Constraint-Based Scheduling for Paint Shops in the Automotive Supply Industry

FELIX WINTER and NYSRET MUSLIU, Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, Institute for Logic and Computation, DBAI, TU Wien

Factories in the automotive supply industry paint a large number of items requested by car manufacturing companies on a daily basis. As these factories face numerous constraints and optimization objectives, finding a good schedule becomes a challenging task in practice, and full-time employees are expected to manually create feasible production plans.

In this study, we propose novel constraint programming models for a real-life paint shop scheduling problem. We evaluate and compare our models experimentally by performing a series of benchmark experiments using real-life instances in the industry. We also show that the decision variant of the paint shop scheduling problem is NP-complete.

CCS Concepts: • **Mathematics of computing → Combinatorial optimization**; • **Applied computing → Industry and manufacturing**.

Additional Key Words and Phrases: Constraint Programming, Paint Shop Scheduling, Exact Methods, NP-complete

ACM Reference Format:
Felix Winter and Nysret Musliu. 0000. Constraint-Based Scheduling for Paint Shops in the Automotive Supply Industry. *ACM Trans. Intell. Syst. Technol.* 0, 0, Article 0 (0000), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Paint shops belonging to the automotive supply industry produce a large number of items requested by car manufacturing companies, on a daily basis. To ensure a cost efficient production, modern factories adopt a high level of automation, including the use of multiple painting robots and conveyor belt systems. This sophisticated production process calls for good painting schedules, which are difficult to establish. Human planners are usually not able to identify optimized production sequences. Therefore, the development of automated techniques for paint shop scheduling in the automotive supply industry is urgently needed.

In the literature, related problems have been studied, and several publications have considered the minimization of color changes for paint shop scheduling in the automotive industry (e.g. [6, 15, 21, 22]). Previously, we introduced a real-life paint shop scheduling problem that appears in the automotive supply industry [26] which includes important practical features, such as the optimized allocation of materials to carrying devices, and considers many sequence and resource constraints. In addressing this problem, the goal is to find a production schedule that fulfills a large set of given

Authors' address: Felix Winter, winter@dbai.tuwien.ac.at; Nysret Musliu, musliu@dbai.tuwien.ac.at, Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, Institute for Logic and Computation, DBAI, TU Wien, Favoritenstraße 9, Vienna, Austria, 1040.

2157-6904/0000/0-ART0 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

demands and to minimize the numbers of color changes and carrying devices that are used to carry materials through the paint shop. Furthermore, the problem considers due date constraints. Therefore, a feasible solution should be able to ensure that all products placed on the carriers are scheduled on time. To solve the problem, we previously proposed a local-search based approach and evaluated its performance on a set of practical benchmark instances that we made publicly available in [26]. However, to the best of our knowledge, exact solution approaches have not been investigated in the literature, and optimal solutions for many instances are not yet known. Furthermore, a complexity analysis of the hardness of the problem has not been considered in the literature.

In the current work, we introduce for the first time exact approaches based on constraint programming (CP) to solve the paint shop scheduling problem. We propose two new CP models that can be used to efficiently formulate the problem. One of them uses a direct modeling approach, and the other one models the sequence constraints of the problem as deterministic finite automatons (DFAs). Furthermore, we evaluate and compare our proposed modeling techniques by performing a series of benchmark experiments using state-of-the-art solvers on known practical benchmark instances of paint shop scheduling. Although the exact methods we describe cannot be used to solve large practical instances, the proposed approaches can provide optimal solutions to nine benchmark instances. Additionally, we analyze the complexity of the paint shop scheduling problem and show that the decision variant of the problem is NP-complete.

In the next section, we provide a review of the related literature. In Section 3, we describe the paint shop scheduling problem as it appears in practice. In Sections 4 and 5, we respectively provide a formal description of the problem's input and propose our modeling techniques. In Sections 6 and Section 7, we present the NP-completeness proof and discuss the experimental results, respectively. In the last section, we provide our concluding remarks.

2 RELATED LITERATURE

Automated solution methods for production scheduling and sequencing problems in the automotive industry have been thoroughly studied. One of the earliest investigated problems from this area is the so-called car sequencing problem, which was first described in [15]. The goal of the original formulation of this sequencing problem is to find an optimized production sequence for a given set of cars; the manufacturing process for each car may require different assembly operations depending on the installation options ordered (e.g., sun roof, air conditioning). As each of these options is installed at a different station, solutions to the car sequencing problem should ensure that the capacity of these stations is never exceeded during production. These constraints are usually expressed as ratio constraints that restrict the number of cars having a certain option appearing in subsequences of the solution.

In [11] the car sequencing problem was identified as an NP-hard problem. Many heuristic and exact solution approaches have been investigated to solve this problem. One of the earliest exact approaches [3] successfully utilized a constraint logic programming approach to solve large practical problem instances. Other exact approaches for car sequencing have since been proposed, and they include mixed integer programming (e.g., [4]) and branch and bound algorithms (e.g., [5]). Metaheuristic and hybrid approaches have also been investigated to tackle extremely large instances for variants of the car sequencing problem (e.g., [13, 18]) in a reasonable runtime. In [21], the authors provided an extensive survey of the solution approaches for the car sequencing problem and described a widely investigated problem extension used in the ROADEF'2005 challenge. The extended problem formulation used in this challenge additionally considers the painting process and includes the minimization of the costs caused by necessary color changes in the car sequence. The investigated solution methods for the ROADEF'2005 problem include exact approaches based

on mixed integer programming (e.g., [17]) as well as metaheuristics such as ant colony optimization and local search (e.g., [7, 17]).

Another sequencing problem originating from automotive paint shops focuses solely on the minimization of costs induced by color changes [6]. The main goal of this paint shop problem is to find an optimal coloring for a given sequence of jobs that minimizes the required color changes. An NP-hardness proof and an exact approach using dynamic programming (under bounded instance parameters) were investigated in [6]. Linear programming and local search based approaches to tackling practically sized instances of the problem were studied in [12].

A sequential ordering problem in automotive paint shops was described in [22]. Similar to other paint shop problems, this sequential ordering problem aims to find a production sequence that minimizes the necessary color changes. However, this variant considers the utilization of so-called selectivity banks, in which multiple cars are grouped together in banks, as is often the case in automotive paint shops. The authors of [22] proposed a model as a sequential ordering problem and introduced an exact method based on a branch and bound algorithm. Further studies on the topic also investigated heuristic techniques to quickly produce efficient solutions (e.g., [23, 24]).

In [26], we proposed a novel paint shop scheduling problem occurring in paint shops of the automotive supply industry. Similar to previous production scheduling problems from the automotive area, our paint shop scheduling problem is aimed at creating an optimized schedule for a paint shop that minimizes color changes in the production sequence. However, as the manufacturing process of this particular problem produces car components that are placed on carrier devices, novel carrier constraints need to be considered, and the number of carrier changes in the schedule should be minimized. In contrast to previous automotive paint shop problems, the proposed paint shop scheduling problem introduces due date constraints and therefore requires a prompt scheduling of the required components. These unique properties make the proposed paint shop scheduling problem considerably different from previous automotive sequencing problems. Hence, we proposed a novel metaheuristic solution approach using simulated annealing in [26]. In the experimental evaluation, the metaheuristic approach was able to produce feasible solutions for large realistic problem instances within a reasonable runtime.

Another scheduling problem from the automotive supply industry deals with component primer painting [20]. The problem was deemed to be NP-hard in [20], and the authors further proposed an exact solution approach that uses mixed integer programming (MIP) and a metaheuristic approach that uses tabu search. Similar to the paint shop scheduling problem, the component primer painting problem deals with the placement of multiple automotive components on hanger devices. However, solution methods for this problem cannot be compared with the methods we investigate in the current work, as they do not consider due dates and they define different constraints and an objective function that includes capacity loss, mixing costs (when different item categories are placed on a hanger) and workload costs. Furthermore, the component primer painting problem does not consider the minimization of color changes or carrier changes and related constraints, all of which appear in the paint shop scheduling problem.

Table 1 presents an overview of the properties of the problems related to the paint shop scheduling problem investigated herein.

Columns 1–5 display from left to right the name of the related problem, considered constraints, considered solution objectives, example papers describing exact solution methods, and example papers investigating heuristic solution methods. The final row of the table shows the properties of the paint shop scheduling problem investigated in the current work.

Problem	Constraints	Objective Function	Exact Methods	Heuristics
Car Sequencing [15, 21]	Capacity Constraints, Color Batch Size	Color Changes, Capacity Violations	[3–5, 17]	[13, 17, 18]
Paint Shop Problem [6]	None	Color Changes	[6, 12]	[12]
Sequential Ordering in Paint Shops [22]	Precedence Constraints	Color Changes	[22]	[23, 24]
Component Primer Painting [20]	Hanger Eligibility, Hanger Capacity	Capacity Loss, Mixing Costs, Workload	[20]	[20]
Paint Shop Scheduling [26]	Due Dates, Resource Capacity, Forbidden Sequences	Color Changes, Carrier Changes	–	[26]

Table 1. An overview on the literature on related problems to the paint shop scheduling problem.

3 PAINT SHOP SCHEDULING PROBLEM

The aim of the paint shop scheduling problem we introduced in [26] is to determine an optimized production schedule that organizes the painting for a large number of automotive components within given due dates.

Two minimization criteria should be considered to reduce waste and save costs. First, the schedule should minimize the required color changes in the production sequences whenever possible. Second, the schedule should ensure the efficient utilization of the carrying devices used to transport the raw material components through the paint shop.

All items scheduled for painting need to be placed on customized carrier devices that move through the paint shop's painting cabins on a conveyor system. In each cabin, several painting robots apply paint on the carried automotive components. Carriers come in many different types, each of which can transport certain configurations of demanded materials. Hence, different carrier device types need to be used in production. Although combinations of different raw material items may be transported by a single carrier, scheduling products that should be painted with different colors on a single carrying device is impossible. Figure 1 shows a schematic of two carrier types and three possible material configurations. The carrier shown on the left uses a material configuration that transports two triangular and two square components, the middle carrier transports two circular and two square components, and the carrier on the right side transports three circular and three square components. The figure illustrates how the same carrier type (the left and middle carriers are similar, whereas the right carrier is of a different type) can be used to transport different material type combinations through the paint shop as long as all pieces on a single carrier are painted with the same color (e.g., white, light-gray).

The paint shops of the automotive supply industry are designed to support an almost fully automated production process. Therefore, any scheduled carrying devices are automatically moved through the paint shop on a circular conveyor belt system. Carriers can be inserted into and removed from the conveyor belt at two carrier gates. One of the gates is used to insert carrying devices while the other one can be used to remove carriers from the circular conveyor belt system. Once a carrier has been inserted, it moves through the cyclic paint shop system, wherein it repeatedly passes by the painting cabins, the carrier gates, and a material gate until the schedule selects it for ejection at the output gate. At the material gate, unpainted raw materials may be placed on

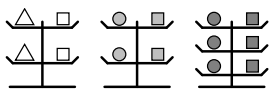

Fig. 1. Schematic showing three carriers of two different carrier types.

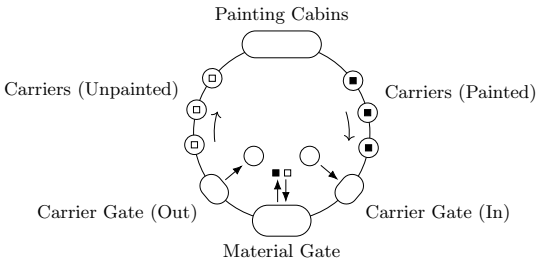

Fig. 2. Schematic showing a paint shop layout that is commonly used in the automotive supply industry.

any empty carrying device by paint shop employees. A loaded carrier then moves to the painting cabins, where the scheduled color is applied on all carried items. Whenever a loaded carrier arrives at the material gate after having completed a full round, another employee takes off the colored material pieces and may place new uncolored raw materials onto the carrier for painting in the succeeding round.

Figure 2 shows a schematic of a paint shop's layout and visualizes the movement of carriers through the paint shop. Carriers are displayed as circles in the figure, and some of them carry unpainted automotive components, which are visualized as small white squares. Carrying devices that have passed by the painting cabins are marked with black squares (which represent painted materials) in the graphic.

As the paint shop maintains a circular layout, the painting schedule is organized in rounds that are processed sequentially. Within each painting round, several carrier units are painted one after the other in a sequence that is predetermined by the schedule. However, the number of processed carriers per round and the exact sequence do not necessarily have to be equal for each round. A schedule therefore sets the painting sequences for multiple rounds and determines the raw material and color configurations for each scheduled carrying device. Note that in practice, the processing of a single paint shop round takes a fixed amount of time that does not depend on the number of carriers scheduled for the round (as long as the number of carriers per round stays within the specified boundaries). Therefore, the scheduling horizon of a problem instance is specified as the number of rounds to schedule, and due dates are expressed as due rounds in the problem input. The planner then needs to schedule carrier configurations into rounds so that all components are produced on time. In practice, schedules are usually planned weekly using a rolling horizon approach.

We derive a table representing a candidate solution to the paint shop scheduling problem. In this table, each column represents the scheduling sequence for a single round. Each table cell then assigns the carrier type, material configuration, and color that should be scheduled in the associated round sequence (from top to bottom). Figure 3 illustrates a toy problem solution for a scheduling horizon of three rounds. In the schedule shown in the figure, the carrier sequence (A1, A1, A2, B1, B2) scheduled for the first round (R1) includes five carriers. The first three carriers of this round sequence use a type A carrier with item configurations 1 and 2 and should be painted in a light gray color. Carriers 4 and 5 in R1 are of type B, and require a dark gray color and item configurations 1 and 2.

When all carrier configurations and colors that can be scheduled for production are considered, a large number of different schedules can be created. However, numerous constraints imposing

	R1	*R2*	*R3*
1	A1	A2	C1
2	A1	A2	C2
3	A2	C1	C3
4	B1	B2	B1
5	B2		B2

Fig. 3. Example of a painting schedule with three rounds. Each column represents the scheduled carrier sequences scheduled within a single round.

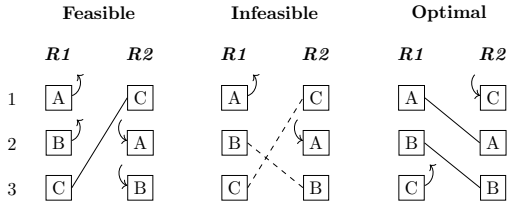

Fig. 4. Three options to reuse carriers between two consecutive rounds.

due dates and technical requirements regarding feasible carrier sequences (sequence constraints also apply to round overlapping sequences) need to be fulfilled.

R.1 All material demands must be scheduled for production on time.

R.2 Carrier type availabilities should be considered in each round (e.g., if 10 physical instances of carrier type A are available, then this carrier can never be scheduled more than 10 times in a single round).

R.3 Minimum/maximum carrier capacities should be considered in each round: The number of carriers scheduled for each round needs to fall within the minimum and maximum boundaries to ensure an efficient production cycle (empty carriers may be scheduled if necessary).

R.4 Forbidden carrier type sequences must not appear in the schedule: For logistical reasons at the material gate, certain carrier types are not allowed to directly follow another carrier type in the production sequence (e.g. a type A carrier should never directly follow a type B carrier in the production sequence).

R.5 Minimum and maximum carrier blocks should be considered: Similar to the consideration of the forbidden carrier type sequence constraints, logistical restrictions impose minimum and maximum block sequence constraints on scheduled carrier sequences.
In other words, whenever a carrier of type t is scheduled, the same carrier type needs to or may be used for the next consecutive carriers until the given minimum/maximum block length is reached (e.g., to illustrate the minimum block length, let the minimum block length for type t_1 be three and let the previously scheduled carrier type sequence be $\langle t_3, t_3, t_2, t_1 \rangle$; to satisfy the minimum block length, at least the next two carriers in the sequence need to be of type t_1).

R.6 Forbidden color sequences should be respected: For certain pairs of colors c_1, c_2, a number of carriers need to be painted in a different color before a switch from color c_1 to another color c_2 is legal in the production sequence.
For example, let this number for colors c_1 and c_2 be three. Then, the color sequences $\langle c_1, c_2 \rangle$ and $\langle c_1, y, c_2 \rangle$ would be illegal while the color sequence $\langle c_1, y, y, y, c_2 \rangle$ would be legal (assuming that $y \neq c_1$ and $y \neq c_2$).

A multiobjective function further includes two minimization criteria for the paint shop scheduling problem. The first optimization goal is to minimize color changes in the production sequence while the second optimization goal is concerned with the efficient utilization of carrying devices. In the following sections, we further explain the second minimization goal.

As a paint shop schedule usually does not use the same carrier type sequence in each round, the carriers need to be removed from and inserted to the conveyor belt system between rounds. However, if carriers of the same type are scheduled in two consecutive rounds some of them may be reused as long as the sequence of the kept carriers is compatible with the scheduled carrier sequence in the succeeding round. As the insertion and removal of carrier units to and from the circular track might lead to delays and are general not doable in parallel, the number of such

operations should be kept as low as possible. Note that for any given two consecutive rounds, the minimum number of required carrier insertions and removals can be calculated by determining the minimum string edit distance (ED) [25] between two carrier type round sequences.

Two consecutive rounds of carrier type sequences may be viewed as two strings: the minimum number of required carrier changes corresponds to the ED with only the insertion and deletion operations considered. Figure 4 visualizes three alternative ways to reuse carriers between two consecutively scheduled round sequences, each of which uses three carriers (R1: A, B, C and R2: C, A, B). The graphic shows how the ED determines the minimum number of required carrier changes.

The feasible option shown on the left side of the figure reuses only a single type C carrier and requires a total of two carrier insertions and two carrier removals. The infeasible option shown in the middle of the figure suggests retaining type B and C carriers between two consecutive rounds. However, this is technically not possible as C cannot be placed on an earlier position than B in the next round if it is reused (no edge crossings are allowed). The option shown on the right side of the figure requires the fewest number of carrier insertions and removals, that is, one insertion and one removal; this requirement corresponds to the minimum string ED in this case.

4 MODELING THE PROBLEM WITH CP

In this section, we briefly provide an overview on CP, prior to proposing a direct CP model for the paint shop scheduling problem. In Section 5, we further propose an alternative model for some of the problem's constraints.

4.1 Preliminaries

CP is a paradigm for solving combinatorial search problems using a wide range of techniques from areas such as artificial intelligence, computer science, and operations research. CP has been successfully applied to solve problems from many domains, including scheduling, vehicle routing, and planning.

In CP, users declaratively state the constraints that restrict feasible solutions to a search problem. In the case of optimization problems, an associated objective function that determines the cost of a solution is declared.

Formally, a constraint optimization problem is defined as follows:
Given a set X of variables $(x_1, x_2, \ldots, x_n)$, a set D of domains for each variable $(D_1, D_2, \ldots, D_n)$, a set C of constraints $(c_i \subseteq (D_1 \times D_2 \times \cdots \times D_n), \forall c_i \in C)$, and an objective function $f : (D_1 \times D_2 \times \cdots \times D_n) \rightarrow \mathbb{R}$, a constraint optimization problem is the problem of finding an assignment $x_i = d_i \in D_i, \forall i \in \{1, 2, \ldots, n\}$ such that all constraints are satisfied and the cost function is optimized.

In solving constraint optimization problems with CP, standard methods utilize a combination of backtracking search and constraint propagation, in which users can specify customized problem-specific branching strategies. Further information about CP is available in [19].

High-Level Modeling. In the following sections, we use a high-level CP modeling notation to propose CP models for the paint shop scheduling problem. Most parts of the models are directly solvable by CP. However, we implicitly make use of *constraint reification* to express conditional sums and logical implications. To illustrate how logical implications can be translated into low-level clauses, we consider a constraint of the form $(x_{1,1} = 0) \Rightarrow (x_{1,2} = 0)$. We can translate this constraint into the following clauses: $b_1 \Leftrightarrow (x_{1,1} = 0), b_2 \Leftrightarrow (x_{1,2} = 0), \neg b_1 \vee b_2$, where b_1, b_2 are Boolean variables.

We also implicitly make use of the *element constraint* to use variables as indices for array access in our models.

Further information on constraint reification and the element constraint is available in [19].

4.2 Input parameters

The following parameters describe instances of the problem:

Set of carrier configurations: K

Set of carrier types: T

Set of colors: C

Set of materials: M

Set of all rounds to schedule: R

Set of carrier positions per round (maximum number of positions per round): S

Minimum number of carriers that have to be scheduled in each round: $q \in \mathbb{N}_{>0}$

Number of available carriers of type t in round r: $a_{r,t} \in \{1, \ldots, |S|\}, \forall r \in R, t \in T$

Set of demands: D

Each demand will request a number a of materials m in color c that have to be scheduled until round r. The set of demands may contain optional demands that are due until future rounds (i.e. rounds lying outside the scheduling horizon).

Number of requested items per demand: $a_d \in \mathbb{N}_{>0}, \forall d \in D$

Material type of demand: $m_d \in M, \forall d \in D$

Due round of demand: $r_d \in \mathbb{N}_{>0}, \forall d \in D$

Color of demand: $c_d \in C, \forall d \in D$

Number of pieces of material type m that can be placed on configuration k: $u_{k,m} \in \mathbb{N}, \forall k \in K, m \in M$

Carrier type of each carrier configuration (v_0 will be set to 0): $v_x \in \{0, \ldots, |T|\}, \forall x \in \{0, \ldots, |K|\}$

Number of carriers scheduled in the round previous to the scheduling horizon (history round): $p \in \mathbb{N}$

As already mentioned, production in the paint shop will process one round of carriers after the other and therefore the conveyor belt system will never be empty. For this reason, whenever we need to create a new paint shop schedule we are given the carriers and colors used in the latest previous round of production as an input, so that the amount of carrier and color changes within the first round of the scheduling horizon can be determined.

Carrier type of the scheduled carrier at position i of the history round: $pt_i \in T, \forall i \in \{1, \ldots, p\}$

Used color at position i of the history round: $pc_i \in C, \forall i \in \{1, \ldots, p\}$

Forbidden carrier type sequences: F

First carrier type of forbidden sequence f: $t_f^1 \in T, \forall f \in F$

Second carrier type of forbidden sequence f: $t_f^2 \in T, \forall f \in F$

Minimum block size of consecutive carriers with type t: $b_t^{\min} \in \mathbb{N}_{>0}, \forall t \in T$
Whenever a carrier of type t is scheduled, the same carrier type needs to be used for the next consecutive carriers until the given minimum block length is reached. (e.g., let $b_{t_1}^{\min} = 3$ and let the previously scheduled carrier type sequence be $\langle t_3, t_3, t_2, t_1 \rangle$; to satisfy the minimum block length, at least the next two carriers in the sequence need to be of type t_1).

Maximum block size of consecutive carriers with type t: $b_t^{\max} \in \mathbb{N}_{>0}, \forall t \in T$

Set of forbidden color sequences: O

First color of forbidden color sequence o: $c_o^1 \in C, \forall o \in O$

Second color of forbidden color sequence o: $c_o^2 \in C, \forall o \in O$

The number of carriers that have to be painted in a different color before a switch from color c^1 *to color* c^2 *becomes legal for sequence o:* $j_o \in \mathbb{N}_{>0}, \forall o \in O$
For example let $j_o = 3$ for colors $c_o^1 = v$ and $c_o^2 = w$. Then the color sequences $\langle v, w \rangle$ and $\langle v, y, w \rangle$ would be illegal while the color sequence $\langle v, y, y, y, w \rangle$ would be legal (assuming that $y \neq v$ and $y \neq w$).

Color transition costs for all pairs of colors: $f_{c_1,c_2} \in \mathbb{N}, \forall c_1, c_2 \in C$

4.3 Decision Variables

Scheduled carrier configuration in round i and position j: $x_{i,j} \in \{0, \ldots, |K|\}, \forall i \in R, j \in S$
If the value 0 is assigned, the position is empty and no carrier will be scheduled at the position.

Scheduled color configuration in round i and position j: $y_{i,j} \in \{0, \ldots, |C|\}, \forall i \in \{0, \ldots, |R|\}, j \in S$
If the value 0 is assigned, the position is empty and will not be painted.

4.4 Auxiliary Variables

Number of scheduled carriers per round: $p_i \in \{0, \ldots, |S|\}, \forall i \in \{0, \ldots, |R|\}$

Number of totally scheduled carriers: $ps \in \{0, \ldots, |S| \cdot |R| + p\}$

Sequence variables that will convert a given round index i and position index j into a one dimensional position index: $seq_{i,j} \in \{0, \ldots, |S| \cdot |R| + p\}, \forall i \in \{0, \ldots, |R|\}, j \in S$
For example let exactly 100 carriers be scheduled in round 1 and the length of the history round p be 5, then $seq_{2,3}$ will be set to the value 108. $seq_{i,j}$ will be set to 0 if and only if no carrier is scheduled at position j in round i.

4.5 Hard Constraints

In the following we propose the set of hard constraints for our model. Note that all of them are essential and that we do not use any redundant constraints in our formulation.

(a) Bind the correct number of scheduled carriers to the associated helper variables:

$$\begin{aligned} p_0 &= p \\ p_i &= \sum_{\{j \in S | x_{r,j} \neq 0\}} 1 \quad \forall i \in R \\ ps &= \sum_{i \in \{0,\ldots,|R|\}} p_i \end{aligned} \tag{1}$$

(b) Set correct values to sequence variables:

$$\begin{gathered} seq_{0,j} = j \quad \forall j \in \{1, \ldots, p\} \\ seq_{0,j} = 0 \quad \forall j \in \{p+1, \ldots, |S|\} \\ seq_{i,1} = p + 1 + \sum_{z \in \{1,\ldots,i-1\}} p_z \quad \forall i \in R \\ seq_{i,j} = seq_{i,j-1} + 1 \quad \forall i \in R, j \in \{2, \ldots, |S|\} \text{ where } x_{i,j} \neq 0 \\ seq_{i,j} = 0 \quad \forall i \in R, j \in S \text{ where } x_{i,j} = 0 \end{gathered} \tag{2}$$

(c) Unplanned carrier positions should always be scheduled last in a round[1]:

$$(x_{i,j} = 0) \Rightarrow (x_{i,j+1} = 0) \quad \forall i \in R, j \in \{1, \ldots, |S| - 1\} \tag{3}$$

(d) Any scheduled carrier position must also assign a color and any unscheduled position must not assign a color:

$$(x_{i,j} \neq 0) \Leftrightarrow (c_{i,j} \neq 0) \quad \forall i \in R, j \in S \tag{4}$$

(e) All demands must be satisfied in time, where overproduction is allowed (this constraints models requirement **R.1** from Section 3):

$$\begin{gathered} \sum_{\{d \in D |\ m_d = m \wedge r_d <= r \wedge c_d = c\}} a_d \leq \sum_{\{x_{i,j} | i \in \{1,\ldots,r\} \wedge j \in \{1,\ldots,|S|\} \wedge y_{i,j} = c\}} u_{(x_{i,j}),m} \\ \forall r \in R, m \in M, c \in C \end{gathered} \tag{5}$$

(f) Carrier availabilities must be respected in each round (models requirement **R.2**):

$$\sum_{\{j | j \in S \wedge v_{(x_{r,j})} = t\}} 1 \leq a_{r,t} \quad \forall r \in R, t \in T \tag{6}$$

(g) The minimum round capacity must be fulfilled in each round (models requirement **R.3**):

$$p_r >= q, \forall r \in R \tag{7}$$

(h) Forbidden carrier type sequences must not appear in the schedule (models requirement **R.4**):

$$\begin{gathered} (v_{(x_{i,j})} \neq t_f^1) \vee (v_{(x_{i,j+1})} \neq t_f^2) \quad \forall f \in F, i \in R, j \in \{1, \ldots, |S| - 1\} \text{ where } j < p_i \\ (v_{(x_{i,(p_i)})} \neq t_f^1) \vee (v_{(x_{i+1,1})} \neq t_f^2) \quad \forall f \in F, i \in \{1, \ldots, |R| - 1\} \\ (pt \neq t_f^1) \vee (v_{(x_{1,1})} \neq t_f^2) \quad \forall f \in F \end{gathered} \tag{8}$$

(i) Minimum carrier type block size restrictions must be fulfilled (models requirement **R.5**)[2]:

[1] This constraint breaks symmetric solutions that would be possible if unused carrier positions could appear anywhere in the variable arrays. However, it is still not a redundant constraint, as other parts of the model rely on this restriction.

[2] For simplicity, we omit an additional corner case that has to be regarded: The last carrier type and color that appears in the history round also needs to be checked regarding the sequence constraints. This can simply be modeled by adding additional constraints for the history round.

$$\bigwedge_{z\in\{j+2,\ldots,|S|\}} (seq_{i,z} = 0 \vee seq_{i,z} \geq seq_{i,j+1} + b_t^{\min} \vee v_{(x_{i,z})} = t) \wedge$$
$$\bigwedge_{y\in\{i+1,\ldots,|R|\},z\in S} (seq_{y,z} = 0 \vee seq_{y,z} \geq seq_{i,j+1} + b_t^{\min} \vee v_{(x_{y,z})} = t) \wedge$$
$$\left(\bigvee_{z\in\{j+1,\ldots,|S|\}} (seq_{i,z} = seq_{i,j} + b_t^{\min} \wedge v_{(x_{i,z})} = t) \vee \right. \tag{9}$$
$$\left. \bigvee_{y\in\{i+1,\ldots,|R|\},z\in S} (seq_{y,z} = seq_{i,j} + b_t^{\min} \wedge v_{(x_{y,z})} = t) \right)$$
$$\forall t \in T, i \in R, j \in \{1,\ldots,|S|-1\} \text{ where } j < p_i \wedge v_{(x_{i,j})} \neq t \wedge v_{(x_{i,j+1})} = t$$

$$\bigwedge_{z\in\{2,\ldots,|S|\}} (seq_{i+1,z} = 0 \vee seq_{i+1,z} \geq seq_{i+1,1} + b_t^{\min} \vee v_{(x_{i+1,z})} = t) \wedge$$
$$\bigwedge_{y\in\{i+2,\ldots,|R|\},z\in S} (seq_{y,z} = 0 \vee seq_{y,z} \geq seq_{i+1,1} + b_t^{\min} \vee v_{(x_{y,z})} = t)$$
$$\left(\bigvee_{z\in\{1,\ldots,|S|\}} (seq_{i+1,z} = seq_{i+1,1} + b_t^{\min} - 1 \wedge v_{(x_{i+1,z})} = t) \vee \right. \tag{10}$$
$$\left. \bigvee_{y\in\{i+2,\ldots,|R|\},z\in S} (seq_{y,z} = seq_{i+1,1} + b_t^{\min} - 1 \wedge v_{(x_{y,z})} = t) \right)$$
$$\forall t \in T, i \in \{1,\ldots,|R|-1\} \text{ where } v_{(x_{i,p_i})} \neq t \wedge v_{(x_{i+1,1})} = t$$

(j) Maximum carrier type block size restrictions must be fulfilled (models requirement **R.5**)[2]:

$$\bigvee_{z\in\{j+1,\ldots,|S|\}} (seq_{i,z} > seq_{i,j} \wedge seq_{i,z} \leq seq_{i,j} + b_t^{\max} \wedge v_{(x_{i,z})} \neq t) \vee$$
$$\bigvee_{y\in\{i+1,\ldots,|R|\},z\in S} (seq_{y,z} > seq_{i,j} \wedge seq_{y,z} \leq seq_{i,j} + b_t^{\max} \wedge v_{(x_{y,z})} \neq t) \tag{11}$$
$$\forall t \in T, i \in R, j \in S, \text{ where } j \leq p_i \wedge v_{(x_{i,j})} = t \wedge seq_{i,j} =\leq ps - b_t^{\max}$$

(k) No forbidden color sequences should occur in the schedule (models requirement **R.6**)[2]:

$$\bigwedge_{z\in\{j+1,\ldots,|S|\}} (seq_{i,z} = 0 \vee seq_{i,z} > seq_{i,j} + j_o \vee y_{i,z} \neq c_o^2) \wedge$$
$$\bigwedge_{x\in\{i+1,\ldots,|R|\},z\in S} (seq_{x,z} = 0 \vee seq_{x,z} > seq_{i,j} + j_o \vee y_{x,z} \neq c_o^2) \tag{12}$$
$$\forall o \in O, i \in R, j \in S \text{ where } j \leq p_i \wedge y_{i,j} = c_o^1$$

4.6 Auxiliary Variables for the Objective Function

The amount of color change costs occurring in round r of the schedule:

$$cc_r \in \mathbb{N}, \forall r \in R$$

The number of required carrier type changes between round r and $r+1$:

$$sc_r \in \{0, \ldots, |S| \cdot 2\}, \forall r \in \{0, \ldots, |R| - 1\}$$

The number of carriers that will be reused between round r and round $r+1$:

$$sk_r \in \{0, \ldots, |S|\}, \forall r \in \{0, \ldots, |R| - 1\}$$

Information on the position of the kept carrier sequence in the next/previous round:

$$kept^1_{i,j} \in \{0, \ldots, |S|\}, \forall i \in \{0, \ldots, |R| - 1\}, j \in S$$
$$kept^2_{i,j} \in \{0, \ldots, |S|\}, \forall i \in R, j \in S$$

4.7 Hard Constraints for Objective Function

Calculate color changes per round: [3]

$$cc_i = \sum_{j \in \{1,\ldots,|S|-1\}} f_{(y_{i,j}),(y_{i,j+1})} + f_{(y_{i-1,p_i-1}),(y_{i,1})} \quad \forall i \in R \tag{13}$$

All kept carrier type sequences between consecutive rounds have to be legal: [4]

In Figure 4 in Section 3, we show which carrier types may be reused between two consecutive rounds by drawing edges that connect the associated positions. We initially experimented with a modeling approach that introduces variables for all possible edges and tries to maximize the number of selected edges without causing any edge crossings to capture the ED between two consecutive rounds. However, using this model in practical instances is not efficient. Therefore, we propose a modeling approach that introduces variables to store the positions of all reused carriers in equations 14, 15, and 16. For example, if the second reused carrier that is scheduled on position four in the current round sequence should be reused in the next round sequence at position three, the associated variables store the values three and four ($kept^1_{i,2} = 3, kept^2_{i,2} = 4$, where i could be any round index). A value of zero is assigned to a $kept^y_{i,x}$ variable when less than x carriers are kept between the corresponding round (where $y \in \{1, 2\}$). Table 2 shows example variable assignments that correspond to the edge examples that are showcased in Figure 4.

$$\begin{gathered} kept^1_{i,j} > kept^1_{i,j-1} \quad \forall i \in \{0, \ldots, |R| - 1\}, i \in \{2, \ldots, |S|\} \text{ where } kept^1_{i,j} \neq 0 \\ kept^2_{i,j} > kept^2_{i,j-1} \quad \forall i \in R, i \in \{2, \ldots, |S|\} \text{ where } kept^2_{i,j} \neq 0 \end{gathered} \tag{14}$$

$$\begin{gathered} kept^2_{1,j} \leq p \quad \forall j \in S \text{ where } kept^2_{1,j} \neq 0 \\ kept^1_{i,j} \leq p_{i+1} \quad \forall i \in \{0, \ldots, |R| - 1\}, j \in S \text{ where } kept^1_{1,j} \neq 0 \\ kept^2_{i,j} \leq p_{i-1} \quad \forall i \in \{2, \ldots, |R|\}, j \in S \text{ where } kept^2_{1,j} \neq 0 \\ kept^1_{0,j} = 0 \wedge kept^2_{1,j} = 0 \quad \forall j \in \{p+1, \ldots, |S|\} \\ kept^1_{i,j} = 0 \wedge kept^2_{i+1,j} = 0 \quad \forall i \in \{1, \ldots, |R| - 1\}, j \in S \text{ where } j > p_i \\ kept^1_{i,j} > 0 \wedge kept^2_{i+1,j} > 0 \quad \forall i \in \{0, \ldots, |R| - 1\}, j \in S \end{gathered} \tag{15}$$

$$\begin{gathered} pt_{(kept^2_{1,j})} = v_{(x_{1,(kept^1_{0,j})})} \quad \forall j \in S \text{ where } kept^2_{1,j} \neq 0 \\ v_{(x_{i,(kept^2_{i+1,j})})} = v_{(x_{i+1,(kept^1_{i,j})})} \quad \forall i \in \{1, \ldots, |R| - 1\}, j \in S \text{ where } kept^1_{i,j} \neq 0 \end{gathered} \tag{16}$$

[3] We assume that color costs from and to 0 will always be 0

[4] For simplicity, we omit a special condition that handles the corner case of an empty history round. In this case one can simply add a constraint that forces all carriers of round 1 to be inserted if $p = 0$.

Feasible		
x	$kept^1_{1,x}$	$kept^2_{2,x}$
1	1	3
2	0	0
3	0	0

Infeasible		
x	$kept^1_{1,x}$	$kept^2_{2,x}$
1	3	3
2	1	2
3	0	0

Optimal		
x	$kept^1_{1,x}$	$kept^2_{2,x}$
1	2	1
2	3	2
3	0	0

Table 2. Table shows the $kept^1_{i,j}, kept^2_{i,j}$ variable values that correspond to the three options to reuse carriers between two consecutive rounds as shown in Figure 4.

Calculate the number of reused carriers after each round:

$$sk_i = \sum_{\{j \mid j \in S \wedge kept^1_{i,j} \neq 0\}} 1 \quad \forall i \in \{0, \ldots, |R| - 1\} \tag{17}$$

Count the total number of required carrier changes between two rounds:

$$sc_i = p_i - sk_i + p_{i+1} - sk_i \quad \forall i \in \{0, \ldots, |R| - 1\} \tag{18}$$

4.8 Objective Function

The objective function of the paint shop scheduling problem aims to minimize the number of carrier changes (sc) and color change costs (cc) per round.

The sums are squared because the required changes should be distributed over the scheduling horizon and peaks of many changes within single rounds should be avoided.

$$\min \sum_{i \in R} cc_i^2 + \sum_{i \in \{0,\ldots,|R|-1\}} sc_i^2 \tag{19}$$

Note that the objective function is essentially a multiobjective function that combines the color change and carrier change objectives as a weighted sum of squared changes per round (using identical weights of value 1). This function was formulated with support from expert-practitioners to reasonably capture the cost factors in real-life automotive paint shop settings; thus, we do not consider additional alternative multiobjective functions in the current work.

We want to point out that color change costs and the number of carrier changes are in similar orders of magnitude for all practical problem instances used in our experimental evaluation. Therefore, we do not discuss the normalization of the two objectives in this paper.

5 MODELING THE PROBLEM WITH DFAS

In this section, we propose a different way to model the sequence constraints (requirements **R.4**, **R.5**, and **R.6** from Section 3) by using DFAs. For this variant of the model, we replace equations 8, 9, 10, 11, and 12 with automaton formulations. All automatons process either the total sequence of scheduled carrier types or the total sequence of scheduled colors. We can provide the total carrier type or color sequence in our model by simply concatenating the values of all two indexed decision variables ($x_{i,j}$ or $y_{i,j}, \forall i \in R, j \in S$) into a one-dimensional list. The automatons can then be used to check whether or not the total color or carrier type sequence can be accepted. Automaton-based models can be directly solved by CP using the *regular constraint* [16].

5.1 Forbidden carrier type sequences

For each forbidden carrier type sequence $f \in F$, we model an automaton that processes the total sequence of scheduled carrier types (this model replaces Equation 8 from Section 4). One state accepts all carrier types, whereas the second state does not accept any carrier type t^2_f that

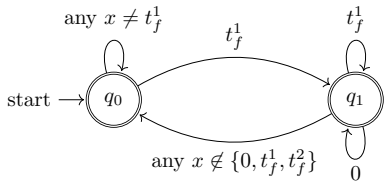

Fig. 5. Automaton generated for each carrier type sequence to check the forbidden carrier type sequence constraint.

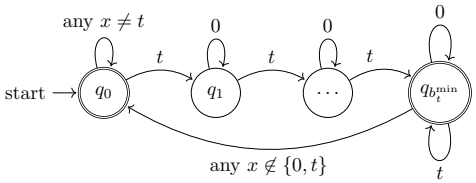

Fig. 6. Automaton constructed for each carrier type $t \in T$ to check the minimum carrier block type size constraint.

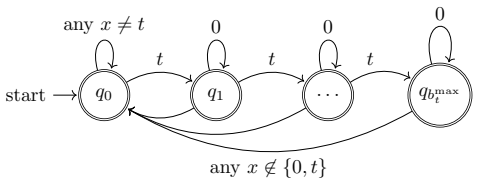

Fig. 7. Automaton generated for each carrier type to check the maximum carrier block type size constraint.

immediately follows a carrier of type t_f^1. Figure 5 shows how automatons can be constructed to check that no forbidden carrier type sequence occurs in the schedule.

The first state q_0 in Figure 5 accepts any carrier type. State q_1 is entered whenever the first carrier type of the forbidden sequence (t_f^1) is encountered and does not accept the second type (t_f^2) before any other type is encountered. Both states are legal final states.

5.2 Minimum carrier type block sizes

For each carrier type $t \in T$, we model an automaton that processes the total sequence of scheduled carrier types (this model replaces Equations 9 and 10 from Section 4). Figure 6 shows how automatons can be constructed to check the minimum carrier type block size constraint.

The first state q_0 of Figure 6 accepts any carrier type that is different from t. States $q_1 - q_{b_t^{\min}}$ are used to count the consecutive assignments of carrier type t. States q_0 and $q_{b_t^{\min}}$ are the only legal final states.

5.3 Maximum carrier type block sizes

For each carrier type $t \in T$, we model an automaton that processes the total sequence of scheduled carrier types (this model replaces Equation 11 from Section 4).

Figure 7 shows how automatons can be constructed to check the maximum carrier type block size constraint.

The first state q_0 in Figure 7 accepts any carrier type that is different from t. States $q_1 - q_{b_t^{\max}}$ are used to count the consecutive assignments of carrier type t. All states are legal final states.

5.4 Forbidden color sequences

For each forbidden color sequence $o \in O$, we model an automaton that processes the total sequence of scheduled colors (this model replaces Equation 12 from Section 4).

Figure 8 shows how automatons can be constructed to check the forbidden color sequence constraint.

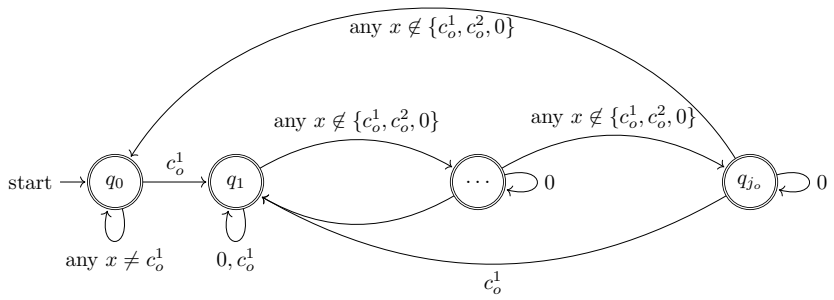

Fig. 8. Automaton generated for each forbidden color sequence to check the forbidden color sequence constraint.

The first state q_0 in Figure 8 accepts any color assignment that is different from c_o^1. States $q_1 - q_{j_o}$ are used to assert that an assignment of color c_o^2 may only occur if color c_o^1 has not been encountered within the previous j positions. All states are legal final states.

6 COMPLEXITY RESULTS

In this section, we show that the decision variant of the paint shop scheduling problem (which asks whether or not a feasible schedule with an objective value $\leq t$ can be found) is NP-complete. We prove the following:

Theorem 6.1. *The decision variant of the paint shop scheduling problem is NP-complete.*

Proof. We provide a polynomial time reduction from the set cover problem [10] to the paint shop scheduling problem.

Consider an arbitrary instance of the set cover problem consisting of the universe of elements $U = \{1, \ldots, w\}$, an integer k, and a set S that denotes the collection of z sets. The union of all sets in S is equal to the universe $U = \{1, \ldots, w\}$. The question is whether a set covering of size k or less is available or not.

We construct an instance of the paint shop scheduling problem as follows. We set the scheduling horizon to a single round ($R = \{1\}$) and set $C = \{1\}$ as only one color to be considered for scheduling. The set of materials M is set to match all items of the universe U, $M = \{1, \ldots, w\}$. The maximum number of allowed carrier devices per round is set to k while the minimum number of required carrier devices per round is set to $q = 1$. We only consider a single carrier type and therefore set $T = \{1\}$. The set of all demands D that need to be scheduled is given as $D = \{(1, i, 1, 1) | i \in \{1, \ldots, w\}\}$ (we schedule each material exactly once until round 1). We further create z carrier configurations, each of which corresponds to a single set in S: $K = \{1, \ldots, z\}$. All configurations belong to the same carrier type. Therefore, we set $v_k = 1, \forall k \in K$. The materials contained within each configuration should be equal to all elements contained in the associated set. Hence, we set $\forall x \in K, m \in M$:

$$u_{x,m} = \begin{cases} 1, & \text{if } m \text{ is contained in the associated set of configuration } x \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, we set the number of carriers in the history round to $p = 0$. Then, we set all color costs to 0 and disable all sequence dependent hard constraints and the carrier availability constraint by setting $f_c(1, 1) = 0, b_1^{\min} = 1, b_1^{\max} = k, o = 0, f = 0$ and $a_{1,1} = k$. Note that by setting color costs

to 0 and not using sequence constraints, we do not interfere with the original specification but instead, we simply build legal instances to the problem by modifying the input parameters.

We now prove the following:

Theorem 6.2. *There exists a set cover of size k or less if and only if there exists a feasible paint shop schedule with total costs lower than or equal to k^2.*

Proof. As we set all color transition costs to 0 and the history round contains 0 carriers, the objective function of the paint shop scheduling problem becomes equal to the squared number of scheduled carriers in round 1 (any carrier needs to be inserted). As we have set the number of maximum carriers per round to k, any feasible paint shop schedule fulfilling the maximum round capacity hard constraint will have an objective value $\leq k^2$. Furthermore, because we have disabled all hard constraints, except the demand constraint, in the paint shop scheduling instance, any schedule that satisfies all demands becomes a feasible schedule.

Let S be a set cover using k' sets (where $k' \leq k$). Then, all elements of the universe U are contained in at least one of the selected sets. Through our reduction, the paint shop scheduling problem is constructed in such a way that each element of the universe has to be scheduled at least once in round 1. For each set $s \in S$, there exists a carrier configuration that carries each element in s exactly once. Therefore, there exists a set of k' carrier configurations that can be scheduled in any sequence to fulfill all demands.

At this point, we prove the opposite direction. Let P be a feasible paint shop schedule. Then, any material should be scheduled exactly once in round 1. Therefore, any repeatedly used configurations can be removed from P in such a way that each carrier configuration scheduled in P is used exactly once without violating the demand constraint. As we have set the maximum carriers per round to k in our reduction, we have k' (where $k' \leq k$) carriers in the schedule. Given such a schedule that uses k' carriers, we can construct a feasible set covering of size k' by using the sets corresponding to the configurations used in P.

We then show that the decision variant of the paint shop scheduling problem is NP-complete. Suppose that we have given a candidate solution P. We show below that we can verify in polynomial time whether or not P is a feasible solution to the problem.

The number of carriers in P cannot be larger than $|S| \cdot |R|$ (see input parameters in Section 4). We can simply check the carrier availability and round capacity constraints by counting the number of carriers in each round. Similarly, the sequence constraints (minimum/maximum block length of consecutive carrier types, forbidden carrier types and color sequences) can be checked by iterating over the scheduled sequence. Furthermore, we can check the demand constraint by verifying that Equation 5 holds. We have to perform not more than $|R| \cdot |M| \cdot |C|$ comparisons to check this equation. For each comparison, we need to consider at most d demands and $|S| \cdot |R|$ positions to calculate the sums.

To calculate the total color change costs of P, we iterate over the scheduled sequence, similar to our approach to the sequence-dependent hard constraints. Finally, we need to determine the maximum number of carriers that can be reused between any two consecutive rounds in the schedule to calculate the total carrier change costs. As mentioned previously, we can establish this number by solving the corresponding string ED problem for each pair of consecutive rounds (the polynomial time algorithms for calculating the minimum ED were described in [25]). □

7 EMPIRICAL EVALUATION

In this section, we provide a detailed description of the experimental evaluation of the CP models proposed in this work. First, we briefly describe the experimental setup and computational environment (Section 7.1). Second, we introduce the search strategies used to program the search for the evaluated CP solvers (Section 7.2). Finally, we provide a summary of the experimental results and discuss the evaluation of the proposed models (Section 7.3).

7.1 Experimental Environment

To evaluate our models, we implemented the direct model proposed in Section 4 and the model using DFAs proposed in Section 5 by using the MiniZinc [14] modeling language, which provides interfaces to state-of-the-art CP and MIP solvers. All experiments were conducted on a computing cluster with 10 identical nodes, each having 24 cores, an Intel(R) Xeon(R) CPU E5–2650 v4 @ 2.20 GHz and 252 GB RAM.

We evaluated the proposed models by using the set of benchmark instances that we previously introduced to evaluate metaheuristic approaches to the paint shop scheduling problem [26][5]. This collection of instances includes 24 instances based on real life planning scenarios. Instances 1–12 model problems for a small paint shop with a maximum capacity of 19 carriers per round. Instances 13–24 describe problems for large-scale paint shops that allow up to 480 carriers per round. The set of small instances and the set of large instances describe six different planning horizons of 7, 20, 50, 70, 100, and 200 rounds (two instances for each horizon: one that does not include forbidden carrier/color sequence constraints and another one that includes forbidden sequence constraints).

Table 3 presents an overview of the size parameters for all instances. The columns show from left to right the instance ID, the number of rounds, the length of the planning horizon in days (in the industry, about five rounds are usually processed within 24 hour shifts), the maximum carrier capacity per round, whether or not forbidden sequence constraints are included, and the number of generated variables and constraints (# vars and # cs, respectively). To determine the number of generated variables and constraints, we analyzed the output of the MiniZinc compiler using the direct model (direct) and the model using DFAs (regular). A – indicates that the compiler ran out of memory on our benchmark machine or could not finish execution within 6 hours.

The size parameters displayed in Table 3 show that the model using DFAs generally leads to a lower number of variables and constraints for instances 1–10 in comparison with the direct model; hence, the model using DFAs has high efficiency. Furthermore, we can see that the MiniZinc compiler was not able to encode the large instances 11–24 on our machine within 6 hours of runtime, thereby indicating the tremendous size of the search space of large practical problem instances.

We conducted a large number of experiments with two state-of-the-art CP solvers: Chuffed [1], which uses lazy clause learning; and Gecode [8], which is a nonlearning solver. In Section 7.2, we provide details about the 13 programmed search strategies that we evaluated for the CP solvers. Both solvers were run on each of the instances using both proposed models and search strategies within a runtime limit of 6 hours. Therefore, a total of 312 experiments were conducted for each CP solver.

Using the MiniZinc compiler we were able to automatically translate the proposed CP models for paint shop scheduling into MIP formulations. Therefore, we could also conduct experiments with the two state-of-the-art MIP solvers Gurobi [9] and CPLEX [2] under the same runtime restrictions we used for evaluating the CP solvers.

[5]All instances, the MiniZinc code, and the solution validator are publicly available online: https://www.dbai.tuwien.ac.at/staff/winter/.

	# rounds	horizon days	capacity	forbidden seq	direct # vars	direct # cs	regular # vars	regular # cs
Instance 1	7	1.4	19	no	283845	289639	9237	12004
Instance 2	7	1.4	19	yes	399066	406964	9369	11499
Instance 3	20	4	19	no	3102108	3121429	25702	31801
Instance 4	20	4	19	yes	2891799	2912206	29312	37257
Instance 5	50	10	19	no	18374644	18435525	94066	124711
Instance 6	50	10	19	yes	19249180	19302056	66212	81411
Instance 7	70	14	19	no	42968894	43101797	210417	289019
Instance 8	70	14	19	yes	34183055	34258489	102820	129488
Instance 9	100	20	19	no	48323257	48439788	204091	282930
Instance 10	100	20	19	yes	84017220	84172576	232421	312301
Instance 11	200	40	19	no	-	-	-	-
Instance 12	200	40	19	yes	-	-	-	-
Instance 13	7	1.4	480	no	-	-	-	-
Instance 14	7	1.4	480	yes	-	-	-	-
Instance 15	20	4	480	no	-	-	-	-
Instance 16	20	4	480	yes	-	-	-	-
Instance 17	50	10	480	no	-	-	-	-
Instance 18	50	10	480	yes	-	-	-	-
Instance 19	70	14	480	no	-	-	-	-
Instance 20	70	14	480	yes	-	-	-	-
Instance 21	100	20	480	no	-	-	-	-
Instance 22	100	20	480	yes	-	-	-	-
Instance 23	200	40	480	no	-	-	-	-
Instance 24	200	40	480	yes	-	-	-	-

Table 3. An overview of the instance size parameters for the 24 instances that have been used for empirical evaluation.

Furthermore, we conducted a set of experiments using the local search based solver we previously proposed in [26] on our benchmark machine to compare the heuristic results with the results produced by the proposed exact methods. As the local search solver randomly selects search moves, we conducted 10 repeated runs for each instance and calculated the arithmetic mean solution costs for our evaluation.

7.2 Programmed Search Strategies

We evaluated the performance of the CP solvers Chuffed and Gecode by using several programmed search strategies, which are based on variable- and value selection heuristics. Such heuristics determine the order of the explored variable and value assignments for a CP solver and can play a critical role in reducing the search space that needs to be enumerated by the solver. For our experiments, we implemented the search strategies directly in the MiniZinc modeling language using search annotations.

7.2.1 Variable Selection. We defined the variable selection strategies for the decision variables that capture the scheduled carrier configurations $x_{i,j}, \forall i \in R, j \in S$ and the scheduled colors $y_{i,j}, \forall i \in R, j \in S$. Additionally, we investigated variable selection heuristics that set an order on the assignment of auxiliary variables that capture the number of required carrier changes between two rounds $sc_i, \forall i \in \{0, \ldots, |R| - 1\}$. We also experimented with search strategies that focus on these sc variables as the calculation of the required carrier changes is one of the more complex parts of the models. Any variables that we do not explicitly mention in the search strategies are selected on the basis of the CP solvers' default strategy after all the mentioned variables have been assigned.

- **default:** No variable and value selection is specified, and the solver uses its default strategy.
- **custom1:** All carrier configuration variables are selected first, followed by all color variables (i.e., $x_{1,1}, x_{1,2}, \ldots, x_{i,j}, y_{1,1}, y_{1,2}, \ldots, y_{i,j}$).

- **custom2:** The carrier change auxiliary variables are selected before the carrier configurations and colors are assigned
  (i.e., $sc_0, sc_1, \ldots, sc_{|R|-1}, x_{1,1}, x_{1,2}, \ldots, x_{i,j}, y_{1,1}, y_{1,2}, \ldots, y_{i,j}$).
- **custom3:** For each position in the schedule the associated carrier configuration variable is selected first, followed by the associated color variable
  (i.e., $x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \ldots, x_{i,j}, y_{i,j}$).
- **custom4:** The carrier change variables are selected first; then, the process continues with **custom3**
  (i.e., $sc_0, sc_1, \ldots, sc_{|R|-1}, x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \ldots, x_{i,j}, y_{i,j}$).
- **smallest:** The variables with the smallest possible domain value are chosen first (ties are broken on the basis of the order of **custom1**).
- **first fail:** The variables with the smallest domains are chosen first (ties are broken on the basis of the order of **custom1**).

7.2.2 Value Selection. We experimented with two different value selection heuristics:

- **min**: The smallest value is first assigned from a variable domain.
- **split**: The variable domain is bisected to first exclude the upper half of the domain.

Using the seven different variable selection strategies together with the two value selection heuristics we conducted experiments with a total of 13 search strategy configurations for each instance (the **default** variable selection uses the solver's default value selection).

7.3 Computational Results

In the following, we present the computational results for the paint shop scheduling problem that we obtained using the evaluated CP and MIP solvers. First, we provide an overview of the detailed CP results produced with Chuffed and Gecode. Second, we present the results produced by the MIP solvers Gurobi and CPLEX in Section 7.3.2. Finally, we present an overall comparison of the best results in Section 7.3.3.

7.3.1 CP Results. We conducted experiments for instances 1–24 by using the introduced search strategies with Chuffed and Gecode (for Chuffed, we activated the free search parameter which allows the solver to alternate between the given search strategy and its default one on each restart). However, Gecode was not able to produce any feasible solution within the runtime limit. Thus, we only discuss the results produced with Chuffed in this section.

Tables 4 and 5 display an overview of results achieved with Chuffed using the direct model and the DFA model, respectively. Both tables present the summarized results for each of the evaluated search strategies and time limit configurations in a single row. The first seven columns display from left to right the used search strategy, the time limit, the number of best results achieved within its group ("group" refers to runs using the same model and runtime), the number of fastest optimality proofs in the group, the number of optimal solutions found, the number of proven optimal solutions, and the number of instances where a feasible solution could be achieved. Columns 8–11 present from left to right the average number of visited nodes in the search tree (only for instances that could be solved to optimality), the average runtime needed for optimality proofs, the standard deviation of visited nodes for proofs, and the standard deviation of the optimality proof time.

The results in both tables show that different search strategies do not exert a large effect on the number of solved instances. Nevertheless, the default search strategy seems to be slightly weaker than the other search strategies, and the smallest variable selection strategy leads the two models to the largest number of best solutions and optimality proofs.

Table 6 displays the overall best costs achieved for instances 1–10 by using Chuffed with the direct model (direct) and the DFA model (reg) (we omitted the results for instances 11–24 as the solution process ran out of memory for these instances). The results formatted in bold face denote

search	time	#best	#fastest	#opt	#proof	#solved	avg nodes	avg rt	std nodes	std rt
custom1 min	6h	2	0	2	2	**4**	287513.50	79.96	302407.65	73.54
custom1 split	6h	2	0	2	2	**4**	258341.00	68.06	260843.21	56.77
custom2 min	6h	3	0	3	**3**	**4**	972613.33	1547.57	1022208.46	2544.58
custom2 split	6h	3	0	3	**3**	**4**	1796283.33	2657.52	1146115.93	4176.63
custom3 min	6h	2	0	2	2	**4**	390482.00	78.61	237821.22	51.71
custom3 split	6h	2	0	2	2	**4**	520798.00	89.18	35434.54	20.68
custom4 min	6h	3	0	3	**3**	**4**	2033199.67	2092.07	1686560.49	3245.35
custom4 split	6h	3	0	3	**3**	**4**	2171945.33	4536.86	1933987.52	7501.61
default	6h	3	0	3	**3**	3	3221216.33	4856.34	4419678.13	6773.03
ff min	6h	2	0	2	2	**4**	2108669.00	470.00	608751.06	63.00
ff split	6h	2	0	2	2	**4**	2204912.00	481.15	472642.90	61.87
smallest min	6h	**4**	1	**4**	**3**	**4**	97822.67	45.25	157924.49	33.66
smallest split	6h	**4**	**2**	**4**	**3**	**4**	110939.00	43.81	182193.95	33.21

Table 4. Table summarizing the experimental results achieved with Chuffed and the direct model.

search	time	#best	#fastest	#opt	#proof	#solved	avg nodes	avg rt	std nodes	std rt
custom1 min	6h	9	1	**9**	**9**	**10**	3133111	1286.46	7235326.43	2274.45
custom1 split	6h	**10**	2	**9**	**9**	**10**	3481850.67	1109.3	6981431.78	1896.39
custom2 min	6h	9	0	8	8	**10**	1013865.75	709.87	1635642.59	1482.07
custom2 split	6h	7	0	7	7	**10**	9037592.57	1055.71	20273523.56	2518.71
custom3 min	6h	9	0	8	8	**10**	1013865.75	709.87	1635642.59	1482.07
custom3 split	6h	9	0	8	8	**10**	1246310.63	855.64	1966662.15	1929.19
custom4 min	6h	8	0	7	7	**10**	13717699.29	1177.45	29593979.97	2751.27
custom4 split	6h	8	0	7	7	9	21727481.14	1497.90	52229928.06	3715.43
default	6h	6	0	6	6	7	887576.83	226.43	1221199.62	458.27
ff min	6h	9	1	8	8	**10**	1025480.88	604.63	1841286.56	1314.26
ff split	6h	9	0	8	8	**10**	1608216.25	939.78	2789085.47	2002.13
smallest min	6h	**10**	2	**9**	**9**	**10**	3443121.67	1796.47	7117553.6	3624.84
smallest split	6h	9	**3**	**9**	**9**	**10**	2324348.44	1614.02	4194724.49	3787.35

Table 5. Table summarizing the experimental results achieved with Chuffed and the DFA model.

	chuffed-6h	proof time	chuffed-reg-6h	proof time
Instance 1	**775***	80.96s	**775***	3.10s
Instance 2	**842***	29.33s	**842***	0.70s
Instance 3	**961***	256.95s	**961***	1.84s
Instance 4	**918**	-	**918***	26.27s
Instance 5	-	-	**530***	33.68s
Instance 6	-	-	**842***	9.78s
Instance 7	-	-	**844***	2410.03s
Instance 8	-	-	**1237***	572.40s
Instance 9	-	-	**975***	3622.07s
Instance 10	-	-	**964**	-

Table 6. Table showing the best costs achieved for instances 1–10 using Chuffed.

the overall best results, a * indicates that the solver could prove optimal costs within the time limit, and a – denotes the instances where no solution could be found.

We can clearly observe in Table 6 that the DFA-based model can solve more instances and prove more optimality results within the runtime limit in comparison with the direct model. For instances 1–3, for which both models were able to prove optimality, the DFA-based model could provide a much faster optimality proof, indicating that this model improves the performance of the solution process.

	cplex-6h	best bd	proof t	cplex-reg-6h	best bd	proof t
I 1	777	744	-	776	643	-
I 2	991	841	-	**842**	842	10992.47s
I 3	-	-	-	2761	961	-
I 4	-	-	-	12920	844	-
I 5	-	-	-	11085	225	-
I 6	-	-	-	1933	730	-
I 7	-	-	-	-	187.4	-
I 8	-	-	-	-	961	-
I 9	-	-	-	-	280.25	-
I 10	-	-	-	-	961	-

Table 7. Table showing the results achieved for instances 1–10 using CPLEX.

	gurobi-6h	best bd	proof t	gurobi-reg-6h	best bd	proof t
I 1	**775**	768	-	**775**	775	6637.99s
I 2	**842**	842	9444.53s	**842**	842	127.56s
I 3	-	-	-	**961**	961	282.80s
I 4	-	-	-	967	862.49	-
I 5	-	-	-	**530**	530	13398.48s
I 6	-	-	-	-	842	-
I 7	-	-	-	904	841	-
I 8	-	-	-	-	964	-
I 9	-	-	-	-	582	-
I 10	-	-	-	-	961	-

Table 8. Table showing the results achieved for instances 1–10 using Gurobi.

7.3.2 Integer Programming Results. As we implemented our models using the MiniZinc constraint modeling language, we could directly use the MiniZinc compiler to convert the direct model and the DFA-based model into MIP encodings. In the following, we present the experimental results produced using these encodings together with the MIP solvers Gurobi and CPLEX.

Table 7 provides an overview of the best results for instances 1–10 with CPLEX (the solution process with encodings for instances 11–24 ran out of memory on our benchmark machine before any result could be produced). The columns on the left side of the table include the results achieved using the direct formulation: the cost of the best solution found within the runtime limit of 6 hours (cplex-6h), the best bound achieved using the direct formulation (best bd), and the required optimality proof time in seconds. The results in bold face denote the overall best results, and a – indicates that no solution could be found within the runtime limit. The right side of the table similarly shows the results achieved with CPLEX using the DFA formulation.

Table 7 shows that the direct formulation is only able produce solutions for the two smallest instances within the time limit, whereas the formulation using DFAs seems to be more efficient as it derived solutions for six instances and generated one optimality proof.

Table 8 provides the results achieved with Gurobi in our experiments. The results are presented in the same way as that in Table 7: The left side shows the best cost solutions produced with the direct model (gurobi-6h), followed by the best objective bound and the required optimality proof time. The right side of the table shows the best results achieved with the DFA-based problem formulation.

solver	# best	# fastest	# opt	# proven	# solved	avg nodes	avg rt	std nodes	std rt
chuffed-6h	4	0	4	3	4	96910	43.01	158722.61	32.53
chuffed-reg-6h	**10**	**9**	**9**	**9**	**10**	984148.89	735.66	1625073.26	1330.49
gecode-6h	0	0	0	0	0	-	-	-	-
gecode-reg-6h	0	0	0	0	0	-	-	-	-
cplex-6h	0	0	0	0	2	-	-	-	-
cplex-reg-6h	1	0	1	1	6	14375	10981.08	-	-
gurobi-6h	2	0	2	1	2	2149	9276.63	-	-
gurobi-reg-6h	4	0	4	4	6	110315.25	5071.33	213079.3	6269.04

Table 9. Table summarizing the experimental results achieved with all evaluated CP and MIP solvers using the proposed models.

The results presented in Table 8 show that Gurobi can reach improved results relative to CPLEX for all instances, except instance 5. Moreover, Gurobi can provide optimality proofs for four instances. The DFA-based model leads to improved results relative to the direct formulation of the problem.

7.3.3 Comparison of Results. The summarized results of the experiments with the evaluated CP and MIP solvers are shown in Table 9. From left to right, columns 1–5 show the solver configuration (*6h* indicates the time limit of 6 hours, the DFA-based model is compared with the direct model indicated by *reg*), the number of overall best cost solutions reached, the number of overall fastest optimality proofs, the number of solutions solved to optimality, and the number of optimal cost proofs. Columns 6–10 show the number of instances where a feasible solution could be obtained, the number of average expanded nodes (only for optimally solved solutions), the average proof time, the standard deviation of expanded nodes for optimally solved solutions, and the standard deviation of proof times.

The results shown in Table 9 indicate that Chuffed based on the DFA formulation clearly produces the best results for the most number of instances and provides the fastest optimality proofs in our experiments. By contrast, the Gecode solver could not reach any feasible solution in our experiments. Furthermore, the DFA formulation generally leads to improved results for the MIP solvers as the number of solved instances is higher than that of the results of the direct model.

Table 10 compares the overall best cost solutions achieved by the solvers within the time limit of 6 hours (6h). The table includes the results produced with the local search method proposed in [26] (localsearch). For a fair comparison, we repeated the experiments with local search on our benchmark machine and calculated the solution costs on the basis of the arithmetic mean of 10 repeated runs as this method depends on randomly generated moves. The results formatted in bold face indicate the best results per instance, and a * denotes proven optimal solutions.

The results presented in Table 10 show that Chuffed produced the best results in our experiments for instances 1–10 and that it can prove optimality for instances 1–9. By contrast, Gecode was not able to solve a single instance. This result indicates that lazy clause learning is able to produce improved results for these instances relative to a nonlearning solver. CPLEX was able to prove a single optimal solution (instance 2), and Gurobi was able to prove three optimal solutions (instances 2, 3, and 5) in our evaluation. Large instances (instances 11–24) could only be solved by local search in our experimental setting as the MiniZinc compiler ran out of memory due to the excessive number of required variables and constraints. We further observed that local search failed to reach any of the optimal solutions for instances 1–10. Therefore, the exact methods in this work are highly effective for reaching optimal quality results for instances 1–10, which could not be reached

	chuffed-6h	cplex-6h	gurobi-6h	localsearch-6h
I 1	**775***	776	**775***	930.9
I 2	**842***	**842***	**842***	1015.5
I 3	**961***	2761	**961***	971.6
I 4	**918***	12920	967	1100.8
I 5	**530***	11085	**530***	551.7
I 6	**842***	1933	-	863.5
I 7	**844***	-	904	912.1
I 8	**1237***	-	-	1529.5
I 9	**975***	-	-	1406.3
I 10	**964**	-	-	1029.9
I 11	-	-	-	**4471.5**
I 12	-	-	-	**4917.9**
I 13	-	-	-	**62816.5**
I 14	-	-	-	**91587.3**
I 15	-	-	-	**136675.8**
I 16	-	-	-	**180608.1**
I 17	-	-	-	**297230.8**
I 18	-	-	-	**526878**
I 19	-	-	-	**460643.5**
I 20	-	-	-	**839361.1**
I 21	-	-	-	**841710.7**
I 22	-	-	-	**1524201.9**
I 23	-	-	-	**1641116.1**
I 24	-	-	-	**2542131.3**

Table 10. Table showing the best results per instance produced by the exact methods proposed in this paper and the state-of-the-art metaheuristic for paint shop scheduling.

by metaheuristic methods. However, local search was the only evaluated method that could solve extremely large practical instances.

8 CONCLUSION

In this work, we proposed CP modeling techniques to solve a real-life paint shop scheduling problem. Additionally, we analyzed the problem's complexity and proved that the decision variant is NP-complete.

Our experimental results showed that the exact methods investigated herein can be used successfully to solve small- to medium-sized instances, which still entail a huge search space. The advantage of these methods is that they could provide previously unknown optimal solutions for some instances. Our experiments showed that the best results could be achieved using the DFA-based model. Based on this model, Chuffed produced better results than Gurobi and CPLEX for the majority of the benchmark instances.

In the future, we intend to consider the hybridization of the proposed modeling techniques together with existing metaheuristic approaches within the framework of large neighborhood search.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

[1] Geoffrey G. Chu. 2011. Improving combinatorial optimization. (2011).

[2] IBM Corporation. 2019. *IBM ILOG CPLEX 12.10 User's Manual.*

[3] Mehmet Dincbas, Helmut Simonis, and Pascal Van Hentenryck. 1988. Solving the Car-Sequencing Problem in Constraint Logic Programming. In *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI'88)*. Pitman Publishing, Inc., USA, 290–295.

[4] Andreas Drexl and Alf Kimms. 2001. Sequencing JIT Mixed-Model Assembly Lines Under Station-Load and Part-Usage Constraints. *Management Science* 47, 3 (2001), 480–491.

[5] Andreas Drexl, Alf Kimms, and Lars Matthießen. 2006. Algorithms for the car sequencing and the level scheduling problem. *J. Sched.* 9, 2 (2006), 153–176.

[6] Th. Epping, W. Hochstättler, and P. Oertel. 2004. Complexity results on a paint shop problem. *Discrete Applied Mathematics* 136, 2 (Feb. 2004), 217–226.

[7] Caroline Gagné, Marc Gravel, and Wilson L. Price. 2006. Solving real car sequencing problems with ant colony optimization. *European Journal of Operational Research* 174, 3 (Nov. 2006), 1427–1448.

[8] Gecode Team. 2019. *Gecode: Generic Constraint Development Environment.* http://www.gecode.org

[9] LLC Gurobi Optimization. 2020. *Gurobi Optimizer Reference Manual.* http://www.gurobi.com

[10] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*. Plenum Press, New York, 85–103.

[11] Tamás Kis. 2004. On the complexity of the car sequencing problem. *Operations Research Letters* 32, 4 (July 2004), 331–335.

[12] Frédéric Meunier and Bertrand Neveu. 2012. Computing solutions of the paintshop–necklace problem. *Computers & Operations Research* 39, 11 (Nov. 2012), 2666–2678.

[13] Ignacio Moya, Manuel Chica, and Joaquín Bautista. 2019. Constructive metaheuristics for solving the Car Sequencing Problem under uncertain partial demand. *Comput. Ind. Eng.* 137 (2019).

[14] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *Principles and Practice of Constraint Programming – CP 2007 (Lecture Notes in Computer Science)*, Christian Bessière (Ed.). Springer, Berlin, Heidelberg, 529–543.

[15] Bruce D. Parrello, Waldo C. Kabat, and L. Wos. 1986. Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem. *Journal of Automated Reasoning* 2, 1 (March 1986), 1–42.

[16] Gilles Pesant. 2004. A Regular Language Membership Constraint for Finite Sequences of Variables. In *CP (Lecture Notes in Computer Science)*, Vol. 3258. Springer, 482–495.

[17] Matthias Prandtstetter and Günther R. Raidl. 2008. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research* 191, 3 (Dec. 2008), 1004–1022.

[18] Markus Puchta and Jens Gottlieb. 2002. Solving Car Sequencing Problems by Local Optimization. In *EvoWorkshops (Lecture Notes in Computer Science)*, Vol. 2279. Springer, 132–142.

[19] Francesca Rossi, Peter van Beek, and Toby Walsh (Eds.). 2006. *Handbook of Constraint Programming*. Foundations of Artificial Intelligence, Vol. 2. Elsevier.

[20] Ivan Kristianto Singgih, Onyu Yu, Byung-In Kim, Jeongin Koo, and Seungdoe Lee. 2020. Production scheduling problem in a factory of automobile component primer painting. *Journal of Intelligent Manufacturing* (Jan. 2020).

[21] Christine Solnon, Van Dat Cung, Alain Nguyen, and Christian Artigues. 2008. The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. *European Journal of Operational Research* 191, 3 (Dec. 2008), 912–927.

[22] S. Spieckermann, K. Gutenschwager, and S. Voß. 2004. A sequential ordering problem in automotive paint shops. *International Journal of Production Research* 42, 9 (May 2004), 1865–1878.

[23] Hui Sun, Shujin Fan, Xianle Shao, and Jiangong Zhou. 2015. A colour-batching problem using selectivity banks in automobile paint shops. *International Journal of Production Research* 53, 4 (Feb. 2015), 1124–1142.

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

[25] Robert A. Wagner and Michael J. Fischer. 1974. The String-to-String Correction Problem. *Journal of the ACM (JACM)* (Jan. 1974).

[26] Felix Winter, Nysret Musliu, Emir Demirović, and Christoph Mrkvicka. 2019. Solution Approaches for an Automotive Paint Shop Scheduling Problem. *Proceedings of the International Conference on Automated Planning and Scheduling* 29 (July 2019), 573–581.