Contents lists available at ScienceDirect

International Journal of Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Acceptance Ordering Scheduling Problem: The impact of an order-portfolio on a make-to-order firm's profitability

Federico Perea [a,*], Juan C. Yepes-Borrero [b], Mozart B.C. Menezes [c]

[a] Dpto. Matemática Aplicada II. Instituto de Matemáticas de la Universidad de Sevilla. Escuela Politécnica Superior. Universidad de Sevilla, Sevilla, Spain
[b] Universidad Internacional de Valencia-VIU, C/Pintor Sorolla 21, 46002, Valencia, Spain
[c] NEOMA Business School, Rue du Maréchal Juin, 76130, Mont-Saint-Aignan, France

ARTICLE INFO

Keywords:
Accepting and scheduling orders
Product proliferation
Metaheuristic
Integer programming
Value chain complexity

ABSTRACT

Firms' growth, the darling measure of investors, comes from higher revenues. Thus, sales and marketing departments make extreme efforts to accept as many customer orders as possible. Unfortunately, not all orders contribute equally to profits, and some orders may even reduce net profits. Thus, saying no (i.e., not accepting an order) may be a necessary condition for net profits growth. For understanding the impact of rejecting orders on profitability, we propose an order acceptance and scheduling problem (OAS). Although the OAS has extensively been studied in the literature, there is still some gap between these papers and real-life problems in industry. In an attempt to close that gap, the OAS we propose considers orders revenues, machines costs, holding costs and tardiness costs. We develop a mixed integer linear programming (MILP) model for solving this problem. Since the complexity of the problem makes it impossible for the MILP to solver large-scale instances, we also propose a metaheuristic algorithm. Numerical experiments show that the metaheuristic finds good quality solutions in short computational times. In the last part of the paper we confirm some managerial insights: higher holding and tardiness costs imply a lower acceptance of orders, forcing production has a concave negative impact on net profits, and accurately estimating costs is essential for good planning.

1. Introduction

In this article, we discuss on the problem of accepting and scheduling orders from customers with the goal of maximizing profits. The work is based on the Acceptance Ordering Scheduling Problem (AOS), where decision makers choose orders to accept for production from a pool of customer orders. We provide different approaches for solving the problem and, simultaneously, take a look at the effect that accepting suboptimal orders has on profitability.

Our environment is a make-to-order setting, which is different from a make-to-stock environment. In a make-to-order production process, goods are manufactured in response to real customer demand: the manufacturing process responds to the demand by producing the goods and delivering the exact quantity demanded to the customers at the origin of the order. That directly contrasts with make-to-stock environment, where the production process is triggered by speculative means (usually a forecast). In a make-to-stock environment, there is more leeway for moving around production orders and adjusting the quantities produced in each order for optimizing the response to a stochastic *speculative process*.

Assuming a make-to-order environment, suboptimal orders are orders that would not be on any optimal production plan, but are included based on other (typically nonquantitative) criteria. To illustrate, consider the following real case. A customer orders 26 times per year the same general product, but the orders' specifications (e.g., color of a detail) slightly changes through time. Consequently, these changes do not allow for making-to-stock, so orders are produced in the quantity requested at each time. Although the total annual volume would be profitable if split in two or three orders, the order size when split in 26 orders is too small to make an order profitable. However, because of the *belief* that the customer could eventually stop making changes in the specifications, allowing for a better scheduling of the orders, the company keeps accepting orders that bring negative contribution to the company's profitability.

This forced, or compulsory, inclusion of orders may increase the number of orders accepted, which originates the so called *structural complexity*. Structural complexity is related to the number of product varieties or orders in the case of a make-to-order environment (see Ruiz-Hernández et al. (2019) for more information on structural complexity). To illustrate, consider the case of Sonoco Product

* Corresponding author.
E-mail addresses: perea@us.es (F. Perea), juancamilo.yepes@universidadviu.com (J.C. Yepes-Borrero), mozart.menezes@neoma-bs.fr (M.B.C. Menezes).

https://doi.org/10.1016/j.ijpe.2023.108977
Received 23 June 2022; Received in revised form 6 June 2023; Accepted 12 July 2023
Available online 20 July 2023

Company, Industrial Core Division. This is a packaging company that throughout the years, has increased its portfolio to unsustainable levels. Competitive pressures and a lack of strategic alignment between operations and sales functions pushed Sonoco to high levels of structural complexity (see Bloemen and Menezes (2019a) and Bloemen and Menezes (2019b)). This happened until December 1, 2017, when Sonoco's CEO announced, in its New York Analysts Meeting that the company would reduce the number of stock-keeping units by 35%. This action would reduce available capacity by 30% and would still increase profits by an estimated 20%: reducing production implied an increase in profits.

Practitioners recognize that structural complexity (see, for example, Adams et al. (2016), Hirose et al. (2017), Mocker and Ross (2017)) is an important factor consistently believed to induce profitability loss. Therefore, researchers and practitioners design strategies for reducing it. Another illustrative example is Carrefour, a large French retailer, which announced in its 2017 Annual Report a decision to reverse its focus away from very large business units to chains operating with small shops since they are "less complex and more adapted to customers behaviour".

Around the same period, the Danish toymaker Lego announced large cuts, alleging bureaucracy and complexity as the root cause of its financial problems; see Mocker and Ross (2017). Lately, Coca-Cola announced that they were "slashing" nearly 50% of the total number of product lines (Grothaus, 2020). This action not only suggests that the firm was suffering the effect of complexity, but also makes us wonder why they achieved that number of products in the first place.

A possible reason to explain why companies allow the number of products and/or orders to grow too much is that growth is often measured by revenue growth, which leads the marketing and sales functions to accept as many orders as possible or to introduce as many products as possible in the portfolio; see Mariotti (2008). A second possible explanation is related to the power of retailers, that seem to have increased with time, to push firms to enlarge the product varieties of the offered portfolio. Lu and Menezes (2022), in a study based on competitive games between a manufacturer and a retailer, find that retailers' negotiating power is a main factor that leads to product-variety proliferation and consequently, structural complexity. When the retailer chooses the assortment depth but has no first-mover pricing advantage, then product proliferation arises moderately. Nevertheless, according to Lu and Menezes (2022), "However, in the case where the retailer also enjoys a high profit margin due to pricing leadership, product proliferation becomes rampant..." We follow the insights discussed in that paper and ask ourselves what happens when the buyer has sufficient negotiating power to impose orders to be satisfied, even if these same orders would never be accepted by the supplier if the optimal decision was unconstrained from the buyer power.

Another explanation is that because accurately forecasting profits is unattainable in complex systems (see Bradley (2022) for an interesting anecdote), managers substitute this performance metric for revenue, a much easier indicator to forecast. Hyndman and Menezes (2021) suggest that this form of salience bias may be common in complex environments. If this is the case, then more products imply more revenues, which is a thought shared by practitioners. Mariotti (2008) mentions that while aiming for higher profitability, firms have "proliferated nearly every-thing: products, customers, markets, suppliers… which leads to higher revenues but too often to operational profit losses." In particular, as a detail important to our work, make-to-order firms have embarked on a large expansion of their product portfolios (Menezes et al., 2021). Although there is some agreement that "overserving" customers may lead to profitability losses, it is not clear how much profitability improvement may potentially be secured by cutting back complexity.

Scheiter et al. (2007) suggests that complexity management could lead to an increase in earnings before interest and taxes (a.k.a. EBIT, a form of operational profits) from 3 to 5 percentage points. Adams et al. (2016) mention the case of a large food manufacturer facing an approximately 10% margin loss due to increased complexity.

The operations management/research literature gives emphasis on solving a problem; that is, finding an optimal solution. The academia has placed less emphasis on understanding the impact of 'pseudo-externalities' on the economics of production at operational level. Our paper brings this novelty. Instead of simply assuming a list of jobs to process, we look on the impact of filtering out profit-wise less attractive jobs. This allowance is equivalent to giving more degrees of freedom to the operations manager as opposed to the marketing/sales group that have as main objective increasing sales revenues (even though sales/marketing functions are called profit centers, they are really aiming at increasing revenues) - see Mariotti (2008) for more on those conflicting goals of marketing and sales functions. We bring to the model the ability to solve real problems: in a recent collaboration with Sonoco Industrial it was estimated a lower bound of 20% for EBIT improvements, and in another work with Danone, the food/dairy producer, we estimated potential gains of similar magnitude. These improvements contrast to most work in the literature that by focusing on reducing costs have less significant impact on the firms' profitability.

In a first work where the level of structural complexity is measured, Menezes et al. (2021) reports, in an empirical study that uses a measure for structural complexity called *pars*-Complexity proposed in Ruiz-Hernández et al. (2019), that a decrease of over 2 percentage points of the EBIT margin may be lost for each additional *pars*-Complexity point. In a simulated study, Chatha and Jalil (2022) found that "structural complexity on the demand side has a negative... influence on the operational performance of the manufacturer". They also observed that complexity has an inverted U-shaped impact on operational performance (as previously suggested by Menezes et al. (2021)) and that demand uncertainty interacts with structural complexity and reduces the overall operational performance of the manufacturer, which is a fact already noted by Menezes and Pinto (2022).

Shifting our attention from the motivational to the methodological aspect of our paper, the scheduling of parallel machines has been deeply studied in the literature. One of its simplest versions consists of deciding which machines perform which orders, see Fanjul-Peyro and Ruiz (2011). A more realistic version of this problem considers machine setups between the processing of the orders, see Fanjul-Peyro et al. (2019). Lately, more and more literature on these problems assume that the functioning of machines needs of additional resources (e.g. workers), both in the processing of the orders and on the machine setups (see e.g. Fanjul-Peyro et al. (2017), Yepes et al. (2020), Lopez-Esteve et al. (2022)). These papers impose that all orders need to be accepted. As we have discussed before, this might be suboptimal, and another decision that should be made is whether or not to accept a given order. Therefore, the OAS (Order Acceptance and Scheduling) problem has also been deeply studied in the literature. However, from what the authors know, only two OAS papers consider machine costs and only one of those two consider fixed costs. The reader is referred to Slotnick (2011) for a review. We now briefly summarize some related literature on problems similar to the one addressed in this paper. Oguz et al. (2010) propose an OAS problem on a single machine. Orders have release dates, due times and a revenue that (piecewise linearly) depends on how tardy they are. These authors also consider setups on the machine. The problem consists of deciding which orders to schedule and their sequence to maximize global revenue. No costs related to machines are considered. Tanaka (2011) shows that some scheduling problems with rejection can be converted into ordinary scheduling problems by making some changes in the order completion costs. Fanjul-Peyro and Ruiz (2012) address the not all machines (NAS) problem and the not all jobs (NAJ) problem. In NAS, only a number of the available machines can be used. In NAJ, at least a fixed number of jobs (for us, order and job are synonyms) need to be processed. No revenues or costs are involved in the objective.

Table 1
Specifications of OAS papers reviewed.

Paper	b_j	r_j	d_j	τ_j	β_j	α_i	δ_i	p_{ij}	s_{ijk}	m
Oguz et al. (2010)	X	X	X	X				X	X	1
Wang et al. (2015)	X		X	X				X		2
Xu et al. (2015)	X		X					X	X	1
Wang and Wang (2018)	X		X	X		X		X		m
Wang and Ye (2019)	X		X		X			X		m
Kong et al. (2020)		X				X	X	X		m
Li and Ventura (2020)	X		X	X				X		1
Naderi and Roshana (2020)	X		X	X				X		m
Yavari et al. (2020)	X		X	X				X		m
Bruni et al. (2020)	X							X	X	m
Tarhan and Oguz (2021)	X	X	X		X			X	X	1

Xu et al. (2015) extend (Oguz et al., 2010) by assuming the stochastic arrival times of orders. Wang et al. (2015) study an OAS with two identical parallel machines, assuming that no costs on the machine are involved, with the objective of total profit maximization. Wang and Wang (2018) consider several parallel machines. For each order, the authors consider production costs, processing times (both are machine dependent), revenues, due dates and unit tardiness costs. In the same paper, a bi-objective approach is considered to simultaneously optimize the makespan and the net profit. Wang and Ye (2019) maximize the total revenue minus a weighted tardiness in an unrelated parallel machine OAS. Kong et al. (2020) consider both orders and machine costs, with budget constraints both in energy consumption and machine launch. Li and Ventura (2020) consider an order acceptance problem with one machine and maximize total revenue minus the tardiness penalty. Naderi and Roshana (2020) address a multiple-machine order acceptance problem by assuming machines are identical. Yavari et al. (2020) study an order acceptance problem that maximizes revenues minus tardiness penalties. Bruni et al. (2020) add uncertainty to an OAS problem in which the total completion time is minimized, which guarantees a certain profit. Tarhan and Oguz (2021) propose a generalized OAS problem with a single machine in which orders are grouped on batches according to clients.

Readers may note that most of the papers related to order acceptance problems assume no costs on the machines. In an attempt to close the gap between industry and academia, in the problem addressed in this paper we do assume both fixed and variable costs on machines. More specifically, and as it will be detailed in Section 2, we consider: orders revenues, release dates, due dates, holding costs, tardiness costs, machines fixed costs, machines variable costs, processing times, and machine setup times. Table 1 in Section 2 compares the specifications of our problem with other OAS problems found in the literature.

The contribution of this paper can be summarized as:

- Algorithmic contribution: We address a new acceptance scheduling problem, which is closer to industry than other similar problems found in the literature and includes several profits and costs. As solution approaches for this problem, we propose both a Mixed Integer Linear Programming (MILP) model and a metaheuristic solution. Both approaches are tested by performing extensive numerical tests on it for assessing its performance.
- Managerial contribution: We use the metaheuristic to derive some managerial insights, among which we emphasize the following:
  - We evaluate the impact of forcing order acceptance and bring insight on the rate of increase of that impact as higher fractions of the orders are imposed.
  - We analyze holding and tardiness costs as drivers of performance.
  - Our analysis brings insights on the relationship between complexity, measured by number of orders, and operations' freedom. The analysis include the possibility that complexity leads to a mis-estimation of some cost drivers.

As explained in the experiments, we find that (as expected) by increasing the fraction of orders that must be accepted, profitability suffers. However, this is only a partial understanding: although average profitability decays, median profitability decays even more, and the interquartile range of profit outcomes shift down, which substantially increases the risk for profit losses. We also find, through numerical experiments, that profit losses concavely decrease in forced order-acceptation. That is, although a few orders imposed in the optimal solution impact just slightly, these same few orders could impact profitability substantially when added to an already suboptimal solution. This is equivalent to saying that relaxing the enforcement of a few orders in a loss situation may already make a large difference on the bottom line (i.e., profitability).

The rest of the paper is structured as follows. In Section 2 we formally state the problem treated in this paper, and formulate it as a MILP. Since that MILP can only solve small-sized instances of the problem, in Section 3, we present a metaheuristic for finding good feasible solutions in reasonable CPU times. In Section 4, we explain the set of numerical experiments conducted and assess the quality of both the MILP and the metaheuristic. In Section 5, we comment on some managerial insights, from the results obtained on another set of instances. Section 6 summarizes some conclusions and further research.

2. Problem statement

This section formally defines the problem treated in this paper. In such problem, there is a set of orders offered to a company. The company has the option of accepting or rejecting the orders. If an order is rejected, no cost or no benefit is incurred. If the order is accepted, the company receives it at a predefined point in time. Orders also come with deadlines. If an order is delivered to the client after the deadline, a penalty fee is paid per unit of time of delay. Holding costs are also paid, since the moment the order is received until delivery to the client. Successful delivery to the client implies a benefit. Orders need to be processed in one of the available machines, which can work in parallel. However, the same machine cannot process two orders at the same time. Machines have associated costs: fixed cost of starting the machine, plus variable costs which depend on the time the machines are processing orders. Machines also need to be adjusted between the processing of two consecutive orders (setups).

The problem consists of deciding

- which orders to accept,
- when to process the accepted orders (time), and
- where to process the accepted orders (machine),

so that net profits (revenues minus costs) are maximized

In the rest of this section we formally define the problem, by proposing a mixed integer linear programming (MILP) model. We will start by defining its input data, followed by variables, objective function and constraints.

2.1. Input data

We begin by defining the sets and indices needed for the formulation proposed:

- $N = \{1,\dots,n\}$ is the set of orders, indexed by j and k. For modeling purposes, we add a dummy order 0 and denote $N_0 = N \cup \{0\}$.
- $M = \{1,\dots,m\}$ is the set of machines, indexed by i.

For each order j, the following input data are needed.

- $b_j \geq 0$ is the revenue obtained if it is processed.
- $r_j \geq 0$ is the time when it is received, also known as *release time*. From that moment on, order j can be processed.
- $d_j \geq 0$ is its due date. Accepted orders cannot be delivered before their due date, and if they are delivered after the due date, a penalty cost is incurred.
- $\tau_j \geq 0$ is its unitary holding cost. Note that this cost is incurred from the release time (r_j) until the delivery to the client.
- $\beta_j \geq 0$ is the unitary tardiness penalty. Note that this penalty is incurred from the due date (d_j) until the order is delivered to the client.

In addition, for each machine i, the following input data are given.

- $\alpha_i \geq 0$ is its processing cost per time unit.
- $\delta_i \geq 0$ is the fixed cost incurred if using it.
- $p_{ij} \geq 0$ is the processing time of order j on it.
- $s_{ijk} \geq 0$ is its setup time, which is needed after processing order $j \in N_0$ if order $k \in N$ is the next order processed on this machine. Note that s_{i0j} is the initial setup needed if the first order processed by this machine is order j.

Table 1 summarizes the main characteristics of the OAS problems treated in the papers revised in the Introduction. For each of the characteristics considered in our problem (by columns), we insert an "X" if the corresponding paper (by rows) does consider it. We do not show any column for the processing times p_{ij}, since all papers analyzed do consider such processing times. The last column refers to the number of machines considered, which is divided into three levels: 1, 2 or any m (no constraint on the number of machines). Note that none of the papers considers all the specifications we include in our problem.

2.2. A mathematical formulation

In this section, we provide an MILP model for our problem. For this, we need the following sets of binary variables and nonnegative variables.

Binary variables:

- Y_{ij} takes a value of one if machine i processes order j and zero otherwise.
- X_{ijk} takes a value of one if orders j and k are consecutively processed on machine i. In this case, we say that $j-k$ are consecutive: j is the predecessor of k or, equivalently, k is the successor of j. Otherwise, the variable takes a value of zero.
- V_i takes a value of one if machine i is activated and zero otherwise.
- W_j takes a value of one if order j is accepted and zero otherwise.

Nonnegative variables:

- $C_j \geq 0$ is the completion time of order j.
- $D_j \geq d_j$ is the moment when order j is given to the client and is defined in the model as the maximum between C_j and d_j (the delivery time).

Note that the delivery date D_j does not necessarily have to coincide with the completion time C_j, since orders processed before their due date need to wait until such due date to be delivered to the client. However, we need to impose that $D_j \geq C_j$ for accepted orders as a constraint in our MILP model. The model consists of the following:

$$\max \sum_{j\in N} b_j W_j - \sum_{i\in M, j\in N} \alpha_i p_{ij} Y_{ij} - \sum_{i\in M} \delta_i V_i - \sum_{j\in N} \tau_j((d_j - r_j)W_j + (D_j - d_j)) - \sum_{j\in N} \beta_j (D_j - d_j) \quad (1)$$

$$\text{s.t.: } \sum_{k\in N} X_{i0k} = V_i, \; i \in M \quad (2)$$

$$\sum_{i\in M} Y_{ij} = W_j, \; j \in N \quad (3)$$

$$Y_{ij} = \sum_{k\in N_0\setminus\{j\}} X_{ijk}, \; i \in M, j \in N \quad (4)$$

$$Y_{ik} = \sum_{j\in N_0\setminus\{k\}} X_{ijk}, \; i \in M, k \in N \quad (5)$$

$$C_j \geq r_j + \sum_{i\in M} p_{ij} Y_{ij}, \; j \in N \quad (6)$$

$$C_k \geq C_j + s_{ijk} + p_{ik} - \bar{M}(1 - X_{ijk}), \; j \in N_0, k \in N \setminus \{j\}, i \in M \quad (7)$$

$$D_j \geq C_j - \bar{M}(1 - W_j), j \in N \quad (8)$$

(1) defines the objective, which consists of maximizing

- total revenue from the orders accepted ($B = \sum_{j\in N} b_j W_j$),
- minus machine variable costs ($V = \sum_{i\in M, j\in N} \alpha_i p_{ij} Y_{ij}$),
- minus machine fixed costs ($F = \sum_{i\in M} \delta_i V_i$),
- minus order holding costs ($H = \sum_{j\in N} \tau_j((d_j - r_j)W_j + (D_j - d_j))$),
- minus order tardiness costs ($T = \sum_{j\in N} \beta_j (D_j - d_j)$).

Constraint (2) ensures that there is a first order processed on machine i only if this machine is activated. (3) states that one machine processes order j if and only if such an order is accepted. These last two sets of constraints ensure that the orders that are accepted are processed by exactly one machine. (4) ensures that if j is processed by i, then j should have exactly one successor on this machine (which could also be the dummy order). (5) ensures that if k is processed by i, then k should have exactly one predecessor on this machine (which could also be the dummy order). (6) state that the completion time of order j should be, at least, its release date plus the processing time on the machine on which it is processed. (7) are MTZ-like subtour elimination constraints that allow us to find the completion time of orders, where $\bar{M}$ is a sufficiently large constant. Finally, constraints (8) impose that the delivery time of accepted order j is, at least, its completion time.

Note that if order j is not accepted, and therefore $W_j = 0$, then (8) do not impose any value on D_j. Therefore, this variable D_j is only constrained by its definition domain, $[d_j, +\infty)$. Since both τ_j and β_j are positive, from the objective function (1) we deduce that an optimal solution would set D_j as small as possible, and therefore we conclude $D_j = d_j$ in these cases.

2.2.1. Complexity

This problem is quite complex, since a much simpler version of it, in which no benefits or costs are considered and the only decision is how to schedule the orders, is already NP-hard (see Garey and Johnson (1979)). Therefore, the problem proposed is also NP-hard.

3. Metaheuristic algorithm

As we will see in the experiments section, the MILP proposed before is only able to solve small-sized instances of our problem. Therefore, in this section we propose a semi-greedy heuristic as an alternative method, see Hart and Shogan (1987), which is expected to provide good feasible solutions to large-scale instances in short computational times.

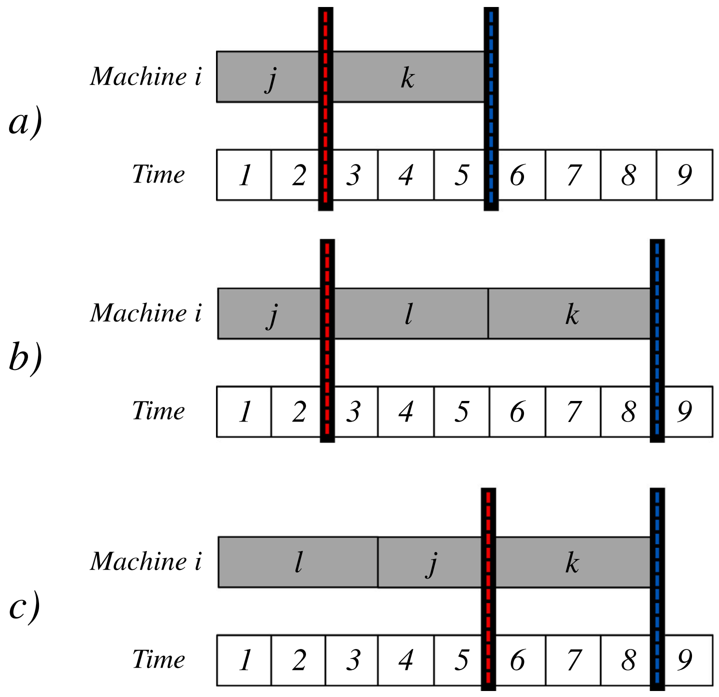

Fig. 1. Example of an insertion into the partial solution.

Semi-greedy algorithms are GRASP algorithms (as introduced in Feo and Resende (1995)) without the local search phase. In our algorithm, preliminary tests with a local search phase did not show improvements in solutions, due to the fact that small changes in the final solution of our semi-greedy heuristic, most of the time generate worse solutions. In addition, local search phases are time-consuming. Since the proposed semi-greedy algorithm is very fast and provides good-quality solutions (see the experiments section), we decided to use all available CPU time to generate different feasible solutions by applying randomization over the constructive rule, without the local search. The results of this preliminary test, comparing our heuristic with and without the local search phase on a set of 90 instances, can be found in the appendix.

3.1. Constructive phase

During the execution of the constructive phase, we define the set of pending orders as the set of orders that have not yet been assigned to a machine. Note that this set changes during the execution of the algorithm. Originally, the set of pending orders is N. We define an *insertion* as a 3-tuple (i, j, k), which summarizes on which machine (i) and at which position (k) the order (j) is inserted in the *current solution*. Given a current solution (which is originally empty), the constructive phase consists of testing the insertion of each pending order into each possible position on each machine. At each possible insertion (i, j, k), we compute a value that we call λ_{ijk} profit, which measures the impact that such machine-order-position insertion will have on the objective value. The insertion with the best λ profit is chosen, and the partial solution is updated. This value is computed as follows:

$$\lambda_{ijk} = b_j - \tau_j(D_j - r_j) - \beta_j(D_j - d_j) - \alpha_i p_j - \gamma_i \delta_i \theta - \tau'_{ijk} + \beta'_{ijk}.$$

Some of the parameters involved in this definition are introduced in Section 2. The others are defined as follows.

- γ_i is a new parameter that takes a value of one if machine i does not have any order assigned yet and zero otherwise. In this way, we only consider the fixed cost of machine i, δ_i, the first time this machine is assigned an order in the algorithm.
- θ is a factor to account for the fixed cost. During the algorithm, this value changes and takes one of the following three values: $\{\theta_1 = 0;\ \theta_2 = \frac{1}{nm};\ \theta_3 = 1\}$. Note that $\theta = 0$ means that the fixed cost of machines is not taken into account for the computation of λ. Note also that θ is used internally in the algorithm, and regardless of its value, in the final solution, the fixed costs are considered.
- τ'_{ijk} is the increase in the holding costs if we insert order j on machine i into position k.
- β'_{ijk} is the increase in the tardiness costs if we insert order j on machine i into position k.

Fig. 1 shows an example of an insertion into a partial solution. In Fig. 1(a), the partial solution has orders j and k assigned. We observe that order j ends at time 2 and that order k ends at time 5. Note that if we insert order l after order j (Fig. 1(b)), then order k ends at time 8, which increases the holding cost for this order (τ') and possibly also the tardiness cost (β'). Finally, in Fig. 1(c), we observe that if we insert order l before orders j and k, then both orders will finish later. In this case, the value τ' is the sum of the increment in the holding cost of orders j and k. Similarly, the β' value is the sum of the increment (if any) in the tardiness cost of orders j and k.

At each iteration, we find the list of insertions that have a λ value greater than or equal to $-\eta$, where η is a threshold factor that is calibrated in the experiments section. This list of insertions is called the *Candidate List*, or CL, and depends on the value of η:

$$CL(\eta) = \{(i, j, k) : \lambda_{ijk} \geq -\eta\}. \tag{9}$$

In this heuristic algorithm, we choose the insertion in the CL with the largest λ value.

After an insertion is chosen and the partial solution is updated (the order is assigned and removed from the list of pending orders), we repeat the process with the remaining pending orders. Then, we repeat this process with the remaining pending orders until no feasible insertion generates a λ value greater than or equal to $-\eta$, and in this case, the algorithm ends.

3.2. Randomization

The previous constructive phase builds a feasible solution in a greedy way: at every iteration, the best insertion (the insertion with the largest λ value) is chosen. To further explore the feasible region, we use randomization when choosing the insertion. Instead of choosing the insertion with the largest λ value in the CL, we randomly choose one element from a list formed by the insertions with λ value greater than a given threshold (named Restricted Candidate List, or RCL). The RCL depends on a parameter $\rho \in [0,1]$ as follows:

$$RCL(\rho,\eta) = \{(i,j,k) \in CL(\eta) : \lambda_{ijk} \in [\lambda_{\max} - \rho(\lambda_{\max} - \lambda_{\min}), \lambda_{\max}]\}, \quad (10)$$

where $\lambda_{\min} = \min_{(i,j,k)\in CL(\eta)} \lambda_{ijk}$ and $\lambda_{\max} = \max_{(i,j,k)\in CL(\eta)} \lambda_{ijk}$. Note that the best value in $CL(\eta)$ would be $\lambda_{\max}$, and the worst value in $CL(\eta)$ would be $\lambda_{\min}$. Note also that when ρ is closer to 1, the size of the RCL is larger. In addition, $\rho = 0$ means that the algorithm is completely greedy (we only keep the insertion with the best value), and $\rho = 1$ means that the algorithm is completely random (considering only insertions that yield a λ value greater than the threshold η).

The value of ρ is calibrated in the experiments section. The algorithm is run during the available CPU time (denoted as $t_{\max}$), and we keep the best solution obtained.

To avoid getting stuck in local optima, we keep track of the number of iterations without an improvement in the objective. If $I \in \mathbb{Z}^+$ iterations of the algorithm are run without improvement in the objective, then the value of θ is updated. The algorithm begins with $\theta = \theta_1$ and, after I iterations without improvement, we update $\theta = \theta_2$. After another I iterations without improvement, we update $\theta = \theta_3$, which remains with the same value until the rest of the run. The value of I is calibrated in the experiments.

Algorithm 1 shows a pseudocode of the metaheuristic proposed. In this pseudocode, we use the following new notation: $BestVal$ refers to the best objective value found; no_imp refers to the number of iterations without finding a better solution; $PendingOrderList$ is the list of orders that have not been assigned to a machine yet; $added$ is a binary variable to check if at least one insertion can be added to the CL; $MachinePosition_i$ is the list of potential positions of machine i where a new order can be inserted; and $NetBenefit$ is the objective value of the feasible solution found. We note that if θ has been updated to θ_3, then θ remains with the same value for the rest of the available time.

```
$BestVal = 0$
$no\_imp = 0$
$\theta = \theta_1$
while $time < t_{\max}$ do
    $no\_imp++$
    $PendingOrderList = N$
    $CL(\eta) = \emptyset$
    $added = 1$
    while $added = 1$ do
        $added = 0$
        foreach $j \in PendingOrderList$ do
            foreach $i \in M$ do
                foreach $k \in MachinePosition_i$ do
                    Compute $\lambda_{ijk}$
                    if $\lambda_{ijk} > -\eta$ then
                        $CL(\eta) = CL(\eta) \cup \{(i,j,k)\}$
                        $added = 1$
                    end
                end
            end
        end
        if $added = 1$ then
            Compute $RCL(\rho,\eta)$
            Randomly choose one element $(i^*, j^*, k^*) \in RCL(\rho)$, and
              update the current solution accordingly
            $PendingOrderList = PendingOrderList \setminus \{j^*\}$
        end
    end
    Compute value of complete solution: $NetBenefit$
    if $NetBenefit > BestVal$ then
        $BestVal = NetBenefit$
        $no\_imp = 0$
    end
    if $no\_imp = I$ and $\theta \neq \theta_3$ then
        Update $\theta$
        $no\_imp = 0$
    end
end
Return $BestSol$
```

Algorithm 1: Pseudocode of the metaheuristic algorithm.

4. Experiments

This section summarizes the computational experiments performed in order to assess the performance of the MILP, and the quality of the solution returned by the metaheuristic. For this aim, random instances were generated with number of orders $n \in \{10, 15, 20\}$ and number of machines $m \in \{2, 3, 4\}$. Each of the 9 possible combinations of n and m was replicated 20 times randomly, which results in a total of $9\times20 = 180$ instances. These instances were divided into two groups, namely, a training set (where the metaheuristic algorithm was calibrated) and an evaluation set (where the metaheuristic algorithm and the MILP were compared). The input data were generated following the distributions detailed below ($\mathcal{U}[u,\ell]$ denotes a discrete uniform distribution between u and ℓ):

1. $p_{ij} = \mathcal{U}[10, 50], \ \forall\, i, j.$
2. $s_{ijk} = \mathcal{U}[10, 50], \ \forall\, i, j, k.$
3. $b_j = \mathcal{U}[300, 350], \ \forall\, j.$
4. $r_j = \mathcal{U}[0, 20], \ \forall\, j.$
5. $d_j = r_j + \mathcal{U}[0, 100], \ \forall\, j$, in this way the due date (d_j) is not before the release date (r_j).
6. $\tau_j = \mathcal{U}[1, 3], \ \forall\, j.$
7. $\beta_j = \mathcal{U}[10, 30], \ \forall\, j.$
8. $\alpha_i = \mathcal{U}[1, 10], \ \forall\, i.$
9. $\delta_i = \mathcal{U}[50, 100], \ \forall\, i.$

We chose the parameters in these ranges to ensure some variability in the results with respect to the number of orders accepted, the number of orders delayed, etc. We consider the initial setup of machine i for order k to be s_{ikk}.

Besides, and in order to better assess the performance of our metaheuristic, we also propose a set of large instances. In this set, the number of jobs varies in $n \in \{60, 80, 100, 120\}$, and the number of machines varies in $m \in \{8, 12, 16\}$. Each combination of n and m is randomly replicated 10 times, the input data for these instances being randomly generated in the same way as explained before. Therefore, this set contains a total of $4 \times 3 \times 10 = 120$ instances.

The MILP was run on a Windows 10 computer with 8 GB of RAM memory and an AMD Opteron 2600 MHz processor with two cores, programmed in GAMS and solved with CPLEX 12.9. The metaheuristic algorithm was coded in Microsoft Visual Studio 2019 using C# and run on virtual machines with 2 virtual processors and 8 GB of RAM memory, which are virtualized in an OpenStack virtualization framework supported by 12 blades. Each blade has four 12-core AMD Opteron Abu Dhabi 6344 processors running at 2.6 GHz and 256 GB of RAM. No parallel computing is performed.

Table 2
Calibration of the size of the RCL.

ρ	0	0.25	0.5	0.75	1
mean	431.55	485.69	489.31	488.86	487.29

Table 3
Calibration of the maximum number of instances without improvement.

I	100	100 nm	1000 nm
mean	475.2218	477.2676	477.1293

Table 4
Calibration of the threshold.

η	0	10	50	100	200
mean	498.4600	497.8126	491.3163	469.9874	425.1215

4.1. Results over the training set

In this section, we calibrate some parameters of the metaheuristic algorithm.

- The size of the restricted candidate list ρ, for which the following five levels are tested: $\{0, 0.25, 0.5, 0.75, 1\}$.
- The number of iterations without improvement I, for which the following three levels are tested: $\{100, 100 \times n \times m, 1000 \times n \times m\}$.
- The threshold η, for which the following five levels are tested: $\{0, 10, 50, 100, 200\}$.

Each of the 90 instances was run for each possible combination of the 3 factors tested, which resulted in a total of $90 \times 5 \times 3 \times 5 = 6750$ runs of the metaheuristic algorithm.

In order to calibrate the metaheuristic, an analysis of variance (ANOVA) was performed, with the objective value as the response variable, the three parameters tested as factors, removing the effect on the variability produced by the instances themselves. The results showed that both ρ and η significantly affect the average objective value (p-value $< 2e{-}16$). However, the number of iterations without improvement does not seem to affect the average objective value (p-value $= 0.361$). We now analyze the three factors independently.

4.1.1. Calibration of the RCL size (ρ)

Table 2 shows the average results for each value of ρ. We observe that the best average value is found for $\rho = 0.5$. Therefore, this level was used to assess the quality of the metaheuristic algorithm in the evaluation set of instances.

4.1.2. Calibration of the maximum number of instances without improvement (I)

Table 3 shows the average objective value for each of the values tested in this factor. Although we have seen that the number of iterations does not significantly affect the average objective value, we chose for the evaluation set the value $100nm$, as it showed the best average results.

4.1.3. Calibration of the threshold (η)

Table 4 shows the average results obtained for the different values of η. We observe that the metaheuristic yields the best average results when this parameter is set to 0. Therefore, we chose this value the metaheuristic will be run with this value in the evaluation set.

4.2. Results over the evaluation set

We now evaluate both the MILP and the metaheuristic over the evaluation set of instances.

4.2.1. Results of the MILP

Table 5 shows the average results obtained by the MILP, grouped according to the number of orders (n) and the number of machines (m). The other columns of the table are the terms that constitute the objective function: Z = the global objective value, B = gross benefit, V = machine variable cost, F = machine fixed cost, H = holding costs and T = tardiness costs. We also show in the last column the percentage of rejected orders in the solutions found. We observe that there is a proportion of rejected orders, which means that the expenses that would be generated when processing them do not compensate for the benefit returned. We also observe that there are tardiness costs, which means that there are orders that cannot be delivered on time, and they are still profitable.

From a computational perspective, we observe in Table 6 that, except in nine instances, the MILP reaches the optimal solution in less than one hour. The nine instances in which the optimality was not found belong to the groups with $n = 20$ orders and $m = \{3, 4\}$ machines.

We run these nine instances again by allowing three hours of CPU time, with the results shown in Table 7. Instance names reflect the number of orders, number of machines, and instance number within its group (in this order). In the table we detail, for each instance, the objective value (Z), the upper bound (UB) and the MILP GAP for both the run with one hour of the maximum time (subindex 1) and the run with three hours of maximum time (subindex 3). The last two columns show the improvements in the objective value and GAP.

We observe an improvement in the upper bounds and, therefore, in the gaps of the nine instances. However, we also observe that only in three of the nine instances there was an improvement in the objective value. This fact suggests that we may have reached the maximum size for which optimal solutions to this problem can be found in a reasonable amount of time.

4.2.2. Metaheuristic results

In Table 8, we see the average results obtained by the metaheuristic algorithm with the optimal factor levels found in the training set ($\rho = 0.5$, $I = 100nm$, $\eta = 0$). For each instance, the metaheuristic algorithm was run for a maximum time of mn seconds. For each group of instances, column RPD shows the average relative percent deviation of the solution, which is computed as follows:

$$RPD = 100\frac{Z_{MILP} - Z_{metaheuristic}}{Z_{MILP}},$$

where Z_{METHOD} refers to the best objective value found by the corresponding method. Column OPT_MH(%) shows the percentage of optimal solutions found by the metaheuristic algorithm (among those instances in which the MILP did prove optimality). Finally, Column t_Best (Sec.) shows the average time when the algorithm found its best solution.

It can be observed that the average RPDs are quite low (all of them were less than 1%, except in the group of 20 orders and 2 machines). In addition, for the groups of instances in which the MILP did not find all optimal solutions, the metaheuristic algorithm already yields better objective values on average (negative RPD). Additionally, in terms of optimal solutions found, the metaheuristic reports satisfactory results and finds on average 95% of the optimal solutions in the instances tested. In addition, even though the metaheuristic algorithm was run for mn seconds, it found the best solution quite rapidly (for most groups of instances, an average of less than a second), as opposed to the much longer running times needed by the MILP.

4.3. Results over the large set

In this section we test the performance of the metaheuristic over the set of large instances. It is worth mentioning that the MILP cannot find a feasible solution in one hour, even for the smallest group of instances of this set. This set of larger instances also allows us to compare

Table 5
Summary of results for MILP.

n	m	Z	B	V	F	H	T	Rejected (%)
10	2	232	716.6	143.6	109.9	221.3	9.8	78
15	2	397.6	1172.5	246.6	131.2	319.2	77.9	76
20	2	321.2	1252.8	377.1	141.5	392.3	20.7	81
10	3	425.9	1451.6	376.5	202.4	431.4	15.4	56
15	3	452.8	1624.4	401.6	213.3	535	21.7	67
20	3	737.5	2120.4	498.5	218.7	637	28.7	67
10	4	515.5	1651.6	353.1	220.9	515.7	46.4	49
15	4	595.6	1805.2	404.8	248	516.9	39.9	63
20	4	790.1	2420	546.3	269.9	777.3	36.4	63
Average		496.5	1579.5	372.0	195.1	482.9	33.0	66.7

Table 6
Summary of the time and GAP results for MILP allowing for a time limit of 60 min of CPU time.

n	m	Z	Seconds	GAP (\%)
10	2	232	2	0.0
15	2	397.6	24	0.0
20	2	321.2	378	0.0
10	3	425.9	3	0.0
15	3	452.8	44	0.0
20	3	737.5	1815	8.8
10	4	515.5	5	0.0
15	4	595.6	123	0.0
20	4	790.1	2859	31.5
Average		496.5	583.6	4.5

our metaheuristic with a state-of-the-art algorithm for a related problem, adapted to our problem. More specifically, we have adapted the memetic algorithm proposed in He et al. (2019). In brief, this algorithm combines a biased random-key genetic algorithm (see Gonçalves and Resende (2011)) and a large neighborhood search algorithm (see Demir et al. (2012)). The main change we had to make to adapt this algorithm was in the crossovers of the genetic algorithm, since the original algorithm was proposed for a single machine problem, and our problem has multiple machines. For this, we use the crossover operator proposed in Vallada and Ruiz (2011), since it showed very good results in a parallel machine problem with sequence-dependent setup times.

Table 9 shows the average RPD of the two algorithms at the different instance sizes (20 instances per group). For each group of instances, column $RPD\_MA(\%)$ shows the average RPD of the memetic algorithm and column RPD_SG shows the average RPD of our semi-greedy algorithm. The last row shows the average RPD over all instances in this set. We observe that for all groups of instances, the semi-greedy algorithm proposed in this paper yields better results than the memetic algorithm with the same computing time, being the average RPD of the memetic 11.59% and the average RPD of our algorithm 0.01%. We also observe that in all groups except 80 × 16 and 120 × 16, the average RPD of our algorithm is 0.00 (meaning that our algorithm always found the best solution). In the other two groups, the average RPD of our algorithm is 0.07%. We checked that only in one instance of each of those group of instances, our algorithm returned a worse solution than the memetic algorithm.

5. Managerial insights

To derive managerial insights, we generate 100 new instances inspired by the mentioned problem at Sonoco Company (Bloemen and Menezes, 2019a,b). All of them have $m = 5, n = 30$. The rest of the input data are generated as follows.

- Processing times. Let $p_j = zeta(shape = 1.5)$. This distribution represents a Pareto-like distribution for the order size: few orders are large, and many orders are small (the case of Sonoco company). We define $p_{ij} = p_j + \mathcal{U}[0,1]$. In this way, we add some noise, which means that not all machines need the same processing time for the same order.
- Setup times. Let $s_{jk} = \mathcal{U}[1,3]$, from which we set $s_{ijk} = s_{jk} + \mathcal{U}[0,1]$. In this way, the machines are unrelated.
- Revenues: $b_j = 100p_j$. Revenue depends on the order size.
- Release times: $r_j = \mathcal{U}[0,5]$. Release times are randomly distributed from 0–5.
- Due dates: $d_j = (r_j + p_j)\mathcal{U}[1,2]$. Due dates depend on the release dates plus the base processing time and are multiplied by 1 or 2 (chosen randomly).
- Holding cost: $\tau_j = \mathcal{U}[1,3]$. The holding cost is randomly chosen.
- Tardiness penalty cost: $\beta_j = \tau_j + \mathcal{U}[1,3]$. The tardiness cost is larger than the holding cost.
- Variable cost of machines: $\alpha_i = \mathcal{U}[70,90]$.
- Fixed cost of machines: $\delta_i = \mathcal{U}(50,100)$.

Machine costs (both variable and fixed) are chosen so that net profit is approximately 80% of revenues.

In the rest of this section, we perform a sensitivity analysis on the holding cost (τ_j) and tardiness penalty cost (β_j) as follows. We simulate that both of these costs are estimated wrongly, by modifying them. For each instance, we generate 49 new instances, in which $\tau_j = \tau_j\gamma_t$ (with $\gamma_t \in \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75\}$) $\beta_j = \beta_j\gamma_b$ (with $\gamma_b = \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75\}$). In this way, for each instance, we have a 7 × 7 table. In each cell, we modify the reference instances (reference instances are those in which $\gamma_t = \gamma_b = 1$).

We will see that changes in these parameters (the holding cost and penalty cost) imply substantial changes in the solutions. Besides, we will also check that forcing orders to be accepted has a negative impact on net profit, and that such impact follows a concave function.

5.1. Impact of holding and tardiness costs

The distribution of profits for a given combination of tardiness and holding costs can be seen in Fig. 2. Figs. 2 (1a), (1b), and (1c) show box-and-whisker plots of profits on the y-axis for the problems defined above. In each of the three plots, we fix holding (tardiness) costs, which are represented by a dark (light) box plot, to the value shown on the legend on the right, while in the x-axis, we vary the tardiness (holding) costs. These box-and-whisker plots show up through the third quartile completely but cut short of showing the last quartile. From Figs. 2 (1a), (1b), and (1c), we can see that profits can be as low as zero. The median profit, which is marked by a white horizontal line, shows the expected trend that as holding and tardiness costs increase, then median profits decrease.

Figs. 2 (1d), (1e), and (1f) show the corresponding acceptance fraction of the total orders (vertical axis) when optimally choosing orders that maximize profits. The same lines of the plots on the left column show the accepted fraction of orders on the y-axis for the problems defined above. In each of the three plots, we fix holding (tardiness) costs, which are represented by a dark (light) box plot, to the value shown on the legend on the right, while in the x-axis, we vary the tardiness (holding) costs. The conclusion is that for lower tardiness

Table 7
Improvement when increasing the maximum CPU time allowed from 60 to 180 min.

Instance	Z_3	UB_3	GAP_3	Z_1	UB_1	GAP_1	$Z_3 - Z_1$	$GAP_1 - GAP_3$
I_020_03_6	1482	2364.23	37.32	1482	2442.29	39.32	0	2.00
I_020_03_9	1313	2255.38	41.78	1190	2341.03	49.17	123	7.38
I_020_04_0	636	636.00	0.00	636	983.58	35.34	0	35.34
I_020_04_1	1021	1626.44	37.22	1021	1723.65	40.77	0	3.54
I_020_04_3	1197	1594.24	24.92	1197	1814.70	34.04	0	9.12
I_020_04_4	1177	2074.34	43.26	1119	2149.66	47.95	58	4.69
I_020_04_5	1095	2277.06	51.91	1095	2342.32	53.25	0	1.34
I_020_04_8	774	1294.47	40.21	717	1463.32	51.00	57	10.79
I_020_04_9	767	1461.85	47.53	767	1637.42	53.16	0	5.63

Table 8
Summary of the results obtained by the metaheuristic algorithm.

n	m	t	RPD (%)	OPT_MH(%)	t_Best (Sec.)
10	2	20	0.00	1	0.01
15	2	30	0.00	1	0.04
20	2	40	2.90	0.9	0.02
10	3	30	0.16	0.9	0.12
15	3	45	0.00	1	0.77
20	3	60	−0.72	0.875[a]	0.20
10	4	40	0.60	0.9	0.12
15	4	60	0.00	1	0.58
20	4	80	−0.31	1[a]	1.47
Average			0.29	0.95	0.37

[a]The average was computed over the instances that were solved to optimality by the MILP.

Table 9
Summary of results obtained by the metaheuristic algorithms over the large set.

n	m	t	RPD_MA (%)	RPD_SG (%)
60	8	60	8.36	0.00
80	8	80	16.07	0.00
100	8	100	8.88	0.00
120	8	120	12.91	0.00
60	12	60	11.63	0.00
80	12	80	13.29	0.00
100	12	100	11.55	0.00
120	12	120	11.97	0.00
60	16	60	9.79	0.00
80	16	80	8.54	0.07
100	16	100	14.42	0.00
120	16	120	11.66	0.07
Average			11.59	0.01

and holding costs, there is a relatively high acceptance of orders (close to 75% of them), while when these costs are higher, the acceptance ratio goes down to 25%.

Fig. 2 shows unconstrained situations. That is, decision makers choose freely which orders to accept or not to optimize the firm's profitability. Such an unconstrained situation may not be the case, because some orders need to be accepted regardless of their impact in the final profits, and this is what we discuss next.

5.2. Required acceptance

As discussed in the Introduction, product proliferation permeates many industries. Menezes et al. (2021) presents the case of a firm in the packaging industry where the number of products grew one order of magnitude in the last two decades. In this case, the firm, under growth pressure, was accepting all orders that the sales department would bring to the productive units.

To ellaborate on this fact, we solve the same set of instances and fix both tardiness and holding cost factors to the value of 1. Besides, we assume that a certain fraction $f \in \{0.0, 0.25, 0.50, 0.75, 1.00\}$ of the orders (randomly chosen) are compulsorily accepted. The other orders are chosen so as to maximize the net profit.

Fig. 3 shows the average profits on the y-axis, as a fraction of the unconstrained profit, for different levels, on the x-axis, of compulsory order acceptance. When the level of compulsory acceptance is 25%, average profits fall to approximately 85% of the unconstrained levels and continue to fall for a higher level of compulsory acceptance until the profits become negative (losses) when it is compulsory to accept all orders. This result reflects what the authors of this article have observed in several companies that they have recently worked with, see Menezes et al. (2021): Profits decay in an increasing loss manner as the constraint tightens the ability of the firm to choose the orders that better fit their situation (capacity installed, tardiness and holding costs, and setup costs).

Consequently, as seen in Fig. 3, profitability decays when companies are forced to accept orders. One common behavior that we encounter too often is when customers have sufficient power to impose a series of less profitable orders. This often happens because they also order profitable orders. Nonetheless, Fig. 3 does not tell the whole story.

Fig. 4 shows, in an analogous plot to the one presented in Fig. 3, the distribution of profits for different levels of order acceptance through box-and-whisker plots. The white line in each box represents the median values, and the small dark square marks the mean values (the values shown in Fig. 3). When the level of compulsory acceptance is zero, profits are at the maximum level and serve as a benchmark for which we normalize all profits.

The average values are similar to Fig. 3, but the distribution of profits presented in Fig. 4 shows that even for relatively low values of compulsory acceptance, the risk of losses is already important. When 25% of orders are compulsorily accepted, there is already over 0.25 probability of losses (profits below the dot-dashed line that marks zero profits). When this fraction increases to 50%, almost half of the outcomes imply losses.

Our insights in this subsection are mediated by different values of tardiness and holding costs. In the results presented in Fig. 4, both tardiness and holding cost factors are 1. However, it is clear from Fig. 2 ($1d$), ($1e$), ($1f$) that optimal solutions have fewer orders in the portfolio when these factors are higher. That is, when the factors are higher, the total fraction of products produced is lower, the average profits are lower (an unsurprising result), and the risk of losses is higher.

These results not only show that risk is substantially increased but also suggest that business unit performance is less dependent on managerial decisions and more dependent on the chance of particular pressures for accepting orders. That is, companies with many comparable business units, such as those presented in Menezes et al. (2021), are likely to present a wide range of performance among the plants. The increase in orders in the order portfolio, when compared to Fig. 2-($1e$), which is also said to increase structural complexity, has been observed by Hyndman and Menezes (2021) to increase the variability of the quality of decision making, which compounds the problem because the increase in range reported herein is induced by the compulsory nature of orders but is fully optimized otherwise.

Fig. 5 shows the distribution of the fraction of accepted orders and the fraction of profits with respect to the unconstrained instance, when 25% of the orders are compulsory and different levels of tardiness and

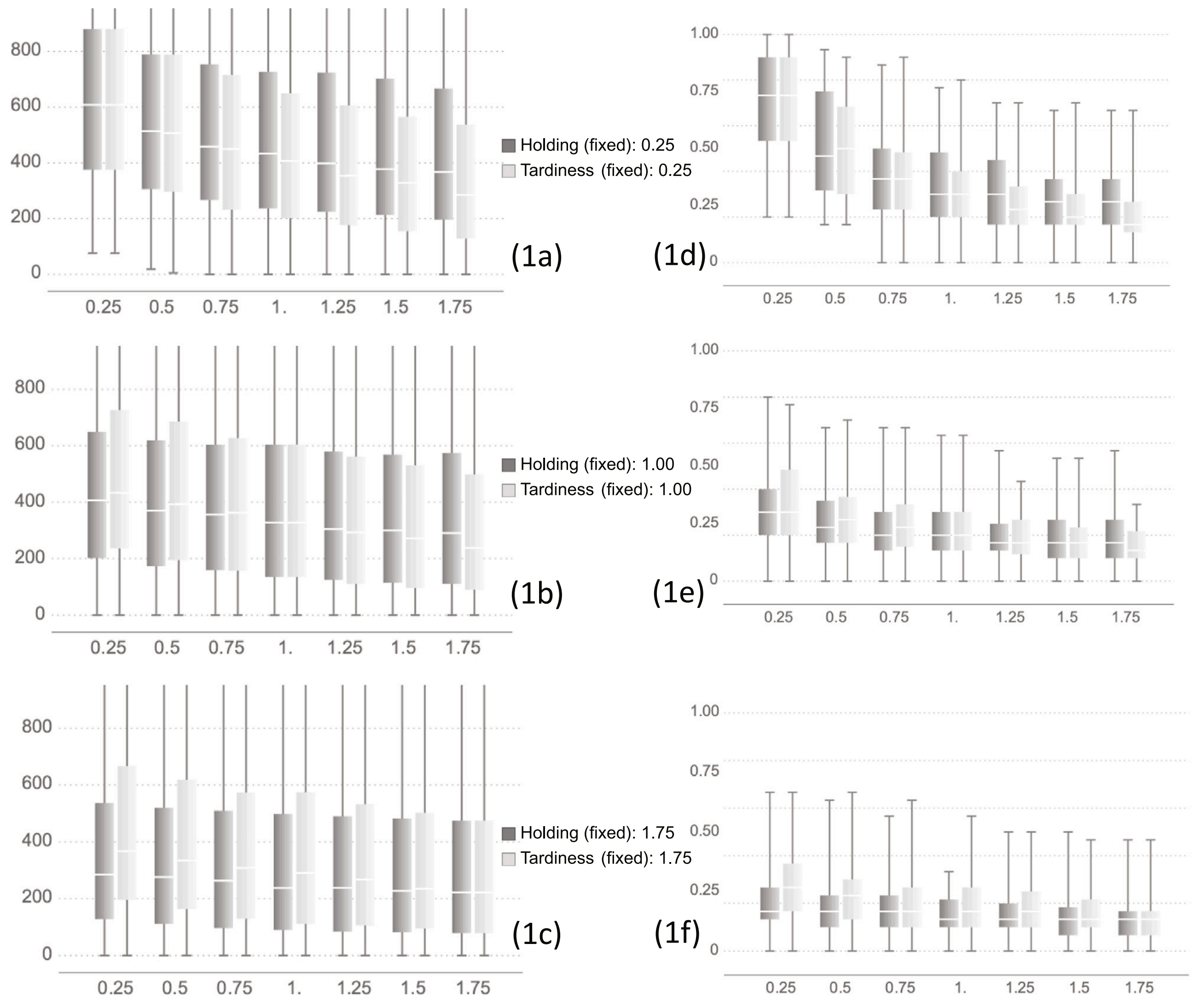

Fig. 2. Profit for different holding and tardiness costs.

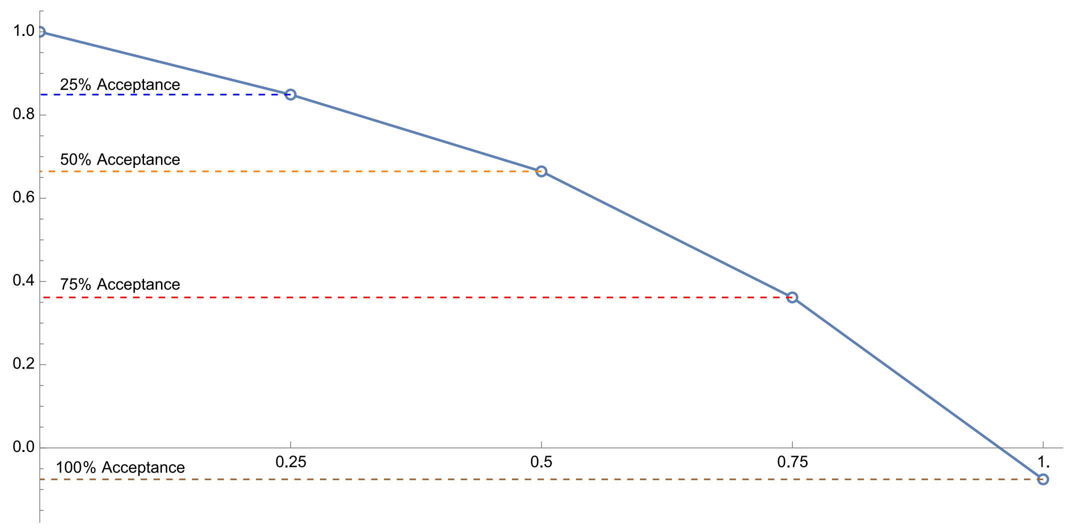

Fig. 3. Average profits on the *y*-axis, as a fraction of the unconstrained profit, for different levels, on the *x*-axis, of compulsory order acceptance.

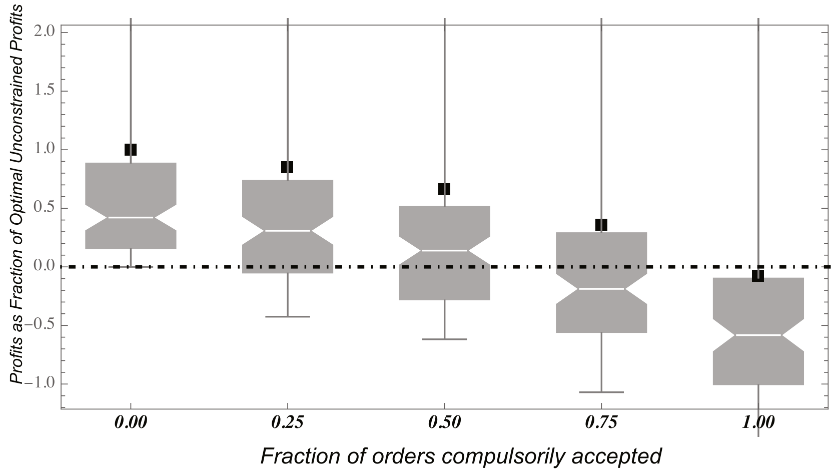

Fig. 4. Average profits on the y-axis, as a fraction of the unconstrained profit, for different levels, on the x-axis, of compulsory order acceptance.

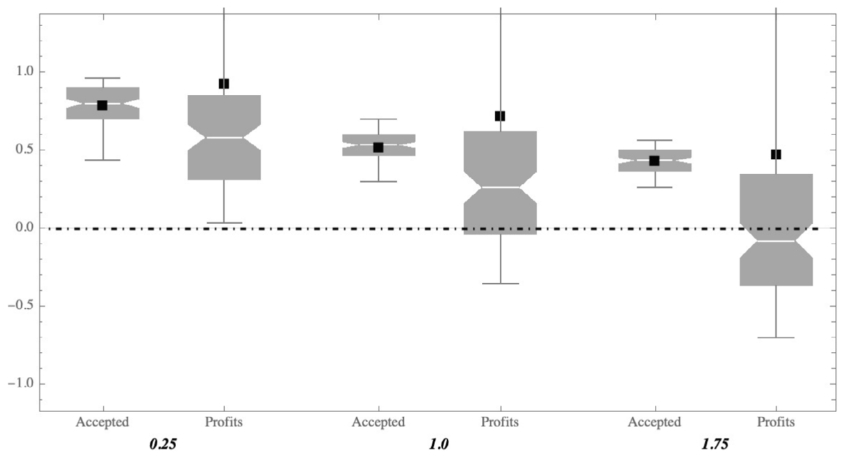

Fig. 5. Distribution of the fraction of accepted orders and profits for different levels of tardiness and holding costs (equally valued). All cases consider that a fraction 0.25 of all orders must be accepted. The profits are normalized to the average profit when no orders must be accepted.

holding costs (equally valued) are considered. The profits are normalized to the average profit when there is no minimum amount of orders to be accepted in each of the different tardiness and holding costs. In this figure, it is possible to see that the number of accepted orders decreases slightly. However, the impact of profitability is substantial as costs increase even when compared to profits obtained without acceptance constraints, even if they are obtained with these same costs.

5.3. *On the costs' accuracy*

A common concern when discussing the impact of complexity on profits, as caused by product proliferation, is the loss of visibility of the true value of operational parameters. The large number of product variants and customers associated with these variants make it difficult to precisely assess the costs. For example, a manufacturer serving different customers may have different penalty fees for the different customers. Even for the same buyer, the type of product and the location of ownership transfer can impact the penalty fees. To complicate matters even more, penalty fees are sometimes related to a shipment that carries several products, which makes any penalty cost allocation nearly impossible. Therefore, we focus in this section on instances where tardiness and holding costs are miscalculated. In particular, we are concerned with underestimating costs, which often happens with products with small contributions. Firms often allocate operational (nonvariable) costs based on each product's contribution to the total revenue. Therefore, products with lower contributions have a smaller share of the cost allocation. Let β represent the error factor with respect to the correct cost. To illustrate, $\beta = 0.25$ implies that a parameter is estimated at a fraction equal to 0.25 of its corrected value. Fig. 6 reports on the top (bottom) plot the profit losses that result when holding (tardiness) costs are estimated as with error factor β as shown on the horizontal axis.

Profit losses due to underestimating tardiness costs come from late delivery of orders whose penalty cost is more than the decision maker thought. These late orders may bring friction with customers and potential litigation.

Profit losses, from holding cost underestimation, come mostly from accepting nonvalue adding orders and holding them for a long period before ownership is transferred. These costs also negatively impact the length of the cash-to-cash cycle time, create a financial burden and increase the firm's operational risk.

Fig. 6 shows that when the value of β is lower, the profit losses are higher. Moreover, profit losses that originate from underestimating holding costs not only are higher in absolute value than those that originate from miscalculating tardiness costs, but also have a higher dispersion, which implies a higher risk exposure.

6. Conclusion

We present in this paper an acceptance order scheduling problem, considering different costs related to both orders and machines. We

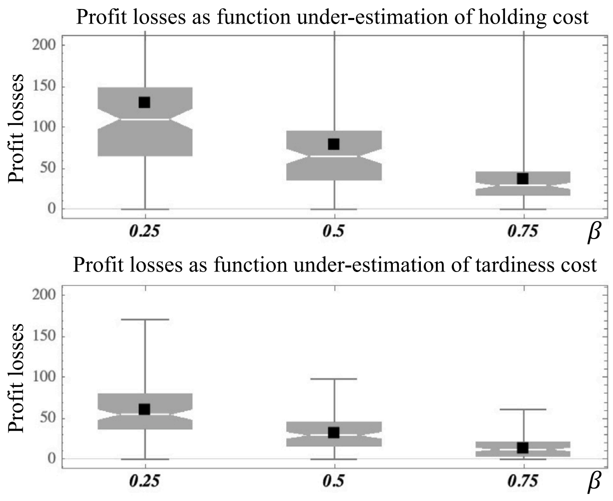

Fig. 6. These plots show the profit losses resulting from solving the heuristic algorithm when holding (tardiness) costs are estimated as β times its correct value. Profit losses are the amount lost beyond the objective function value using the correct value of each parameter.

have formulated it as a MILP model and have introduced an efficient metaheuristic heuristic approach for addressing reasonably large instances. Extensive numerical tests empirically prove that the metaheuristic returns good-quality solutions in short CPU times. Besides, we also compare our metaheuristic with a state-of-the art algorithm for a similar problem, and conclude that our approach finds better solutions. Therefore, our paper contributes to numerical approaches for addressing the problem.

Based on the quality of the results in the numerical tests, we focus on the impact of compulsory order acceptance and the effect that it has on portfolio size and on both profitability and risk. We find that, as expected, by increasing the fraction of orders that must be accepted, profitability suffers. We note that even though profitability loss may not be dramatic in expectation (i.e., not leading to negative profits), the fraction of instances that end up with negative profits can be substantial. As an example from our reported numerical experiments, when some 25% of the potential orders are imposed to be accepted, profit losses are less than 20% of the optimal profit. This could be a reasonable price to pay, for example, to keep some customers happy. However, over 25% of the cases indeed result in negative profits. We have other insights from the managerial perspective, such as the misjudgment of the values of tardiness and holding costs.

The important managerial insights obtained call attention to the fact that when order acceptance is imposed, the financial performance of the business units can go from being minimally impacted to strongly affected. That is, the randomness of the profile of the orders imposed may be a much more important factor that impacts performance than the quality of the managerial decisions. That is, management performance becomes a matter of luck! This insight explains some significant gaps in performance in real business units that we recently encountered in our collaboration with some companies, where managers show frustration for not being able to better explain why their business unit is not performing at par with other business units that belong to the same company. This issue seriously impacts motivation and may affect entire career progressions.

We close this paper by pointing out that our work has taken a static point of view, which is sometimes the case. That is, all customers express their needs before the first unit is produced so the decision maker can select which orders to accept or not. Some other times, orders come at random times, and the acceptance or not of an order will impact, in a probabilistic manner, future profits. This dynamic setting has not been explored and may lead to even more interesting insights. Another direction to explore in the future is related to the mathematical properties of our objective function. Although we have not been able to prove any particular property, we believe based on the numerical results that profits are a submodular decreasing function in the set of orders accepted. This is an important mathematical result and could constitute a paper by itself.

Table 10
Local search preliminary test.

Jobs	Machines	RPD without LS (%)	RPD with LS (%)
10	2	0.00	0.00
	3	0.00	0.00
	4	0.00	0.00
15	2	0.00	0.00
	3	0.00	0.00
	4	0.00	0.15
20	2	0.00	4.26
	3	0.00	0.13
	4	0.00	0.05

Data availability

Data will be made available on request

Acknowledgments

The first author acknowledges the financial from the Spanish Agencia Estatal de Investigación (AEI) and the European Regional Development's fund (ERDF): PID2020-114594GB-C21; the Regional Government of Andalusia: projects FEDERUS-1256951, AT21_00032. The first and second authors also acknowledge the Spanish AEI (grant number PID2021-124975OBI00), partially financed with FEDER funds. The third author is grateful to the full support from the Area of Excellence The Complexity Advantage and the subarea The Value Chain Complexity at NEOMA Business School. Special thanks are due to the editor and the anonymous referees for their comments and suggestions, which helped improve the paper.

Appendix

Table 10 shows the preliminary tests performed with the local search. We observe that the average RPD is better for the algorithm without a local search when the size of the instances increases.

References

Adams, C., Alldredge, K., Mueller, C., Whitmore, J., 2016. Simpler Is (Sometimes) Better: Managing Complexity in Consumer Goods. McKinsey & CO INC.
Bloemen, R., Menezes, M.B., 2019a. Roger Bloemen Video 1: Complexity at SOnoco Industrial. Youtube.com, URL: https://youtu.be/vhoOI-grbJI.
Bloemen, R., Menezes, M.B., 2019b. Roger Bloemen Video 2: Complexity at SOnoco Industrial. Youtube.com, URL: https://youtu.be/FW8X3mGBhIg.
Bradley, C., 2022. Hockey stick dreams, hairy back reality. https://www.linkedin.com/pulse/hockey-stick-dreams-hairy-back-reality-chris-bradley, Retrieve May, 26, 2022.
Bruni, M., Khodaparasti, S., Demeulemeester, E., 2020. The distributionally robust machine scheduling problem with job selection and sequence-dependent setup times. Comput. Oper. Res. 123, 105017.
Chatha, K.A., Jalil, M.N., 2022. Complexity in three-echelon supply chain network and manufacturing firm's operational performance. Comput. Ind. Eng. 108196. http://dx.doi.org/10.1016/j.cie.2022.108196.
Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. European J. Oper. Res. 223, 346–359. http://dx.doi.org/10.1016/j.ejor.2012.06.044.
Fanjul-Peyro, L., Perea, F., Ruiz, R., 2017. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. European J. Oper. Res. 260, 482–493.

Fanjul-Peyro, L., Ruiz, R., 2011. Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput. Oper. Res. 38, 301–309.

Fanjul-Peyro, L., Ruiz, R., 2012. Scheduling unrelated parallel machines with optional machines and jobs selection. Comput. Oper. Res. 39, 1745–1753.

Fanjul-Peyro, L., Ruiz, R., Perea, F., 2019. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Comput. Oper. Res. 101, 173–182.

Feo, T., Resende, M., 1995. Greedy randomized adaptive search procedures. J. Global Optim. 6, 109–133. http://dx.doi.org/10.1007/BF01096763.

Garey, M., Johnson, D., 1979. Computers and Intractability. A Guide to the Theory of NP-Completeness. Freeman, San Francisco.

Gonçalves, J., Resende, M., 2011. Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics 17, 487–525. http://dx.doi.org/10.1007/s10732-010-9143-1.

Grothaus, M., 2020. Coke is killing 200 brands: Here's the list of canceled products announced so far. URL: https://www.fastcompany.com/90567765/coke-is-killing-200-brands-heres-the-list-of-canceled-products-announced-so-far, Retrieve November, 30, 2020.

Hart, J.P., Shogan, A.W., 1987. Semi-greedy heuristics: An empirical study. Oper. Res. Lett. 6 (3), 107–114.

He, L., Guijt, A., de Weerdt, M., Xing, L., Yorke-Smith, N., 2019. Order acceptance and scheduling with sequence-dependent setup times: A new memetic algorithm and benchmark of the state of the art. Comput. Ind. Eng. 138, 106102. http://dx.doi.org/10.1016/j.cie.2019.106102.

Hirose, R., Sodhi, D., Thiel, A., 2017. How Do Winning Consumer-Goods Companies Capture Growth? McKinsey & CO INC.

Hyndman, K., Menezes, M.B., 2021. Behavioral pitfalls of product proliferation in supply chains: An experimental study. Decis. Sci. 240, 12542.

Kong, M., Pei, J., Liu, X., Lai, P.C., Pardalos, P.M., 2020. Green manufacturing: Order acceptance and scheduling subject to the budgets of energy consumption and machine launch. J. Clean. Prod. 248, 119300.

Li, X., Ventura, J.A., 2020. Exact algorithms for a joint order acceptance and scheduling problem. Int. J. Prod. Econ. 223, 107516.

Lopez-Esteve, A., Perea, F., Yepes-Borrero, J.C., 2022. GRASP algorithms for the unrelated parallel machines scheduling problem with additional resources during processing and setups. Int. J. Prod. Res. 1–17.

Lu, L., Menezes, M.B., 2022. Supply Chain Vertical Competition and Product Proliferation under Different Power Structures. Mimeo.

Mariotti, J., 2008. The Complexity Crisis. Why Too Many Products, Markets, and Customers are Crippling Your Company. Platinum Press.

Menezes, M.B., Pinto, R., 2022. Product proliferation, cannibalisation, and substitution: A first look into entailed risk and complexity. Int. J. Prod. Econ. 108327.

Menezes, M.B., Ruiz-Hernández, D., Chen, Y.T., 2021. On the validity and practical relevance of a measure for structural complexity. Int. J. Prod. Econ. 240, 108243.

Mocker, M., Ross, J., 2017. The problem with product proliferation. Harv. Bus. Rev. 95 (3), 104–110.

Naderi, B., Roshana, V., 2020. Branch-relax-and-check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. European J. Oper. Res. 286, 811–827.

Oguz, C., Salman, F., Yalcin, Z.B., 2010. Order acceptance and scheduling decisions in make-to-order systems. Int. J. Prod. Econ. 125, 200–211.

Ruiz-Hernández, D., Menezes, M.B., Amrani, A., 2019. An information-content based measure of proliferation as a proxi for structural complexity. Int. J. Prod. Econ. 212, 78–91.

Scheiter, S., Scheel, O., Klink, G., 2007. How Much Does Complexity Really Cost? ATKearney.

Slotnick, S.A., 2011. Order acceptance and scheduling: A taxonomy and review. European J. Oper. Res. 212, 1–11.

Tanaka, S., 2011. A unified approach for the scheduling problem with rejection. In: IEEE International Conference on Automation Science and Engineering. Trieste, Italy, pp. 369–374.

Tarhan, I., Oguz, C., 2021. Generalized order acceptance and scheduling problem with batch delivery: Models and metaheuristics. Comput. Oper. Res. 134, 105414.

Vallada, E., Ruiz, R., 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European J. Oper. Res. 211 (3), 612–622. http://dx.doi.org/10.1016/j.ejor.2011.01.011, URL: https://www.sciencedirect.com/science/article/pii/S0377221711000142.

Wang, X., Huang, G., Hu, X., Cheng, T.C.E., 2015. Order acceptance and scheduling on two identical parallel machines. J. Oper. Res. Soc. 66 (10), 1755–1767.

Wang, B., Wang, H., 2018. Multiobjective order acceptance and scheduling on unrelated parallel machines with machine eligibility constraints. Math. Probl. Eng. 2018, 1–12.

Wang, S., Ye, B., 2019. Exact methods for order acceptance and scheduling on unrelated parallel machines. Comput. Oper. Res. 104, 159–173.

Xu, L., Wang, Q., Huang, S., 2015. Dynamic order acceptance and scheduling problem with sequence-dependent setup time. Int. J. Prod. Res. 53 (19), 5797–5808.

Yavari, M., Marvi, M., Akbari, A.H., 2020. Semi-permutation-based genetic algorithm for order acceptance and scheduling in two-stage assembly problem. Neural Comput. Appl. 32, 2989–3003.

Yepes, J.C., Villa, F., Perea, F., Caballero, J.P., 2020. GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. Expert Syst. Appl. 141, 1–12.