Vol. 35, No. 1 (2025) | DOI: 10.37190/ord250101

OPEN ACCESS

Operations Research and Decisions

www.ord.pwr.edu.pl

An efficient heuristic for a real-life OAS problem

Marcin Anholcer[1] Andrzej Żak[2*]

[1] *Institute of Informatics and Electronic Economics, Poznań University of Economics and Business, Poznań, Poland*

[2] *Faculty of Applied Mathematics, AGH University, Kraków, Poland*

[*] *Corresponding author, email address: zakandrz@agh.edu.pl*

Abstract

Inspired by a real-life manufacturing problem, we present a mathematical model and a heuristic that solves it. A desired solution needs not only to maximize the company's profit but must also be easy to interpret by the members of the management. The considered problem is thus a variant of the order acceptance and scheduling (OAS) problem, which can be solved using known heuristics. Our approach is different because we study the mechanism by which setup times arise, unlike other approaches where setup times are treated as parts of the instance. This enables us to develop a very fast and efficient heuristic, formulate a MILP model that can be applied to solve much larger problems than previously known methods, and ultimately meet decision-makers' expectations. We prove the efficiency of the presented method by comparing its results with the optimum obtained by a state-of-the-art solver. We also briefly discuss a case study that arose in a food industry company in Poland.

Keywords: *order acceptance and scheduling, sequence-dependence, greedy heuristics*

1. Introduction

The problem presented in this article is a real-life problem that appeared in the food production industry. The details will be described in the following sections, as for now let us mention only that the company under discussion produces various kinds of chips for several customers, being food retailers in Poland. It possesses one production line that can be used for production, however it requires changeover/setup almost whenever it switches from one order to another. The company's primary objective is to maximize profit, however, for several business reasons, the management is interested not only in an optimal solution but also in being able to understand, what is the influence or significance of every single order on this solution. For that reason, it is preferred to obtain some easily interpretable evaluation assigned to every order, even if in consequence the obtained solution is only close to the global optimum. The main constraint is the capacity of the production line, which implies that not all the orders can be fulfilled.

Received 18 May 2024, accepted 16 November 2024, published online 8 February 2025
ISSN 2391-6060 (Online)/© 2025 Authors

The costs of publishing this issue have been co-financed by the Department of Operations Research and Business Intelligence at the Faculty of Management, Wrocław University of Science and Technology, Wrocław, Poland

Putting this together we can see that the problem we address is an order selection and scheduling problem on a single machine to maximize the manufacturer's profit, defined as total revenue. This problem has been previously studied by Charnsirisaksul et al. [8] and Oğuz et al. [35]. Both studies applied heuristics, particularly the dynamic release first-sequence best (d-RFSB) heuristic with a complexity of $O(n^2)$. They also provided a mixed-integer linear programming (MILP) formulation, but could only solve very small instances with up to 15 orders using CPLEX [10]. Larger instances were not solved optimally. Our approach differs in the way we deal with changeover times (setup times), which are the periods wasted on setting up the machine when switching from one order to another. In previous studies, these times were included as part of the problem instance. Motivated by a real-life case study, we investigated the mechanisms causing these changeover times and found that they are driven by changes of a relatively small number of features characterizing the orders, significantly fewer than the number of orders. As a result, we achieved the following goals:

- We developed an efficient heuristic with computational complexity $\max(O(n \log n), O(nm))$, where m is the number of features. Given that $m = 3$ in the real-life problem we examined, the complexity was essentially $O(n \log n)$.
- We formulated a MILP model capable of solving very large problems in a reasonable time. For instance, problems with 10,000 orders were solved optimally by the GUROBI [18] solver in an average of 40 seconds, which was even faster than the d-RFSB heuristic.
- We provided a solution that is easily interpretable by management, meeting their strong preference for understandable outcomes.

Let us briefly discuss the former research on these two topics. The order acceptance and scheduling problem, sometimes referred to as the scheduling problem with rejections, has been widely studied at least for the last thirty years. An extensive review of this topic collecting the research up to 2011 was published by Slotnick [37]. The author presented a systematic review of various kinds of problem and solution approaches. Earlier, two papers were published that attracted our interest. One of them is [9], where the authors, motivated by a steel production problem, minimize the total cost, including the rejection cost and the transition cost. The solution method applied in their research is a hybrid evolutionary algorithm joining a genetic algorithm and extreme optimization. In the other paper Oğuz et al. [35] also consider the variant with sequence-dependent setup times and other constraints, like time windows, to find the optimal solutions of small instances with at most 15 orders and present heuristics solving larger problems. More recently the order acceptance and scheduling with setup times was considered in [36], where the authors focused, among others, on introducing several assumptions making the problem more consistent with real-life problems. Other papers treating this topic are [41] and [42], where the authors deal with the generalized order acceptance and scheduling problem. A variant similar to the one considered in our work has been analyzed by Ou [33], who considers batch processing on a single machine with setup times, rejections, and the objective being the weighted sum of the makespan of the accepted jobs and the total cost computed as the sum of the setup costs and rejection penalties. Another similar setting was considered in [43]. In this case, the objective is equal to the total cost of earliness and tardiness penalties, changeover costs, and rejection penalties. The variants of the problem with optimization of the weighted sum of revenue and tardiness penalty on distinct parallel machines were solved by Emami et al. by Lagrangian relaxation [14] and Benders decomposition [13]. A dynamic version of the problem, where the arrival times of orders are stochastic and follow a Poisson distribution, and the objective is the expected revenue was studied in [17] and [47].

Some authors included the setup times in the processing times. Wang et al. [46] considered a job shop problem in a make-to-stock / make-to-order manufacturing company. Another example is [29], where the variant with time windows and the possibility of the capacity extension were included in the problem set. Exact algorithms for this variant of the problem were presented in [28]. There are also papers, where the setup/changeover times are not considered at all [19, 22, 26, 30, 34, 38].

One of the first papers where the scheduling problem with changeover times was analyzed is the article of Lockett and Muhlemann [31], where the authors consider a branch-and-bound approach and several heuristics. Another early paper on this topic is the one of Geoffrion and Graves [16], where the authors applied a solution strategy using the quadratic assignment problem assisted with some linear programming adjustments.

Several MIP formulations for the scheduling problems with sequence-dependent changeovers have been proposed by Velez et al. [45]. Lee and Maravelias [25] mentioned the changeover times as a possible extension of their MILP models for short-term scheduling. Blocher et al. [1] studied a discrete time single machine scheduling problem with sequence-independent changeover times and applied a branch -and-bound method to solve it. Hu et al. [21] studied the properties of a batch scheduling problem with subassemblies. In particular, they characterized a set of optimal schedules. Konge and Subramanian [23] presented and evaluated a MILP model for a scheduling problem with changeovers in an ice cream production facility.

Bowers et al. [2] applied cluster analysis to reduce the sequence-dependent changeover times in a manufacturing short-term scheduling problem. Clustering helped in reducing the complexity of the model, otherwise equivalent to an appropriately defined instance of TSP. Hong et al. [20] applied accelerated dynamic programming for car resequencing in automotive paint shops. In this case, the changeover times depended on the sequence of orders, too. Also Sun and Han [40] studied scheduling with changeovers in automotive paint shops. In particular, they applied two batching strategies to obtain color-oriented batches before painting. Sun and Fan [39] in turn considered the car sequencing problem for mixed-assembly lines, in particular applying multiple objective and colony optimization to find an approximate solution to the problem.

Casado et al. [4] analyzed a scheduling problem with changeovers originating from the steel manufacturing industry. After presenting an MIP formulation, they proposed a heuristic approach to solve it and showed that some simplified variants of the problem are equivalent to clique partition and graph coloring problems. Montoya-Torres et al. [32] used a randomized heuristic to solve scheduling problems with sequence-dependent setup times, motivated by real-life manufacturing systems. Duncan in her thesis [12] focused on the scheduling problem with changeovers present in a company manufacturing industrial equipment. In particular, she analyzed some pairing and grouping methods.

Many publications involving changeovers are connected with the applications in the chemical industry. Castro et al. [6] modeled the sequence-dependent changeovers using TSP constraints in multiperiod refinery planning. Castro [5] considered scheduling with changeovers in a multiproduct batch chemical plant, using among others the resource-task network formulation of the problem. Castro et al. [7] presented several mathematical models of batch scheduling problems with sequence-dependent setup times and solved them with a greedy algorithm. A continuous time scheduling problem with changeovers and its application to the chemical industry was presented by Díaz-Ramírez and Huertas [11]. Li and Milne [27] presented a formulation of a real-life production scheduling problem that appeared in a chemical company. They also presented a three-step heuristic involving in particular a neighborhood search strategy. Brunaud et al. [3] studied batch scheduling with quality-based changeovers, i.e., the situation present in the chemical industry, where cleaning/setup can be avoided if enough batches of the second product are processed consecutively. The authors considered in particular the state-task network, resource-task network, and unit-operation-port-state-superstructure frameworks. It is worth mentioning that also collecting data about the changeover times can be a challenge in complex manufacturing systems. In this context, Engelmann et al. [15] developed a machine learning model for the identification and characterization of machine setups in cyberphysical production systems.

The current article is organized as follows. In the following section, we describe the problem and present its mathematical model, together with an illustrative example. In Section 3, we present the naive method used initially to solve the problem and the new algorithm, also illustrated with an example. In Section 4, the results of numerical experiments performed on a set of randomly generated test problems have been presented. Section 5 contains a description of the application of the presented method in a company operating in the food industry located in southwestern Poland. We conclude the paper with some final remarks.

2. Description of the problem

The company is considering the execution of n distinct orders $O_1, \ldots, O_n$, each of which can be sold on the market based on the offers collected from potential buyers. Each product is characterized by m features $F_1, \ldots, F_m$, with values uniquely assigned to the products. Each feature can take multiple values, and these values are determined by the specific settings of the machines' working parameters. Any change in the value of a feature requires corresponding retooling of the production line components. Therefore, to alter the value of any feature, a retooling process is necessary. Each retooling has a fixed processing time, which remains constant regardless of the specific value change for that feature.

Production follows a cyclic pattern over a given period, meaning the same schedule is repeated weekly throughout the planning horizon. Multiple feature changes can be executed simultaneously. In such cases, the overall production time is only affected by the longest of the simultaneous retoolings. Consequently, moving forward, we define a changeover as the retooling that impacts total production time, disregarding others that do not contribute to this delay. If the longest retooling pertains to feature F_i, we refer to it as a changeover caused by feature F_i (or simply a changeover when no ambiguity arises).

To illustrate, suppose the features F_1, F_2, and F_3 for order O_j are $(A, a, 1)$, and for order $O_{j'}$, they are $(B, b, 2)$. The retooling times for F_1, F_2, and F_3 are 2, 1, and 0.5, respectively. When switching production from O_j to $O_{j'}$, three separate retoolings are required (since all features change). However, as these can occur simultaneously, the changeover is determined by the longest retooling. Hence, the duration time of the changeover in this case is 2, and the changeover is caused by feature F_1. If the features of $O_{j'}$ were $(A, b, 2)$, then the changeover time would be 1, driven by feature F_2.

Interestingly, although the total changeover time depends on the sequence in which the orders are executed, the way changeovers occur allows us to approach the problem in a quasi-sequence-independent manner. As we will see, once a subset of orders is selected, the optimal sequence that minimizes total changeover time is simply to sort the orders lexicographically by their feature values. This approach will be elaborated in Theorem 1 in the following subsection.

2.1. Mathematical model

The parameters are:

- E – the production line's efficiency, not depending on the order, kg/h,
- T – time capacity, i.e., the maximum working time of the production line during one week, h/week,
- q_j, $j = 1, \ldots, n$ – sizes of the order O_j, kg/week,
- $t_i, i = 1, \ldots, m$ – the duration of changeover corresponding with feature F_i, h; in the remainder of the paper we will assume without loss of generality that the features are indexed so that $t_i \leq t_j$ when $i > j$; for the sake of notation convenience we set $t_{m+1} = 0$,
- t_0 – cleaning time; at the beginning of every week the production line has to be cleaned, h,
- $c_j, j = 1, \ldots, n$ – unit profit for order O_j, USD/kg.

Let us denote by

$$a_j = \frac{q_j}{E} \tag{1}$$

the time, h/week used for processing order O_j during one week, $j = 1, \ldots, n$. Moreover, for every feature F_i, let f_i denote the number of all possible combinations of the features with indices at most i, i.e., f_1 will be the number of all values of F_1, f_2 the number of all possible pairs of values of F_1 and F_2 and so on. Additionally let $f_0 = 1$ stand for cleaning.

The decision variables are:

- x_j, $j = 1, \ldots, n$ – binary variable equal to 1 if the company decides to execute order O_j and 0 otherwise,
- $y_{ik}, i = 1, \ldots, m, k = 1, \ldots, f_i$ – binary variable equal to 1 if the company needs at some moment to set up the production line to the state where features $F_1, F_2, \ldots, F_i$ take the kth combination of their possible values and 0 otherwise.

The core concept of the forthcoming MILP model is that we can predict the minimal total time spent on changeovers based solely on the selected set of orders. An optimal sequence, which minimizes the time spent on changeovers is determined by arranging the chosen orders in lexicographic order by the names of their attributed features (in short lexicographic order).

Theorem 1. Let σ be an ordering of accepted orders and $z(\sigma)$ be the total time lost for changeovers forced by σ. Let g_i be the number of distinct tuples $(F_1, \ldots, F_i)$, $i = 1, \ldots, m$, among accepted orders and let $\chi_i(\sigma)$ be the number of changeovers caused by feature F_i. Furthermore, let σ^* be the lexicographical order of accepted orders. Then

$$z(\sigma) \geq z(\sigma^*)$$

with equality if

$$\chi_i(\sigma) = \chi_i(\sigma^*) = \begin{cases} g_i - 1, & \text{if } i = 1 \\ g_i - g_{i-1}, & \text{if } i \in \{2, \ldots, m\} \end{cases} \tag{2}$$

Proof. If the orders are sorted in lexicographic order then $\sum_{k=1}^{i} \chi_k(\sigma^*) = g_i - 1$ since there are g_i distinct tuples $(F_1, \ldots, F_i)$. Thus $\chi_i(\sigma^*) = g_i - g_{i-1}$, $i = 1, \ldots, m$. On the other hand, if σ is any ordering of the orders then $\sum_{k=1}^{i} \chi_k(\sigma) \geq g_i - 1$. Then $z = z(\sigma)$ satisfies

$$\begin{aligned} & z = \sum_{k=1}^{m} t_k \chi_k(\sigma) \to \min \\ & \sum_{k=1}^{i} \chi_k(\sigma) \geq g_i - 1, \quad i = 1, \ldots, m \quad \chi_k(\sigma) \geq 0, \quad k = 1, \ldots, m. \end{aligned} \tag{3}$$

The problem dual to (3) has the form

$$\begin{aligned} & w = \sum_{l=1}^{m} (g_l - 1) y_l \to \max \\ & \sum_{l=i}^{m} y_l \leq t_i, \quad i = 1, \ldots, m \quad y_l \geq 0, \quad l = 1, \ldots, m. \end{aligned} \tag{4}$$

Consider $\chi^* = (\chi_1^*, \ldots, \chi_m^*)$ with $\chi_k^* = g_k - g_{k-1}$, $k = 1, \ldots, m$, and $y^* = (y_1^*, \ldots, y_m^*)$ with $y_l^* = t_l - t_{l+1}$. Since $t_l \geq t_{l+1}$, χ^* and y^* are feasible solutions of (3) and (4), respectively. Moreover, $z(\chi^*) = w(y^*)$. Hence, by the duality χ^* is an optimal solution of (3). □

Therefore in the sequel we may assume that the orders are sorted in lexicographic order by the values of the features. Let $\mathbb{J}_{ik}$ denote the set of the indices of orders requiring the setup with feature combination defined with $y_{ik} = 1$. The mathematical model of the problem takes the form

$$z = \sum_{j=1}^{n} c_j x_j \longrightarrow \max, \tag{5}$$

$$\sum_{j=1}^{n} a_j x_j + (t_0 - t_1) + \sum_{i=1}^{m} (t_i - t_{i+1}) \sum_{k=1}^{f_i} y_{ik} \leq T, \tag{6}$$

$$|\mathbb{J}_{ik}| y_{ik} \geq \sum_{j \in \mathbb{J}_{ik}} x_j, \tag{7}$$

$$y_{ik} \leq \sum_{j \in \mathbb{J}_{ik}} x_j,$$

$$x_j \in \{0, 1\} \text{ for } j = 1, \ldots, n, \tag{8}$$

$$y_{ik} \in \{0, 1\} \text{ for } i = 1, \ldots, m, \quad k = 1, \ldots, f_i.$$

Indeed, the objective is simply maximization of the profit. Furthermore, by Theorem 1, in the best schedule of accepted orders the number of changeovers caused by feature i is equal to $\sum_{k=1}^{f_1} y_{1k} - 1$ if $i = 1$, is equal to $\sum_{k=1}^{f_i} y_{ik} - \sum_{k=1}^{f_{i-1}} y_{i-1,k}$ if $i \in \{2, \ldots, , m\}$. The duration time of such changeover is equal to t_i.

Therefore, the time limit constraint takes the form:

$$\sum_{j=1}^{n} a_j x_j + t_0 + t_1 \left(\sum_{k=1}^{f_1} y_{1k} - 1 \right) + t_2 \left(\sum_{k=1}^{f_2} y_{2k} - \sum_{k=1}^{f_1} y_{1k} \right) + t_3 \left(\sum_{k=1}^{f_3} y_{3k} - \sum_{k=1}^{f_2} y_{2k} \right) + \cdots \leq T$$

which is equivalent to

$$\sum_{j=1}^{n} a_j x_j + (t_0 - t_1)1 + (t_1 - t_2) \sum_{k=1}^{f_1} y_{1k} + (t_2 - t_3) \sum_{k=1}^{f_2} y_{2k} + (t_3 - t_4) \sum_{k=1}^{f_3} y_{3k} + \cdots \leq T$$

which agrees with (6)

The constraints (7) connect x_j to y_{ik} and assure that $y_{ik} = 1$ if and only if at least one $x_j = 1$ for $j \in \mathbb{J}_{ik}$. All the variables are binary, as defined by (8).

2.2. Illustrative example

To illustrate the problem and its MILP model, let us consider an example described with the data presented in Table 1.

Table 1. Input data for the example
$t_1 = 2$, $t_2 = 1$, $t_3 = 0.5$, $t_0 = 2$, $T = 12$, $E = 20$

Order	F_1	F_2	F_3	c_j	q_j	a_j
O_1	A	a	0	3.0	14.0	0.7
O_2	A	a	1	5.0	19.0	0.95
O_3	A	a	3	4.0	22.0	1.1
O_4	A	b	2	2.0	24.0	1.2
O_5	A	b	3	6.0	20.0	1.0
O_6	A	b	3	3.0	32.0	1.6
O_7	A	c	1	4.0	47.0	2.35
O_8	B	b	2	2.0	27.0	1.35
O_9	B	c	0	4.0	9.0	0.45
O_{11}	C	b	2	2.0	27.0	1.35
O_{12}	C	c	1	5.0	15.0	0.75
O_{13}	C	c	2	1.0	32.0	1.6
O_{14}	C	c	3	3.0	15.0	0.75

Let us start with constraints (7). If $i = 1$ then since the possible states of feature $F_i = F_1$ are A, B or C, we have

$$\mathbb{J}_{1,1} = \{1, \ldots, 7\}, \ \mathbb{J}_{1,2} = \{8, \ldots, 10\}, \ \mathbb{J}_{1,3} = \{11, \ldots, 14\}.$$

Hence, equations (7) corresponding to F_1 have the following form

$$7y_{1,1} \geq x_1 + \cdots + x_7, \qquad (A)$$

$$3y_{1,2} \geq x_8 + \cdots + x_{10}, \qquad (B)$$

$$4y_{1,3} \geq x_{11} + \cdots + x_{14}. \qquad (C)$$

If $i = 2$, then all possible states of F_1 and F_2 are Aa, Ab, Ac, Bb, Bc, Cb, Cc. Hence,

$$\mathbb{J}_{2,1} = \{1, 2, 3\}, \quad \mathbb{J}_{2,2} = \{4, 5, 6\}, \quad \mathbb{J}_{2,3} = \{7\}$$

$$\mathbb{J}_{2,4} = \{8\}, \quad \mathbb{J}_{2,5} = \{9, 10\}, \quad \mathbb{J}_{2,6} = \{11\}, \quad \mathbb{J}_{2,7} = \{12, 13, 14\}$$

Therefore, equations (7) that correspond with all the possible settings of F_1 and F_2 are as follows:

$$3y_{2,1} \geq x_1 + \cdots + x_3, \qquad (Aa)$$

$$3y_{2,2} \geq x_4 + \cdots + x_6, \qquad (Ab)$$

$$y_{2,3} \geq x_7, \qquad (Ac)$$

$$y_{2,4} \geq x_8, \qquad (Bb)$$

$$2y_{2,5} \geq x_9 + x_{10}, \qquad (Bc)$$

$$y_{2,6} \geq x_{11}, \qquad (Cb)$$

$$3y_{2,7} \geq x_{12} + \cdots + x_{14}, \qquad (Cc)$$

Analogously, equations (7) corresponding to the settings of F_1, F_2 and F_3 have the form

$$y_{3,1} \geq x_1, \qquad (Aa0)$$

$$y_{3,2} \geq x_2, \qquad (Aa1)$$

$$\ldots$$

$$2y_{3,5} \geq x_5 + x_6, \qquad (Ab3)$$

$$\ldots$$

$$y_{3,13} \geq x_{14}. \qquad (Cc3)$$

Finally the constraint (6) takes the form

$$\sum_{j=1}^{14} a_j x_j + (2-2) + (2-1)\sum_{k=1}^{3} y_{1,k} + (1-0.5)\sum_{k=1}^{7} y_{2,k} + 0.5\sum_{k=1}^{13} y_{3,k} \leq 12.$$

3. Solution

The primary objective of the company is the maximization of the profit. The instances of the problem described in the previous sections can be solved efficiently using well-known optimization solvers. However, the solutions obtained this way cannot be accepted by the management of the firm. As we already mentioned in the introduction, managers need clear and easy-to-compute indicators that allow them to prepare a ranking of the orders so that the accepted orders will be the ones with the highest value of the indicator. It is preferred to obtain a worse solution, provided that such an interpretable indicator will be assigned to every order.

For that reason, our efforts focused on finding a method that would be on one hand easy to apply and whose results would be easy to interpret, while on the other hand, it would produce a solution as close as possible to the optimal one. These efforts resulted in the heuristic that we are going to present in one of the following subsections. We will start, however, by describing the method that was used in the company before our activity started. (In reality, this method served only as the initial foundation for the analysis. Subsequently, it underwent substantial modifications based on business experience and intuition. However, these modifications were made without explicit support from numerical computations). This one, which we call naive heuristic, produces a solution that can be easily interpreted, however, as we will see later, does not need to be satisfactorily close to the actual optimum.

3.1. Naive heuristic

The method presented in the current subsection is essentially the same as the well-known greedy heuristic for the Knapsack Problem (see, e.g., [24]). Nevertheless, we present it for the sake of completeness, using the notation from the current paper.

The naive approach consists in adding greedily the orders according to the non-increasing order of unit profit. Let A_0 denote this heuristic and let z_0 be the value of the objective function (5) obtained using this heuristic. For that purpose, we first compute the unit profits according to the equation

$$d_j = \frac{c_j q_j}{a_j}, \quad j = 1, \ldots, n \tag{9}$$

```
**PHASE 1: ORDERING**
**for** $j = 1, \ldots, n$ **do**
    Compute $d_j$ using equation (9)
**end**
Sort orders according to the non-increasing order of $d_j$ and mark all of them as unchecked
**PHASE 2: SELECTION**
Set all the variables $x_j$ and $y_{ik}$ to 0.
**while** *there is at least one unchecked order* **do**
    Select the first unchecked order $j$ and mark it as checked.
    Set the temporary value of $x_j$: $x_j^{temp} \leftarrow 1$.
    Calculate the corresponding temporary values $y_{ik}^{temp}$ of $y_{ik}$ using equation (7).
    **if** *constraint (6) is satisfied with temporary values of variables* **then**
        Set $x_j \leftarrow x_j^{temp}$.
        Set $y_{ik} \leftarrow y_{ik}^{temp}$ for each $i, k$ such that $y_{ik} \neq y_{ik}^{temp}$.
    **end**
**end**
```

Algorithm 1. Naive heuristic

This could be expressed as $d_j = c_j E$ but we present it in the form as above for the sake of consistency with the notation used in the remainder of this paper, in particular in the descriptions of the methods. Then, the naive heuristic A_0 can be written in the form presented as Algorithm 1.

3.2. The new method

We denote the heuristic described in this section by A_1 and the value of the objective function (5) obtained using A_1 by z_1.

Let

$$Q_{ik} = \sum_{j \in \mathbb{J}_{ik}} q_j, \quad i = 1, \ldots, m, \ k = 1, \ldots, f_i \tag{10}$$

denote the sum of quantities corresponding with the orders included in set $\mathbb{J}_{ik}$. We also introduce unit penalties for each kind of changeover (e.g. changeover corresponding with every feature). It will be defined with the equation

$$p_i = \frac{f_i - f_{i-1}}{f_i} t_i, \quad i = 1, \ldots, m \tag{11}$$

where we assume $f_0 = 1$ for consistency.

The idea of this equation follows from the fact that by Theorem 1, the number of necessary changeovers caused by F_i is equal to $f_i - f_{i-1}$ (assuming all orders are preliminarily admissible). On the other hand, the penalties should be uniformly distributed among f_i combinations of settings, which defines the denominator.

Having defined the unit penalties, we distribute them proportionally among the orders according to the equation:

$$a^\star_{ij} = \frac{q_j}{Q_{ik}} p_i, \quad i = 1, \ldots, m, k = 1, \ldots, f_i, j \in \mathbb{J}_{ik} \tag{12}$$

By including both nominal production time a_j and the penalties, we define the adjusted production time of every order:

$$a^\star_j = a_j + \sum_{i=1}^{m} a^\star_{ij}, \quad j = 1, \ldots, n \tag{13}$$

Finally, we can compute the adjusted unit profits per hour with the equation

$$d^\star_j = \frac{c_j q_j}{a^\star_j}, \quad j = 1, \ldots, n \tag{14}$$

Note that our method is somewhat similar to the d-RFSB heuristic, as both approaches involve adding an additional element to the processing time in the denominator of (14). Specifically, in the d-RFSB heuristic, the next order chosen is the one that minimizes the value $\frac{c_j q_j}{a_j + s_{ij}}$ over all unscheduled orders j, where i is the last scheduled order and s_{ij} is the respective set-up time. This incorporation of changeover times is the major difference of our new algorithm from the naive one.

Then the orders are included in the schedule greedily, according to the non-increasing order of $d^\star_j$. Taking into account the above considerations, the heuristic (denoted A_1) can be summarized in the form presented as Algorithm 2.

```
PHASE 1: ORDERING
for $i = 1, \dots, m, k = 1, \dots, f_i$ do
    Compute $Q_{ik}$ using equation (10)
end
for $i = 1, \dots, m$ do
    Compute $p_i$ using equation (11)
end
for $i = 1, \dots, m, k = 1, \dots, f_i, j \in \mathbb{J}_{ik}$ do
    Compute $a^\star_{ij}$ using equation (12)
end
for $j = 1, \dots, n$ do
    Compute $a^\star_j$ using equation (13)
    Compute $d^\star_j$ using equation (14)
end
Sort orders according to the non-increasing order of $d^\star_j$ and mark all of them as unchecked.
PHASE 2: SELECTION
Perform the PHASE 2: SELECTION like in the Algorithm 1
```

Algorithm 2. New heuristic

3.3. Illustrative example – continued

In this subsection, we illustrate all the significant steps of the presented algorithm on the same example that has already been introduced in Subsection 2.2. The necessary data and partial computations are collected in Tables 2 and 3. Let us start with Table 2.

Table 2. Data and computations for the example

Parameter	F_1	F_{12}	F_{123}	Sum
Number of combinations (f_i)	3	7	13	23
Number of changeovers $(f_i - f_{i-1})$	2	4	6	12
Time lost for changeovers (t_i)	4	4	3	11
Penalty for a changeover (p_i)	1.33	0.57	0.23	

Recall that three features are given: F_1, F_2 and F_3, having 3, 3 and 4 possible values, respectively, while the numbers of possible combinations of F_1, (F_1, F_2) and (F_1, F_2, F_3) are $f_1 = 3$, $f_2 = 7$ and $f_3 = 13$. The corresponding changeover times are $t_1 = 2$, $t_2 = 1$ and $t_3 = 0.5$, and the production line efficiency $E = 20$. The unit penalties are obtained by using equations (11):

$$p_1 = \frac{f_1 - f_0}{f_1} t_1 = \frac{3-1}{3} \times 2 = \frac{4}{3}$$

$$p_2 = \frac{f_2 - f_1}{f_2} t_2 = \frac{7-3}{7} \times 1 = \frac{4}{7}$$

$$p_3 = \frac{f_3 - f_2}{f_3} t_3 = \frac{13-7}{13} \times 0.5 = \frac{3}{13}$$

The remaining data and partial computations have been presented in Table 3. Let us discuss its selected entries in a more detailed way.

Table 3. Data and computations for the example.

j	F_1	F_2	F_3	c_j	q_j	a_j	Q_{1k}	Q_{2k}	Q_{3k}	a^*_{1j}	a^*_{2j}	a^*_{3j}	a^*_j
1	A	a	0	3.0	14.0	0.7	178.0	55.0	14.0	0.1	0.15	0.23	1.18
2	A	a	1	5.0	19.0	0.95	178.0	55.0	19.0	0.14	0.2	0.23	1.52
3	A	a	3	4.0	22.0	1.1	178.0	55.0	22.0	0.16	0.22	0.23	1.72
4	A	b	2	2.0	24.0	1.2	178.0	76.0	24.0	0.17	0.18	0.23	1.79
5	A	b	3	6.0	20.0	1.0	178.0	76.0	52.0	0.14	0.15	0.08	1.38
6	A	b	3	3.0	32.0	1.6	178.0	76.0	52.0	0.23	0.24	0.14	2.22
7	A	c	1	4.0	47.0	2.35	178.0	47.0	47.0	0.35	0.57	0.23	3.5
8	B	b	2	2.0	27.0	1.35	57.0	27.0	27.0	0.63	0.57	0.23	2.78
9	B	c	0	4.0	9.0	0.45	57.0	30.0	9.0	0.21	0.17	0.23	1.06
10	B	c	1	3.0	21.0	1.05	57.0	30.0	21.0	0.49	0.39	0.23	2.17
11	C	b	2	2.0	27.0	1.35	89.0	27.0	27.0	0.4	0.57	0.23	2.55
12	C	c	1	5.0	15.0	0.75	89.0	62.0	15.0	0.22	0.13	0.23	1.34
13	C	c	2	1.0	32.0	1.6	89.0	62.0	32.0	0.47	0.29	0.23	2.6
14	C	c	3	3.0	15.0	0.75	89.0	62.0	15.0	0.22	0.13	0.23	1.34
Sum					324.0	16.2				4.0	4.0	3.0	27.2

Recall that j is the number of orders to be processed, while the columns F_1, F_2, and F_3 contain the possible values of the features in the corresponding setup. c_j represents the unit profit, while q_j is the order size. Using equation (1) the values of a_j were computed, like

$$a_1 = \frac{q_1}{E} = \frac{14}{20} = 0.7, \quad a_2 = \frac{q_2}{E} = \frac{19}{20} = 0.95$$

and so on. Using equation (10) the values of Q_{1k} have been computed. In particular, for the value A of the first feature (i.e., for $j = 1, \dots, 7$) we have

$$Q_{11} = \sum_{j \in \{1,\dots,7\}} q_j = 14 + 19 + 22 + 24 + 20 + 32 + 47 = 178$$

Using the same equation, the values of Q_{2k} corresponding with the pairs of features including A, i.e., the pairs Aa (corresponding with orders 1, 2, and 3), Ab (corresponding with orders 4, 5, and 6), and Ac (order 7) were obtained as follows:

$$Q_{21} = \sum_{j \in \{1,2,3\}} q_j = 14 + 19 + 22 = 55$$

$$Q_{22} = \sum_{j \in \{4,5,6\}} q_j = 24 + 20 + 32 = 76,$$

$$Q_{23} = \sum_{j \in \{7\}} q_j = 47$$

Finally, the values of Q_{3k}, associated with the triples $Aa0$ ($j = 1$), $Aa1$ ($j = 2$), $Aa3$ ($j = 3$), $Ab2$ ($j = 4$), $Ab3$ ($j = 5, 6$) and $Ac1$(j=7) were computed as follows:

$$Q_{31} = \sum_{j\in\{1\}} q_j = 14, \quad Q_{32} = \sum_{j\in\{2\}} q_j = 19, \quad Q_{33} = \sum_{j\in\{3\}} q_j = 22$$

$$Q_{34} = \sum_{j\in\{4\}} q_j = 24, \quad Q_{35} = \sum_{j\in\{5,6\}} q_j = 20 + 32 = 52, \quad Q_{36} = \sum_{j\in\{7\}} q_j = 47$$

The remaining entries of the columns have been computed analogously. In the next step, equations (12) and (13) have been used to compute the penalties for the changeovers and the adjusted production times:

$$a^\star_{11} = \frac{q_1}{Q_{11}} p_1 = \frac{14}{178} \times \frac{4}{3} = 0.1, \quad a^\star_{21} = \frac{q_1}{}Q_{21} p_2 = \frac{14}{55} \times \frac{4}{7} = 0.15, \quad a^\star_{31} = \frac{q_1}{Q_{31}} p_3 = \frac{14}{14} \times \frac{3}{13} = 0.23$$

$$a^\star_1 = a_1 + a^\star_{11} + a^\star_{21} + a^\star_{31} = 0.7 + 0.1 + 0.14 + 0.23 = 1.18$$

$$a^\star_{12} = \frac{q_2}{Q_{11}} p_1 = \frac{19}{178} \times \frac{4}{3} = 0.14, \quad a^\star_{22} = \frac{q_2}{Q_{21}} p_2 = \frac{19}{55} \times \frac{4}{7} = 0.2, \quad a^\star_{32} = \frac{q_2}{Q_{32}} p_3 = \frac{19}{19} \times \frac{3}{13} = 0.23$$

$$a^\star_2 = a_2 + a^\star_{12} + a^\star_{22} + a^\star_{32} = 0.95 + 0.14 + 0.2 + 0.23 = 1.52$$

and so on. The following steps of Algorithm 2 have been illustrated in Table 4. The values of $d^\star_j$ have been computed using equation (14). For instance, the first two values are obtained as

$$d^\star_1 = \frac{c_1 q_1}{a^\star_1} = \frac{3 \times 14}{1.18} = 35.56, \quad d^\star_2 = \frac{c_2 q_2}{a^\star_2} = \frac{5 \times 19}{1.52} = 62.47$$

Table 4. Data and computations for the example.
Source: own elaboration

j	F_1	F_2	F_3	c_j	q_j	a_j	a^*_j	$d^\star_j$	A_0	A_1	Optimal
5	A	b	3	6.0	20.0	1.0	1.38	86.39	1.0	1.0	1.0
2	A	a	1	5.0	19.0	0.95	1.52	62.47	1.0	1.0	1.0
12	C	c	1	5.0	15.0	0.75	1.34	55.81	1.0	1.0	0.0
7	A	c	1	4.0	47.0	2.35	3.5	53.64	0.0	1.0	1.0
3	A	a	3	4.0	22.0	1.1	1.72	51.04	1.0	0.0	1.0
6	A	b	3	3.0	32.0	1.6	2.22	43.19	0.0	0.0	1.0
1	A	a	0	3.0	14.0	0.7	1.18	35.56	0.0	0.0	0.0
9	B	c	0	4.0	9.0	0.45	1.06	33.87	1.0	0.0	0.0
14	C	c	3	3.0	15.0	0.75	1.34	33.48	0.0	0.0	0.0
10	B	c	1	3.0	21.0	1.05	2.17	29.0	0.0	0.0	0.0
4	A	b	2	2.0	24.0	1.2	1.79	26.8	0.0	0.0	0.0
11	C	b	2	2.0	27.0	1.35	2.55	21.12	0.0	0.0	0.0
8	B	b	2	2.0	27.0	1.35	2.78	19.39	0.0	0.0	0.0
13	C	c	2	1.0	32.0	1.6	2.6	12.28	0.0	0.0	0.0

Table 5. Computations for the example

	A_0	A_1	Optimal
z	414	478	587
z, %	71	81	100
Time lost for changeovers	7.5	6	4.5
Producing time	4.25	5.05	7.00
Time total	11.75	11.05	11.50

The solutions have been sorted with the non-increasing values of $d_j^\star$. Corresponding values of variables x_j are presented in column A_1. As one can see, in the first four main steps of the new method, the value 1 was assigned to the variables corresponding with orders $5, 2, 12, 7$. No other order could be included in the plan because of the violation of constraint (6). In order to make it possible to compare the results, we presented also the plan obtained by the naive heuristic (in column A_0) and the optimal solution generated by Gurobi (in the last column). The performance of the algorithms on the example has been summarized in Table 5.

4. Numerical experiments

In order to examine the method's performance, we run 2000 of numerical experiments on randomly generated test problems. The average running time for a single experiment was 0.02 s, 0.045 s, 1.46 s, 5.1 s, respectively, for algorithms A_0, A_1, d-RFSB and the exact solution by the Gurobi solver.

The problems were characterized with the following values of parameters ($x \in [a, b]$ means that the value of x was an integer chosen uniformly at random from the interval $[a, b]$):

- the number of features $m \in [2, 7]$,
- the number of orders $n \in [200, 2000]$
- the number of possible values of features $F_1, \ldots, F_m$ were chosen randomly from the intervals $\lambda_i \in \left[\left\lfloor \sqrt[m]{i \cdot n} \right\rfloor, 2\left\lfloor \sqrt[m]{i \cdot n} \right\rfloor\right]$, and then the values of $F_1, \ldots, F_m$ were chosen uniformly at random from the sets $F_i \in [1, \lambda_i]$, for $i = 1, \ldots, m$ independently for every order; this implies in particular that $1 \le f_1 \le \lambda_1$, $1 \le f_2 \le \lambda_1\lambda_2$ and $1 \le f_3 \le \lambda_1\lambda_2\lambda_3$ etc.,
- the production line's efficiency $E \in [500, 6000]$,
- sizes of orders $q_j \in [400, 6000]$,
- time capacity

$$T = \frac{1}{2}\sum_{j=1}^{n} a_j = \frac{1}{2}\sum_{j=1}^{n} \frac{q_j}{E}$$

- durations of changeovers $t_i \in [1/m, 10/m]$ with $t_i \ge t_{i+1}$,
- cleaning time $t_0 = 8$,
- unit profits $c_j \in [20, 70]$.

This choice of parameters serves two key purposes. First, we aimed to ensure that each feature significantly impacts the problem's solution. Otherwise, the orders could be distinguished mainly by the first few features, making the rest negligible, as the corresponding changeovers would nearly always occur alongside those with longer durations. Second, we wanted to independently analyze the influence of changeover duration and the number of features. Therefore, as we increased the number of features, we reduced their duration to avoid a strong correlation between these two factors.

The above was the so-called unbiased setting. We also performed several experiments for biased settings. The unit profits were generated differently, while the remaining parameters were defined in the same way as before. The change in the method of obtaining the values of c_j was motivated by the fact that the smaller orders correspond with higher unit profits. The exact method of generating the profits was:

$$b_j = \frac{40000}{q_j}, \quad c'_j \in [20,70], \quad c_j = \frac{1}{10}(c'_j + b_j)$$

where b_j defines the bias corresponding with the order quantity, and the division by 10 scales the value of c_j to fit the ranges of other parameters.

The heuristics were implemented in Python 3.7 [44], while the exact solution was obtained with Gurobi 9.1.2 [18]. The computations were performed on a PC with Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz with 16GB RAM and a 64-bit operating system.

Taking for the notation convenience $t_{m+1} = 0$ and $f_0 = 1$, the total time lost for changeovers in the case when all orders are performed is equal to

$$\begin{aligned} t_{ch}(P) := (t_0 - t_1) \times 1 + \sum_{i=1}^{m}\sum_{k=1}^{f_i}(t_i - t_{i+1})y_{ik} &= (t_0 - t_1) \times 1 + \sum_{i=1}^{m}\sum_{k=1}^{f_i}(t_i - t_{i+1}) \times 1 \\ = (t_0 - t_1) + \sum_{i=1}^{m}(t_i - t_{i+1})f_i = \sum_{i=0}^{m}(t_i - t_{i+1})f_i \end{aligned}$$

where the equality in the first line follows from the fact that $y_{ik} = 1$ if and only if $\mathbb{J}_{ik} \neq \emptyset$, cf. equations (6) and (7). Similarly, the working time in the case when all orders are performed is equal to

$$t_w(P) := \sum_{j=1}^{n} a_j \tag{15}$$

Let

$$r(P) = \frac{t_{ch}(P)}{t_w(P)} \tag{16}$$

It is natural to expect that the more time is spent on the changeovers, the more the approximate character of the presented methods will influence their efficiency. In other words, we expect that with the increase of $r(P)$, the performance of the presented heuristics should worsen.

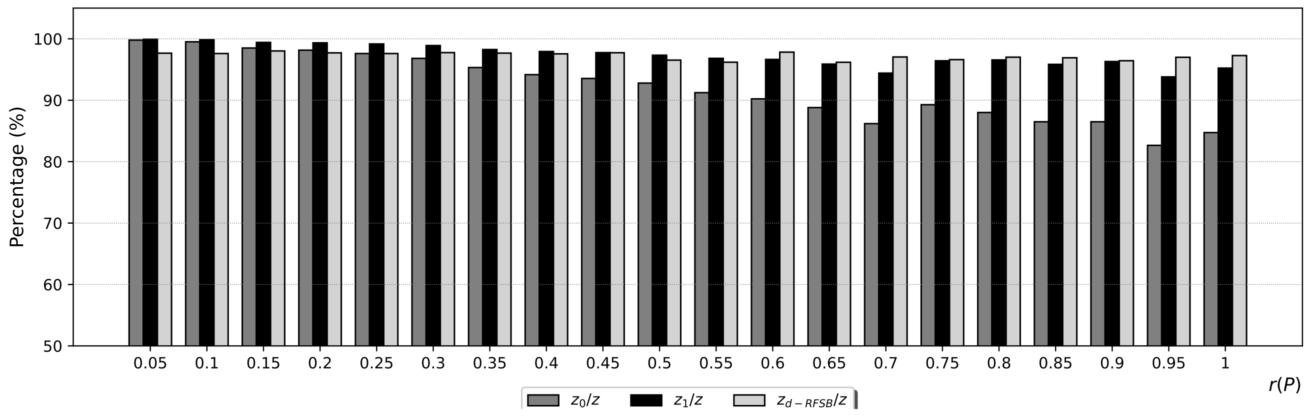

Figure 1. Results of the experiments – algorithms performance, unbiased case

Figure 1 illustrates the performance of algorithms A_0 and A_1 concerning the value of $r(P)$ on the randomly generated instances following the unbiased setting. The labels of the horizontal axis are the left ends of the left-side closed intervals of length 0.05 (e.g. 0.35 denotes the interval $[0.35, 0.4)$. The value on the vertical axis is the percentage of the total profit obtained by the heuristics concerning the exact optimum generated with Gurobi.

As we can see, the accuracy of both heuristics decreases as the ratio calculated with equation (16) goes up. This confirms our suppositions presented above. On the other hand, even for very elevated values of $r(P)$, the average accuracy of the second heuristics remains at the level of approximately 6$\%$ (the heuristic reaches about 94$\%$ of the optimal profit gained by Gurobi), while in the entire sample, it is only 3.1$\%$ (96.9$\%$ of the optimal profit). The naive heuristic performs much worse, its results are on average by 9$\%$ worse than the optimal profit of Gurobi in the entire sample, while in the worst case, the average difference reached the value of 17$\%$ (the profit equals to 83$\%$ of its optimal value). The performance of the d-RFSB algorithm remains consistently above 96.5 $\%$.

Analogous results for the instances following the biased setting have been presented in Figure 2. The labels of the axes are the same as in the previous case.

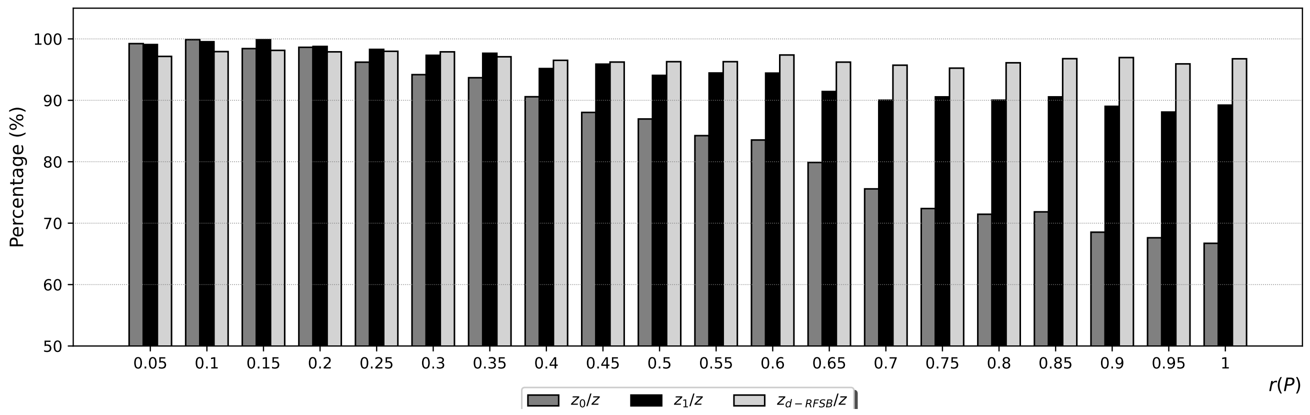

Figure 2. Results of the experiments – algorithms performance, biased case

The performance of both heuristics is much worse in this case: A_1 goes down even to 88.5$\%$ on average for the second last interval of the values of $r(P)$ (94.2$\%$ in the entire sample, i.e., the mean accuracy equals to 5.8$\%$). Things go even worse for the naive heuristic, where the lowest average (in the last considered interval) is 66$\%$, while in the entire sample, it is only 85.0$\%$ (accuracy of 15.0$\%$). As before, the performance of the d-RFSB algorithm remains consistently above 96.5 $\%$.

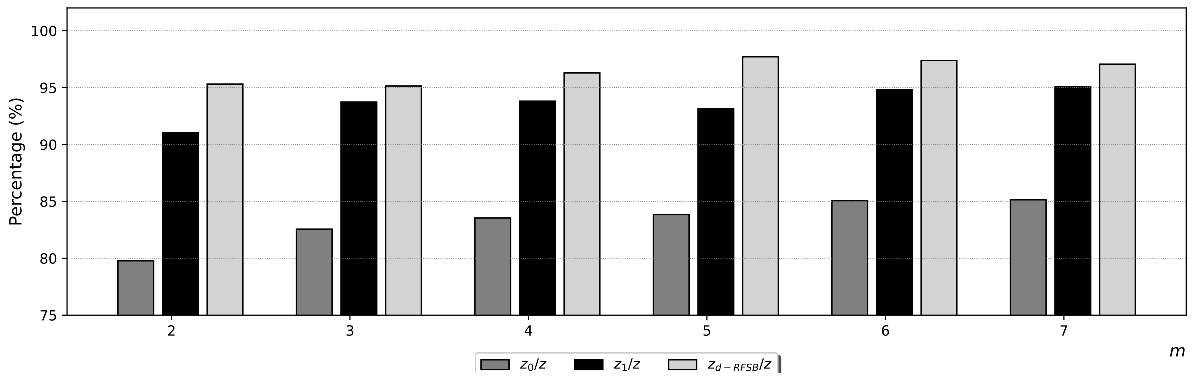

Figure 3. Results of the experiments – performance depending on the number of features, both cases

Figure 3 illustrates the algorithm performances based on the number of features, showing a slight increase as the number of features grows—an initially unexpected result. However, recall that we designed the experiments to ensure that the duration time and number of features are not dependent. Since we have already demonstrated the significant impact of duration time on the problem's solution, this relationship would have been evident if these two factors were correlated. Instead, our experiments indicate no significant dependence between the two properties.

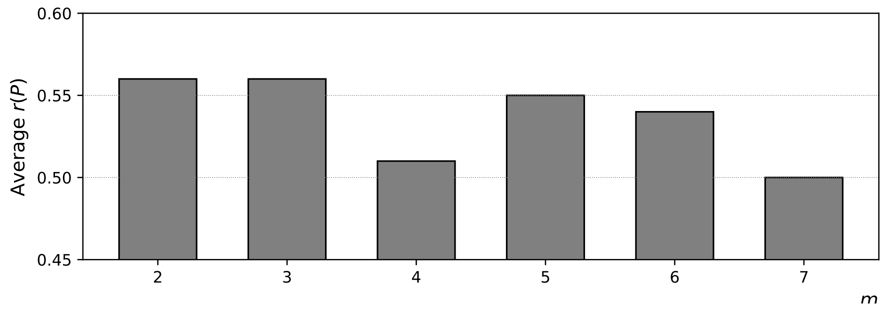

Figure 4. Results of the experiments – average value of $r(P)$ across experiment groups defined by m, both cases

As shown in Figure 4, the average value of $r(P)$ remains nearly constant across experiment groups defined by the number of features.

5. Case study

Such a problem arose in a branch of the company Intersnack located in Nysa in Poland which is a chips production plant. At that time the overall equipment effectiveness index (OEE index) in this branch was low compared to the other branches of Intersnack, but the assortment included many profitable indices (in terms of margin per kilogram). Nevertheless, due to the low OEE index, the management began to question whether the general strategy for selecting offers on the market was appropriate and started to work on algorithms to improve the strategy. With cooperation with the management of this branch of Intersnack, we formulated the mathematical model (5)–(8). To do it, we distinguished three main changeovers that affected the OEE index. These changeovers were forced by two different oils used in the frying process, seven different body types of the shapes of chips, and 28 different tastes of chips. The number of orders was equal to 290 (i.e. $n = 290$) and the number of distinct products was equal to 101 (more specific: $|F_1| = 2$, $|F_2| = 7$, $|F_3| = 28$, $f_1 = 2$, $f_2 = 11$ and $f_3 = 101$). The parameter $r(P)$, which in some way predicts the quotient of the time needed for changeovers to the time needed for actual production, was 0.32.

Furthermore, we observed a bias of the margin caused by the number of orders (Figure 5). Performance of A_0 was 71.33% and of A_1 was 93.01%.

While the optimization model (5)–(8) can be precisely solved using mixed integer linear programming (MILP) solvers, the management expressed a strong preference for developing an alternative approach – a greedy algorithm to rank the orders in question. Their rationale was that such an algorithm would offer a deeper understanding of the complex decision space, which might extend beyond the confines of the mathematical model. This deeper understanding could prove invaluable in adapting the solution to address additional, potentially critical, business intricacies that were not explicitly considered in the mathematical model but may gain significance in the future. This motivated the creation of algorithm A_1.

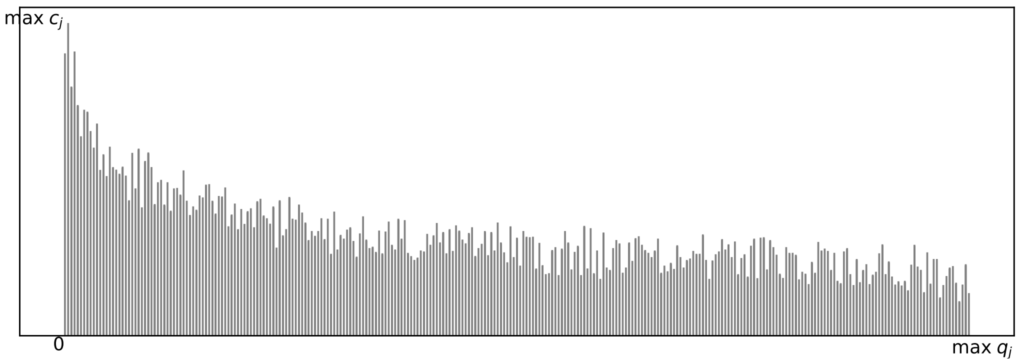

Figure 5. Bias of the margin

On the flip side, exact methods could face limitations when dealing with significantly larger instances of the model due to computational complexity. In contrast, the computational complexity of algorithm A_1 is relatively low, primarily dependent on sorting, and operates with a time complexity of $O(n \log n)$ (assuming the number of features is low compared to the number of orders which is natural in these kind of settings).

6. Conclusions

Starting from a real problem that occurred in a food industry company located in Poland, we developed a model and algorithm for solving this type of task , which was the main goal of the current paper. The problem in question is, in fact, a hybrid of two others known from the literature: the Problem of Accepting and Scheduling Orders and the Changeover Problem.

Although the method never found the global optimum, it achieved 99% of the optimal performance in 10% of the experiments. Nonetheless, it meets the primary condition specified by the company's management. It assigns easily interpreted indicators to orders, which, on the one hand, can be used to build order rankings, and on the other hand, allow for the analysis of orders and thus facilitate strategic business decisions.

A possible direction for further research that could be developed in the future is to adapt the method so that it can be used in situations where switching different features cannot be performed simultaneously. In such a situation, the order in which orders are executed could be important. In particular, one can imagine a situation in which, in an optimal solution, it would be profitable to change the configuration of the production line from one feature value to another and then change it back again. This would, of course, require at least some changes in the calculation of penalties and adjusted production times.

Acknowledgement

We sincerely thank the managers of the Intersnack – Renata Galas-Nocoń, Joanna Rychlewska-Guzik, and Piotr Choroszy for their invaluable contributions to this article. Their guidance, expertise, and deep understanding of the company's operations and industry trends were crucial in shaping the research direction and enriching the study with critical insights. This work would not have been possible without their dedicated support and collaboration. We also thank the anonymous reviewers for their meaningful comments that improved this manuscript, particularly for drawing our attention to the d-RFSB heuristic, which provided an interesting and valuable comparison to our approach.

References

[1] Blocher, J. D., Chand, S., and Sengupta, K. The changeover scheduling problem with time and cost considerations: Analytical results and a forward algorithm. *Operations Research 47*, 4 (1999), 559–569.

[2] Bowers, M. R., Groom, K., Ng, W. M., and Zhang, G. Cluster analysis to minimize sequence dependent changeover times. *Mathematical and Computer Modelling 21*, 11 (1995), 89–95.

[3] Brunaud, B., Perez, H. D., Amaran, S., Bury, S., Wassick, J., and Grossmann, I. E. Batch scheduling with quality-based changeovers. *Computers & Chemical Engineering 132* (2020), 106617.

[4] Casado, S., Laguna, M., Pacheco, J., and Puche, J. C. Grouping products for the optimization of production processes: A case in the steel manufacturing industry. *European Journal of Operational Research 286*, 1 (2020), 190–202.

[5] Castro, P. M. Optimal scheduling of a multiproduct batch chemical plant with preemptive changeover tasks. *Computers & Chemical Engineering 162* (2022), 107818.

[6] Castro, P. M., Grossmann, I. E., and Zhang, Q. Expanding scope and computational challenges in process scheduling. *Computers & Chemical Engineering 114* (2018), 14–42.

[7] Castro, P. M., Harjunkoski, I., and Grossmann, I. E. Greedy algorithm for scheduling batch plants with sequence-dependent changeovers. *AIChE Journal 57*, 2 (2011), 373–387.

[8] Charnsirisakskul, K., Griffin, P. M., and Keskinocak, P. Order selection and scheduling with leadtime flexibility. *IIE Transactions 36*, 7 (2004), 697–707.

[9] Chen, Y.-W., Lu, Y.-Z., and Yang, G.-K. Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling. *The International Journal of Advanced Manufacturing Technology 36*, 9-10 (2008), 959–968.

[10] Cplex, I. I. IBM ILOG CPLEX V12.1: User's Manual for CPLEX. International Business Machines Corporation Armonk, NY, USA, 2009.

[11] Díaz-Ramírez, J., and Huertas, J. I. A continuous time model for a short-term multiproduct batch process scheduling. *Ingeniería e Investigación 38*, 1 (2018), 96–104.

[12] Duncan, W. P. *Methods for Reducing Changeover Times Through Scheduling*. Master's thesis, University of Rhode Island, Kingston, RI, USA, 2011.

[13] Emami, S., Moslehi, G., and Sabbagh, M. A Benders decomposition approach for order acceptance and scheduling problem: a robust optimization approach. *Computational and Applied Mathematics 36*, 4 (2017), 1471–1515.

[14] Emami, S., Sabbagh, M., and Moslehi, G. A Lagrangian relaxation algorithm for order acceptance and scheduling problem: A globalised robust optimisation approach. *International Journal of Computer Integrated Manufacturing 29*, 5 (2016), 535–560.

[15] Engelmann, B., Schmitt, S., Miller, E., Bräutigam, V., and Schmitt, J. Advances in machine learning detecting changeover processes in cyber physical production systems. *Journal of Manufacturing and Materials Processing 4*, 4 (2020), 108.

[16] Geoffrion, A. M., and Graves, G. W. Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/LP approach. *Operations Research 24*, 4 (1976), 595–610.

[17] Germs, R., and van Foreest, N. D. Order acceptance and scheduling policies for a make-to-order environment with family-dependent lead and batch setup times. *International Journal of Production Research 51*, 3 (2013), 940–951.

[18] Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual (accessed on 18 October 2023).

[19] He, C., Leung, J. Y.-T., Lee, K., and Pinedo, M. L. Improved algorithms for single machine scheduling with release dates and rejections. *4OR 14*, 1 (2016), 41–55.

[20] Hong, S., Han, J., Choi, J. Y., and Lee, K. Accelerated dynamic programming algorithms for a car resequencing problem in automotive paint shops. *Applied Mathematical Modelling 64* (2018), 285–297.

[21] Hu, X., Blocher, J. D., Heese, H. S., and Zhou, F. Scheduling products with subassemblies and changeover time. *The Journal of the Operational Research Society 67*, 8 (2016), 1025–1033.

[22] Kapadia, M. S., Uzsoy, R., Starly, B., and Warsing Jr., D. P. A genetic algorithm for order acceptance and scheduling in additive manufacturing. *International Journal of Production Research 60*, 21 (2022), 6373–6390.

[23] Konge, U., and Subramanian, S. Scheduling of process plants with equipment changeover. *Computers & Chemical Engineering 162* (2022), 107812.

[24] Lawler, E. L. Fast approximation algorithms for Knapsack problems. *Mathematics of Operations Research 4*, 4 (1979), 339–356.

[25] Lee, H., and Maravelias, C. T. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. *Computers & Chemical Engineering 107* (2017), 171–183.

[26] Lei, D., and Guo, X. A parallel neighborhood search for order acceptance and scheduling in flow shop environment. *International Journal of Production Economics 165* (2015), 12–18.

[27] Li, Q., and Milne, R. J. A production scheduling problem with sequence-dependent changeover costs. *International Journal of Production Research 52*, 13 (2014), 4093–4102.

[28] Li, X., and Ventura, J. A. Exact algorithms for a joint order acceptance and scheduling problem. *International Journal of Production Economics 223* (2020), 107516.

[29] Li, X., Ventura, J. A., and Bunn, K. A. A joint order acceptance and scheduling problem with earliness and tardiness penalties considering overtime. *Journal of Scheduling 24*, 1 (2021), 49–68.

[30] Liu, P., and Lu, X. New approximation algorithms for machine scheduling with rejection on single and parallel machine. *Journal of Combinatorial Optimization 40*, 4 (2020), 929–952.

[31] Lockett, A. G., and Muhlemann, A. P. Technical note—a scheduling problem involving sequence dependent changeover times. *Operations Research 20*, 4 (1972), 895–902.

[32] Montoya-Torres, J. R., Soto-Ferrari, M., and González-Solano, F. Production scheduling with sequence-dependent setups and job release times. *DYNA 77*, 163 (2010), 260–269.

[33] Ou, J. Near-linear-time approximation algorithms for scheduling a batch-processing machine with setups and job rejection. *Journal of Scheduling 23*, 5 (2020), 525–538.

[34] Ou, J., Lu, L., and Zhong, X. Parallel-batch scheduling with rejection: Structural properties and approximation algorithms. *European Journal of Operational Research 310*, 3 (2023), 1017–1032.

[35] Oğuz, C., Salman, F. S., and Yalçin, Z. B. Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics 125*, 1 (2010), 200–211.

[36] Perea, F., Yepes-Borrero, J. C., and Menezes, M. B. C. Acceptance ordering scheduling problem: The impact of an order-portfolio on a make-to-order firm's profitability. *International Journal of Production Economics 264* (2023), 108977.

[37] Slotnick, S. A. Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research 212*, 1 (2011), 1–11.

[38] Spindler, S., Steinhardt, C., and Klein, R. Exact solution approaches for order acceptance and scheduling decisions in m-machine open shops. *Computers & Industrial Engineering 176* (2023), 108952.

[39] Sun, H., and Fan, S. Car sequencing for mixed-model assembly lines with consideration of changeover complexity. *Journal of Manufacturing Systems 46* (2018), 93–102.

[40] Sun, H., and Han, J. A study on implementing color-batching with selectivity banks in automotive paint shops. *Journal of Manufacturing Systems 44,* (2017), 42–52.

[41] Tarhan, I., and Oğuz, C. Generalized order acceptance and scheduling problem with batch delivery: Models and metaheuristics. *Computers & Operations Research 134* (2021), 105414.

[42] Tarhan, I., and Oğuz, C. A matheuristic for the generalized order acceptance and scheduling problem. *European Journal of Operational Research 299*, 1 (2022), 87–103.

[43] Thevenin, S., and Zufferey, N. Learning variable neighborhood search for a scheduling problem with time windows and rejections. *Discrete Applied Mathematics 261* (2019), 344–353.

[44] Van Rossum, G., and Drake Jr., F. L. *Python Reference Manual*. Centrum voor Wiskunde en Informatica, Amsterdam, 1995.

[45] Velez, S., Dong, Y., and Maravelias, C. T. Changeover formulations for discrete-time mixed-integer programming scheduling models. *European Journal of Operational Research 260*, 3 (2017), 949–963.

[46] Wang, Z., Qi, Y., Cui, H., and Zhang, J. A hybrid algorithm for order acceptance and scheduling problem in make-to-stock/make-to-order industries. *Computers & Industrial Engineering 127* (2019), 841–852.

[47] Xu, L., Wang, Q., and Huang, S. Dynamic order acceptance and scheduling problem with sequence-dependent setup time. *International Journal of Production Research 53*, 19 (2015), 5797–5808.