International Journal of Industrial Engineering Computations 12 (2021) 249–272

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Two meta-heuristic algorithms for optimizing a multi-objective supply chain scheduling problem in an identical parallel machines environment

Nima Farmand[a*], Hamid Zarei[b] and Morteza Rasti-Barzoki[c]

[a]*M.Sc., Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*
[b]*Ph.D. Candidate, Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*
[c]*Associate Professor, Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*

C H R O N I C L E

Article history:
Received January 25 2021
Received in Revised Format March 17 2021
Accepted March 21 2021
Available online March, 21 2021

Keywords:
Multi-objective optimization
Supply chain scheduling
NSGA-II
MOPSO
Supply chain management

A B S T R A C T

Optimizing the trade-off between crucial decisions has been a prominent issue to help decision-makers for synchronizing the production scheduling and distribution planning in supply chain management. In this article, a bi-objective integrated scheduling problem of production and distribution is addressed in a production environment with identical parallel machines. Besides, two objective functions are considered as measures for customer satisfaction and reduction of the manufacturer's costs. The first objective is considered aiming at minimizing the total weighted tardiness and total operation time. The second objective is considered aiming at minimizing the total cost of the company's reputational damage due to the number of tardy orders, total earliness penalty, and total batch delivery costs. First, a mathematical programming model is developed for the problem. Then, two well-known meta-heuristic algorithms are designed to spot near-optimal solutions since the problem is strongly NP-hard. A multi-objective particle swarm optimization (MOPSO) is designed using a mutation function, followed by a non-dominated sorting genetic algorithm (NSGA-II) with a one-point crossover operator and a heuristic mutation operator. The experiments on MOPSO and NSGA-II are carried out on small, medium, and large scale problems. Moreover, the performance of the two algorithms is compared according to some comparing criteria. The computational results reveal that the NSGA-II performs highly better than the MOPSO algorithm in small scale problems. In the case of medium and large scale problems, the efficiency of the MOPSO algorithm was significantly improved. Nevertheless, the NSGA-II performs robustly in the most important criteria.

1. Introduction

It is essential for supply chains that customers and manufacturers work together in a coordinated approach in order to achieve information sharing, communication, and high efficiency. For the sake of these objectives, there should be a quick flow of information between manufacturers, retailers, distribution centers, delivery systems, and customers (Simchi-Levi, 2004). The challenge of the integrated supply chain scheduling at different levels has been one of the most significant issues in Supply Chain Management (SCM) and Operations Research (OR) (Thomas & Griffin, 1996). In classical production environments, schedules and delivery dates are fixed and certain, and the transportation unit is not taken into account. Moreover, decisions on production scheduling and delivery schedules are considered separately (Ho & Chang, 1995). Thus, the Integrated Production and Distribution Scheduling (IPDS) model refers to issues that consider production and distribution scheduling simultaneous and integrated (Chen & Vairaktarakis, 2005; Wang et al., 2014). In IPDS problems concerning the batch delivery system, dispatching jobs should wait until all of them are processed in the same batch. Accordingly, the shipping date of each batch has to be equal to the last completed job, which belongs to the batch, and the number of batches that are sent for the customer affects delivery costs. On the one hand, the waiting time of the jobs for processing the batches may raise the number

* Corresponding author
E-mail: n.farmand@in.iut.ac.ir (N. Farmand)

2021 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.3.002

of tardy jobs, resulting in an increase in tardiness costs. On the other hand, sending several jobs in batches brings about a decline in delivery costs and earliness penalties (Chen, 2010). Therefore, there is a trade-off between the tardiness penalties, the operation time of the entire process (makespan), and the earliness penalties and delivery costs. In this regard, simultaneous optimization of this trade-off can make the problem more complicated, but makes it closer to reality. One of the conventional production systems in the supply chain scheduling problems is the "parallel machines" environment. A parallel production planning problem consists of a set of n jobs which should be processed and scheduled on a set of m parallel machines. Hence, a sequence of processing jobs must be found in order to optimize the total operation time and other supply chain costs using one or more objective functions (Cheng & Sin, 1990; Sivrikaya-Şerifoğlu & Ulusoy, 1999). In this article, the sequential processing of jobs is considered as the production ability, in which each job has only one action that can be processed by each machine. Then, the other jobs from the same batch are sequentially processed; thereby, the batch processing time equals the sum of processing times of all jobs belonging to it.

Wang and Cheng (2000) performed the first research on parallel machines scheduling with batch delivery aiming at minimizing the total operation time and delivery costs. Lenstra et al. (1977) have shown that parallel production scheduling is a strongly NP-hard problem for $m \geq 3$. Furthermore, if the problem is considered with multiple objectives, it will become more complicated (Wang & Liu, 2015). Hence, researchers have begun to develop approximated solution methods, and as a result, several meta-heuristic algorithms have been introduced for solving multi-objective scheduling problems (Coello, 2002; Tyagi & Gupta, 2018). As claimed by Deb (2014), three solution methods, including a priori, a posteriori, and interactive approaches are suggested to solve multi-objective optimization problems (MOPs). According to the second approach, the Pareto curve is created by a set of Pareto solutions. So, decision-makers are able to choose some solutions from the Pareto curve as the optimal solution. In the last decade, researchers have taken into account the second approach in an attempt to resolve multiple objectives problems (Ojstersek et al., 2020). In this study, a bi-objective scheduling for production and distribution will be integrated in a production environment with identical parallel machines. Moreover, minimizing total weighted tardiness and total operation time is taken into account as the first objective function. In addition, minimizing the total weighted number of tardy orders, total earliness penalty, and total batch delivery costs is considered as the second objective function. Then, a bi-objective problem is modeled using a mixed integer linear programming (MILP) approach. Subsequently, two meta-heuristics algorithms are proposed to solve the problem. The first one is the multi-objective particle swarm optimization (MOPSO), and uses a mutation function. The second one is the non-dominated sorting genetic algorithm (NSGA-II), and uses a one-point crossover operator and a heuristic mutation operator. The Taguchi parameter tuning method is applied to adjust the algorithms' parameters. After setting the parameters, the performance of the two algorithms is examined in small, medium, and large scale problems, using five comparing criteria. Finally, the interpretations of the experiments are presented.

The rest of the article is organized as follows: In Section 2, the literature on the IPDS problem is reviewed to identify the research gap and explain the main contributions of this article. In Section 3, the problem definition is presented, and the notations used throughout the paper are given. Section 4 provides the mathematical model for this problem, including the objective functions and constraints, along with their explanations. In Section 5, the details of the MOPSO and the NSGA-II algorithms are explained. In Section 6, experimentation and computational results are provided. The last section (Sect.7) draws conclusions and proposes future research directions.

2. Literature review

A review of the studies on the IPDS problem in the parallel machines environment is provided. Furthermore, the researchers' achievements in the meta-heuristic algorithms in this field, as well as the main contributions of this article, are also presented in this section.

2.1. Integrated production and distribution scheduling

Most studies have focused on the IPDS problem with respect to maximizing customer service levels and minimizing customer delivery time, where delivery costs were not considered. Chen (2010) conducted a proper review of scheduling problems with regard to dispatching. He introduced the symbol $\alpha/\beta/\mu/\delta/\gamma$ to illustrate related problems. Here, α represents the structure of machines, β represents the constraints and specific conditions. Besides, μ denotes the characteristics of the delivery process shown as $v(x, y)$, which x and y represent the number of vehicles and their capacity, respectively. Furthermore, δ indicates the number of customers, and finally, γ denotes the objective function(s). To cite an example, Potts (1980) was among the first to investigate an IPDS problem. This problem involves a type of delivery time regardless of transportation costs, and is in the category of single and immediate delivery. As one of the first studies on batch delivery and transportation costs, Cheng and Kahlbacher (1993) examined the $1//v(\infty,\infty), direct/1/\sum E_j + TC$ problem aiming at minimizing the total weighted earliness and delivery costs, and revealed that it is an ordinarily NP-hard problem. Chang et al. (2014) investigated an IPDS problem in order to minimize total job delivery time and batch delivery costs. Besides, an ant colony optimization (ACO) was designed to solve the problem. Gao et al. (2015) studied the IPDS problem concerning batch delivery with limited vehicle capacity to multiple customers aiming at minimizing the makespan. They also proved that the general version of the IPDS problem is strongly NP-hard. Aminzadegan et al. (2019) investigated the $1/batch/v(\infty,\infty), direct/n/\theta \sum_{b=1}^{N} Y_b + e\sum f_j +$

$\sum W_j T_j + \sum \alpha_j U_j$ problem with a single machine and two types of customers. To this end, they integrated production, distribution, and resource allocation to minimize tardiness, lost sale and batch delivery costs, and resource allocation. Moreover, an ant lion optimization (ALO) algorithm and adaptive genetic algorithm (AGA) were customized to solve the problem. In a make-to-order (MTO) strategy, Liu et al. (2020) considered the IPDS problem with vehicle routing problem as transportation stage aiming at minimizing the total order delivery time. To solve the problem, the variable neighborhood search (VNS) algorithm was applied using ten neighborhood structures and a local search algorithm. As a result, the effectiveness of the algorithm was superior to memetic and large neighborhood search (LNS) algorithms.

The problem of supply chain scheduling in parallel machines environment has been studied by researchers in recent years (Munoz-Villamizar et al., 2019; Nikabadi & Naderi, 2016). Lin and Jeng (2004) examined the $P_m/S_{tb}/v(\infty,\infty), direct/1/Lmax$ and $P_m/S_{tb}/v(\infty,\infty), direct/1/\sum U_j$ problems to minimize tardiness and the total number of late orders. Their computational results indicated that the dynamic programming algorithm was able to solve the small size problems, and problems up to 500 jobs and five machines were solved using heuristic algorithms. Considering the different setup times for unrelated parallel machines, Liu and Lu (2016) investigated the $R_m \to D, h/r-a, v=1, c \geq 1/Dmax$ and $R_m \to D, h/nr-a, v=1, c \geq 1/Dmax$ problems. In their model, orders are processed on parallel machines and then they are delivered to customers. They also considered the availability and unavailability of the machines and interruption of jobs constraints for these problems. Joo and Kim (2017) considered the IPDS problem with sequence-dependent setup time and minimized the makespan to ascertain machine scheduling, batching, and distribution schedule. The problem was solved by several types of the GA framework and the analysis indicated the outperformance of the GA_CRC type algorithm. The work of Ekici et al. (2019) proposed an MTO strategy aiming at minimizing total tardiness and earliness penalties in the attendance of sequence-dependent setup time and the other features of unrelated parallel machines. To address the problem, three heuristic methods and a Tabu search-based meta-heuristics were designed.

In studies carried out on the IPDS problem with identical parallel machines, Shim and Kim (2008) examined an identical parallel machines scheduling problem to minimize total tardiness of orders with the notation of $P_m//V(\infty,\infty), split, direct/1/\sum T_j$. Ullrich (2013) considered the IPDS problem in an identical parallel machines environment with machine ready times, and delivery time windows for the vehicle routing problem. Afterward, a GA was adopted to handle the problem. Schaller (2014) solved the scheduling problem with an objective function in an attempt to entirely eliminate total tardiness in identical parallel machines by taking into account setup times for the machines. By using Tabu-search and GA, and the optimal branch and bound method, he showed that GA performed much better in problem-solving. Chen et al. (2015) studied the $P_m/pmtn/Dmax$ problem, which was a two-stage scheduling problem. That is, the orders are processed on identical parallel machines, and afterward are delivered to customers by a single vehicle. They solved the problem by using an approximate algorithm with a constrained boundary to minimize the maximum delivery time. Moreover, Chen et al. (2016) developed the work of Chen et al. (2015) by considering a set of n due dates assigned to n jobs. Wang et al. (2019) solved an IPDS problem with some characteristics such as machine-dependent ready times, and multi-trip vehicle routing with time windows and uncertain travel times. They proposed a robustness approach to handle uncertain travel times, and a memetic algorithm for minimizing the tardiness and delivery costs. Wang et al. (2019) considered an IPDS problem in the hybrid flow-shop environment including identical parallel machines in three stages. To address the problem, a four-layered constructive (CH_{VNS}), a hybrid heuristic method ($CONS_{VNS}$), and a VNS algorithm were proposed to minimize the delivery completion time.

2.2. Use of the meta-heuristic algorithms

Synchronization of production and distribution scheduling is an NP-hard problem. Accordingly, several meta-heuristics algorithms (i.e., PSO, GA, ACO) have been designed and customized to solve these supply chain scheduling problems. Subsequently, multi-objective meta-heuristics algorithms (i.e., NSGA-II, MOPSO, SPEA-II, MOACO) have also been proposed by researchers to optimize the trade-off between various objectives in supply chain scheduling problems. Cakici et al. (2012) proposed a bi-objective GA with different heuristics to minimize total weighted tardiness and delivery costs for an MTO scheduling problem. Besides, the work of Cakici et al. (2014) solved an IPDS problem in a parallel machines production environment with a single vehicle and multiple customers. Hamidinia et al. (2012) examined the $1//\sum\sum \alpha_{ij}T_{ij} + \beta_{ij}E_{ij} + h_{ij}H_{ij} + D_jY_{jk}$ problem with a single machine for processing jobs and multiple customers. Also, mathematical programming was designed to minimize total earliness, tardiness, holding penalties, and batch delivery cost. Furthermore, they used a GA and integer programming to solve the problem. The findings demonstrated that the proposed GA had high performance. Rajkanth et al. (2017) examined the single machine and parallel machines scheduling problems, in which the parallel machines problem was solved using two models including the jobs sequence and jobs assignment. As a result, an appropriate upper limit was achieved from a GA to assess the efficiency of the meta-heuristic algorithm provided for parallel machine scheduling. Raghavan et al. (2018) examined a scheduling problem for challenges faced by factories that produced certain electronic components, including reworking and re-processing. They showed that, compared to the CPLEX solver and modified shortest total estimated processing time (MSTEPT) algorithm, the modified GA had the best quality solutions in small to medium size problems. However, the MSTEPT outperformed the modified GA in large-scale problems.

Assarzadegan and Rasti-Barzoki (2016) investigated the $1/v(\infty,\infty), direct/k/T\,max + \sum\sum \alpha_{jk} \times max(0, d_{jk} - A_{jk}) + \sum \theta_k B_k$ scheduling problem based on the due date assignment for a single machine and several customers. Then, they solved the problem by using the parallel simulated annealing (PSA) and AGA algorithms. Jiang et al. (2017) examined the $P_m/batch, t_i, p_j, s_j \leq c/Cmax$ problem, which is a scheduling coordination problem on identical parallel machines along with batch delivery. They applied some constraints such as different processing times and sizes, as well as a smaller number of jobs than the capacity of the machine. To solve the problem, they used a combination of the DPSO and GA algorithms. Saeidi (2016) solved a multi-objective scheduling problem on parallel machines with the intention to minimize both the makespan and machine costs by using MOPSO and NSGA-II. By comparing the obtained solutions, the NSGA-II demonstrated a higher performance than the MOPSO algorithm. Hassanzadeh et al. (2016) addressed a bi-objective IPDS problem in the flow-shop environment using the HCMOSPO and HBNSGA-II algorithms. Then they compared the results with the MOPSO algorithm and the NSGA-II. They reported that the developed algorithms had a superior performance than the other algorithms. Sheikh et al. (2018) developed multiple-objectives scheduling problem in a flow-shop environment. Additionally, three objective functions were considered, including minimizing total tardiness, total completion time, and the makespan. Also, the problem was solved using heuristic and meta-heuristics algorithms, i.e., the NSGA-III and the MOPSO algorithm. The results indicated the superiority of the meta-heuristics algorithms, with the NSGA-III exceeding the MOPSO algorithm in one of the three performance criteria. Ganji et al. (2020) applied three meta-heuristic algorithms, including MOACO, NSGA-II, and MOPSO, to address a green multi-objective IPDS problem to minimize customer dissatisfaction, total tardiness penalty, fuel costs, emitted carbon from vehicles, and distribution costs. The outcomes of experiments revealed the outperformance of NSGA-II compared to the other two algorithms. Some studies were performed in the context of multi-objective optimization parallel machines scheduling problems with meta-heuristic algorithms. Guo et al. (2016) studied a bi-objective IPDS problem in an unrelated parallel machines environment. The study aims to minimize the total number of delayed orders and the sum of holding costs, labor costs, tardiness and earliness costs, and batch delivery costs. A bi-level evolutionary optimization (BLEO) algorithm was used to solve the problem. Shahidi-Zadeh et al. (2017) developed a multi-objective problem in the unrelated parallel machines environment, and considered characteristics such as ready jobs, release time, and the batch's capacity constraint. Two objective functions were considered, including minimization of total tardiness, total earliness, and costs of purchasing the machinery, and minimization of the makespan. The results showed a better performance of the bi-objective harmonic search (MOHS) algorithm compared to the other algorithms. Zhou et al. (2018) considered the $P_m/B, r_j/Cmax, TEC$ problem. They defined two objective functions, including minimization of total electricity consumption costs, which is an environmental sustainability indicator, and minimization of the makespan. They solve the large instances by using a multiple-objectives discrete differential evolution algorithm. Afterward, the performance was compared to the NSGA-II and AMGA (the archive-based micro-genetic algorithm). It was revealed that the suggested algorithm had a preferable performance regarding the quality of solutions. Wu and Che (2019) considered a multi-objective energy-efficient scheduling problem for unrelated parallel machines. Afterward, they employed two objective functions in an attempt to minimize the total energy consumption and the makespan. They developed a memetic differential evaluation (MDE) algorithm and showed that the MDE algorithm was considerably superior than the multi-objective SPEA-II and NSGA-II.

2.3. Main contributions

There is a significant research gap according to the literature review. As can be seen from Table 1, optimizing the trade-off between crucial criteria of decision-making in the multi-objective IPDS problem has not been considered, to the best of our knowledge. So, a new bi-objective IPDS problem in the identical parallel machine environment will be optimized to fill the research gap. The first objective is considered as a measure to increase customer satisfaction (minimizing tardiness penalty and makespan). The second one is considered as a measure to reduce some manufacturer's costs such as warehouse cost, increasing the company's reputation, and transportation costs. Accordingly, the main contributions are as follows:

- *Concept*: The idea of integrated production and distribution scheduling is conceptualized so as to help decision-makers for evaluating the trade-off between customer service costs and manufacturer's costs. After that, the idea is operationalized in a problem of supply chain scheduling in an identical parallel machines environment.
- *Model:* A new multi-objective MILP model is developed to assess the trade-off between two crucial objective functions. The first objective minimizes the total weighted lateness and total operation time. The second one minimizes the total earliness penalty, total weighted number of tardy orders, and total batch delivery costs.
- *Solution approach:* Two meta-heuristics algorithms are suggested to address the small, medium, and large scale problems. First, the MOPSO algorithm with a mutation function is proposed, followed by the NSGA-II with a one-point crossover operator and a heuristic mutation operator. Afterward, the algorithms are compared using five comparing criteria. It should be noted that a trade-off between batch delivery costs and the customer's contractual due date is examined in detail as using these algorithms.

Table 1
The literature review of associated researches and this study

Reference	Production system type	Tardiness penalty	Earliness penalty	Reputational damage for tardiness	Total operating time	Batching	Capacity-constrained vehicle	Setup time	MOP	Objective(s)	Solution method(s)
(Chang et al., 2014)	RM	-	-	-	-	✓	✓	-	-	$\sum w_j D_j + T$	ACO_DPA
(Gao et al., 2015)	Single	-	-	-	✓	✓	✓	-	-	C_{max}	HA, DP
(Aminzadegan et al., 2019)	Single	✓	-	✓	-	✓	✓	-	-	$\theta \sum Y_b + e \sum f_j + \sum w_j T_j + \sum \alpha_j U_j$	AGA, ALO, HA
(Liu et al., 2020)	Single	-	-	-	-	✓	✓	-	-	$\sum D_j$	VNS, LNS, MA
(Lin & Jeng, 2004)	PM	✓	-	✓	-	-	-	✓	-	$\sum U_j$ and L_{max}	DP, HA
(Liu & Lu, 2016)	PM	-	-	-	-	✓	✓	-	-	D_{max}	HA
(Joo & Kim, 2017)	RM	-	-	-	✓	✓	✓	✓	-	C_{max}	GA
(Ekici et al., 2019)	RM	✓	✓	-	-	-	-	✓	-	$\sum(E_i + T_i)$	SHA, TSA, RSPA, TSBM
(Ullrich, 2013)	PM	✓	-	-	-	✓	✓	-	-	$\sum T_j$	GA, DA
(Schaller, 2014)	PM	✓	-	-	-	✓	-	✓	-	$\sum T_j$	GA, TS, B&B
(Chen et al., 2015)	PM	-	-	-	✓	-	✓	-	-	D_{max}	GA, APP
(Wang et al., 2019)	PM	✓	-	-	-	✓	✓	-	-	$\sum T_j + D_j$	MA, GA
(Wang et al., 2019)	HFS	-	-	-	-	-	✓	✓	-	D_{max}	VNS, CH_{VNS}, $CONS_{VNS}$
(Cakici et al., 2012)	Single	✓	-	-	-	✓	✓	-	✓	$\sum w_j T_j$, TC	NSGA-II
(Cakici et al., 2014)	PM	-	-	-	-	✓	✓	-	-	$\sum w_j D_j$	HA
(Hamidinia et al., 2012)	Single	✓	✓	-	-	✓	✓	-	-	$\sum \sum \alpha_{ij} T_{ij} + \beta_{ij} E_{ij} + h_{ij} H_{ij} + D_j Y_{jk}$	GA
(Raghavan et al., 2018)	Single, RM	✓	-	-	-	-	-	✓	-	$\sum w_i T_i$	MSTEPT, GA
(Assarzadegan & Rasti-Barzoki, 2016)	Single	✓	-	-	-	✓	✓	-	-	$T_{max} + \sum \sum \alpha_{jk} (d_{jk} - A_{jk}) + \sum \theta_k B_k$	AGA, PSA
(Jiang et al., 2017)	QM	-	-	-	✓	✓	✓	-	-	C_{max}	DPSO-GA
(Saeidi, 2016)	PM	-	-	-	✓	-	-	✓	✓	C_{max}, $\sum C_i$	MOPSO, NSGA-II
(Hassanzadeh et al., 2016)	FS	✓	✓	✓	✓	✓	✓	-	✓	$\sum \alpha_j T_j + C_{max}, \sum(\beta_j E_j + w_j U_j + h_j H_j + \gamma_j I_j) + \sum \theta\, y_b$	HCMOSPO, HBNSGA-II
(Sheikh et al., 2018)	FS	✓	-	-	✓	-	-	✓	✓	C_{max}, TT, TC	MOPSO, NSGA-III
(Ganji et al., 2020)	Single	✓	-	-	-	✓	✓	-	✓	$TC + FC + CC$, $\sum T_j$, CD	MOPSO, NSGA-II, MOACO
(Guo et al., 2016)	RM	✓	✓	✓	-	✓	✓	-	✓	$\sum U_k$, $LC + TC + \sum(R_k + E_k + T_k)$	BLEO, TGA, GAMA, ESMO
(Shahidi-Zadeh et al., 2017)	RM	✓	✓	-	✓	✓	✓	-	✓	C_{max}, $\sum \gamma_i L_i + \sum(\beta_j T_j + \theta_j E_j)$	MOHS, MOPSO, NSGA-II, MOACO
(Zhou et al., 2018)	PM	-	-	-	✓	✓	-	-	✓	$Cmax, TEC$	NSGA-II, AMGA, MODDE
(Wu & Che, 2019)	RM	-	-	-	✓	-	-	-	✓	$Cmax, E$	MDE, SPEA-II, NSGA-II
This Study	PM	✓	✓	✓	✓	✓	✓	✓	✓	$\sum \alpha_i T_i + C_{max}$, $\sum(w_i U_i + \beta_i E_i) + \theta \sum Z_b$	NSGA-II, MOPSO

3. Problem description

In this section, we explain the notations applied in the whole article, followed by describing the details and assumptions of the problem.

Notations

Indices

i	Job (order) index, $i \in \{1, \dots, n\}$
b	Batch index, $b \in \{1, \dots, n\}$
k	Machine index, $k \in \{1, \dots, m\}$

Parameters

n	Total number of jobs (orders)
m	Total number of machines
d_i	Contractual delivery time of job i
w_i	Reputational weight of the job i (The cost of the company's reputational damage for job i)
p_i	Processing time of the job i
S_b	Setup time of sub-batch from batch b
θ	Delivery cost
α_i	The weight of lateness for job i per elongated time unit
β_i	The penalty of earliness for job i
M	A huge positive number

Decision variables

C_{max}	Makespan
C_i	Completion time of the processed job i (delivery time)
$C_{b,k}^{SB}$	Completion time of sub-batch from batch b on the machine k
C_b	Completion time of the batch b
T_i	Lateness of the job i
E_i	Earliness of the job i
Z_b	1 if the batch b is formed and otherwise 0
x_{ibk}	1 if the job i is in the batch b and on the machine k and otherwise 0
a_{bk}	1 if a sub-batch of the batch b is on the machine k and otherwise 0
U_i	1 if the job i is tardy and otherwise 0

Details and assumptions of the problem

This study presents a bi-objective scheduling problem of production and distribution in the production environment with identical parallel machines, as defined by Chen (2010): $P_m/S_b, batch/v(\infty, n), direct/1/\sum_{i=1}^{n} \alpha_i T_i + C_{max}, \sum_{i=1}^{n} w_i U_i + \sum_{i=1}^{n} \beta_i E_i + \theta \sum_{b=1}^{n} Z_b$. It means that there is a processing system for n jobs that includes m parallel machines and the sequence-independent setup time to form each sub-batch of batch b; here, orders are delivered directly in batches to the customer. Finally, two objective functions are addressed. The following assumption are considered for the problem:

- The number of n independent jobs must be processed. Each job is processed at a time on one of the m machines.
- At time zero, all jobs are available.
- The processing time for each job (p_i) is definite and above zero.
- Each machine k can only process one job i at a time and all machines are accessible at all time.

- The preemption of the job is not allowed.
- Setup times are independent of the sequence for forming the sub-batches of each batch on the parallel machines.

Consider a supply chain scheduling problem in an identical parallel machines environment, which includes a production system with m machines and n jobs. A customer orders n jobs to a manufacturer which each job i has a priority weight of w_i, a processing time of p_i, and a contractual due date of d_i. If the job's completion time (C_i) is longer than its contractual due date (d_i), the job is tardy, and a job is early if the job's completion time (C_i) is lower than its contracted due date (d_i). Subsequently, an earliness penalty of β_i for early jobs and a tardiness cost of α_i for tardy jobs are considered. Besides, the number of tardy jobs delivered to the customer has a notable impact on the manufacturer's reputation. Hence, there is another type of tardiness cost called "the cost of the company's reputational damage" that only depends on the number of tardy jobs based on their priority weight (w_i) and not depends on the amount of tardiness.

In the manufacturing unit, jobs are sequentially processed in batches, and each batch is delivered to the customer by a vehicle with the capacity of n. In this problem, when a job is completed in a batch, it waits for all jobs to be completed. Each batch can include one or several sub-batches that are processed on different machines. Afterward, the sequence-independent setup time (S_b) is imposed when a sub-batch on any machine is created from each batch and it is applied before the beginning of the sub-batch processing. After this setup time, jobs in the same sub-batch are sequentially processed. So, the variable $C_{b,k}^{SB}$ is a time when all the sub-batch (SB) jobs of the batch b on the machine k are processed and equals to the last completed job which belongs to the sub-batch of batch b. Moreover, variable C_b is a time that all jobs in batch b are processed and it equals to the latest completion time of the sub-batch $(C_{b,k}^{SB})$. Here, x_{ibk} is a binary variable and if the job i is in the batch b and on machine k, its value is 1, and 0 otherwise. Similarly, in the case of U_i, if the job i is tardy, the U_i value is 1 and otherwise 0. Moreover, in the case of Z_b, if batch b is formed, i.e., it contains the job, its value is 1 and 0 otherwise. The batch delivery date to the customer is the maximum value of the jobs' completion times (C_i) in that batch. So, the maximum completion time of all orders is equal to the total operation time (makespan). Later on, delivery costs can be reduced by batch delivery, in which the delivery system has one or several batches. In this problem, there are sufficient vehicles with the capacity of n, and the delivery cost for each batch is equal to θ. For instance, if the batch delivery cost (θ) is raised by a third-party logistics company, the manufacturer will decide to decrease the dispatching number of batches; thereby, the tardiness penalty can increase, and vice versa. As a result, finding an optimal sequence of jobs processing on the parallel machines and assigning them to the batch are the primary purposes of this IPDS problem in order to optimize the trade-off between the decision variables. Fig. 1 indicates the schematic view of the problem.

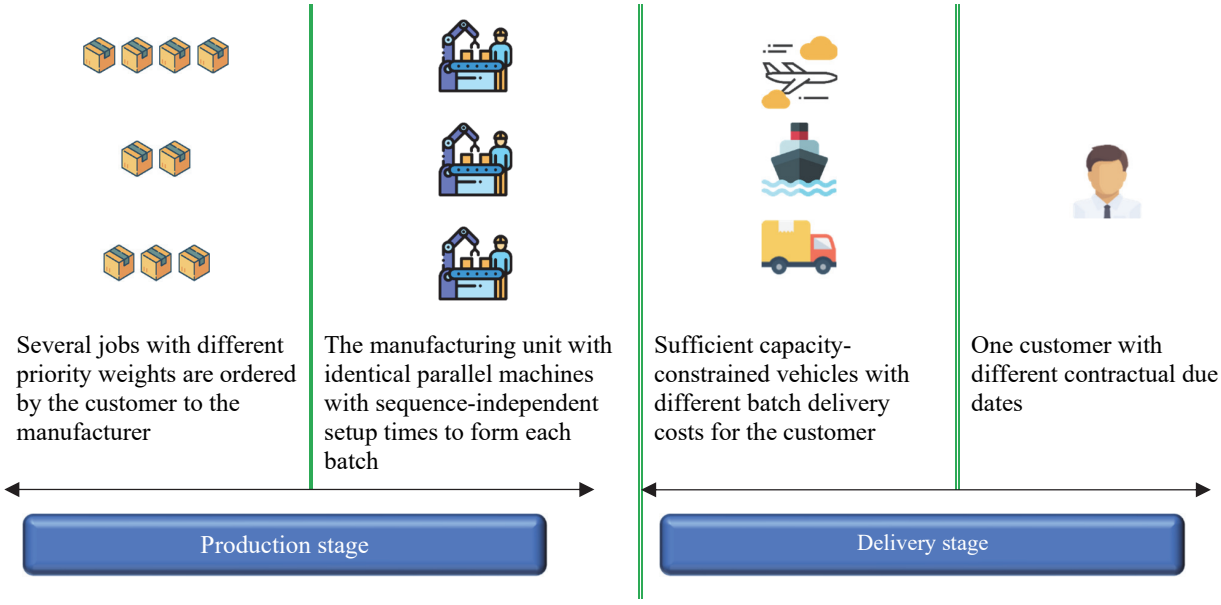


Fig. 1. The problem schematic view

4. Mathematical modeling

In this study, we consider the production environment with identical parallel machines and present a multi-objective model with mixed integer linear programming. Moreover, we attempt to minimize the most important costs based on the completion time of the orders and batch delivery. Hence, we attempt to minimize the two objective functions simultaneously. The objectives are (1) minimizing the total tardiness penalty and makespan, and (2) minimizing the total cost of the company's

reputational damage due to the number of tardy orders, total earliness penalty and total batch delivery costs. The multi-objective problem is as follows:

$$\text{Minimize} \sum_{i=1}^{n} \alpha_i T_i + C_{max} \tag{1}$$

$$\text{Minimize} \sum_{i=1}^{n} w_i U_i + \sum_{i=1}^{n} \beta_i E_i + \theta \sum_{b=1}^{n} Z_b \tag{2}$$

s.t.

$$M(1 - U_i) + T_i \geq C_i - d_i \qquad \forall i \tag{3}$$

$$MU_i + E_i \geq d_i - C_i \qquad \forall i \tag{4}$$

$$M.U_i \geq C_i - d_i \qquad \forall i \tag{5}$$

$$\sum_{k=1}^{m} \sum_{b=1}^{n} x_{ibk} = 1 \qquad \forall i \tag{6}$$

$$M.a_{bk} \geq \sum_{i=1}^{n} x_{ibk} \qquad \forall k, b \tag{7}$$

$$C_{b,k}^{SB} \geq \sum_{i=1}^{n} x_{ibk}.p_i + a_{bk}.S_b + C_{(b-1),k}^{SB} \qquad \forall k, b \tag{8}$$

$$C_{0,k}^{SB} = 0 \qquad \forall k \tag{9}$$

$$C_b \geq C_{b,k}^{SB}.a_{bk} \qquad \forall k, b \tag{10}$$

$$C_i = \sum_{b=1}^{n} \sum_{k=1}^{m} C_b.x_{ibk} \qquad \forall i \tag{11}$$

$$C_{max} \geq C_i \qquad \forall i \tag{12}$$

$$M.Z_b \geq \sum_{k=1}^{m} \sum_{i=1}^{n} x_{ibk} \qquad \forall b \tag{13}$$

$$M(1 - Z_b) \geq -\sum_{k=1}^{m} \sum_{i=1}^{n} x_{ibk} \qquad \forall b \tag{14}$$

$$C_{max}, C_i, C_{b,k}^{SB}, C_b, E_i, T_i, H_{b,k}^{SB}, O_{ibk} \geq 0 \qquad \forall i, k, b \tag{15}$$

$$x_{ibk}, Z_b, a_{bk}, U_i \in \{0,1\} \qquad \forall i, k, b \tag{16}$$

The first objective function, presented in Relation (1), minimizes the total weighted tardiness and total operating time. Tardy jobs are crucial for producer in terms of the level of customer service and customer satisfaction with the timely delivery of orders. Therefore, one of our goals is to minimize total weighted tardiness ($\alpha_i T_i$) and total operation time (C_{max}). The second objective function, presented in Relation (2), minimizes the total cost of the company's reputational damage ($w_i U_i$); if the job is tardy, it imposes a reputation cost on the company based on its weight or priority, total earliness penalty ($\beta_i E_i$) and total batch delivery cost to the customer. Relations (3) and (4) calculate the amount of tardiness and earliness for job i, respectively. Relation (5) determines whether or not job i is tardy. Equation (6) assures that each job is only processed in one batch and on one machine because no preemption is allowed. Relation (7) specifies whether a sub-batch of batch b is located on machine k. Constraint (8) calculates the completion time for a sub-batch of batch b processed on machine k. Equation (9) considers the starting schedule at time zero. The completion time of batch b is calculated in Constraint (10), and the delivery time of job i is calculated in equation (11). Relation (12) calculates the makespan and constraints (13) and (14) guarantee that if the number of members in the batch is zero, it does not form and otherwise, the batch is formed and the cost of delivery is included in the objective function (2). Relations (15) and (16) are used to demonstrate the natures of the decision variables.

As observed above, the model is nonlinear since some variables are multiplied in relations (10) and (11). Thus, auxiliary variables are defined, including $H_{b,k}^{SB} = C_{b,k}^{SB}.a_{bk}$ in relation (10) and $O_{ibk} = C_b.x_{ibk}$ in equation (11), and some relations are used to linearize constraints (10) and (11). The relations are defined as follows:

$$C_b \geq H_{b,k}^{SB} \qquad \forall k,b \qquad (10\text{-}1)$$

$$H_{b,k}^{SB} \leq M.a_{bk} \qquad \forall k,b \qquad (10\text{-}2)$$

$$C_{b,k}^{SB} - M(1-a_{bk}) \leq H_{b,k}^{SB} \qquad \forall k,b \qquad (10\text{-}3)$$

$$H_{b,k}^{SB} \leq C_{b,k}^{SB} \qquad \forall k,b \qquad (10\text{-}4)$$

$$C_i = \sum_{b=1}^{n}\sum_{k=1}^{m} O_{ibk} \qquad \forall i \qquad (11\text{-}1)$$

$$O_{ibk} \leq M.x_{ibk} \qquad \forall i,k,b \qquad (11\text{-}2)$$

$$C_b - M(1-x_{ibk}) \leq O_{ibk} \qquad \forall i,k,b \qquad (11\text{-}3)$$

$$O_{ibk} \leq C_b \qquad \forall i,k,b \qquad (11\text{-}4)$$

The nonlinear model converts to a MILP model by replacing the above relations instead of constraints (10) and (11).

5. Solution approach

Solving real-life problems is impossible or takes a very long time, due to their large scales. Moreover, the mentioned problem is a bi-objective optimization problem and is strongly NP-hard (Lenstra et al., 1977). Hence, two multi-objective meta-heuristics, including MOPSO and NSGA-II, are designed for solving the problem as approximated solution methods since the exact solution methods cannot efficiently solve this problem.

This section includes defining and explaining the details of the MOPSO and the NSGA-II algorithms presented in this article as efficient meta-heuristics algorithms for optimizing the mentioned bi-objective mathematical model.

5.1. The MOPSO approach

Eberhart and Kennedy (1995) first introduced the particle swarm optimization (PSO) algorithm. This algorithm is based on an uncertain search method to optimize the function inspired by the collective movement of birds or fishes seeking food. In the PSO algorithm, each particle has a fitness value calculated by a fitness function. Whatever a particle is closer to the objective (optimization) in the solution space, it has more competency. Coello et al. (2004) introduced a multi-objective type of the PSO algorithm called the MOPSO in 2004 and developed it in 2006 and 2007 (Coello et al., 2007; Coello, 2006). Since a unique optimal solution cannot be defined in the multi-objective optimization structure, the MOPSO uses a non-dominated solution procedure where particles randomly choose their leaders from an approximated Pareto curve. In this algorithm, a grid structure is defined to search for the solution space. Moreover, the best non-dominated solution is stored in an external memory (repository). After the particle updates its velocity and location, the best solution should be optimized. The pseudo-code of the MOPSO in detail is demonstrated in Appendix A.

5.1.1. The MOPSO algorithm characteristics

In this sub-section, the particles' structure is defined, and then the fitness function, and the mutation operator applied on the MOPSO are explained in detail.

Particles' structure

The MOPSO algorithm begins its work with a set of particles, which are the initial solution. Due to the need for a feasible solution, at first, the feasible solution must be saved according to a specific structure called particles' structure. Since the PSO algorithm has a continuous solution space, an initial solution is generated with random numbers between 0 and 1.

Fig. 2 shows three strings generated by random numbers between 0 and 1 for an initial solution. The first string indicates the job number; the second string indicates the machine number allocated to the i^{th} job. The batch number allocated to the i^{th} job is also displayed in the third string; the number of jobs from 1 to n is equal to the length of the strings.

Job number	1	2	3	4	5	6
Machine number	0.53	0.86	1	0.41	0	0.68
Job Batch	0.47	1	0.74	0	0.52	0.83

Fig. 2 The strings generated by random numbers between {0, 1}

Fitness function

Particles are created for the initial solution in a continuous space. Thus, the numbers between 0 and 1 must be transformed to integer numbers to turn the continuous solution space into a discrete solution space. Finally, the values of the objective functions can be computed.The first decision is to assign the machine number to the i^{th} job. To do so, the random numbers in interval [0,1] are generated and equation (17) is used to transform random numbers between 0 and 1 to integer numbers for the string of machine numbers:

$$i_2 = 1 + [(m-1) \times \delta(2,i)], where\ \delta \in \{0,1\} \tag{17}$$

According to Eq. (17), m is the number of machines and δ is the random number between 0 and 1, which assigned to the i^{th} job in the second string. Thus, i_2 illustrates the integer machine number, which is assigned to the i^{th} job in the second string. The second decision is to assign the batch number to the i^{th} job. To do so, the random numbers in interval [0,1] are generated and Eq. (18) is used to transform numbers between 0 and 1 to integer numbers for the string of batch number:

$$i_3 = 1 + [(b-1) \times \delta(3,i)], where\ \delta \in \{0,1\} \tag{18}$$

According to Eq. (18), b is the number of batches and δ is the random number between 0 and 1, which assigned to the i^{th} job in the third string. So, i_3 indicates the integer batch number, which is assigned to the i^{th} job in the third string. According to Fig. 2, suppose that the customer orders six jobs that must be processed on three machines and delivered to the customer in three batches. Figure 3 shows the method is used to transform the random numbers to integer numbers in the strings of Fig. 2 and also, both the machine number and batch number are assigned to each job.

Job number	1	2	3	4	5	6
Machine number	2	2	3	1	1	2
Job Batch	1	3	2	1	2	2

Fig. 3 Random numbers are converted to integer numbers

Use of a mutation function

A mutation operator is utilized to intercept premature convergence, and better Pareto solutions can be obtained. In the MOPSO algorithm, after a mutation operation is performed on any of the genes in the structure, a new particle is created, in which the mutation rate (mu) is applied to each gene and the particle mutates with the probability of mutation (pm). In the mutation, a gene may be removed from a set of particle genes or a gene is added to the collection that has not so far existed in the population. Figure 4 indicates the mutation of genes in the strings of Fig. 2. Moreover, Fig. 5 shows a new mutated particle after the conversion from continuous to discrete operation.

The mutation function is performed in accordance with the following steps:

1. The genes are selected randomly in the strings based on a function of the number of jobs.
2. A random number with a uniform distribution is allocated to each selected gene. Random numbers are obtained in the interval $\{lb, ub\}$ with uniform distribution according to the following Eq. (19) to Eq. (22):

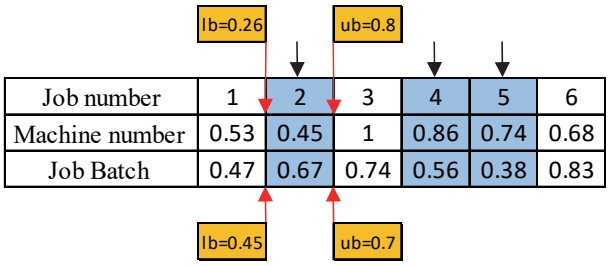
$$pm = |(1 - (iteration - 1)/(MaxIteration - 1))^{(1/mu)}| \tag{19}$$

$$dx = pm \times (Var_{Max} - Var_{Min}) \tag{20}$$

$$lb = |x(i,j) - dx| \tag{21}$$

$$ub = |x(i,j) + dx| \tag{22}$$

3. The batch and machine numbers associated with the i^{th} job are determined, and then allocated to their positions in the strings.
4. Previous steps are repeated until a new mutated population is completed.



Job number	1	2	3	4	5	6
Machine number	0.53	0.45	1	0.86	0.74	0.68
Job Batch	0.47	0.67	0.74	0.56	0.38	0.83

Fig. 4 Random selection of genes and performing a mutation operator on them and new genes generated by random numbers in created intervals

Job number	1	2	3	4	5	6
Machine number	2	1	3	2	2	2
Job Batch	1	2	2	2	1	2

Fig. 5 A new particle created after conversion of random numbers to integer numbers

5.2. The NSGA-II approach

Goldberg and Holland (1988) were the first scholars to generalize the concept of multi-objective fitness problems. They used the non-dominated solution sorting concept and approached the algorithm for the Pareto solution set in consecutive iterations. Goldberg's idea in the concept of the Pareto set was directly used by Deb et al. (2002). Actually, by using the idea of preference for non-dominated solutions and assigning them more fitness, a GA was created with non-dominated sorting. The NSGA-II is among the most well-known algorithms for multiple objectives optimization. Following the introduction of the first version of the algorithm to provide a variety of optimal solutions for the Pareto curve in 2002, the developers of the algorithm generated an elitist mechanism based on the importance of non-dominated queues under the name NSGA-II. The pseudo-code of the NSGA-II in detail is shown in Appendix A.

5.2.1. The NSGA-II characteristics

In this sub-section, the structure of the chromosome is defined, and then a one-point crossover operator and a mutation operator are explained in detail.

Chromosomes' structure

Each repetitive meta-heuristic algorithm requires a structure (encoding) for representing solutions. A GA begins with the initial solutions of a set of chromosomes. Each of these chromosomes has a value for the objective function, called the fitness value. Subsequently, the chance of survival and reproduction is higher for a chromosome if it has a better fitness function value. These chromosomes are referred to as initial populations and can be produced in various ways, such as random reproduction, heuristic methods, and so on. In this paper, each chromosome has a length equal to the number of jobs from 1 to n and has three rows: the first row indicates the job number, the second and third rows illustrate the machine number and batch number, respectively, which are randomly determined and assigned to the i^{th} job. Therefore, the initial population is randomly generated. Figure 6 indicates the chromosome structure.

Job number	1	2	3	4	5	6
Machine number	1	2	2	2	1	1
Job Batch	3	1	1	2	2	3

Fig. 6 A sample chromosome

Suppose there is a problem with six jobs, three batches, and two machines. Figure 6 represents the i^{th} job is processed on the k^{th} machine and delivered to the customer in the b^{th} batch. For example, the second job is processed on the second machine and delivered to the customer in the first batch.

The crossover operator

The most important operator in genetic algorithms is the crossover operator. Crossover is a process in which the old chromosome generation is mixed and combined to form a new chromosome generation. The couples considered as the parent in the selection section will share their genes and create new members. The crossover in a GA eliminates the genetic dispersion or genetic diversity of the population, as it provides a condition so that good genes can be found by other genes. In this study, we use a one-point crossover operator used in the literature on the scheduling problems (please see (Bose et al., 2019; Shen,

2019)). The one-point crossover is applied to both second and third rows for machine assignment and batching the jobs. The one-point crossover operator on chromosomes is shown in Fig. 7.

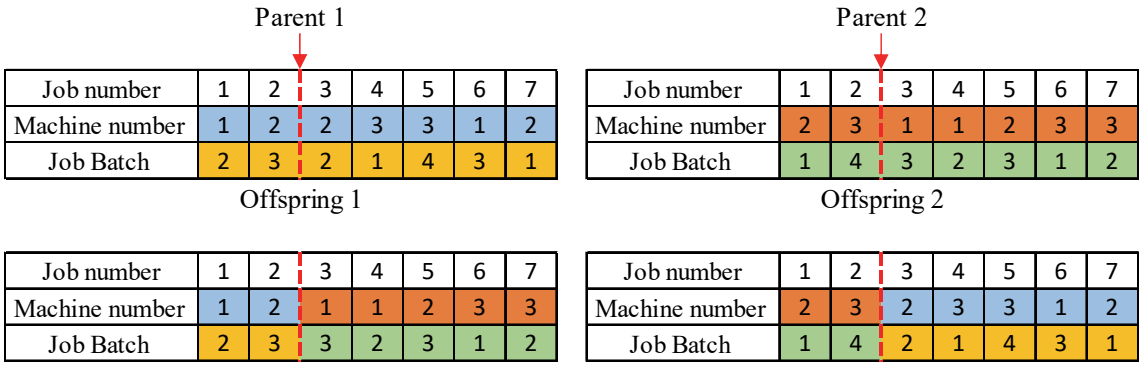


Fig. 7 The crossover operator

The mutation operator

A mutation is another operator that creates other feasible solutions. In genetic algorithms, after generating a member in a new population, each gene mutates with the probability of mutation and produces a mutated population. In this paper, the mutation operator is applied according to the following steps:

1. The genes are selected randomly on the chromosome and the number of selected genes is determined by a function of the number of jobs.
2. To mutate the genes in the second row, random numbers are selected from the interval $[0, M]$, where M is the number of machines and assigned to the machine number for the i^{th} job.
3. To mutate the genes in the third row, random numbers are selected from the interval $[0, B]$, where B is the number of batches and assigned to the batch number for the i^{th} job.
4. Previous steps are repeated to generate new offsprings until a new mutated population is completed.

Fig. 8 indicates the mechanism of mutation operator for machine assignment to the i^{th} job and batching the jobs.

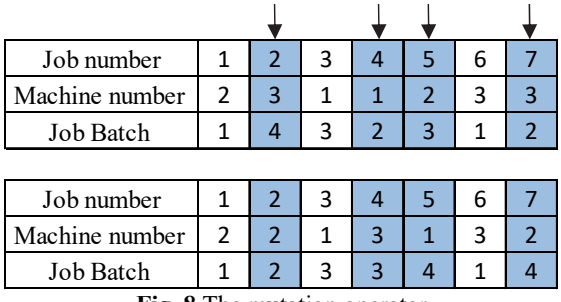


Fig. 8 The mutation operator

6. Experimentation and computational results

In this section, computational results are used to compare the efficiency and effectiveness of the NSGA-II and the MOPSO algorithm to solve small, medium, and large scale problems for nine classes presented in the following section (please see Sect. 6.2.). Firstly, the algorithm's parameters are tuned using the Taguchi design method. Secondly, the effectiveness of both algorithms is compared using the comparison criteria, and the results of the meta-heuristic algorithms are finally analyzed.

6.1. Parameter tuning

The efficiency of meta-heuristic algorithms is directly related to their parameters so that the correct choice of parameters increases the efficiency of the algorithm and also increases the quality of solutions. There are several methods for setting the parameters of the algorithms. In this research, the Taguchi method is used to ascertain the optimum value of parameters with their levels in algorithms. The controlled parameters of the Taguchi method are shown in Tables 2 and 3 for tuning parameters with their levels for the NSGA-II and the MOPSO algorithm, respectively. The problem is divided into three scales and the experiments are implemented for nine classes. Moreover, Minitab 16 was used to carry out all experiments related to the Taguchi method. For this purpose, a problem with 15 jobs and five machines is considered as a small scale problem, and a problem with 100 jobs and 15 machines is considered as a medium and large scale problem to adjust the parameters. Each

experiment number included nine classes and five random instances were generated for each class; each sample was solved five times with algorithms, and the average is presented. Therefore, 405 small scale and 405 medium and large scale sample problems are created to tune the parameters of the NSGA-II. In addition, the number of 1215 small scale and 1215 medium and large scale sample problems are created to adjust the parameters of the MOPSO. Finally, a total of 3240 sample problems were created.

Table 2
Factors and their level in the NSGA-II

		Small scale			Medium and large scale		
Name	Notation	level 1	level 2	level 3	level 1	level 2	level 3
Population size	A	50	80	100	200	300	350
Maximum iteration	B	100	150	200	100	200	250
Crossover Probability	C	0.65	0.7	0.8	0.75	0.8	0.85
Mutation Probability	D	0.3	0.4	0.5	0.15	0.2	0.25

Table 3
Factors and their level in the MOPSO

		Small scale			Medium and large scale		
Name	Notation	level 1	level 2	level 3	level 1	level 2	level 3
Population size	A	50	80	100	200	300	350
Capacity of repository	B	50	80	100	200	300	350
nGrid	C	8	10	12	8	10	12
Maximum iteration	D	100	150	200	100	200	250
C1, C2	E	0.5	1	2	0.5	1	2
W	F	0.8	0.9	1	0.8	0.9	1
Mutation rate	G	0.3	0.4	0.5	0.15	0.2	0.25
Alpha	H	0.08	0.1	0.2	0.08	0.1	0.2

To use the Taguchi method, several comparison criteria are used so that the efficiency and effectiveness of each algorithm can be assessed (please see Sect. 6.3.). Moreover, a weight is allocated to each criterion based on its importance. Finally, each criterion is unscaled by the related deviation index (RDI), and then a response variable for the Taguchi method is achieved. Fig. 9 shows the signals to noise charts from the implementation of these tests. Based on the charts in Fig. 9, the selected NSGA-II and MOPSO algorithm factors levels are demonstrated in Tables 4 and 5, respectively.

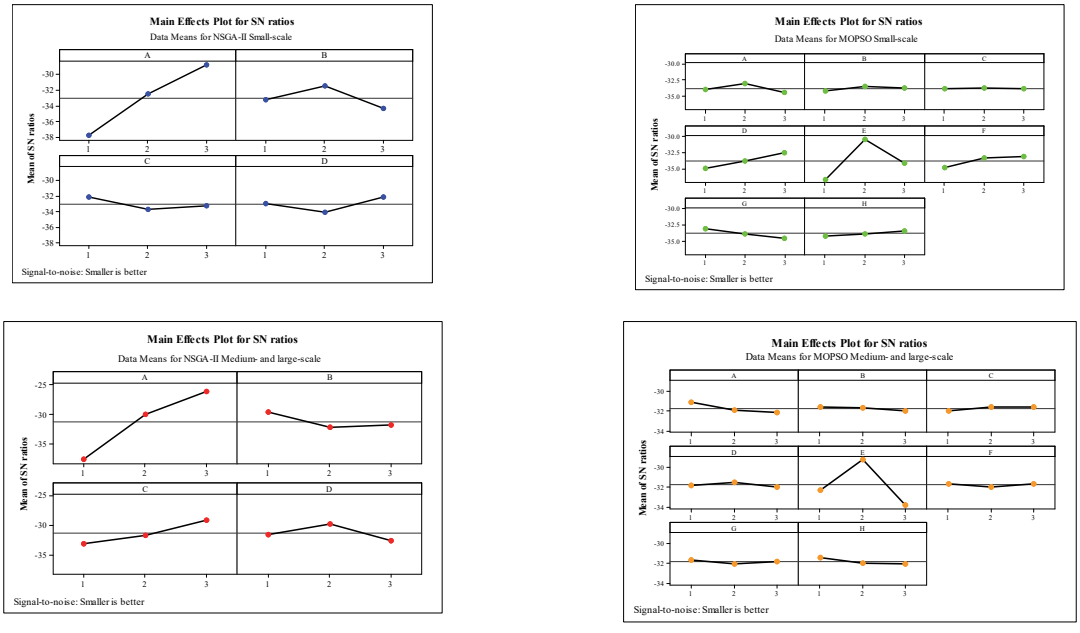


Fig. 9 Signal to noise charts by the Taguchi method

Table 4
The factors and their level for NSGA-II

Factors	Small scale	Medium and large scale
Population size	100	200
Maximum iteration	150	100
Crossover Probability	0.65	0.85
Mutation Probability	0.5	0.2

Table 5
The factors and their level for MOPSO

Factors	Small scale	Medium and large scale
Population size	80	120
Capacity of repository	80	120
nGrid	10	10
Maximum iteration	200	200
C1, C2	1	1
W	1	1
Mutation rate	0.3	0.15
Alpha	0.2	0.08

6.2. Data generation

For the aim of evaluating the effectiveness of the NSGA-II and the MOPSO algorithm, problems are designed into two scales, then the performance of the algorithms is compared and analyzed using the five criteria. The meta-heuristics algorithms MOPSO and NSGA-II were encoded in MATLAB 2015b, and a Core i7 CPU and 8 GB RAM computer with Windows 10 (64 bit) was used as a test system. In small scale sample problems, the number of jobs is assumed to be 8, 12, 16, 20 and 24, and the number of machines is assumed to be 2, 4 and 6. In the medium and large scale sample problems, the number of jobs is considered to be 40, 80, 120, 180 and 250, and the number of machines is considered to be 8, 16, and 27. Other parameters were produced randomly based on a uniform distribution in the following intervals.

- The contractual due dates (d_i) for customers in the intervals on $[0.25P/m, P/m]$, $[1, P/m]$, and $[1,0.75P/m]$, where $P = \sum p_i$
- The batch delivery cost (θ) unit in the intervals on $[1, W/n]$, $[10, W/n]$, and $[30, W/n]$, where $W = \sum w_i$
- The processing time (p_i) in the interval [1,100]
- The priority weight of the job i (w_i) in the interval [1,100]
- The weight of tardiness (α_i) in the interval [5,10]
- The earliness cost (β_i) in the interval [5,10]
- The setup time of the sub-batch from each batch (S_b) in the interval $[1,0.2P/m]$

A summary of the information above can be found in Table 6. Additional problem information, such as the number of machines and jobs, is also shown in Table 7.

Table 6
Parameters and their values

Parameter	Value	Level
contractual due date (d_i)	$[0.25P/m, P/m]$	Loose
	$[1, P/m]$	Moderate
	$[1,0.75P/m]$	Tight
Batch delivery cost (θ)	$[1, W/n]$	Low
	$[10, W/n]$	Medium
	$[30, W/n]$	High
Processing time (p_i)	[1,100]	
Reputational weight of job i (w_i)	[1,100]	
Weight of tardiness (α_i)	[5,10]	
Earliness cost (β_i)	[5,10]	
Setup time (S_b)	$[1,0.2P/m]$	

Table 7
The Number of machines and jobs

Factors	Small scale	Medium and large scale
Number of jobs	8, 12, 16, 20, 24	40, 80, 120, 180, 250
Number of machines	2, 4, 6	8, 16, 27

The batch delivery costs can be associated with the priority of jobs (w_i) and therefore, the delivery costs are introduced according to the weight of jobs. In addition, since the complexity of the problem could be affected by the delivery cost, three classes of the problem with low, medium, and high costs of delivery are generated: Classes A, B, and C represent low,

medium, and high costs of delivery (θ), respectively. Furthermore, due to the impact of contractual due dates of jobs on the complexity of the problem, three sub-classes of problems of classes A, B and C are considered. Each sub-class 1, 2, and 3 represents the loose, moderate, and tight contractual due date (d_i), respectively. Table 8 shows all nine classes. For any combination of jobs and machines (five jobs×three machines) in each class; from A1 to C3, there are 15 problems that five random instances generated for each problem. In the case of small scale, the number of sample problems is 675 ($9 \times 15 \times 5$), and 675 sample problems are also generated for medium and large scale. Each sample is run five times by the algorithms. The ultimate result equals the average of five runs for each class.

Table 8
The classes generated for the problem

	Factors	
Class Name	Contracted due date	Batch delivery cost
A1	$[0.25P/m, P/m]$	$[1, W/n]$
A2	$[1, P/m]$	$[1, W/n]$
A3	$[1,0.75P/m]$	$[1, W/n]$
B1	$[0.25P/m, P/m]$	$[10, W/n]$
B2	$[1, P/m]$	$[10, W/n]$
B3	$[1,0.75P/m]$	$[10, W/n]$
C1	$[0.25P/m, P/m]$	$[30, W/n]$
C2	$[1, P/m]$	$[30, W/n]$
C3	$[1,0.75P/m]$	$[30, W/n]$

6.3. Comparing criteria

In single objective problems, an optimal solution is chosen for the purpose of the problem, while in multiple objective problems, we encounter a collection of Pareto solutions at the end of solving problems and the performance of the algorithms cannot be easily evaluated. Evaluation criteria are important since they can be used to examine the performance of algorithms. In this research, we apply five comparison criteria from the literature to assess the performance of algorithms (Attar et al., 2014; Piroozfard et al., 2018; Zarei & Rasti-Barzoki, 2018). The comparing criteria are including quality metric criterion *(QM)*, spacing metric criterion *(SM),* diversification metric *(DM),* CPU time *(CPUT),* and mean ideal distance *(MID)*. A lower value of *SM*, *MID*, and *CPUT* criteria, and a higher percentage of *QM* criterion, and a higher value of *DM* criterion indicate better efficiency of the algorithm. The characteristics of comparing criteria are described in detail in Appendix B.

6.4. Comparison results and analysis for small scale samples

In this sub-section, the performance of NSGA-II and MOPSO algorithms is investigated for small scale problems according to the preceding criteria. Here, the charts of the three classes A, B, and C are examined, each being derived from the mean of classes A1 to C3 (The resulting data of each class is provided in Appendix C).

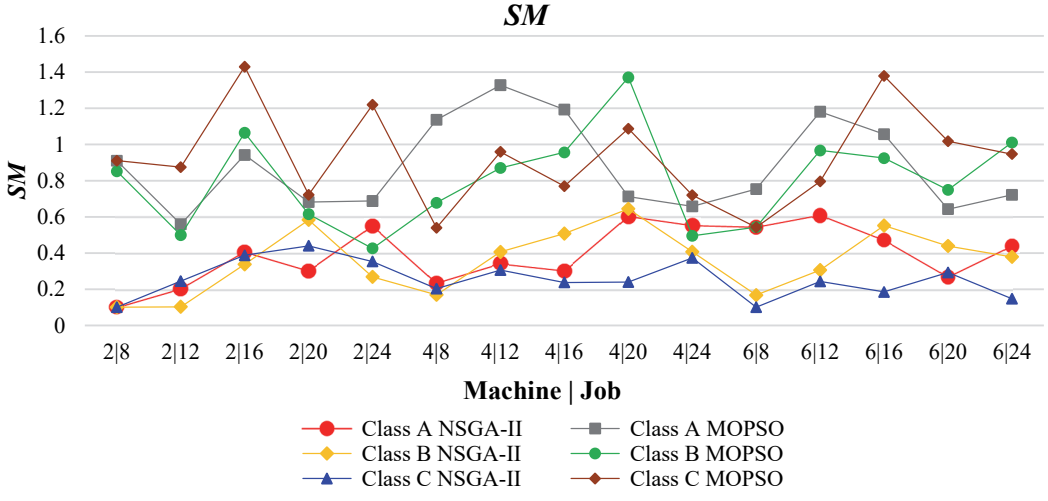


Fig. 10. SM criterion for the two algorithms in the small scale sample problems

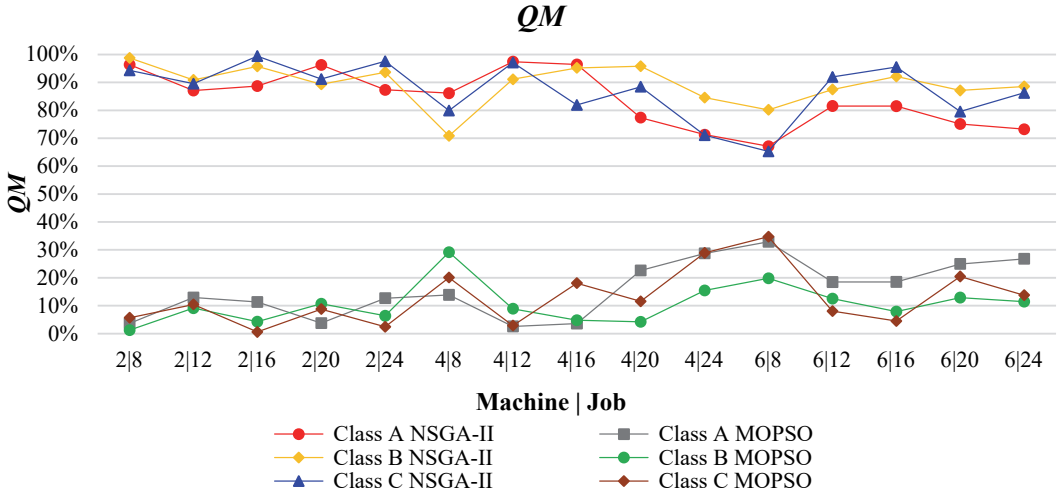


Fig. 11. QM criterion for the two algorithms in the small scale sample problems

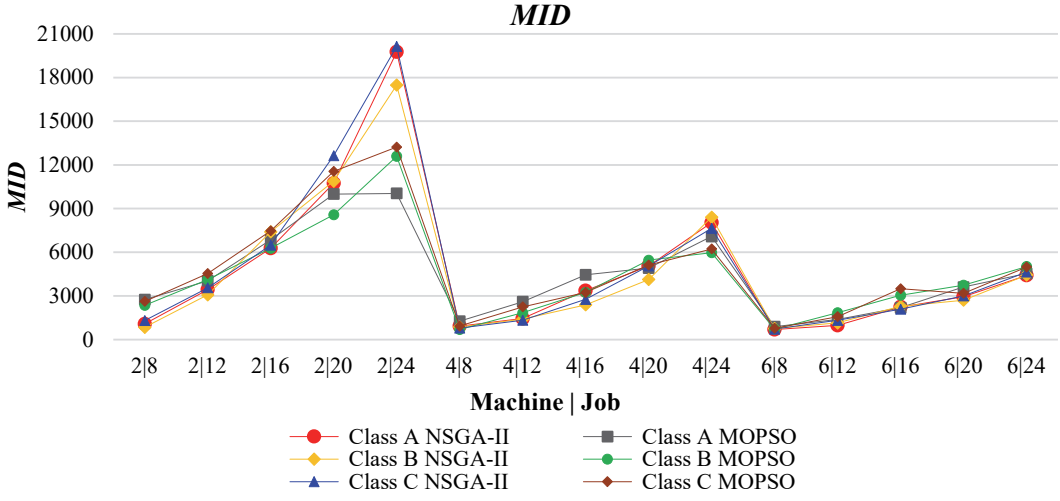


Fig. 12. MID criterion for the two algorithms in the small scale sample problems

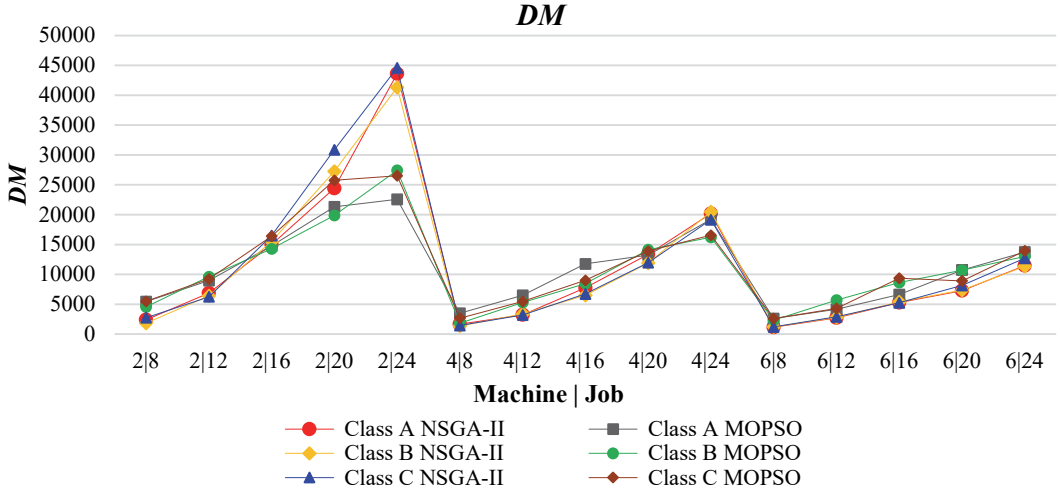


Fig. 13. DM criterion for the two algorithms in the small scale sample problems

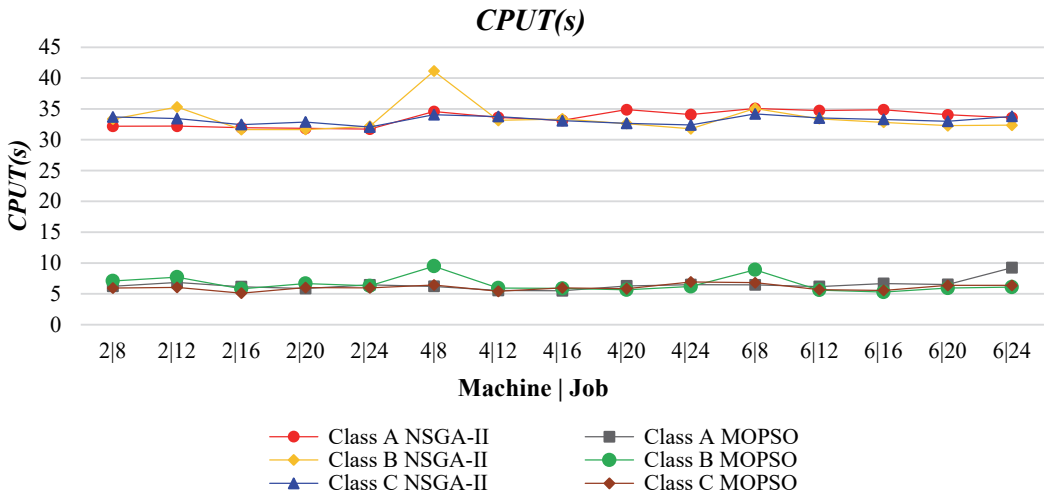
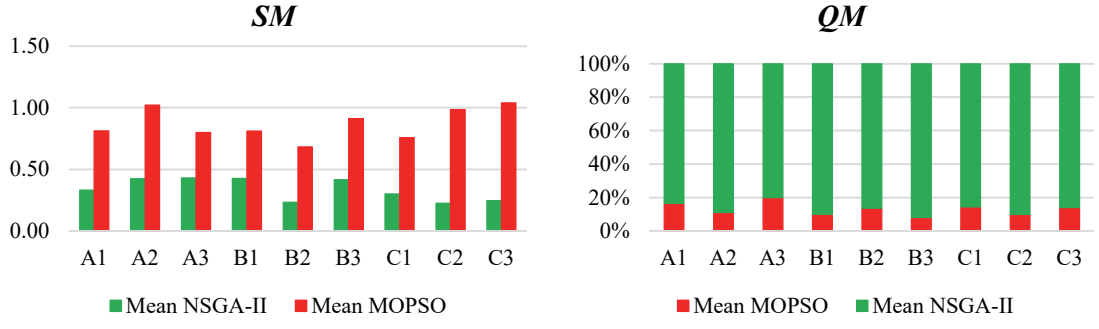


Fig. 14. CPUT criterion for the two algorithms in the small scale sample problems

According to Figs. 10, 11 and 12, the NSGA-II performs better than the MOPSO algorithm due to its smaller values in the *SM* criterion. In the important *QM* criterion, it is shown that the NSGA-II reaches a major portion of the Pareto solutions. The graph shows that increasing the number of machines and jobs gradually decreases the percentage of Pareto solutions in the NSGA-II, while the MOPSO algorithm shows an upward trend. In the *MID* criterion, the NSGA-II had better results than the MOPSO algorithm in 66 percent of class A, 60 percent of class B and 80 percent of class C. It is noteworthy that the MOPSO algorithm is able to outperform the NSGA-II in the experiments 2|20, 2|24, and 4|24 in all the three classes, and in general, the MOPSO algorithm shows a weaker performance when the number of machines and jobs increases. As revealed in Figs. 13 and 14, in the *DM* criterion, the MOPSO algorithm performed superior to the NSGA-II in a variety of Pareto-solutions in 66 percent of class A as well as in 73 percent of classes B and C. It is noteworthy that the performance of NSGA-II is improved when the number of jobs increase from 16 to 24 in the number of machines from 2 and 4. In the *CPUT* criterion, it can be easily concluded that the solving time by the MOPSO in all the classes is about one-fifth of the NSGA-II solving time.

6.4.1. Comparison of algorithms by Standard Error and average value

The Standard Error *(SE)* and the average value *(Mean)* for each of the criteria are calculated for better analysis and assessment of the quality of the presented solutions. The quality of statistics is usually measured by using the. Thus, the performances of the two algorithms are compared in each of the nine classes A1 to C3 using the *SE* and the average value. The lower value of the *SE* is preferable. According to Fig. 15 and Table 9, the mean value and *SE* for the *SM* criterion are lower in all the classes in the NSGA-II than in the MOPSO algorithm. The NSGA-II has a higher average value in the *QM* criterion than in the MOPSO algorithm, and the *SE* is the same for both algorithms. In the *MID* criteria, it can be deduced by comparing the two algorithms that the MOPSO algorithm has a lower average value except for classes A2, B3, and C3. Moreover, the MOPSO algorithm has a lower *SE* in all the classes compared to the NSGA-II, meaning that the MOPSO algorithm is preferable in overall performance in this criterion. In the *DM* criterion, although the NSGA-II is more diverse except for classes A2, B3, C1, and C3, the *SE* in this criterion is lower for the MOPSO algorithm than for the NSGA-II in the rest of the classes. This means that the MOPSO algorithm has a more robust solution with a lower mean value for the *DM* criterion. Eventually, the MOPSO algorithm is shown to perform well on both factors in the *CPUT* criterion except for class C3. Table 10 indicates the best class in each of the criteria based on the *SE* and average value achieved by the MOPSO algorithm and the NSGA-II for small scale problems.



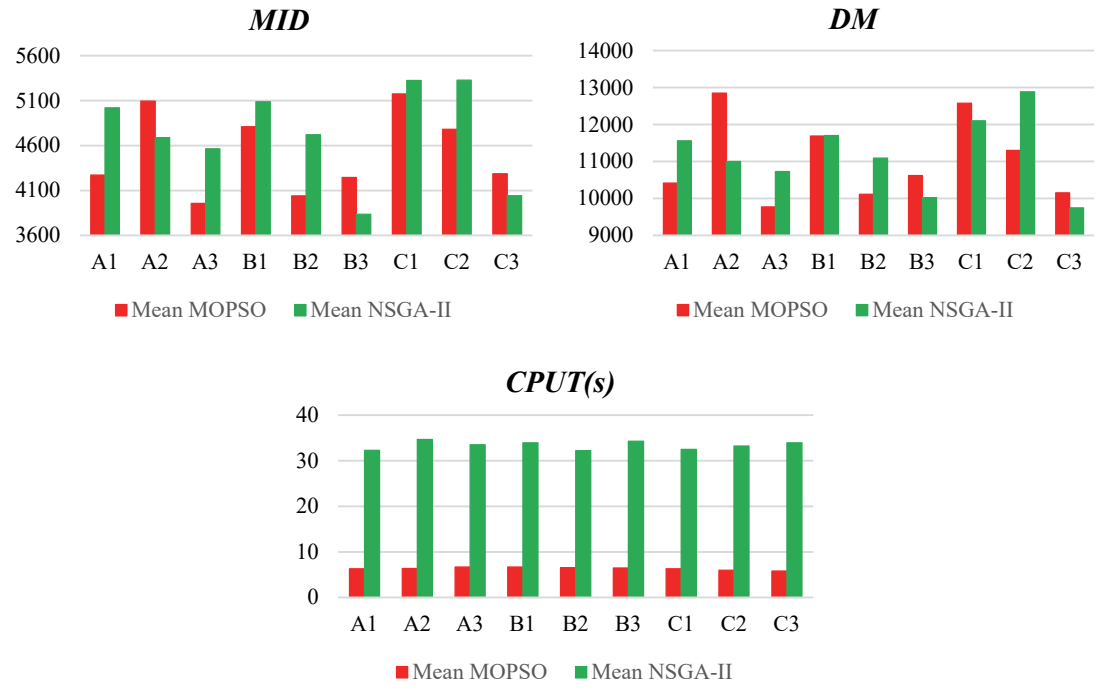


Fig. 15 Mean value for the two algorithms and small scale sample problems for each class

Table 9
SE and Mean values for the two algorithms in small scale sample problems for each class

A1_NSGA-II						A1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.332	83.5	5019	11559	32	Mean	0.812	16.5	4271	10413	6.3
SE	0.0200	1.937	610	1393	0.167	SE	0.0413	1.937	346	755	0.079
A2_NSGA-II						A2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.423	89.0	4688	11002	35	Mean	1.021	11.0	5094	12852	6.4
SE	0.0282	1.388	539	1290	0.334	SE	0.0377	1.388	434	1030	0.245
A3_NSGA-II						A3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.429	80.1	4563	10727	34	Mean	0.799	19.9	3957	9771	6.7
SE	0.0274	1.672	614	1270	0.104	SE	0.0589	1.672	307	680	0.101
B1_NSGA-II						B1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.425	89.9	5086	11699	34	Mean	0.810	10.1	4809	11686	6.7
SE	0.0270	1.424	576	1339	0.323	SE	0.0423	1.424	430	878	0.271
B2_NSGA-II						B2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.233	86.3	4720	11092	32	Mean	0.682	13.7	4040	10113	6.6
SE	0.0191	2.041	634	1473	0.128	SE	0.0321	2.041	393	895	0.127
B3_NSGA-II						B3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.415	92.0	3836	10021	34	Mean	0.912	8.0	4245	10617	6.5
SE	0.0280	0.673	430	1127	0.618	SE	0.0407	0.673	322	719	0.209
C1_NSGA-II						C1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.3	85.6	5324	12101	33	Mean	0.757	14.4	5173	12578	6.3
SE	0.0213	1.297	713	1527	0.092	SE	0.0477	1.297	487	992	0.091
C2_NSGA-II						C2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.226	90.2	5328	12881	33	Mean	0.986	9.8	4781	11301	6.0
SE	0.0190	1.208	669	1657	0.142	SE	0.0488	1.208	407	847	0.089
C3_NSGA-II						C3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.246	86.0	4042	9745	34	Mean	1.039	14.0	4286	10150	5.8
SE	0.0164	2.548	463	1049	0.086	SE	0.0441	2.548	410	862	0.114

Table 10
The best classes achieved by Mean value and Standard Error for small scale sample problems

	NSGA-II	MOPSO
Criterion	Class Name	Class Name
SM	C3	B2
QM	B3	A3
MID	B3	A3
DM	C3	A3
CPUT	C3	A1

6.5. Comparison results and analysis for medium and large scales samples

With regard to the mentioned criteria, this sub-section examines the performance of the NSGA-II and the MOPSO algorithm for problems with the medium and large scales. The charts of the three classes A, B, and C, each respectively, derived from the mean of classes A1 to C3, are investigated (The resulting data of each class is provided in Appendix C).

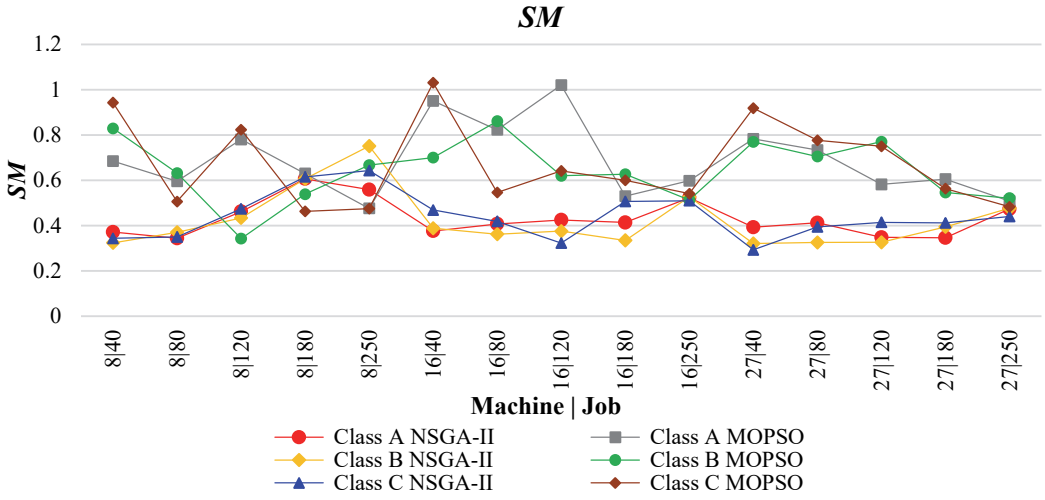


Fig. 16 SM criterion for the two algorithms in the medium and large scale sample problems

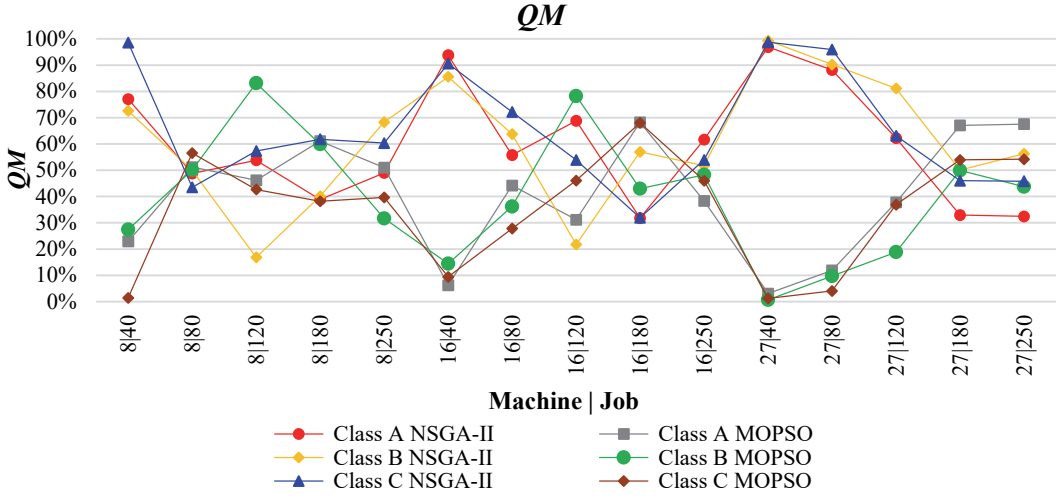


Fig. 17 QM criterion for the two algorithms in the medium and large scale sample problems

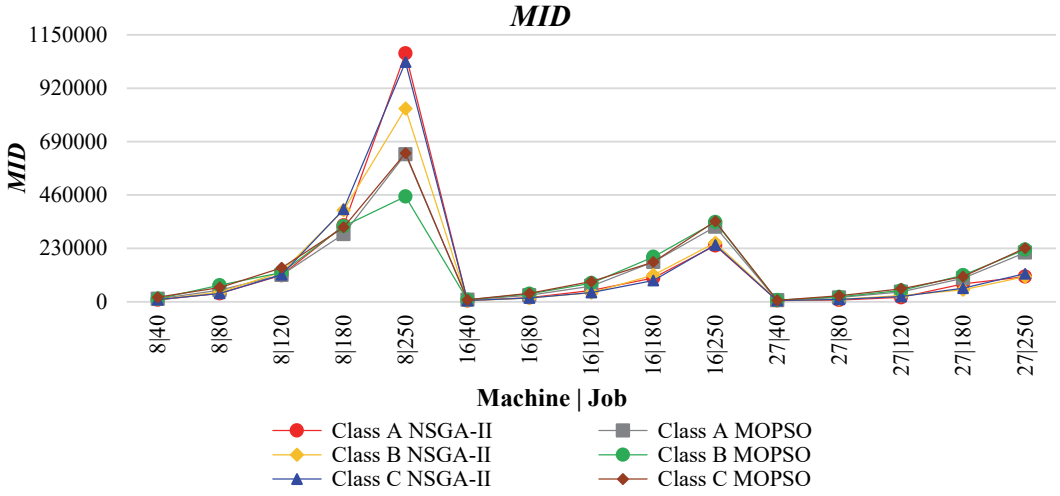


Fig. 18 MID criterion for the two algorithms in the medium and large scale sample problems

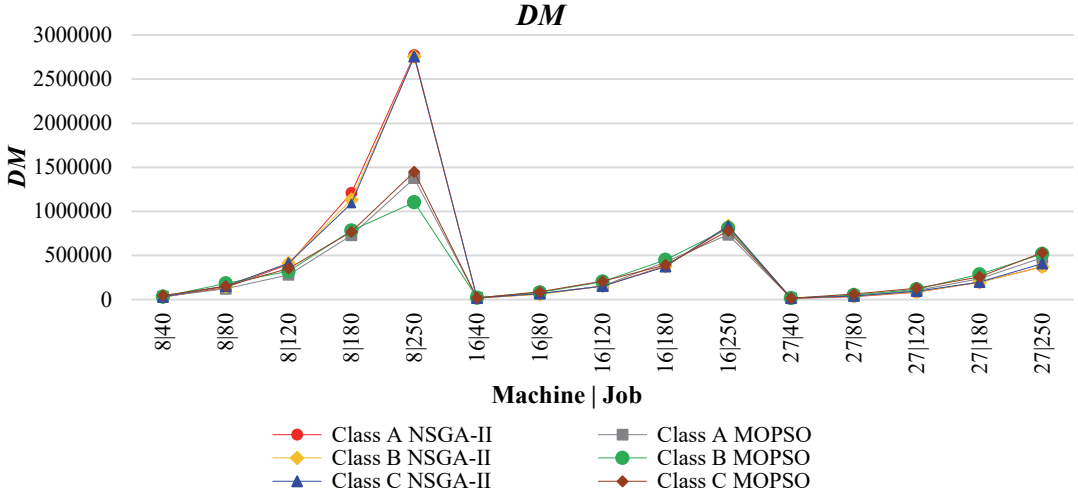


Fig. 19 DM criterion for the two algorithms in the medium and large scale sample problems

According to Figs. 16, 17, and 18, it could be noticed that the performance of the NSGA-II in the *SM* criterion is better than that of the MOPSO algorithm except in the tests of 8|180 and 8|250 for all classes. In addition, considering the MOPSO diagram, it can be inferred that the performance of this algorithm is enhanced with an increasing problem scale to medium and large. With increasing the number of jobs and machines in medium and large scale in the *QM* criterion, the difference between the two algorithms is less than that in the small scale for covering the non-dominated solutions, meaning the performance of the MOPSO algorithm improved greatly in high dimensions. In other words, with moving from medium to large in the number of jobs and machines, the efficiency of the two algorithms became closer to each other. However, at 60% problems of class A, 77% problems of class B, and 73% problems of class C, the NSGA-II had better performance than the MOPSO algorithm. In the *MID* criterion, the NSGA-II included 86%, 80%, and 86% of the solutions in classes A, B, and C, respectively, and had better performance than the MOPSO algorithm. In experiments such as 8|180 and 8|250, the MOPSO algorithm was able to perform better in all the classes than the NSGA-II, demonstrating that the performance of the NSGA-II was reduced with an increase in the number of jobs. However, when the number of machines increases, the NSGA-II again showed better performance. Compared to small scale problems, it can also be concluded that the performance of the NSGA-II improved in the *MID* criterion with increasing problem dimensions. Regarding Figs. 19 and 20, in the *DM* criterion, the MOPSO algorithm had a greater diversity of solutions than the NSGA-II in 66% problems of class A, 73% problems of class B and 66% problems of class C in all the tests. On the other hand, the NSGA-II had more diversification than the MOPSO algorithm in all the three classes in the tests such as 8|120, 8|180, 8|250, and 16|250, meaning that the NSGA-II had better efficiency in the fewer number of machines and more jobs, but had decreased efficiency with increasing the number of machines. At last, it can be realized that the MOPSO algorithm solves problems six times faster than the NSGA-II in all the classes.

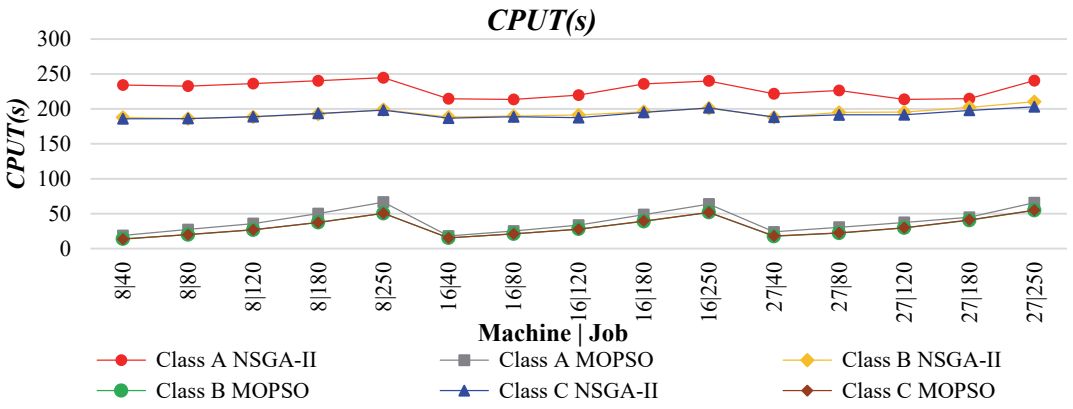


Fig. 20 CPUT criterion for the two algorithms in the medium and large scale sample problems

6.5.1. Comparison of algorithms by Standard Error and average value

The performance of the two algorithms in each of the nine classes A1 to C3 is compared using the *SE* and the average value for medium and large scale sample problems.

Table 11
SE and Mean values for the two algorithms in medium and large scale sample problems for each class

A1_NSGA-II						A1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.392	62.3	131016	456631	245.6	Mean	0.547	37.7	146382	341563	42.7
SE	0.0126	2.500	28526	86980	1.801	SE	0.0178	2.500	21632	47529	2.074
A2_NSGA-II						A2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.451	59.2	145218	448873	209.3	Mean	0.615	40.8	162139	359917	35.8
SE	0.0136	2.769	31091	84569	3.448	SE	0.0227	2.769	24408	50962	2.056
A3_NSGA-II						A3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.500	56.9	161546	430415	230.7	Mean	0.847	43.1	107758	262403	39.8
SE	0.0334	3.317	35119	80225	2.296	SE	0.0332	3.317	13800	33950	1.799
B1_NSGA-II						B1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.432	66.6	148050	446751	185.8	Mean	0.576	33.4	148333	354175	30.9
SE	0.0192	2.952	28886	86537	0.993	SE	0.0213	2.952	18807	44664	1.557
B2_NSGA-II						B2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.402	59.3	127944	443956	199.2	Mean	0.582	40.7	138171	328547	32.0
SE	0.0142	2.855	22886	80562	0.764	SE	0.0286	2.855	17134	40141	1.593
B3_NSGA-II						B3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.428	55.0	131148	435360	197.1	Mean	0.769	45.0	126261	312042	31.1
SE	0.0209	3.798	25327	81331	0.683	SE	0.0244	3.798	14706	36280	1.583
C1_NSGA-II						C1_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.423	67.2	143677	443802	195.3	Mean	0.603	32.8	165076	385931	31.4
SE	0.0189	2.721	28219	80543	0.613	SE	0.0302	2.721	23737	54239	1.566
C2_NSGA-II						C2_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.475	63.7	156685	439195	187.7	Mean	0.556	36.3	164746	374370	31.7
SE	0.0152	2.765	34131	80958	0.940	SE	0.0210	2.765	23657	52417	1.599
C3_NSGA-II						C3_MOPSO					
	SM	QM	MID	DM	CPUT		SM	QM	MID	DM	CPUT
Mean	0.4221	63.9	140925	445023	193.8	Mean	0.8519	36.1	123001	290038	31.2
SE	0.0162	3.250	30639	85577	0.591	SE	0.0361	3.250	13589	31535	1.603

According to Fig. 21 and Table 11, the NSGA-II has a lower average value and lower *SE* in the *SM* criterion than the MOPSO algorithm and is more efficient. The NSGA-II allocates more non-dominated solutions than the MOPSO algorithm in the *QM* criterion, but both algorithms have the same *SE*. In the *MID* criterion, the comparison of the two algorithms shows that the average value in the NSGA-II is the lowest except for classes A3, B3, and C3. However, the MOPSO algorithm has more robust solutions with higher mean value since the *SE* value for all MOPSO classes is lower than the NSGA-II. In the *DM* criterion, it could be noticed that the MOPSO algorithm provides solutions with more robust features than the NSGA-II because of having a lower *SE* with a lower average. By comparing the two algorithms in the *CPUT* criterion, it can be

perceived that the NSGA-II has robust Pareto solutions with a higher average due to having a lower *SE* in all the classes except for classes A2 and A3, as compared to the MOPSO algorithm. Table 12 shows the best class in each of the criteria based on the *SE* and average value of the NSGA-II and the MOPSO algorithm for medium and large scale problems.

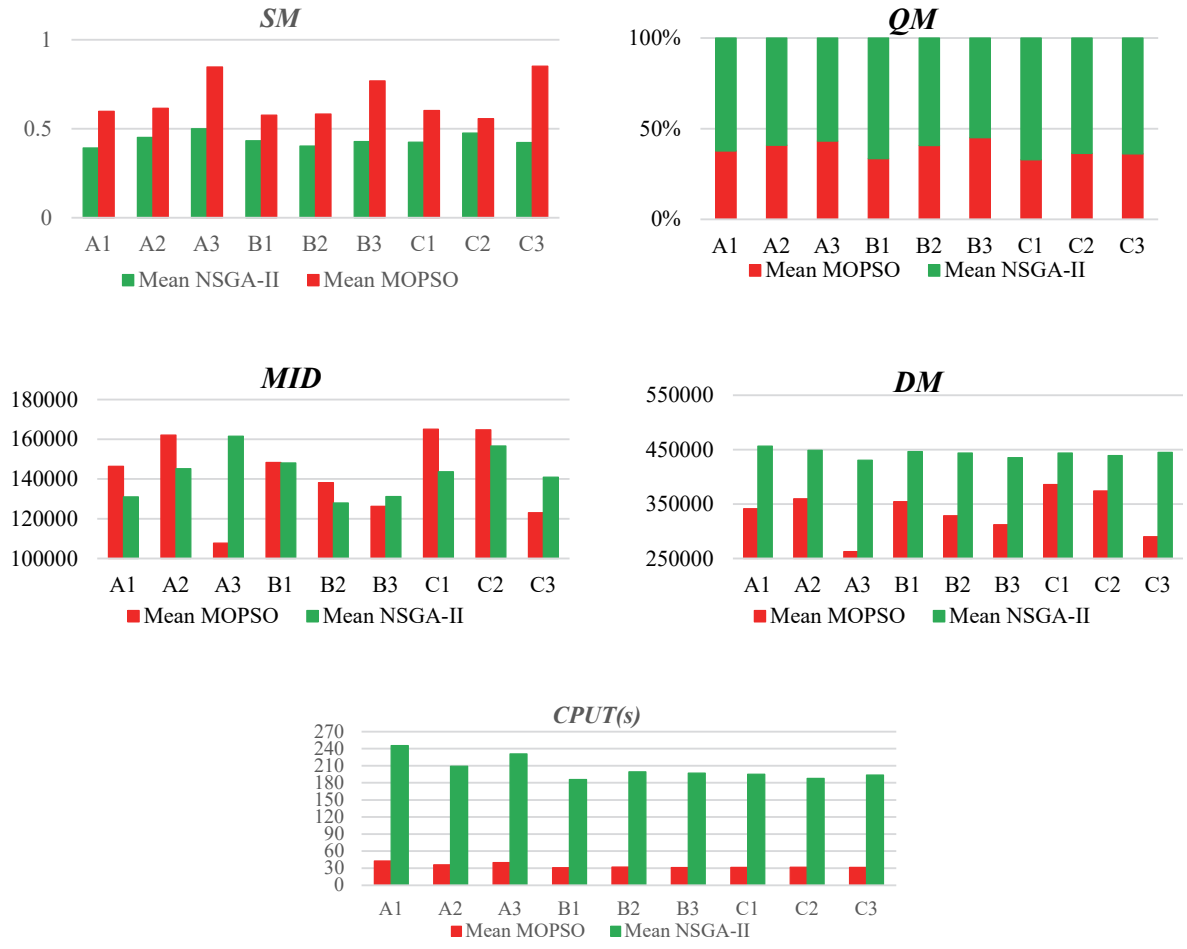


Fig. 21 Mean value for the two algorithms in medium and large scale sample problems for each class

Table 12
The best classes achieved by Mean value and Standard Error for medium and large scale sample problems

	NSGA-II	MOPSO
Criterion	Class Name	Class Name
SM	A1	A1
QM	A1	A1
MID	B2	C3
DM	A3	C3
CPUT	C3	B1

7. Conclusion and suggestions

In this research, a bi-objective scheduling problem of production and distribution was integrated in an identical parallel machines environment. First, a mathematical model was presented for the problem. Then, two meta-heuristics algorithms, including the MOPSO algorithm and the NSGA-II were customized to solve the problem. The NSGA-II was used with a one-point crossover operator and a mutation operator, in which the number of mutations is a function of the number of jobs to generate more mutated offsprings to spot the best optimum solution in the solution space. Also, the MOPSO algorithm was used with a mutation function for generating new particles to find better quality solutions. In the second step, the parameters tuning of the two algorithms was performed for small, medium, and large scale problems using the Taguchi method so that the algorithms could improve problem-solving. After tuning the parameters, the sample problems were solved in small, medium, and large scale problems. Besides, five comparing criteria including *SM*, *QM*, *MID*, *DM*, and *CPUT* were utilized to compare the performance and efficiency of the meta-heuristic algorithms in nine classes of problems by multiplying three levels of delivery costs and three levels of the contractual due dates. Finally, the Standard Error (*SE*) and the average value (*Mean*) of each criterion were used for the precise analysis of the classes. The results of the small scale sample problems

showed that the NSGA-II was completely superior to the MOPSO algorithm in the five criteria except for the *CPUT* and diversity of Pareto solutions in most of the generated samples. Moreover, in the medium and large scale sample problems, the results revealed the significantly improved performance and efficiency of the MOPSO algorithm in some performance criteria such as *QM*, and *DM*. However, generally, the NSGA-II performed better than the MOPSO algorithm in solving problems in all the tests except for the *CPUT* and *DM*. For future research, the corresponding model can be extended with constraints better describing real-life conditions. Additionally, a vehicle routing problem for transportation can be investigated as a more detailed examination of distribution costs. Moreover, other multi-objective algorithms can be used to address the problem and their performance can be evaluated by comparing with the algorithms presented in the current study.

References

Aminzadegan, S., Tamannaei, M., & Rasti-Barzoki, M. (2019). Multi-agent supply chain scheduling problem by considering resource allocation and transportation. *Computers & Industrial Engineering, 137*, 106003.

Assarzadegan, P., & Rasti-Barzoki, M. (2016). Minimizing sum of the due date assignment costs, maximum tardiness and distribution costs in a supply chain scheduling problem. *Applied Soft Computing, 47*, 343-356.

Attar, S., Mohammadi, M., Tavakkoli-Moghaddam, R., & Yaghoubi, S. (2014). Solving a new multi-objective hybrid flexible flowshop problem with limited waiting times and machine-sequence-dependent set-up time constraints. *International Journal of Computer Integrated Manufacturing, 27*(5), 450-469.

Bose, A., Biswas, T., & Kuila, P. (2019). A Novel Genetic Algorithm Based Scheduling for Multi-core Systems *Smart Innovations in Communication and Computational Sciences* (pp. 45-54): Springer.

Cakici, E., Mason, S. J., Geismar, H. N., & Fowler, J. W. (2014). Scheduling parallel machines with single vehicle delivery. *Journal of Heuristics, 20*(5), 511-537.

Cakici, E., Mason, S. J., & Kurz, M. E. (2012). Multi-objective analysis of an integrated supply chain scheduling problem. *International Journal of Production Research, 50*(10), 2624-2638.

Chang, Y.-C., Li, V. C., & Chiang, C.-J. (2014). An ant colony optimization heuristic for an integrated production and distribution scheduling problem. *Engineering Optimization, 46*(4), 503-520.

Chen, Y., Lu, L., & Yuan, J. (2015). Preemptive scheduling on identical machines with delivery coordination to minimize the maximum delivery completion time. *Theoretical Computer Science, 583*, 67-77.

Chen, Y., Lu, L., & Yuan, J. (2016). Two-stage scheduling on identical machines with assignable delivery times to minimize the maximum delivery completion time. *Theoretical Computer Science, 622*, 45-65.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: review and extensions. *Operations research, 58*(1), 130-148.

Chen, Z.-L., & Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science, 51*(4), 614-628.

Cheng, T., & Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research, 47*(3), 271-292.

Cheng, T. C. E., & Kahlbacher, H. G. (1993). Single-machine scheduling to minimize earliness and number of tardy jobs. *Journal of optimization theory and applications, 77*(3), 563-573.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering, 191*(11-12), 1245-1287.

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems* (Vol. 5): Springer, 79-104.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on evolutionary computation, 8*(3), 256-279.

Coello, C. C. (2006). Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine, 1*(1), 28-36.

Deb, K. (2014). Multi-objective optimization *Search methodologies* (pp. 403-449): Springer.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation, 6*(2), 182-197.

Eberhart, R., & Kennedy, J. (1995). *A new optimizer using particle swarm theory.* Paper presented at the Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on.

Ekici, A., Elyasi, M., Özener, O. Ö., & Sarıkaya, M. B. (2019). An application of unrelated parallel machine scheduling with sequence-dependent setups at Vestel Electronics. *Computers & Operations Research, 111*, 130-140.

Ganji, M., Kazemipoor, H., Molana, S. M. H., & Sajadi, S. M. (2020). A green multi-objective integrated scheduling of production and distribution with heterogeneous fleet vehicle routing and time windows. *Journal of Cleaner Production, 259*, 120824.

Gao, S., Qi, L., & Lei, L. (2015). Integrated batch production and distribution scheduling with limited vehicle capacity. *International Journal of Production Economics, 160*, 13-25.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning, 3*(2), 95-99.

Guo, Z., Zhang, D., Leung, S. Y.-S., & Shi, L. (2016). A bi-level evolutionary optimization approach for integrated production and transportation scheduling. *Applied Soft Computing, 42*, 215-228.

Hamidinia, A., Khakabimamaghani, S., Mazdeh, M. M., & Jafari, M. (2012). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers & Industrial Engineering, 62*(1), 29-38.

Hassanzadeh, A., Rasti-Barzoki, M., & Khosroshahi, H. (2016). Two new meta-heuristics for a bi-objective supply chain scheduling problem in flow-shop environment. *Applied Soft Computing, 49*, 335-351.

Ho, J. C., & Chang, Y.-L. (1995). Minimizing the number of tardy jobs for m parallel machines. *European Journal of Operational Research, 84*(2), 343-355.

Jiang, L., Pei, J., Liu, X., Pardalos, P. M., Yang, Y., & Qian, X. (2017). Uniform parallel batch machines scheduling considering transportation using a hybrid DPSO-GA algorithm. *The International Journal of Advanced Manufacturing Technology, 89*(5-8), 1887-1900.

Joo, C. M., & Kim, B. S. (2017). Rule-based meta-heuristics for integrated scheduling of unrelated parallel machines, batches, and heterogeneous delivery trucks. *Applied Soft Computing, 53*, 457-476.
Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems *Annals of discrete mathematics* (Vol. 1, pp. 343-362): Elsevier.
Lin, B., & Jeng, A. (2004). Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs. *International Journal of Production Economics, 91*(2), 121-134.
Liu, L., Li, W., Li, K., & Zou, X. (2020). A coordinated production and transportation scheduling problem with minimum sum of order delivery times. *Journal of Heuristics, 26*(1), 33-58.
Liu, P., & Lu, X. (2016). Integrated production and job delivery scheduling with an availability constraint. *International Journal of Production Economics, 176*, 1-6.
Munoz-Villamizar, A., Santos, J., Montoya-Torres, J., & Alvaréz, M. (2019). Improving effectiveness of parallel machine scheduling with earliness and tardiness costs: A case study. *International Journal of Industrial Engineering Computations, 10*(3), 375-392.
Nikabadi, M., & Naderi, R. (2016). A hybrid algorithm for unrelated parallel machines scheduling. *International Journal of Industrial Engineering Computations, 7*(4), 681-702.
Ojstersek, R., Brezocnik, M., & Buchmeister, B. (2020). Multi-objective optimization of production scheduling with evolutionary computation: A review. *International Journal of Industrial Engineering Computations, 11*(3), 359-376.
Piroozfard, H., Wong, K. Y., & Wong, W. P. (2018). Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources, Conservation and Recycling, 128*, 267-283.
Potts, C. N. (1980). Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Operations Research, 28*(6), 1436-1441.
Raghavan, V. A., Yoon, S. W., & Srihari, K. (2018). A Modified Genetic Algorithm Approach to Minimize Total Weighted Tardiness with Stochastic Rework and Reprocessing Times. *Computers & Industrial Engineering*, *123*, 42-53.
Rajkanth, R., Rajendran, C., & Ziegler, H. (2017). Heuristics to minimize the completion time variance of jobs on a single machine and on identical parallel machines. *The International Journal of Advanced Manufacturing Technology, 88*(5-8), 1923-1936.
Saeidi, S. (2016). A Multi-objective Mathematical Model for Job Scheduling on Parallel Machines Using NSGA-II. *International Journal of Information Technology and Computer Science (IJITCS), 8*(8), 43-49.
Schaller, J. E. (2014). Minimizing total tardiness for scheduling identical parallel machines with family setups. *Computers & Industrial Engineering, 72*, 274-281.
Shahidi-Zadeh, B., Tavakkoli-Moghaddam, R., Taheri-Moghadam, A., & Rastgar, I. (2017). Solving a bi-objective unrelated parallel batch processing machines scheduling problem: a comparison study. *Computers & Operations Research, 88*, 71-90.
Sheikh, S., Komaki, G., & Kayvanfar, V. (2018). Multi objective two-stage assembly flow shop with release time. *Computers & Industrial Engineering, 124*, 276-292.
Shen, J. (2019). An uncertain parallel machine problem with deterioration and learning effect. *Computational and Applied Mathematics, 38*(1), 3.
Shim, S.-O., & Kim, Y.-D. (2008). A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. *Computers & Operations Research, 35*(3), 863-875.
Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2004). *Managing the Supply Chain: Definitive Guide*: Tata McGraw-Hill Education.
Sivrikaya-Şerifoğlu, F., & Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research, 26*(8), 773-787.
Thomas, D. J., & Griffin, P. M. (1996). Coordinated supply chain management. *European journal of operational research, 94*(1), 1-15.
Tyagi, R., & Gupta, S. K. (2018). A Survey on Scheduling Algorithms for Parallel and Distributed Systems *Silicon Photonics & High Performance Computing* (pp. 51-64): Springer.
Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research, 227*(1), 152-165.
Wang, D., Zhu, J., Wei, X., Cheng, T., Yin, Y., & Wang, Y. (2019). Integrated production and multiple trips vehicle routing with time windows and uncertain travel times. *Computers & Operations Research, 103*, 1-12.
Wang, D. Y., Grunder, O., & Moudni, A. E. (2014). Integrated scheduling of production and distribution operations: a review. *International Journal of Industrial and Systems Engineering, 19*(1), 94-122.
Wang, G., & Cheng, T. E. (2000). Parallel machine scheduling with batch delivery costs. *International Journal of Production Economics, 68*(2), 177-183.
Wang, S., & Liu, M. (2015). Multi-objective optimization of parallel machine scheduling integrated with multi-resources preventive maintenance planning. *Journal of Manufacturing Systems, 37*, 182-192.
Wang, S., Wu, R., Chu, F., & Yu, J. (2029). Variable neighborhood search-based methods for integrated hybrid flow shop scheduling with distribution. *Soft Computing*, *24*, 8917–8936.
Wu, X., & Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega, 82*, 155-165.
Zarei, H., & Rasti-Barzoki, M. (2019). Mathematical programming and three metaheuristic algorithms for a bi-objective supply chain scheduling problem. *Neural Computing and Applications*, *31*, 9073–9093.
Zhou, S., Li, X., Du, N., Pang, Y., & Chen, H. (2018). A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost. *Computers & Operations Research, 96*, 55-68.