A mathematical model was proposed by Kyparisis and Koulamas (1997) to minimise the completion time of an OSSP. Bräsel and Hennes (2004) studied an open shop problem, assuming that preemptions are allowed. They considered the average completion time in their problem and proposed new scheduling models assuming that preemptions are permitted.

Mosheiov and Yovel (2004) conducted a study on flow and open shop scheduling problems (OSSP) considering a binding machine. They assumed that each job consisted of a maximum of two operations. One of these two operations is common to all jobs. Cheng and Shakhlevich (2005) worked on minimising independent objective functions for OSSP simultaneously. Their study examined both the permissibility and the absence of preemptions. The researchers calculated the completion times for a set of feasible sequences. The resulting multidimensional scheduling properties showed that creating an optimal sequence problem could be solved in polynomials to minimise a cost function independent of completion time. Sedeño-Noda et al. (2006) proposed network flow approaches for scheduling problems with preemption permissions.

Chen et al. (2008) examined mass sequences for OSSP. Their research investigated the problem, assuming that there are restrictions on release times. Also, they have studied the OSSP with the limitation or the absence of idle time of machines. Sedeño-Noda et al. (2009) proposed a method based on network flow, assuming the existence of a time window and the ability to pre-empt jobs. Their time window limits must also be strictly enforced. They simultaneously selected two criteria for minimisation. Chen et al. (2013) considered batch scheduling and delivery coordinates on two machines in the open shop problem to minimise the completion time by developing an approximate algorithm.

Kyparisis and Christos (2015) first considered the usual three-machine open-shop problem to minimise completion time when all machines are loaded; otherwise, they used an approximate solution to the three-machine problem. They also considered the problem as a mixed shop and processed the remaining work with a flow shop. Bai et al. (2016) investigated the static and dynamic state of the flexible open-shop problem with the objective function of minimising the makespan.

Zhang et al. (2019) proposed a new integer programming model for an open-shop problem using the clinic appointment schedule. The objective function was to minimise makespan and total job processing time. In other words, they considered a combination of minimising the clinic closure time and total patient waiting time. In their method, possible solutions were obtained with a two-step heuristic method, which also provides a lower bound for determining the quality of the solution. Next, they presented a two-stage stochastic optimisation model in which the expected value solution was used to generate two different patient entry patterns. Chen et al. (2020) studied open shop problems for single jobs under precedence constraints to minimise makespan.

Abreu et al. (2022) studied an OSSP in which intermediate storage is forbidden between adjacent production stages (zero buffers or machine blocking constraints). Their goal was to complete jobs in the shortest amount of time possible (makespan). The NP-hardness of this problem led them to propose a constraint programming method with two stages. Dong et al. (2022) generalised the open shop scheduling and parallel machine scheduling problems by introducing parallel multi-stage open shops. Under the constraint that job preemption is not allowed, they attempted to process all jobs on identical k-stage open shops with the minimum possible makespan. They developed an effective polynomial-time approximation scheme (EPTAS). The EPTAS is based on a combination of categorisation, scaling, and linear programming. Before scheduling different types of jobs and/or operations, they are categorised carefully into multiple types before being scaled and categorised into multiple types. An OSSP involving two machines was studied by Yuan et al. (2022) provided that one machine is subject to a fixed maintenance period. Minimising the makespan was the goal. They considered scenarios that were non-resumable, meaning that if the job was started before the maintenance period, but couldn't be completed before the maintenance period, the job had to be restarted after the maintenance period. They discussed only the maintenance period of the first machine, whereas the maintenance period of the second machine was symmetrical.

Although the open-shop scheduling problem has a considerably ample solution space, many researchers employed heuristic and meta-heuristic algorithms to solve it even for the same jobs and machines. Even though binomial time algorithms can solve some open-shop scheduling problems with unique structures, for many OSSPs, given their NP-hard nature, using such algorithms to achieve optimal or near-optimal solutions is inefficient. As such, many researchers employed various metaheuristics to solve OSSPs with different complex structures. For instance, Fang and Ross (1994) proposed a hybrid genetic algorithm with simple heuristic scheduling construction rules. They used this algorithm to minimise the completion time. Błażewicz et al. (2004) considered the time lag criterion in the jobs and used a genetic algorithm to solve it. Andresen et al. (2008) proposed an algorithm based on simulated annealing and a genetic algorithm to minimise total weighted tardiness in the open-shop scheduling problem. Tavakkoli-Moghaddam and Seraj (2009) presented a new Tabu search algorithm for the bi-objective OSSP based on a fuzzy multi-objective decision-making approach. They considered deterministic parameters, in which they minimised the mean delay and the mean completion time at the same time. In their proposed model, setup times were deemed to be independent of the sequence. In a study by Zobolas et al. (2009), a hybrid meta-heuristic approach was used for an OSSP. The optimisation scale in this paper was to minimise the completion time. The solution method consisted of three steps:

1 generating a random initial population

2 employing a heuristic solution to obtain the initial population

3 using a combination of the variable neighbourhood search and genetic algorithms.

Doulabi et al. (2010) developed a mixed-integer programming mathematical model for the open-shop scheduling problem with the least total weighted completion time objective function. They solved the problem with a Tabu search algorithm. Naderi et al. (2010) introduced a heuristic algorithm to improve the performance of the solution algorithm. Roshanaei et al. (2010) employed a simulated annealing algorithm to solve the OSSP problem. The ant colony optimisation method for the OSSP problem was introduced by Panahi et al. (2011). Tanimizu et al. (2017) used a co-evolutionary algorithm to obtain disassembly sequences and post-processing operations to solve the open-shop problem. Benziani et al. (2018) presented a genetic algorithm for the open-shop scheduling problem. They used a simple and efficient chromosome based on the job occurrence, in which the fitness function represents the duration of the schedule. They also developed heuristic approaches to generate the initial population and improve the solutions obtained. Recently, Shareh et al. (2021) used the bat algorithm based on a meta-heuristic function to solve OSSPs. Their heuristic function was designed to increase the optimal solution convergence rate. For the OSSP, Kurdi (2022) proposed a new metaheuristic algorithm (ACONEH). Using the proposed heuristic, ACO's exploration capability was enhanced as well as its ability to solve problems effectively. As a result of this approach, ACONEH would avoid premature convergence and maintain an exploration-exploitation balance. As a proportionate case, Adak et al. (2022) studied the case where a task requires a fixed amount of processing time regardless of the job identity. They proposed a model to minimise the makespan and to solve this problem, they used an ant colony optimisation algorithm.

Researchers in the above works considered different processing restrictions. Research on the open-shop scheduling problem with specific setup times is still limited despite its theoretical and practical importance. A recent literature review on setup time and cost scheduling issues highlighted this limitation (Allahverdi, 2015) and open-shop scheduling problems. For example, Abreu et al. (2020) proposed a hybrid genetic algorithm for OSSPs with SDST to minimise the time completion of the job. They also used two new constructive heuristic methods to produce the initial population. Zhuang et al. (2019) introduced a mixed-integer linear programming (MILP) model for an OSSP with two SDSTs and transportation time) to get closer to real-world industrial environments. They then introduced an improved artificial bee colony algorithm to solve the problem. Noori-Darvish et al. (2012) considered OSSP with SDSTs, fuzzy processing times, and fuzzy due dates.

Mosheiov and Oron (2008) discussed OSSP involving m machines and n jobs. They assumed the processing times would be similar, and set-up times would depend on the sequence based on the assumptions. Also, in the scenario they studied, the groups were ready to process at any time. Their objective function was to reduce completion and flow times as much as possible. They proposed a solution algorithm using a time complexity of o(n). Roshanaei et al. (2010) investigated a non-pre-emptive open-shop scheduling problem with SDSTs that minimised the completion time. To solve the problem, the researchers developed two new meta-heuristic algorithms called multi-neighbourhood search simulated annealing and hybrid simulated annealing.

Additionally, they used two constructive heuristics methods called the longest total processing time (LTMPT) and the longest total remaining processing time (LTMPT and LTRMPT). Cankaya et al. (2019) proposed a mixed-integer programming model and limited programming model for OSSPs with sequence-dependent post-setup times to minimise (C_{max}). The integer programming model performed better for short-term decisions based on their computational results, while the limited programming model performed better for long-term decisions. Naderi et al. (2011) proposed an integer linear programming model for OSSP with SDSTs to minimise the makespan. Due to the complexity of the case study, the authors used the electromagnetic algorithm (EH). Based on their computational results, EH and the proposed model performed better than the other algorithms. Low and Yeh (2009) approached OSSP as a binary integer programming model, minimising total job tardiness by considering independent setup times and dependent removal times. Also, they proposed some hybrid genetic-based heuristics to solve the problem in a logical computation time. Behnamian et al. (2021) presented a bi-objective flexible open shop scheduling method with independent setup time. The goal was to minimise the maximum completion time of jobs and total tardiness. Using mixed integers, they developed a nonlinear programming model. Then, the weighted Lp-metric method was used to address the multi-objective problem. They also developed a scatter search algorithm to achieve near-optimal solutions.

Moradi and Yazdini (2021) proposed a mixed-integer programming model for the bi-objective OSSP with limited human and machine dual resources. They employed two Pareto-based meta-heuristic algorithms, namely non-dominated sorting genetic algorithm II (NDSGAII) and multi-objective vibration damping optimisation (MOVDO), to solve this problem.

Pastore et al. (2022) addressed a SDST scheduling problem. Using a mixed integer linear programming (MILP) model, they developed a novel heuristic approach. To find the exact solution of the model, the solution approach alternates between continuous relaxation and rounding off a set of variables (the sequence variables). As a local search phase in open shop scheduling with non-anticipatory SDST, Abreu and Nagano (2022) hybridised an adaptive large neighbourhood search (ALNS) with constraint programming (CP). Their objective function was makespan minimisation. They proposed a non-anticipatory CP model based on the classic open shop model and to obtain reliable solutions in time, they tested many approximations and exact algorithms due to the NP-hardness of the problem.

The review concerning open-shop scheduling problems made by Ahmadian et al. (2021) shows that there is no work in the literature on the re-entry open-shop scheduling problems. However, certain products must be remanufactured, such as those in the electronic manufacturing industry. Re-entering a product into a production line is classified as reverse flow in production scheduling. Flow shops and job shops address this problem. For example, Dehghan-Sanej et al. (2021) investigated a job shop scheduling problem with reverse flows. Through robust programming, they were able to account for uncertainties associated with processing time in real-world applications and solve problems using simulation annealing (SA) and discrete harmony search (DHS).

Nevertheless, there is only one study on the reverse flow of the OSSP by Aghighi et al. (2021). They proposed a mathematical model and used the VDO algorithm to solve it.

The above literature review indicates that OSSP with reverse flows and SDSTs, which occurs in many industrial environments, has not yet been studied. This paper aims to minimise completion time and total tardiness by addressing these aspects. Numerous studies discussed above demonstrate how heuristic methods can provide satisfactory solutions to various problems. In contrast, setting appropriate heuristic rules for large-scale problems remains challenging. Alternatively, meta-heuristic algorithms are faster and more accurate. Meta-heuristic algorithms are being developed to solve OSSP problems. While some meta-heuristic algorithms have been applied, others have not (Anand and Panneerselvam, 2015). The MOVDO algorithm is used in this study to solve the mentioned problem, which has been less used in previous research.

The next section considers OSSP with SDST and reverse flow. Then a bi-objective integer programming is proposed to model the problem at hand.

3 Problem statement

An open shop consists of scheduling n jobs on m machines, with each job being processed on each machine according to its objectives. Jobs are assigned to machines in no particular order in this problem. The number of possible solutions to this problem is $(n!)^m$ (Ahmadian et al., 2021).

1 When OSSP involves reverse flow, it involves a set of machines and jobs with assembly and disassembly operations. The disassembly flow is the exact opposite of the assembly flow, and both flows are performed on the same machine. Moreover, these jobs have two operating ranges, which are as follows:

2 The jobs on the machines in the order $M_1, M_2, \ldots, M_m$, respectively are arranged (direct/straight jobs).

3 The jobs on the machines in the opposite direction as $M_m, M_{m-1}, \ldots, M_1$, respectively are arranged (indirect/reverse jobs).

Here, M denotes a machine and $\{1,2,\ldots, m\}$ is the counter. Meanwhile, J represents a job with an index in $\{1,2,\ldots, n\}$. Besides, $E1$ is the set of direct jobs, and $E2$ are the set involving reverse jobs. Moreover, jobs $\{1,2,\ldots,s\}$ are direct jobs and reverse jobs are in the set $\{s + 1, s + 2,\ldots, n\}$. Abdeljaouad et al. (2015) showed that the optimal solution to an open shop problem involving jobs on different machines could be obtained when all direct jobs are prerequisites for reverse jobs on the first machine and all reverse jobs are prerequisites for direct jobs on the last machine. This certainly reduces the solution space for an OSSP with reverse flow.

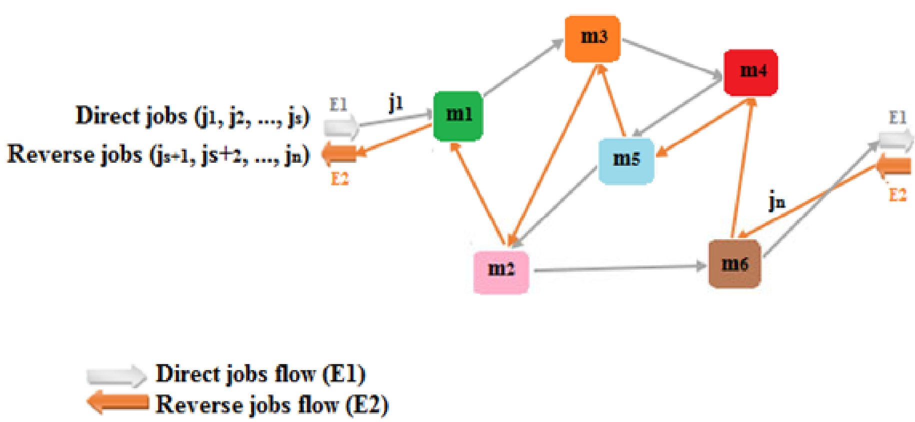
Figure 1 (Aghighi et al., 2021) illustrates OSSP with reverse flows. This figure illustrates a six-machine $\{m_1, m_2,\ldots, m_6\}$ OSSP in which jobs have two flows (direct and reverse), and they are processed on the same machines. According to the rule proposed by Abdeljaouad et al. (2015), in this figure, direct jobs $\{j_1, j_2, \ldots, j_s\}$ are processed on the first machine (m_1) before reverse jobs $\{j_s + 1, j_s + 2, \ldots, j_n\}$, and reverse jobs are processed on the last machine (m_6) before direct jobs. At the same time, intermediary machines perform direct and reverse operations in undetermined sequences and processing routes.

In other words, all direct and reverse jobs on machines are scheduled in an open shop environment. This means there is no predetermined processing route or sequence. It differs from one-flow open shop scheduling in the sense that a direct job is scheduled on the first machine before a reverse job is scheduled on this machine, and a direct job on the first machine is processed earlier than a reverse job. Solving the proposed mathematical model determines the order and sequence of their processing. Also, reverse jobs are scheduled on the last machine before direct jobs, and reverse jobs on the last machine are processed earlier than direct jobs, but by solving the proposed mathematical model, the order and sequence of their processing can be determined. In Figure 1, orange and gray arrows represent the processing route for a direct (j_1) and reverse (j_n) job, respectively. Besides, Figure 2 illustrates the Gantt chart related to Figure 1 with two direct jobs $\{j_1, j_2\}$ and two reverse jobs $\{j_3, j_4\}$.

3.1 Assumptions

Based on the optimality condition proposed by Abdeljaouad et al. (2015), to reduce the solution space, one of the main assumptions of the problem being investigated in this paper is that in the first processing stage, direct jobs precede reverse jobs, and in the last processing stage, reverse jobs precede direct jobs (on each machine).

Figure 1 Overview of job processing route with two opposite flows on the same machines in an open shop environment (see online version for colours)



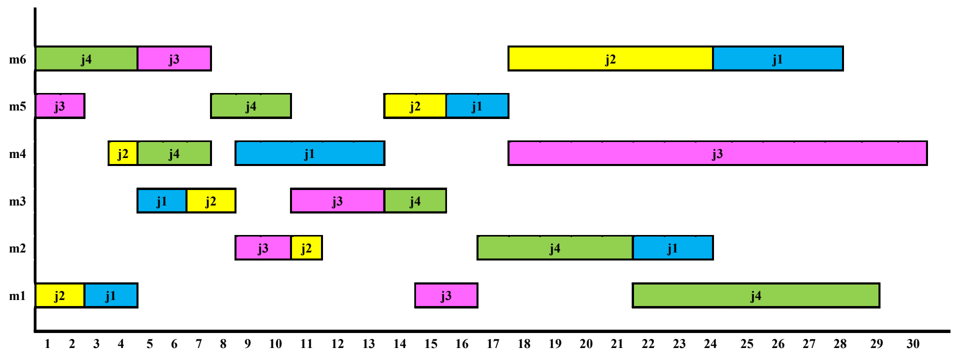
Source: Aghighi et al. (2021)

Furthermore, each job has a specific due date and must be processed by all machines. In this case, jobs can be processed on intermediate machines in any order (except the first and last machines).

However, only one job can be processed on one machine at a time, and each machine operation can only be performed once at a time. However, processing times may vary depending on the machine. The work environment has only one type of machine, and machine preemption and breakdown are not permitted. All machines are available from the start. Processing times and delivery times are deterministic (rather than uncertain). Jobs have reverse flow, and the setup times are sequence-dependent.

In the next section, MILP is utilised to develop the OSSP to minimise both total tardiness and completion time (makespan). Each open-shop production schedule should include two decisions in the decision variables introduced to model the problem. Sequencing the jobs on each machine and sequencing the machines for each job are two of these decisions.

Figure 2 Gantt chart example of Figure 1 (see online version for colours)



3.2 *Mathematical modelling*

In this section, a MILP model is developed to formulate the problem. In many manufacturing workshops, preparation must be done before a job can be handled on a machine. Depending on their order, there may be some setup required between two consecutive jobs on the same machine. These operations are heavily dependent on what preceded them on a single machine. This is known as SDST (Naderi et al., 2011).

This research aims to develop a mathematical model for open-shop scheduling problems with reverse flow and SDST restrictions (based on the model presented in Tavakkoli-Moghaddam and Seraj, 2009)). In this model, reverse flow is applied based on the principle that all direct jobs on the first machine are prerequisites to reverse jobs. In addition, all reverse jobs on the last machine are prerequisites to direct jobs. For this purpose, the indices, sets, parameters, and decision variables are first defined.

3.2.1 *Indices and sets*

i, j, g, j': indices for jobs; $i, j, g, j' \in \{1, 2, \ldots, n\}$ and n is the number of jobs.

$E_1 = \{J_1, J_2, \ldots, J_s\}$ is the set of direct jobs

$E_2 = \{J_{s+1}, J_{s+2}, \ldots, J_n\}$ is the set of reverse jobs

k, l: indices for machines; $k, l \in \{1, 2, \ldots, m\}$ and m is the number of machines.

3.2.2 *Parameters*

m The number of machines

n The number of jobs

p_{ik} The processing time of the job J_i on machine M_k

d_i The due date of the job J_i

se_{jik} The setup time of the machine M_k for job J_i, if the job J_j precedes job J_i on machine M_k

M A large positive number

3.2.3 Variables

T_i Tardiness of job J_i

st_{ik} The starting time of job J_i on machine M_k

c_{ik} The completion time of job J_i on machine M_k

c_{max} The makespan

Y_{ikl} f job J_i on machine M_k precedes the same job on the machine M_l, then $Y_{ikl} = 1$; otherwise $Y_{ikl} = 0$

X_{jik} If job J_j precedes job J_i on machine M_k, then $X_{jik} = 1$; otherwise $X_{jik} = 0$

3.2.4 Problem formulation

The mathematical model of the problem mentioned in this paper is as follows:

$$\min\ z = \left(c_{max}, \sum_i T_i\right) \tag{1}$$

Subject to:

$$st_{ik} + p_{ik} - d_i \le T_i; \qquad \forall i,k \tag{2}$$

$$st_{ik} + p_{ik} \le c_{ik}; \qquad \forall i,k \tag{3}$$

$$st_{i'1} \ge c_{i1}; \qquad \forall i \in E1, i' \in E2 \tag{4}$$

$$st_{im} \ge c_{i'm}; \qquad \forall i \in E1, i' \in E2 \tag{5}$$

$$st_{ik} + p_{ik} + \sum_{j \ne i} se_{jik} x_{jik} - M\left(1 - Y_{ikl}\right) \le st_{il}; \qquad \forall i,k,l \tag{6}$$

$$st_{il} + p_{il} + \sum_{j \ne i} se_{jil} x_{jil} - MY_{ikl} \le st_{ik}\ ; \qquad \forall i,k,l \tag{7}$$

$$st_{ik} + p_{ik} + \sum_{j' \ne i} s_{ej'ik} x_{j'ik} - M\left(1 - x_{ijk}\right) \le st_{jk}; \qquad \forall i,j,k,j' \tag{8}$$

$$st_{jk} + p_{jk} + \sum_{g \ne j} se_{gjk} x_{gjk} - Mx_{ijk} \le st_{ik}; \qquad \forall i, j, k, g \tag{9}$$

$$x_{ijk} + x_{jik} = 1 \qquad \forall i,j,k \tag{10}$$

$$y_{ikl} + y_{ilk} = 1 \qquad \forall i,k,l \tag{11}$$

$$c_{max} \ge c_{ik} \qquad \forall i,k \tag{12}$$

$$c_{max}, c_{ik}, st_{ik}, T_i \geq 0. x_{ijk}, y_{ikl} = \{0,1\} \qquad \forall i,k,l,j \qquad (13)$$

Equation (1) describes the objective functions of the problem for minimising the maximum completion time and minimising the total tardiness. According to equation (2), tardiness at each job is calculated as follows: $T_i = max\{0, max\ \{C_{ik}\} - d_i\}$; $i = 1, 2, \ldots, n$; $k = 1,2, \ldots, m$ (Tavakkoli-Moghaddam and Seraj, 2009). Equation (3) considers the relationship between starting, processing, and completion times. Equations (4) and (5) establish the relationship between a direct and reverse job on the first and last machine in the shop where similar machines are used for direct and reverse jobs. Assuming the disassembly flow is the opposite of the assembly flow, equation (4) ensures that all direct jobs are processed on the first machine before reverse jobs (the time to complete a direct job on this machine is longer than the time to start a reverse job on this machine). In addition, equation (5) ensures that all reverse jobs on the last machine are processed before direct jobs (the time to complete the reverse jobs on this machine exceeds the start time of direct jobs). According to equations (6) and (7), the processing route of direct and reverse jobs on intermediate machines is determined by the SDST and the job starting time. Equations (8) and (9) illustrate the sequence of direct and reverse jobs on each intermediate machine based on the sequence-dependent setup and job start times. In other words, equation (6) requires the completion time of job i on machine k to be less than the start time of job i on the next machine l, while constraint 7 requires the completion time of job i on machine l to be greater than the starting time of job i on the previous machine k. Furthermore, equation (8) requires that the completion time of job i on machine k should be less than the start time of the next job on machine k and according to equation (9), the completion time of job j on machine k should be longer than the time it takes to start the job on this machine. The order in which two jobs are processed on a machine is determined by equation (10). Equation (11) specifies the order between two consecutive operations of a job. Equation (12) calculates the maximum completion time, and equation (13) indicates the non-negativity and integral conditions of the variables used.

4 Problem-solving method

The optimisation field requires practical methods for solving problems after presenting the model. Researchers are increasingly interested in combinatorial optimisation problems today. Numerous optimisation methods such as linear, nonlinear, and dynamic programming have been applied as exact methods, and various heuristic methods have also been proposed to solve them. The computation time of exact methods is usually very long. With the increasing complexity of the problem, it becomes impossible to solve this type of problem.

The OSSP with reverse flows is an NP-hard problem with a high complexity (Dondo and Méndez, 2016). It is challenging to provide an accurate way to optimise the problem in a reasonable time. In this research, a multi-objective vibrating damping optimisation (MOVDO) meta-heuristic algorithm is used to solve the OSSP with reverse flows in medium and large sizes when the setup times are sequence-dependent. The results obtained from this algorithm are compared with the results obtained from other competing meta-heuristic algorithms, including MO-Cuckoo search, MOACO, and NSGA-II. The Taguchi approach is used in the design of experiments to calibrate the parameters of all the solution algorithms. The algorithms are compared using five indicators, including the number of Pareto solutions (NPS), mean ideal distance (MID), diversification metric (DM), computation time, and spacing.

4.1 Multi-objective vibration-damping optimisation algorithm

Mehdizadeh and Tavakkoli-Moghaddam (2009) proposed a vibration-damping optimisation (VDO) algorithm. The idea was inspired by the damping of oscillation amplitude in vibration theory. In damping, the oscillation amplitude is reduced over time until it tends to zero. The method starts from a random initial solution (in the initial domain). The new solution is generated randomly and compared to the previous solution using a neighbourhood structure. When the new solution is worse, it is accepted by the Rayleigh probability distribution (Rayleigh's probability distribution allows the system to escape the local solution). When it is better, the new solution is selected as acceptable. This process continues until the stop condition is reached.

A multi-objective version of VDO referred to as MOVDO, was proposed by Hajipour et al. (2014) to solve multi-objective optimisation problems. It is based on two concepts;

1 fast non-dominated sorting (FNDS)

2 crowding distance (CD).

The algorithm begins by identifying all non-dominated primary chromosomes and then selects them based on the concept of dominance. To find successive layers of non-dominated chromosomes, one temporarily ignores the solutions for the previous layer until all chromosomes are layered. Finally, the tournament method is used to find the next generation of solutions. This method involves selecting *n* initial populations at random. The non-dominated solutions are ranked. The parameter of solutions with the same rank is determined. Lower-ranked solutions are selected. The CD method is used to select the solution with the highest CD among those with the same rank. In terms of elitism, one selects the n population of new generations from the obtained population and then continues until the stop condition for this operation is reached.

A MOVDO algorithm includes the following parameters: number of iterations, population size, primary domain (A_0), maximum number of iterations per domain (L), damping coefficient (γ), and standard deviation. An algorithm's quality depends greatly on the number of iterations and population size. The solution time is reduced when these parameters are lower. Nevertheless, low-quality solutions can be obtained. Additionally, high values can improve solutions but will take more time. To accept worse solutions, the initial domain and damping coefficient are important parameters. The probability that the worse solution will be accepted is as follows:

$$1 - \exp\left(-\frac{A^2}{2\sigma^2}\right) \tag{14}$$

where A is obtained by equation (15) in each iteration's domain.

$$A = A_0 \exp\left(\frac{-\gamma t}{2}\right) \tag{15}$$

Clearly, in lower iterations, there is a higher domain value, and that increases the probability that the worse solutions will be accepted. The probability of accepting the worst solutions increases as the number of iterations increases. Each iteration's domain value is directly affected by the damping coefficient parameter. It decreases the domain and makes it more likely that worse solutions will be accepted. With further iterations, this probability decreases, but initially, the probability of accepting a worse solution is high. As a result, initial iterations search a wide space, and final iterations can find a solution that is congruent with the initial one. Rather than getting rid of near-optimal solutions (by accepting the worst possible ones), the value of these solutions should be improved. In contrast, this incrementing in each domain lengthens the search time (Yazdi and Moghaddam, 2018).

4.2 Pseudo-code

The pseudo-code of the MOVDO algorithm for further clarification of its solution process is shown in Figure 3.

Figure 3 Pseudo-code of MOVDO algorithm

```
Start
Setting values for maximum iteration (MaxIt), Initial Population (nPop), Initial Domain (A0), maximum iteration in each
  domain (L), damping coefficient (γ) and standard deviation (σ)
Generating initial population P and t=1
Assessing initial population
Performing non-dominant sorting (FNDS) and calculating ranks
Calculating the crowding distance (CD)
Sorting the population based on the CD and ranks
1. Do while the stopping condition is not met
 2. Repeat for each particle (X ∈ P)
        3. Let l=1 and Do
        4. Create a neighbourhood (Y) and assess it
                5. If Y dominates X, let X=Y; otherwise, go to the next step
                        6. Randomly select a number from (0, 1), if it is less than a specific number, let X=Y; otherwise,
                          go to next step
                7. If l=L then go to next step; otherwise, l=l+1 and go to Step 4
 8. Performing the FNDS and calculating ranks
 9. Calculating the CD
 10. Sorting the population based on the CD and ranks
 11. Updating the domain and t=t+1
12. If t=MaxIt, then go to the next step; otherwise, go to Step 2
Show the first Pareto frontier
End
```

Source: Yazdi and Moghaddam (2018)

4.3 Main steps in the MOVDO algorithm

The previous section described the general steps of the MOVDO. Below is how the bi-objective problem stated in this article is solved using this algorithm. A description of the solution display method is first provided to achieve this. After that, the procedure for obtaining the initial population and the neighbourhood solutions are discussed. Finally, we discuss the process of stopping the mentioned algorithm following the description of the fitness function.

4.3.1 Solution representation

In the MOVDO algorithm proposed in this research, to determine the assignment of jobs to machines, the solutions are considered an $n \times m$ matrix (n is the total number of jobs and m is the total number of machines). The rows indicate the assignment of each job to the machines. Therefore, the first row represents the first job, the second row represents the second job, etc., and the last row represents the nth job on each machine. Table 1 shows the solution for the processing routes of all jobs (direct $\{j_1, j_2, j_3\}$ and reverse jobs $\{j_4, j_5\}$). For example, the number 4 in the second position in row three indicates that job 3 is processed on the first machine and then on the fourth machine. In other words, in Table 1, the order of processing job 3 on machines is M_1-M_4-M_3-M_2-M_5. On the other hand, to sequence jobs on machines, the solutions are displayed as a matrix $m \times n$, which has m rows for the total number of machines and n columns for the total number of jobs. Accordingly, the rows represent the sequence of jobs on each machine, i.e., the first row represents the sequence of jobs on the first machine, the second row represents the sequence of jobs on the second machine, etc., and the last row also represents the sequence of jobs on the last machine. In Table 2, each job on the machines is shown in sequence; for example, the number 3 in column two indicates that job 3 is processed after job 2 on machine 1. In other words, in Table 2, the order in which jobs are assigned to the machine is J_2-J_3-J_1-J_5-J_4.

Table 1 Order of machines for each job

Jobs			*Machines*		
1	2	1	5	3	4
2	5	2	4	1	3
3	1	4	3	2	5
4	4	3	1	5	2
5	3	5	2	4	1

Table 2 Order of jobs on each machine

Machines			*Jobs*		
1	2	3	1	5	4
2	1	2	4	3	5
3	5	4	2	1	3
4	3	1	5	4	2
5	4	5	3	2	1

4.3.2 Initial solution and neighbouring structure

It is crucial to select appropriate operators to move in the search space and extract and explore better solutions. The neighbourhood structure in the vicinity of the previous solution is used to generate and evaluate a random new solution. If a new solution reduces oscillation energy, it will be accepted as an acceptable solution in the search space, while if it increases the objective function, it will be accepted with the possibility of Rayleigh distribution.

In this paper, the initial solution is generated randomly, and the new solution is generated using the 'adding-or-subtracting-a-small-value' method. As a result of using this method, some values of the previous solution are changed by adding or subtracting small values calculated using equation (16) (Aghighi et al., 2021)

$$d \leftarrow d \pm \frac{(rand().range)}{10}. \tag{16}$$

In equation (16), the function rand () provides a uniformly distributed random number in (0, 1), the range is the range of possible values for the parameter, and 10 (the denominator value) ensures a small change in d (Aghighi et al., 2021).

Figures 4 and 5 show a method of finding a new solution to a problem with six jobs and three machines. In Figure 4, the number 0.97 is randomly selected, from which the value of 0.57 is reduced, and the value of 0.31 is added to the number 0.46. Figure 5 also shows a value of 0.41 is added to 0.11, and 0.2 is subtracted from 0.43. Figure 4 illustrates sequencing on each machine, the previous solution is 5-4-3-6-1-2, and the new solution is 5-4-2-6-1-3, respectively. Figure 5 shows the processing route of each job on the machines. Accordingly, the previous solution is 4-1-3-2, and the new solution is 3-1-4-2.

Figure 4 Neighbourhood structure to find a new solution (sequencing) (see online version for colours)

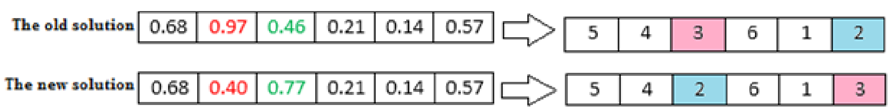
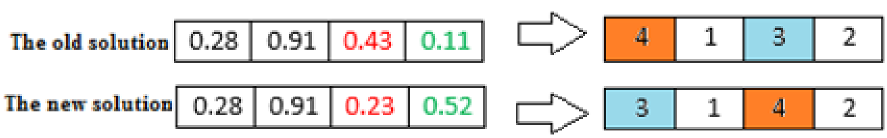


Figure 5 Neighbourhood structure to find a new solution (processing route) (see online version for colours)



In Figures 4 and 5, decimal numbers are numbered from small to large, and these numbers indicate the number of each entry. For instance, in Figure 4, the decimal numbers are arranged from small to large, which is 5, 6, 3, 2, 1, 4. In the new solution vector, the decimal numbers are arranged from small to large, which is 5, 3, 6, 2, 1, 4. According to the number of the entry change, a new solution is derived based on the obtained numbers. In Figure 6, the steps shown as entries 6, 3 change to entries 3, 6.

Figure 6 An example of finding a new solution (sequencing) (see online version for colours)

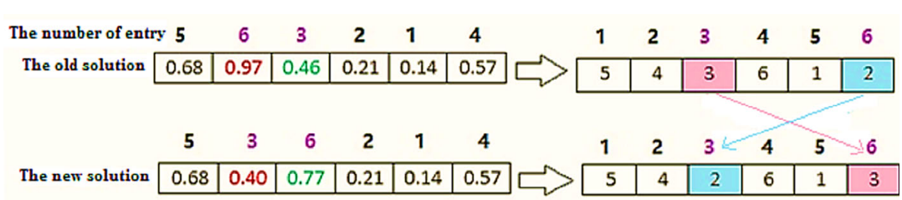


Figure 6 shows the sequence of jobs in the previous solution as 5-4-3-6-1-2, while in the new solution, they are 5-4-2-6-1-3.

4.3.3 Fitness function

The fitness function is only one criterion for guiding the algorithm in searching for suitable solutions. Usually, it is taken directly from the target performance and is defined to evaluate the set of solutions obtained randomly or described using the neighbourhood structure. The fitness value is stored to be used later in the selection method. In this research, fitness is the objective function of the algorithm.

MOVDO determines the order and route of direct and reverse jobs on machines to solve the problem. Based on the order of jobs on the machines according to the processing time, due date, and SDST, C_{max} and total tardiness are calculated. Considering this is a bi-objective open-shop scheduling problem, the Epsilon-constrained method (Bérubé et al., 2009) is utilised to obtain Pareto solutions (best values for objective functions).

4.3.4 Stop criterion

Different criteria can be considered for stopping the algorithm. In this paper, the MOVDO and other comparative algorithms are stopped once the maximum number of iterations has been reached.

4.4 Parameter tuning

Due to the highly dependent nature of the output of meta-heuristic algorithms on the parameters entered, 30 examples of small and large scales are solved in this section using the MOVDO algorithm. Because of its stochastic nature, each instance is tackled 20 times; based on the average solution, Taguchi's experimental design is employed to tune the parameters. An orthogonal array (OA) L9 is used based on the number and level of parameters.

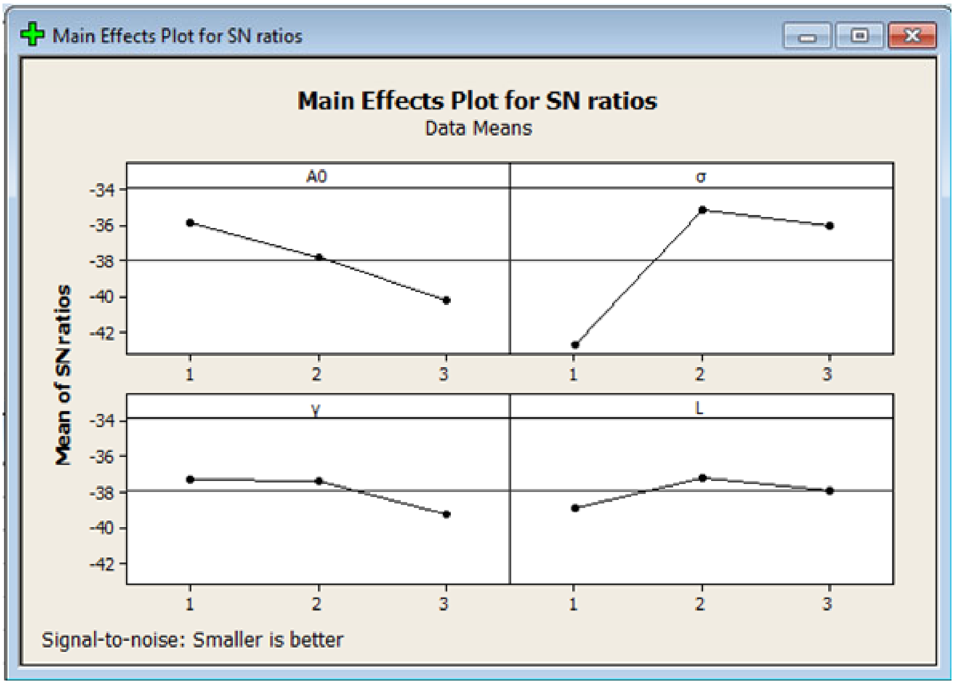
Due to the multi-objective nature of the model, the distance from the ideal point is used to determine the optimal values of the parameters. The input parameters include γ damping coefficient, Rayleigh distribution σ, initial amplitude $A0$, and the maximum number of iterations at each amplitude L. The proposed levels of the MOVDO parameters are shown in Table 3, and the diagrams obtained from the parameter setting results are demonstrated in Figure 7. Statistical computations and diagrams are performed using Minitab 16 software.

Table 3 The investigated levels of MOVDO parameters

	Level 1	*Level 2*	*Level 3*
A0	5	10	20
σ	1	5	10
γ	30	50	100
L	40	60	80

As seen in Figure 7, the parameters σ, γ and L in their second level and parameter $A0$ in its first level determine the best condition. In other words, the best combination of the MOVDO algorithm includes $\sigma = 5$, $\gamma = 50$, $A0 = 5$, and $L = 60$. Also, the results of setting the parameters of the other competing algorithms are as follows. The MOACO algorithm's conversion fitness rate to pheromones = 0.7, and the distance deviation rate = 1. In the NSGA-II algorithm, the percentage of mutation, the crossover percentage, and the tournament size are 2, 0.3, and 0.8, respectively. In the MO-Cuckoo search algorithm, the probability of identifying cuckoo eggs and moving the nest to a new location (P_a) is 0.25, and the step size is 0.01.

Figure 7 Computational results of parameter tuning in S/N ratio plot (see online version for colours)



4.5 *The efficiency of the solution algorithms*

A multi-objective meta-heuristic solution algorithm's two main objectives are convergence to Pareto optimal solutions and providing density and variability to the obtained solutions. Consequently, the multi-objective solution algorithms presented in this paper are compared using some multi-objective performance measures. These measures are the NPS, MID, DM, CPU time (TIME), and spacing metric (SM), one at a time. They are briefly described below.

- NPS index: The NPS found by the algorithm is considered. The higher this measure number is, the better the algorithm's performance.
- MID index: This indicator indicates the Pareto distance from the ideal solution. An ideal solution is the best solution for each objective function. The lower the index value, the better the algorithm performs (Czyżak and Jaszkiewicz, 1998).
- SM index: The distance between the non-dominated solutions is the standard deviation of the index. In other words, it calculates the relative distance between

consecutive Pareto solutions. Generally, the higher the index value, the more efficient the algorithm is Chambari et al. (2012).

- DM index: The Euclidean distance between the initial and final solutions in the Pareto solution set is represented by this metric. This index measures the space cube diameter of the objective's set of non-dominated solutions. The higher the value of this index, the better the algorithm's performance (Zitzler and Thiele, 1998).
- TIME index: This index represents the CPU time an algorithm requires to find a solution. It is one of the most important metrics for comparing algorithms. Lower values indicate better performance.

5 Computational results

In this section, the MOVDO algorithm is compared to an Epsilon-constrained method (Bérubé et al., 2009) to analyse Pareto solutions.

The steps of the ε-constraint method are as follows:

1. Select one of the objective functions as the main objective function
2. Each time according to one of the objective functions, solve the problem and obtain the optimal values of each objective function.
3. Divide the interval between two optimal values of sub-objective functions by a predetermined number and obtain a table of values for $\varepsilon_2, ..., \varepsilon_n$.
4. Each time, solve the problem with the main objective function with each of the values $\varepsilon_2, ..., \varepsilon_n$.
5. Find the Pareto solutions and report them.

This study analyses problems using the Epsilon-constraint method by minimising makespan as the main objective. Two categories of small and medium size problems are solved using MOVDO in MATLAB software version 2016b and the Epsilon-constrained method in GAMS software version 3.1.25. Four to six jobs with two to four machines are considered for small-size problems, and between eight and forty jobs for medium problems are considered with three to eleven machines. Randomly generated problems consist of processing times between [1–13], as well as due date and SDSTs that are randomly generated proportional to the processing time, with uniform distributions. A notebook with five cores and 5.2 GHz and 6 GB of memory is used to solve the problems. In the Epsilon-constraint method, three intervals were considered for each objective function, giving a maximum of 6 Pareto solutions. Table 4 shows the best solution among the Pareto solutions for the Epsilon constraint method and MOVDO algorithm, along with the total time it takes to solve the problem.

Table 4 Comparison between MOVDO algorithm and ε-constraint method results

Problem size	*Problem information*				*ε-constraint method results*			*MOVDO results*		
					Objective function values			*Objective function values*		
	Examples	*Number of direct jobs*	*Number of reverse jobs*	*Number of machines*	*The value of the first objective function*	*The value of the second objective function*	*Time(s)*	*The value of the first objective function*	*The value of the second objective function*	*Time(s)*
Small	1	2	2	2	30	45	0.034	30	45	0.752201
	2	2	2	3	32	62	0.58	32	62	0.865841
	3	2	3	3	41	51	4.82	41	51	4.549987
	4	2	3	4	40	62	141	40	62	4.123658
	5	3	2	4	48	53	359	51	61	3.912548
	6	3	3	3	63	84	20.64	72	93	5.102487
Medium	7	3	5	3	101	201	3,600	112	298	5.774263
	8	4	4	5	102	199	3,600	99	193	6.565658
	9	5	3	5	106	372	3,600	103	194	6.225484
	10	5	5	6	158	237	3,600	166	254	5.154875
	11	6	7	7	259	274	3,600	224	286	6.865154
	12	10	10	8	621	802	3,600	533	608	15.874645
	13	21	14	9	NA	NA	3,600	1,274	1,489	48.256358
	14	24	16	11	NA	NA	3,600	1,457	2,001	51.254585

The results in Table 4 show that medium-sized examples take longer to solve. To solve the problems within a reasonable time limit, the GAMS software is employed for a time limit of 3,600 seconds. There is an average gap of 80% between near-optimal and optimal solutions during this time. As shown in Table 4, GAMS cannot deal with larger problems (problems 13–14) in this timeframe when the problem size increases. In the meantime, the MOVDO algorithm takes longer to solve a problem as its size increases. It can be seen from this table that GAMS software takes longer to solve problems (3–14). However, the proposed algorithm performs similarly to GAMS software in small problems. It is also possible to determine the effects of increasing jobs and machines by considering the time it takes to solve problems.

Table 5 Instance generation

	Problem number	*Problem*	*Number of direct jobs*
Small	1	6 × 2a	j(1, 2)
	2	6 × 2b	j[1–3]
	3	6 × 2c	j[2–4]
	4	6 × 2d	j(2, 3)
	5	5 × 3a	j[1–3]
	6	5 × 3b	j(1, 2)
	7	5 × 3c	j(1, 2)
	8	5 × 3d	j(4, 5)
	9	4 × 4a	j[2–4]
	10	4 × 4b	j[1–3]
	11	4 × 4c	j(1, 2)
	12	4 × 4d	j(3)
Medium and large	13	10 × 6a	j[4–10]
	14	10 × 6b	j[1–5]
	15	10 × 6c	j[1–6]
	16	10 × 6d	j[2–5]
	17	12 × 7a	j[1–10]
	18	12 × 7b	j[1–3]
	19	12 × 7c	j[1–5]
	20	12 × 7d	j[2–10]
	21	15 × 8a	j[1–8]
	22	15 × 8b	j[3–12]
	23	15 × 8c	j[1–9]
	24	15 × 8d	j[5–10]
	25	30 × 9a	j[1–22]
	26	30 × 9b	j[1–24]
	27	30 × 9c	j[1–15]
	28	30 × 9d	j[1–14]

Source: Tavakkoli-Moghaddam and Seraj (2009)

Table 6 The performances of the solution algorithms in solving sample problems

Examples	*Algorithms and indicators*											
	NPS				*SM*				*MID*			
	MOVDO	*MOACO*	*NSGA-II*	*MO-CUCKOO search*	*MOVDO*	*MOACO*	*NSGA-II*	*MO-CUCKOO search*	*MOVDO*	*MOACO*	*NSGA-II*	*MO-CUCKOO search*
1	2	1	1	1	0	NAN	NAN	0	2.0409	3.6826	2.8487	3.6826
2	4	1	1	1	0.92542	NAN	NAN	NAN	2.1359	2.5254	2.5254	2.5254
3	2	2	2	2	0	0	0	0	2.1299	2.1305	2.1251	2.1274
4	3	2	2	2	0	0	0	0	2.2581	2.4147	2.41747	2.4147
5	2	4	2	4	0	0.61865	0	0.62016	2.1457	2.5784	2.5234	2.7554
6	2	3	3	3	0	0.34888	0.34888	0.34888	4.5827	5.5284	5.5284	5.5284
7	3	2	2	2	0.52361	0	0	0	2.875	14.355	14.355	14.225
8	2	2	2	2	0	0	0	0	2.417	1.303	2.303	17.303
9	4	2	1	3	0.64647	0	NAN	0.51464	3.504	4.6789	5.1232	6.48
10	3	4	4	2	0.52212	1.1117	0.70952	0	2.9607	2.2828	3.0139	2.1897
11	4	4	2	4	0.79907	0.66571	0	0.66571	2.4293	2.3615	2.3404	2.3615
12	2	2	1	2	0	0	NAN	0	3.6634	3.3588	2.4841	2.3654
13	4	1	2	4	0.61452	NAN	0	0.64115	3.7451	4.263	9.38	14.457
14	5	4	4	3	0.7547	0.7851	0.2247	0.7481	3.2148	25.526	9.3895	14.447
15	5	2	2	2	0.50131	0	0	0	1.3514	1.3256	1.2474	1.1223
16	9	4	7	4	0.6487	0.66741	0.3347	0.5482	12.5478	2.4312	11.0454	4.3245
17	6	3	4	2	0.93252	0	0.68415	0	14.2574	1.204	19.2354	1.2009
18	2	2	3	4	0	0	0.7458	0.64746	3.4156	2.1354	5.3655	1.1338
19	5	2	3	2	0.5784	0	0.49947	0	1.3573	1.1793	1.203	1.1651
20	4	3	4	3	0.61007	0.6357	0.6252	0.7551	1.0751	1.10478	11.2259	1.1122
21	6	4	1	5	0.5962	0.6251	NAN	0.91648	1.4265	12.1355	1.0245	1.1033
22	6	3	5	2	0.9255	0.6924	0.6127	0	2.4001	1.0142	1.0011	1.1164
23	11	2	9	1	0.9542	0.5263	0.9825	NAN	11.1459	5.2486	1.2458	1.192
24	5	1	2	3	0.2589	NAN	0.2368	0.6625	1.5241	1.7843	3.2789	2.8452
25	7	2	4	4	0.92364	0	0.7142	0.7236	1.0478	1.0408	1.3247	1.2576
26	3	3	1	2	0.49985	0.3666	NAN	0	1.1057	1.1254	1.1243	1.1858
27	9	2	4	7	1.0562	0	0.6395	0.91755	1.1426	1.1926	1.0456	1.0418
28	5	3	4	4	0.77718	0.5243	1.0406	0.8236	1.0426	1.1847	1.0401	1.0257

Table 6 The performances of the solution algorithms in solving sample problems (continued)

Examples	*Algorithms and indicators*							
	DM				*TIME*			
	MOVDO	*MOACO*	*NSGA-II*	*MO-CUCKOO search*	*MOVDO*	*MOACO*	*NSGA-II*	*MO-CUCKOO search*
1	10.69	0	0	2.913	3.095774	3.934631	9.121558	3.784648
2	21.265	0	0	0	14.603801	3.604553	3.065661	7.810854
3	5.7564	5.817	5.817	5.6747	14.385248	9.676872	2.849309	7.624940
4	20.201	2.4484	2.4484	2.4484	15.421235	9.86446	7.852904	8.098721
5	16.709	9.6428	4.1478	9.5399	15.611318	9.936084	7.481646	8.163974
6	7.3491	6.3535	6.3535	6.3535	15.078087	9.990786	7.398535	8.237594
7	5.3315	4.9159	4.9159	5.006	15.357442	10.148029	8.202490	8.327524
8	2.2361	4.5929	4.5265	4.5929	15.350075	10.225025	7.694163	8.157975
9	14.568	5.4833	0	9.8871	15.066909	10.118312	8.099014	8.526214
10	12.188	13.542	16.67	7.5335	15.316498	9.928256	7.593492	8.329530
11	5.5198	7.1117	4.918	7.1117	24.789875	16.885130	12.755909	13.184400
12	5.5327	7.0943	0	7.5753	23.545190	16.215680	12.494537	13.545548
13	20.124	0	5.7979	7.183	28.268854	19.60674	14.905905	19.100264
14	12.116	10.2414	5.9479	25.1245	37.757871	22.618866	18.732249	24.588119
15	10.1114	6.7541	0	16.7895	38.231825	22.244417	18.926514	24.235882
16	16.1236	16.3532	21.471	27.3485	37.506807	22.726411	18.219076	24.098767
17	20.1789	15.446	32.458	20.154	45.160086	17.026985	14.175330	19.726862
18	24.1789	15.326	34.145	48.365	28.241851	17.027874	13.254548	16.856686
19	24.7987	15.109	34.982	22.019	29.131094	16.302430	13.794132	19.050472
20	26.2547	18.495	33.4126	47.101	28.740371	16.146237	13.595797	19.354201
21	26.2566	18.1024	0	131.94	34.875730	18.574237	16.797791	24.705052
22	28.4569	19.4782	37.1221	61.636	34.828312	20.129228	17.099161	25.614596
23	65.2631	25.111	46.2154	0	43.12568	20.178932	17.840231	25.167865
24	73.2548	0	60.015	82.412	51.07529	20.756132	22.089456	57.555123
25	77.124	25.495	64.257	102.357	62.017812	30.814469	28.540486	71.456851
26	84.896	26.546	0	131.785	95.358872	47.223076	44.499012	73.991184
27	183.524	29.692	72.35	140.09	94.966894	47.975479	44.797308	73.874358
28	204.69	33.947	77.234	49.2357	97.089690	47.304027	44.209573	46.734746

Figure 8 The Pareto solutions obtained by the MOVDO algorithm for example 28 (see online version for colours)

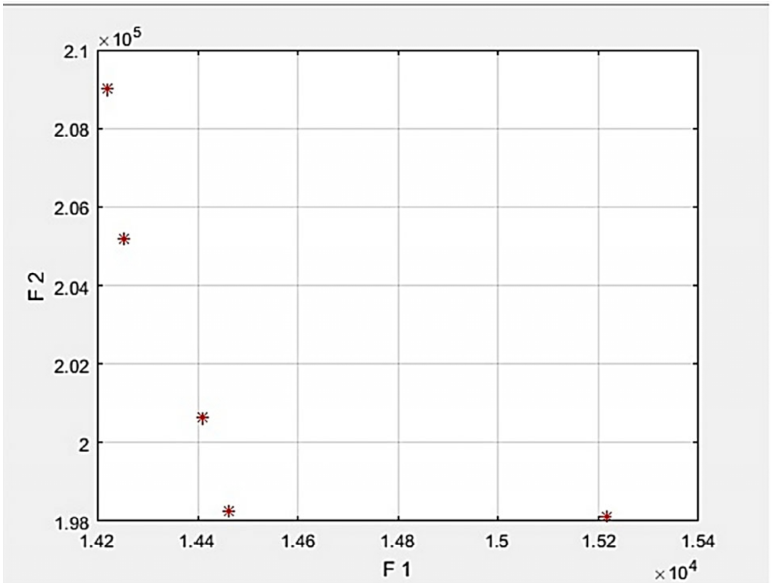
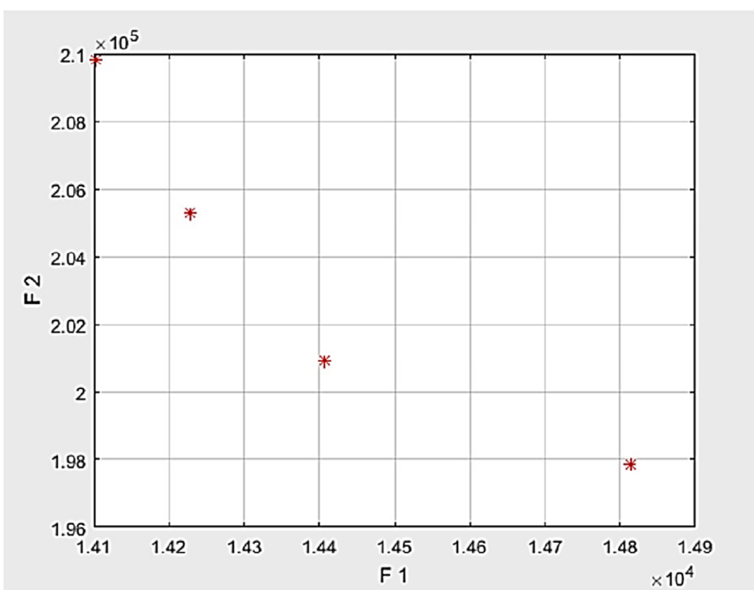


Figure 9 The Pareto solutions obtained by the MO-CUCKOO search algorithm for example 28 (see online version for colours)



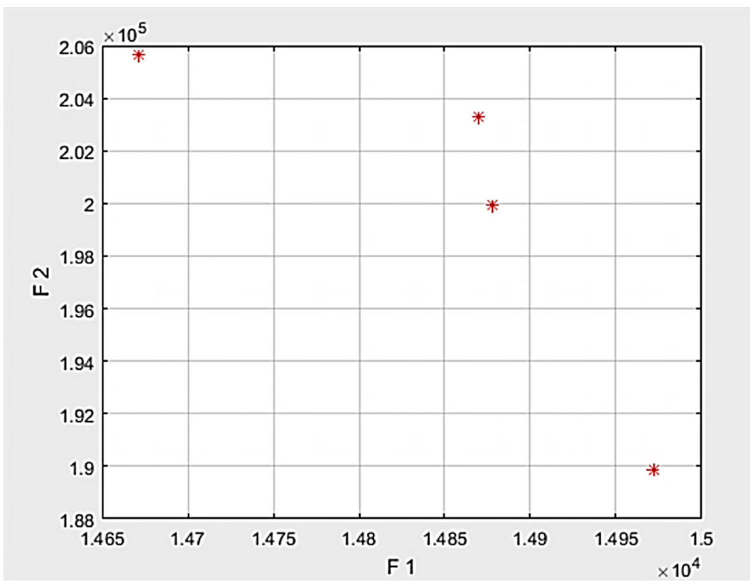
5.1 Evaluating the solution algorithms

Table 5 tests several problems generated in small, medium, and large dimensions. Table 6 gives the averages of the performance metrics for each algorithm after solving each problem 10 times.

5.1.1 Instance generation

In the literature, n jobs and m machines scheduling problems are generated randomly by a classical method. As shown below, this method is implemented.

Figure 10 The Pareto solutions obtained by the NSGA-II algorithm for example 28 (see online version for colours)



The processing times and due dates are uniformly distributed in the intervals [0, 100] and $\left[p\left(1-T-\frac{R}{2}\right), P\left(1-T+\frac{R}{2}\right)\right]$ respectively. The sets {0.2, 0.6, 1.0} and {0.4, 0.6, 0.8} each take into account two parameters, R and T. The mean of total processing times $\overline{p}$ for m machines and *n* jobs scheduling can also be calculated as $P=(m+n-1)\overline{p}$. A small-sized problem can have 4 to 6 jobs and 2 to 4 machines using this instance generation method. Additionally, the number of jobs for medium-sized and large problems can range from 10, 12, 15, and 30, and the number of machines can range from 6 to 9. Whenever a manufacturing environment is specified for these random instances, the manufacturing system is defined as the number of jobs × the number of machines, for example, 15 × 8 indicates an environment with 15 jobs × 8 machines (Tavakkoli-Moghaddam and Seraj, 2009). Due to the lack of set-up times in this method, the duration of sequence-dependent set-up times is defined as a uniform distribution over the range [1, 100], and also for each example, the number of direct and reverse jobs was determined randomly as shown in Table 5. The Pareto diagrams for example 28, (with the results in Table 6) can be found in Figures 8 through 11 using MOVDO, NSGA-II, MOACO, and MO-CUCKOO Search algorithms, respectively.

Based on the results shown in Table 6, the values obtained for each index (NPS, MID, DM, TIME, and SM) are compared in Figures 12 to 16.

Figure 11 The Pareto solutions obtained by the MOACO algorithm for example 28 (see online version for colours)

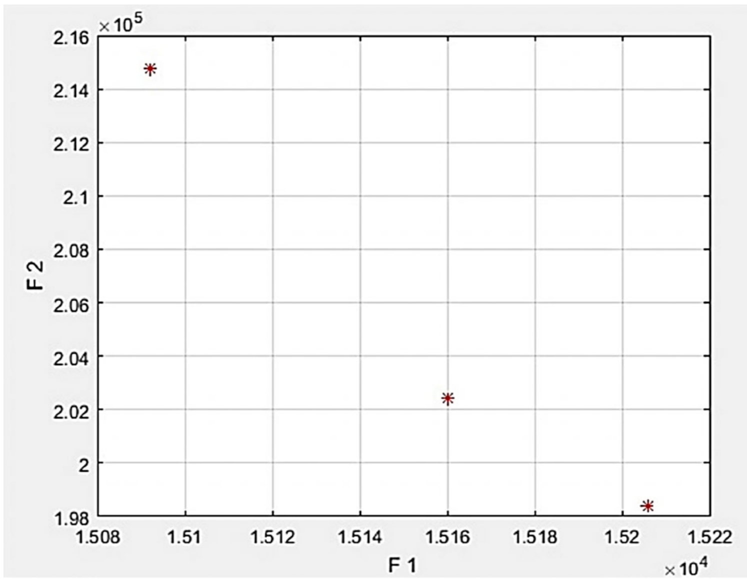
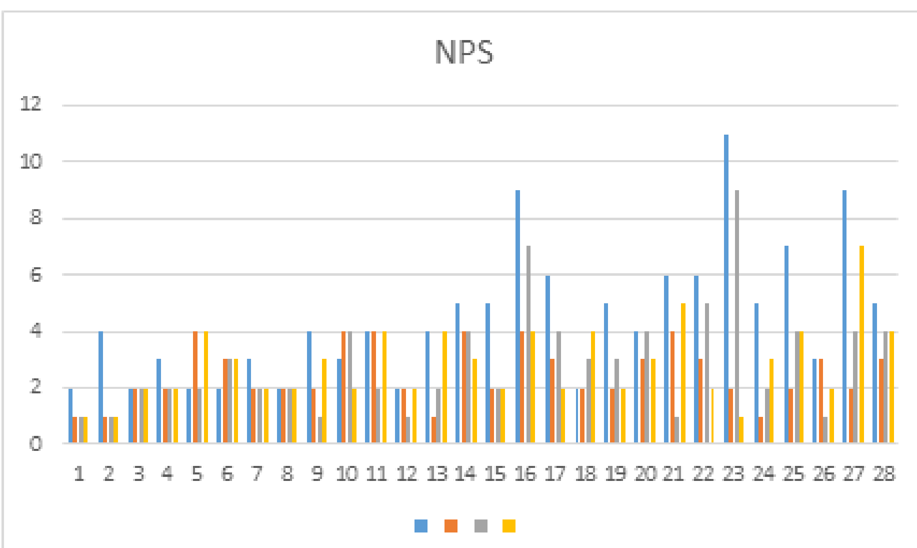


Figure 12 NPS diagram (see online version for colours)



As shown in Figure 12, the MOVDO algorithm performs better in finding more Pareto solutions for most test problems.

As shown in Figure 13, both MOVDO algorithms perform better than the other algorithms in terms of solution uniformity.

Figure 14 shows the performances in terms of the MID index. The lower the index is, the closer the solutions to the ideal point are. As shown in Figure 14 the MOVDO performs well in comparison with the other algorithms discussed in this study.

Figure 13 SM diagram (see online version for colours)

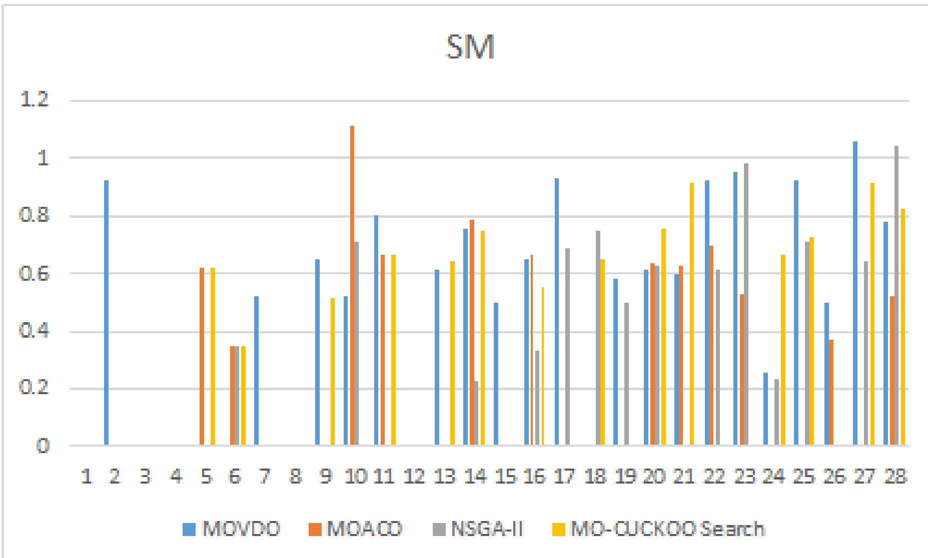
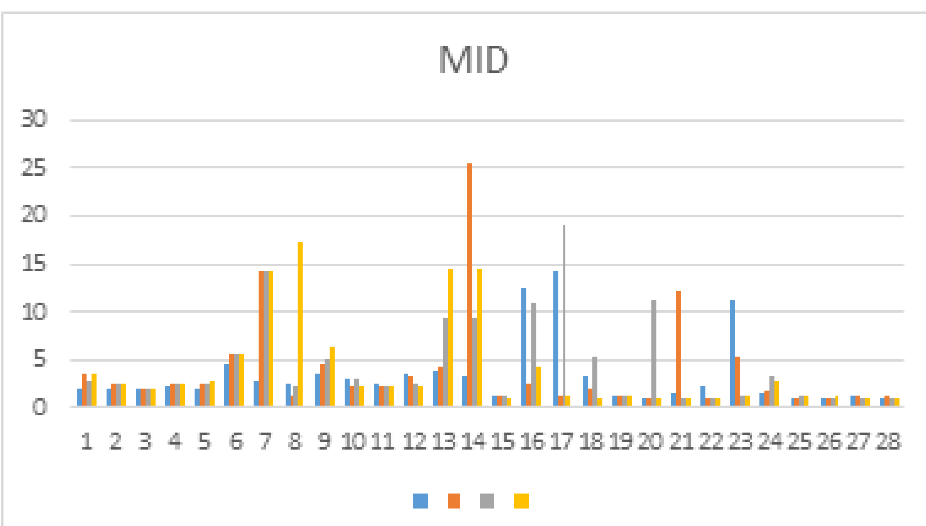
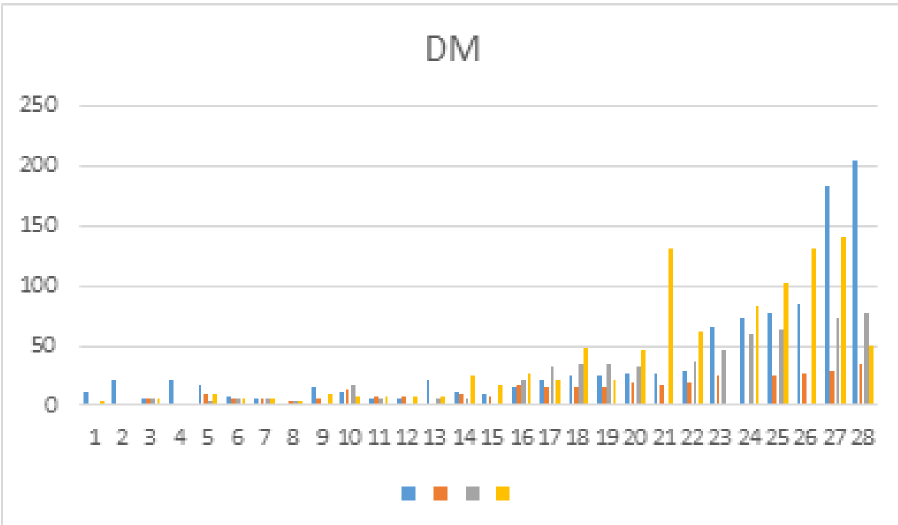


Figure 14 MID diagram (see online version for colours)



As immense diversity is desired for the solutions, Figure 15 shows that MOVDO, MO-Cuckoo search, and NSGA-II algorithms perform better than the MOACO algorithm in terms of DM.

Figure 15 DM diagram (see online version for colours)



The algorithms' solution time increases as the problem size increases, as shown in Figure 16. NSGA-II takes less time than the other algorithms in this diagram.

Figure 16 TIME diagram (see online version for colours)

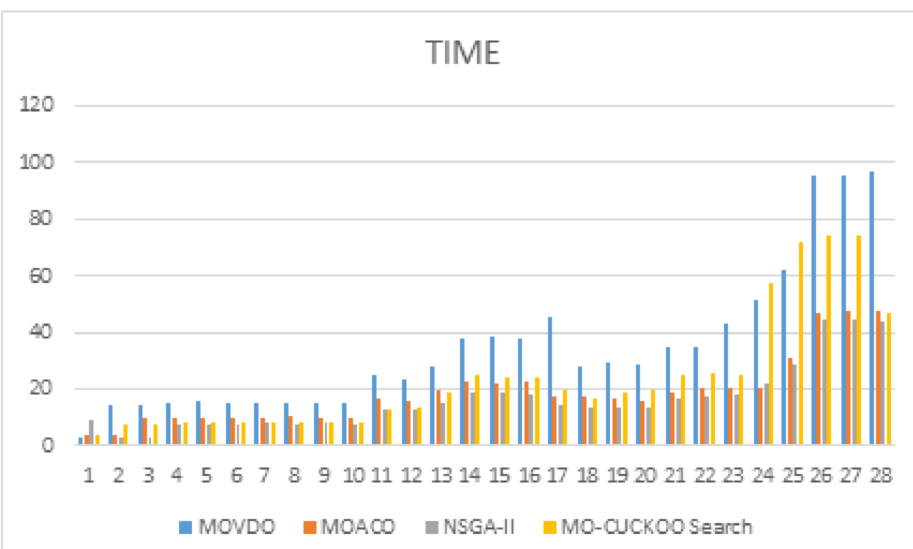


Table 7 Computational results from the case study obtained by MOVDO and GAMS

Problem information							MOVDO results		Real data	
Examples	*Number of direct jobs*	*Number of reverse jobs*	*Number of machines*	*Process time matrix/(s)*	*due date/(s)*	*Solution number*	*Z1*	*Z2*	*Z1*	*Z2*
1	2	2	13	10 200 60 7 7 15 8 14 52 40 30 22 110	j1 = 2730	1	5,035	8,386	13,549	24,687
				10 200 90 7 7 15 8 14 52 40 30 22 110	j2= 4940					
				10 200 90 7 7 15 8 14 52 40 30 22 110	j3= 3459					
				10 200 90 7 7 15 8 14 52 40 30 22 110	j4= 2442					
2	2	3	13	10 200 60 7 7 15 8 14 52 40 30 22 110	j1 = 2730	1	5,168	7,986	17,471	23,655
				10 200 60 7 7 15 8 14 52 40 30 22 110	j2 = 4940					
				10 200 60 7 7 15 8 14 52 40 30 22 110	j3 = 3459					
				10 200 60 7 7 15 8 14 52 40 30 22 60	j4 = 2442					
				10 200 60 7 7 15 8 14 52 40 30 22 60	j5= 3214					
3	3	6	13	10 200 90 7 7 15 8 14 52 40 30 22 60	j1 = 2080	1	11,011	59,770	34,032	73,256
				10 200 90 7 7 15 8 14 52 40 30 22 110	j2 = 5651					
				10 200 90 7 7 15 8 14 52 40 30 22 60	j3 = 3996					
				10 200 90 7 7 15 8 14 52 40 30 22 120	j4 = 2989					
				10 200 90 12 7 15 8 14 52 40 30 22 110	j5= 3626					
				10 200 60 12 7 15 8 14 52 40 30 22 120	j6= 2894	2	12,148	57,482		
				10 200 60 7 7 15 8 14 52 40 30 22 170	j7= 3204					
				10 200 60 7 7 15 8 14 52 40 30 22 170	j8= 4442					
				10 200 60 7 7 15 8 14 52 40 30 22 150	j9= 5723					
4	7	5	13	10 200 90 7 7 15 8 14 52 50 30 22 120	i1 = 2585	1	15,605	122,845	47,121	311,458
				10 200 90 7 7 15 8 14 52 40 30 22 110	i2= 5441					
				10 200 90 7 7 15 8 14 52 40 30 22 110	i3 = 3894					
				10 200 90 7 7 15 8 14 52 40 30 22 120	i4 = 2400					
				10 200 90 12 7 15 8 14 52 40 30 22 120	i5 = 3877					
				10 200 90 12 7 15 8 14 52 40 30 22 120	i6 = 2879					
				10 200 90 7 9 15 8 14 52 40 30 22 120	i7 = 3244					
				10 200 90 7 9 15 8 14 52 40 30 22 170	i8 = 3634	2	14,847	125,482		
				10 200 90 7 9 15 9 14 52 40 30 22 200	i9 = 4517					
				10 200 90 7 9 15 9 14 52 50 30 22 110	i10 = 5066					
				10 200 90 7 7 15 9 14 52 50 30 31 110	i11= 544					
				10 200 90 12 7 15 9 14 52 50 30 31 200	i12= 4585					

Table 7 Computational results from the case study obtained by MOVDO and GAMS (continued)

Problem information							MOVDO results		Real data	
Examples	*Number of direct jobs*	*Number of reverse jobs*	*Number of machines*	*Process time matrix/(s)*	*due date/(s)*	*Solution number*	*Z1*	*Z2*	*Z1*	*Z2*
5	10	8	13	10 200 60 7 7 15 8 14 52 40 30 22 60	i1 = 2585	1	25,914	330,658	84,663	596,100
				10 200 60 7 7 15 8 14 52 40 30 22 110	i2 = 5441					
				10 200 60 7 7 15 8 14 52 40 30 22 110	i3 = 3894					
				10 200 60 7 7 15 8 14 52 40 30 22 120	i4 = 2400					
				10 200 60 7 7 15 8 14 52 40 30 22 120	i5 = 3877					
				10 200 60 7 7 15 8 14 52 40 30 22 120	i6 = 2879					
				10 200 60 7 7 15 8 14 52 40 30 22 150	i7 = 344					
				10 200 60 7 7 15 12 14 52 40 30 22 170	i8 = 3634	2	23,671	34,210		
				10 200 60 7 7 15 12 14 52 40 30 22 200	i9 = 4517					
				10 200 60 7 7 15 12 14 52 40 30 22 200	i10 = 5066					
				10 200 90 12 7 15 12 14 52 40 30 31 110	i11 = 5441					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i12 = 4585					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i13 = 3244					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i14 = 3634	3	25,411	36,214		
				10 200 90 12 7 15 12 14 52 50 30 31 110	i15 = 4517					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i16 = 5066					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i17 = 5441					
				10 200 90 12 7 15 12 14 52 50 30 31 110	i18 = 4585	4	28,014	34,118		

Table 7 Computational results from the case study obtained by MOVDO and GAMS (continued)

Problem information							MOVDO results		Real data	
Examples	*Number of direct jobs*	*Number of reverse jobs*	*Number of machines*	*Process time matrix/(s)*	*due date/(s)*	*Solution number*	*Z1*	*Z2*	*Z1*	*Z2*
6	10	10	13	10 200 60 7 7 15 8 14 52 40 30 22 60	i1 = 2303	1	26,502	371,667	110,035	707,032
				10 200 90 7 7 15 8 14 52 40 30 22 110	i2 = 5169					
				10 200 90 7 7 15 8 14 52 40 30 22 110	i3 = 3142	2	32,154	36,577		
				10 200 90 7 7 15 8 14 52 40 30 22 120	i4 = 2541					
				10 200 60 12 7 15 8 14 52 40 30 22 120	i5 = 3410	3	24,512	31,563		
				10 200 60 12 7 15 8 14 52 40 30 22 120	i6= 5134					
				10 200 60 7 7 15 8 14 52 40 30 22 150	i7= 4849					
				10 200 60 7 7 15 8 14 52 40 30 22 170	i8 = 3440					
				10 200 60 7 7 15 8 14 52 40 30 22 200	i9 = 4576					
				10 200 60 7 7 15 8 14 52 40 30 22 200	i10 = 5999					
				10 200 90 7 7 15 8 14 52 40 30 31 110	i11 = 2771					
				10 200 90 12 7 15 8 14 52 50 30 31 110	i12 = 3731					
				10 200 90 12 7 15 8 14 52 50 30 31 110	i13 = 4608					
				10 200 90 12 7 15 8 14 52 50 30 22 110	i14 = 5187					
				10 200 90 12 7 15 8 14 52 50 30 22 110	i15 = 2180					
				10 200 90 12 7 15 8 14 52 50 30 22 150	i16 = 4790					
				10 200 90 12 7 15 8 14 52 50 30 22 110	i17 = 3200					
				10 200 90 12 7 15 8 14 52 50 30 22 110	i18 = 4246					
				10 200 60 7 7 15 8 14 52 50 30 22 110	i19 = 2868					
				10 200 60 7 7 15 8 14 52 50 30 22 170	i20 = 2979					

In general, based on the above graphical comparison, when each of the four algorithms solved each of the 28 problems of different sizes 10 times, the parameter-tuned MOVDO outperforms each of the other algorithms in terms of four performance metrics.

5.2 A case study

Partial data from the Arman Shahr Atrin Company from the burglar alarm panel production workshop is used to demonstrate the validity and applicability of the proposed model. This workshop presents an OSSP with the reverse flow and SDST. Ten models of alarms are assembled in this workshop according to customer orders. In addition, defective models returned from the market are disassembled to reuse their parts. According to the parts used and their features, there are 10 different models, ranging from SP1 to SP10.In this paper, assembling each model is considered a direct job, whereas disassembling each model is considered a reverse job, and each machine should process all jobs.

For direct jobs, the first machine engraves the logo and product model on the chassis of the alarm panel. For reverse jobs, the previous information is erased or rewritten. The last machine determines the quality of the final or re-entered product. In addition, intermediate machines are used to assemble input parts or disassemble re-entered products.

During two weeks of studies, six assembly and disassembly cases were ordered to the workshop. Table 7 lists the cases that have been solved with the mathematical model proposed in this research. This table considers setup time between 60 and 180 seconds, where all times are expressed in seconds.

It can be seen in Table 7 that the assembly and disassembly times of products are reduced using the suggested model, comparing the results of the proposed model with those observed in the real system. This proves that the proposed model efficiently reduces total tardiness and C_{max}.

6 Conclusion and suggestions for future research

The open-shop scheduling problem under SDSTs was investigated in this study. There was no model for this problem in the literature, so a mathematical formulation was first proposed by considering the objective function of minimising the maximum completion time and minimising the total tardiness of all jobs. Due to the NP-hard nature of the problem, it was solved with the multi-objective MOVDO meta-heuristic. After that, the MO-Cuckoo Search, MOACO, and NSGA-II algorithms were used to compare the results. In terms of four multi-objective performance criteria, the MOVDO algorithm performed better than the other proposed algorithms. Furthermore, the proposed model was implemented in a workshop of Arman Shahr Atrin Company to produce burglar alarm panels. Results obtained from implementing the proposed model show an improvement compared to those obtained from the real world.

We recommend the following for future research: developing a mathematical model of the problem by taking into account other processing constraints such as breakdowns, machine eligibility restrictions, and batch processing. Under uncertainty, we suggest considering other objective functions or using heuristics or meta-heuristics to solve the problem.

References

Abdeljaouad, M.A., Bahroun, Z., Omrane, A. and Fondrevelle, J. (2015) 'Job-shop production scheduling with reverse flows', *European Journal of Operational Research*, Vol. 244, No. 1, pp.117–128.

Abreu, L.R. and Nagano, M.S. (2022) 'A new hybridization of adaptive large neighbourhood search with constraint programming for open shop scheduling with sequence-dependent setup times', *Computers and Industrial Engineering*, 1 June, Vol. 168, p.108128, https://doi.org/10.1016/j.cie.2022.108128.

Abreu, L.R., Cunha, J.O., Prata, B.A. and Framinan, J.M. (2020) 'A genetic algorithm for scheduling open shops with sequence-dependent setup times', *Computers and Operations Research*, 1 January, Vol. 113, p.104793, https://doi.org/10.1016/j.cor.2019.104793.

Abreu, L.R., Nagano, M.S. and Prata, B.A. (2022) 'A new two-stage constraint programming approach for open shop scheduling problem with machine blocking', *International Journal of Production Research*, 15 December, pp.1–20, https://doi.org/10.1080/00207543.2022.2154404.

Adak, Z., Arıoğlu, M.Ö. and Bulkan, S. (2022) 'An ant colony optimization approach for the proportionate multiprocessor open shop', *Journal of Combinatorial Optimization*, Vol. 43, No. 4, pp.785–817.

Aghighi, S., Niaki, S.T.A., Mehdizadeh, E. and Najafi, A.A. (2021) 'Open-shop production scheduling with reverse flows', *Computers and Industrial Engineering*, 1 March, Vol. 153, p.107077, https://doi.org/10.1016/j.cie.2020.107077.

Ahmadian, M.M., Khatami, M., Salehipour, A. and Cheng, T.C.E. (2021) 'Four decades of research on the open-shop scheduling problem to minimize the makespan', *European Journal of Operational Research*, Vol. 295, No. 2, pp.399–426.

Allahverdi, A. (2015) 'The third comprehensive survey on scheduling problems with setup times/costs', *European Journal of Operational Research*, Vol. 246, No. 2, pp.345–378.

Allahverdi, A., Ng, C.T., Cheng, T.C.E. and Kovalyov, M.Y. (2008) 'A survey of scheduling problems with setup times or costs', *European Journal of Operational Research*, Vol. 187, No. 3, pp.985–1032.

Amin, S.H. and Zhang, G. (2013) 'A multi-objective facility location model for closed-loop supply chain network under uncertain demand and return', *Applied Mathematical Modelling*, Vol. 37, No. 6, pp.4165–4176.

Anand, E. and Panneerselvam, R. (2015) 'Literature review of open shop scheduling problems', *Intelligent Information Management*, Vol. 7, No. 1, p.33.

Andresen, M., Br¨asel, H., Plauschin, M. and Werner, F. (2008) 'Using simulated annealing for open shop scheduling with sum criteria', *Simulated Annealing, In-Teh*, pp.49–76 [online] https://www.math.ovgu.de/math_media/Lisa/p08_05.pdf (accessed April 2021).

Bai, D., Zhang, Z-H. and Zhang, Q. (2016) 'Flexible open shop scheduling problem to minimize makespan', *Computers and Operations Research*, 1 March, Vol. 67, pp.207–215, https://doi.org/10.1016/j.cor.2015.10.012.

Baker, K.R. and Trietsch, D. (2009) *Principles of Sequencing and Scheduling*, A John Wiley and Sons. Inc. Publication, SPi Global, Pondicherry, India, Printed in the USA.

Behnamian, J., Memar Dezfooli, S. and Asgari, H. (2021) 'A scatter search algorithm with a novel solution representation for flexible open shop scheduling: a multi-objective optimization', *The Journal of Supercomputing*, Vol. 77, No. 11, pp.13115–13138.

Benziani, Y., Kacem, I. and Laroche, P. (2018) 'Genetic algorithm for open shop scheduling problem', in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT),* IEEE.

Bérubé, J.F., Gendreau, M. and Potvin, J.Y. (2009) 'An exact ε-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits', *European Journal of Operational Research*, 1 April, Vol. 194, No. 1, pp.39–50, https://doi.org/10.1016/j.ejor.2007.12.014.

Błażewicz, J., Pesch, E., Sterna, M. and Werner, F. (2004) 'Open shop scheduling problems with late work criteria', *Discrete Applied Mathematics*, Vol. 134, Nos. 1–3, pp.1–24.

Bräsel, H. and Hennes, H. (2004) 'On the open-shop problem with preemption and minimizing the average completion time', *European Journal of Operational Research*, Vol. 157, No. 3, pp.607–619.

Cankaya, B., Wari, E. and Tokgoz, B.E. (2019) 'Practical approaches to chemical tanker scheduling in ports: a case study on the Port of Houston', *Maritime Economics and Logistics*, Vol. 21, No. 4, pp.559–575.

Chambari, A., Rahmati, S.H.A. and Najafi, A.A. (2012) 'A bi-objective model to optimize reliability and cost of system with a choice of redundancy strategies', *Computers and Industrial Engineering*, Vol. 63, No. 1, pp.109–119.

Chen, R., Huang, W. and Tang, G. (2008) 'Dense open-shop schedules with release times', *Theoretical Computer Science*, Vol. 407, Nos. 1–3, pp.389–399.

Chen, Y., Goebel, R., Lin, G., Su, B. and Zhang, A. (2020) 'Open-shop scheduling for unit jobs under precedence constraints', *Theoretical Computer Science*, 10 January, Vol. 803, pp.144–151, https://doi.org/10.1016/j.tcs.2019.09.046.

Chen, Y., Zhang, A., Chen, G. and Dong, J. (2013) 'Approximation algorithms for parallel open shop scheduling', *Information Processing Letters*, Vol. 113, No. 7, pp.220–224.

Cheng, J., Zhang, G., Li, Z. and Li, Y. (2012) 'Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems', *Soft Computing*, Vol. 16, No. 4, pp.597–614.

Cheng, T.E. and Shakhlevich, N.V. (2005) 'Minimizing non-decreasing separable objective functions for the unit-time open shop scheduling problem', *European Journal of Operational Research*, Vol. 165, No. 2, pp.444–456.

Czyzżak, P. and Jaszkiewicz, A. (1998) 'Pareto simulated annealing–a metaheuristic technique for multiple-objective combinatorial optimization', *Journal of Multi-Criteria Decision Analysis*, Vol. 7, No. 1, pp.34–47.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M. (2002) 'TA fast and elitist multi-objective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp.182–197.

Dehghan-Sanej, K., Eghbali-Zarch, M., Tavakkoli-Moghaddam, R., Sajadi, S.M. and Sadjadi, S.J. (2021) 'Solving a new robust reverse job shop scheduling problem by meta-heuristic algorithms', *Engineering Applications of Artificial Intelligence*, 1 May, Vol. 101, p.104207, https://doi.org/10.1016/j.engappai.2021.104207.

Dolgui, A., Soldek, J. and Zaikin, O. (2006) *Supply Chain Optimisation: Product/Process Design, Facility Location and Flow Control,* Vol. 94, Springer Science and Business Media [online] https://link.springer.com/book/10.1007/b101812.

Dondo, R.G. and Méndez, C.A. (2016) 'Operational planning of forward and reverse logistic activities on multi-echelon supply-chain networks', *Computers and Chemical Engineering,* 8 May, Vol. 88, pp.170–184, https://doi.org/10.1016/j.compchemeng.2016.02.017.

Dong, J., Jin, R., Lin, G., Su, B., Tong, W. and Xu, Y. (2022) 'An efficient polynomial-time approximation scheme for parallel multi-stage open shops', arXiv preprint arXiv:2205.14407.

Doulabi, S.H.H., Jaafari, A.A. and Shirazi, M.A. (2010) 'Minimizing weighted mean flow time in open shop scheduling with time-dependent weights and intermediate storage cost', *Semantic Scholar*, Vol. 2, No. 3, pp.457–460 [online] http://www.enggjournals.com/ijcse/doc/IJCSE10-02-03-11.pdf.

Dror, M. (1992) 'Openshop scheduling with machine dependent processing times', *Discrete Applied Mathematics*, Vol. 39, No. 3, pp.197–205.

Eydi, A., Fazayeli, S. and Ghafouri, H. (2020) 'Multiperiod configuration of forward and reverse integrated supply chain networks through transport mode', *Scientia Iranica. Transaction E, Industrial Engineering*, Vol. 27, No. 2, pp.935–955.

Fang, H. and Ross, P. (1994) 'A promising hybrid GA/heuristic approach for open-shop scheduling problems', in *Proceedings of the 11th European Conference on Artificial Intelligence*, John Wiley and Sons, pp.590–594.

Gonzalez, T. and Sahni, S. (1976) 'Open shop scheduling to minimize finish time', *Journal of the ACM (JACM)*, Vol. 23, No. 4, pp.665–679.

Hajipour, V., Mehdizadeh, E. and Tavakkoli-Moghaddam, R. (2014) 'A novel Pareto-based multi-objective vibration damping optimization algorithm to solve multi-objective optimization problems', *Scientia Iranica. Transaction E, Industrial Engineering*, Vol. 21, No. 6, p.2368.

Khuri, S. and Miryala, S.R. (1999) 'Genetic algorithms for solving open shop scheduling problems', in *Portuguese Conference on Artificial Intelligence*, Springer.

Koulamas, C. and Kyparisis, G.J. (2015) 'The three-machine proportionate open shop and mixed shop minimum makespan problems', *European Journal of Operational Research*, Vol. 243, No. 1, pp.70–74.

Kurdi, M. (2022) 'Ant colony optimization with a new exploratory heuristic information approach for open shop scheduling problem', *Knowledge-Based Systems*, 22 April, Vol. 242, p.108323, https://doi.org/10.1016/j.knosys.2022.108323.

Kyparisis, G.J. and Koulamas, C. (1997) 'Open shop scheduling with maximal machines', *Discrete Applied Mathematics*, Vol. 78, Nos. 1–3, pp.175–187.

Low, C. and Yeh, Y. (2009) 'Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated', *Robotics and Computer-Integrated Manufacturing*, Vol. 25, No. 2, pp.314–322.

Mehdizadeh, E. and Tavakkoli-Moghaddam, R. (2009) 'Vibration damping optimization algorithm for an identical parallel machine scheduling problem', in *Proceeding of the 2nd International Conference of Iranian Operations Research Society*, Babolsar, Iran.

Moradi, F. and Yazdani, M. (2021) 'Modeling and solving multi-objective dual-resource constrained open shop scheduling problem', *Sharif Journal of Industrial Engineering and Management*, Vol. 36, No. 2.1, pp.53–63.

Mosheiov, G. and Oron, D. (2008) 'Open-shop batch scheduling with identical jobs', *European Journal of Operational Research*, Vol. 187, No. 3, pp.1282–1292.

Mosheiov, G. and Yovel, U. (2004) 'Comments on flow shop and open shop scheduling with a critical machine and two operations per job', *European Journal of Operational Research*, Vol. 157, No. 1, pp.257–261.

Naderi, B., Fatemi Ghomi, S.M.T., Aminnayeri, M. and Zandieh, M. (2011) 'Modeling and scheduling open shops with sequence-dependent setup times to minimize total completion time', *The International Journal of Advanced Manufacturing Technology*, Vol. 53, Nos. 5–8, pp.751–760.

Naderi, B., Fatemi Ghomi, S.M.T., Aminnayeri, M. and Zandieh, M. (2010) 'A contribution and new heuristics for open shop scheduling', *Computers and Operations Research*, Vol. 37, No. 1, pp.213–221.

Noori-Darvish, S., Mahdavi, I. and Mahdavi-Amiri, N. (2012) 'A bi-objective possibilistic programming model for open shop scheduling problems with sequence-dependent setup times, fuzzy processing times, and fuzzy due dates', *Applied Soft Computing*, Vol. 12, No. 4, pp.1399–1416.

Panahi, H. and Tavakkoli-Moghaddam, R. (2011) 'Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization', *Expert Systems with Applications*, Vol. 38, No. 3, pp.2817–2822.

Pastore, E., Alfieri, A., Castiglione, C., Nicosia, G. and Salassa, F. (2022) 'A matheuristic approach to the open shop scheduling problem with sequence-dependent setup times', *IFAC-PapersOnLine*, Vol. 55, No. 10, pp.2167–2172.

Roshanaei, V., Esfehani, M.S. and Zandieh, M. (2010) 'Integrating non-preemptive open shops scheduling with sequence-dependent setup times using advanced metaheuristics', *Expert Systems with Applications*, Vol. 37, No. 1, pp.259–266.

Sedeño-Noda, A., Alcaide, D. and González-Martín, C. (2006) 'Network flow approaches to pre-emptive open-shop scheduling problems with time-windows', *European Journal of Operational Research*, Vol. 174, No. 3, pp.1501–1518.

Sedeño-Noda, A., de Pablo, D.A.L. and González-Martín, C. (2009) 'A network flow-based method to solve performance cost and makespan open-shop scheduling problems with time-windows', *European Journal of Operational Research*, Vol. 196, No. 1, pp.140–154.

Shareh, M.B., Bargh, S.H., Hosseinabadi, A.A.R. and Slowik, A. (2021) 'An improved bat optimization algorithm to solve the tasks scheduling problem in open shop', *Neural Computing and Applications*, March, Vol. 33, pp.1559–1573 [online] https://link.springer.com/article/10.1007/s00521-020-05055-7.

Shen, L., Dauzère-Pérès, S. and Neufeld, J.S. (2018) 'Solving the flexible job shop scheduling problem with sequence-dependent setup times', *European Journal of Operational Research*, Vol. 265, No. 2, pp.503–516.

Tanimizu, Y., Sakamoto, M. and Nonomiya, H. (2017) 'A co-evolutionary algorithm for open-shop scheduling with disassembly operations', *Procedia CIRP*, Vol. 63, No. 1, pp.289–294.

Tavakkoli-Moghaddam, R. and Seraj, O. (2009) 'A tabu search method for a new bi-objective open shop scheduling problem by a fuzzy multi-objective decision making approach', *International Journal of Engineering*, Vol. 22, No. 3, pp.269–282.

Tibben-Lembke, R.S. and Rogers, D.S. (2002) 'Differences between forward and reverse logistics in a retail environment', *Supply Chain Management: An International Journal*, 1 December, Vol. 7, No. 5, pp.271–282, https://doi.org/10.1108/13598540210447719.

Yang, X.S. and Deb, S. (2013) 'Multi-objective cuckoo search for design optimization', *Computers and Operations Research*, Vol. 40, No. 6, pp.1616–1624.

Yazdi, M.K. and Moghaddam, R.T. (2018) 'A multi-objective vibration damping meta-heuristic algorithm for multi-objective p-robust supply chain problem with travel time', *Journal of Industrial and Systems Engineering,* (Special issue: *14th International Industrial Engineering Conference*), Vol. 11*,* pp.176–189.

Yuan, Y., Lan, Y., Ding, N. and Han, X. (2022) 'A PTAS for non-resumable open shop scheduling with an availability constraint', *Journal of Combinatorial Optimization*, Vol. 43, No. 2, pp.350–362.

Zhang, P., Bard, J.F., Morrice, D.J. and Koenig, K.M. (2019)' Extended open shop scheduling with resource constraints: appointment scheduling for integrated practice units', *IISE Transactions*, Vol. 51, No. 10, pp.1037–1060.

Zhuang, Z., Huang, Z., Lu, Z., Guo, L., Cao, Q. and Qin, W. (2019) 'An improved artificial bee colony algorithm for solving open shop scheduling problem with two sequence-dependent setup times', *Procedia CIRP*, 1 January, Vol. 83, pp.563–568, https://doi.org/10.1016/j.procir.2019.04.119.

Zitzler, E. and Thiele. L. (1998) 'Multiobjective optimization using evolutionary algorithms–a comparative case study', in *International Conference on Parallel Problem Solving from Nature,* Springer.

Zobolas, G., Tarantilis, C.D. and Ioannou, G. (2009) 'Solving the open shop scheduling problem via a hybrid genetic-variable neighbourhood search algorithm', *Cybernetics and Systems: An International Journal*, Vol. 40, No. 4, pp.259–285.