Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

Solution Approaches for an Automotive Paint Shop Scheduling Problem

Felix Winter, Nysret Musliu
Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling
DBAI, TU Wien, Karlsplatz 13, 1040 Vienna, Austria
{winter,musliu}@dbai.tuwien.ac.at

Emir Demirović
School of Computing and Information Systems
University of Melbourne, Parkville, Victoria, Australia
emir.demirovic@unimelb.edu.au

Christoph Mrkvicka
MCP GmbH
Canovagasse 7, 1010 Vienna, Austria
christoph.mrkvicka@mc-partners.at

Abstract

In the paint shops of the automotive supply industry, a large number of synthetic material pieces need to be painted every day to provide the large variety of items required for car manufacturing. Because of the sophisticated automated production process and the tight due dates requested by car manufacturers, finding an optimized production schedule becomes a challenging task that is at the present time performed by multiple human planners.

In this paper, we formulate and solve a novel real-life paint shop scheduling problem from the automotive supply industry which introduces unique constraints and objectives that do not appear in the existing literature. Additionally, we provide a new collection of benchmark instances based on real-life planning scenarios that can be used to evaluate solution techniques for the problem.

An exact approach based on constraint programming is able to provide optimal solutions for smaller instances, but many larger instances could not be solved yet. Therefore, we propose a metaheuristic method based on local search that uses novel neighborhood relations and various ways to escape local optima. Our approach is able to provide feasible solutions for all instances within reasonable running time.

Introduction

Painting large amounts of synthetic material pieces that are required for car manufacturing is a cost-intensive and time-consuming production process. Therefore, paint shops of the automotive supply industry utilize a high level of automation including a conveyor belt system and multiple painting robots. The sophisticated production process introduces the need for a production schedule that has to fulfill many constraints and minimization objectives. However, at the present time, planning is usually done by humans and therefore it is currently not possible to efficiently find optimized production sequences. For this reason, there is a strong need to develop efficient automated scheduling techniques for paint shop scheduling in the automotive supply industry.

In this paper, we introduce a novel scheduling problem that appears in the paint shops of the automotive supply industry. The main goal of the problem is to find a production sequence that fulfills a large set of given demands and minimizes the number of color changes as well as the number of carrying devices that are required to transport demanded materials through the paint shop. Previous work has dealt with production scheduling problems from the automotive industry that consider the minimization of color changes in the production sequence (e.g. (Spieckermann, Gutenschwager, and Voß 2004), (Solnon et al. 2008), (Prandtstetter and Raidl 2008)) and several variants have shown to be NP-complete (Epping, Hochstättler, and Oertel 2004). However, minimizing the number of color changes forms only one part of the multi-objective optimization problem we introduce in this work. The problem we describe further includes the need for finding an optimized allocation of materials onto carrying devices while fulfilling many problem-specific sequence and resource constraints, that to the best of our knowledge have not been described before in connection with paint shop scheduling. Furthermore, previous scheduling problems from the automotive area usually incorporate an objective function that uses linear sums that capture color change and setup costs (e.g. (Prandtstetter and Raidl 2008)). However, the paint shop scheduling problem we introduce in this paper includes a unique aspect of balancing the changes over the scheduling horizon that is expressed with a quadratic objective function which to the best of our knowledge has not been considered for automotive scheduling problems. Additionally, to evaluate the problem's objective function, multiple instances of the longest common subsequence problem (LCS) (Hirschberg 1977) have to be solved. Another challenge that has to be tackled to efficiently solve the problem we describe is caused by the very large-scale planning scenarios that usually appear in practice.

In this paper, we introduce a greedy algorithm and novel neighborhood operators that can be used together with a local search based approach to tackle very large practical problems. Furthermore, we propose innovative extensions to metaheuristic techniques that incorporate elements of simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983), a min-conflicts heuristic (Minton et al. 1990) and tabu search (Glover 1986) to escape local optima while searching for an optimal solution. We provide a set of 24 instances that are closely based on real-life paint shop scheduling prob-

lems and can be used to benchmark the solution approaches to the problem. Additionally, we show that our metaheuristic methods are able to provide promising solutions for all of the benchmark instances.

The following points summarize the main contributions:

- We specify a novel scheduling problem that appears in the paint shops of the automotive supply industry.
- We generate 24 benchmark instances closely based on real-life planning scenarios. These instances will be made publicly available to serve as benchmarks for the scientific community.
- We propose a metaheuristic method that utilizes novel neighborhood moves to solve the problem.
- We propose a novel adaptive neighborhood move acceptance function that can be used together with simulated annealing and the quadratic objective function that is used for the paint shop scheduling problem.
- We propose a greedy algorithm that can be used to generate an initial solution for our metaheuristic approach to solve very large practical instances in reasonable time.
- We perform a series of benchmark experiments for evaluation and provide new upper bounds for large practical problem instances using our metaheuristic approach.

In the next section, we describe the problem as it appears in paint shops of the automotive supply industry. Afterwards, we give a formal problem specification of the problem followed by a description of the solution approaches we propose in this paper. We then provide an overview of the experiments that we conducted and evaluate the results before we make concluding remarks and talk about future work.

The Paint Shop Scheduling Problem

A typical automotive supply company will serve not only one, but many different car manufacturing companies and therefore produce a large variety of different products that need to be painted before delivery (e.g. bumpers and other exterior systems). Because of the short manufacturing cycles caused by the commonly used concepts of just in time manufacturing in the automotive industry (Sugimori et al. 1977), it is of high importance to create production schedules that are able to fulfill all due dates requested by car manufacturers. Therefore, the main goal of the paint shop scheduling problem we describe in this paper is to determine a technically feasible production sequence that produces all ordered products within the given due dates. Furthermore, two minimization criteria should be considered to reduce waste and save costs: Firstly, the schedule should group orders that request similar colors to minimize required color changes in the production sequence whenever possible. The second minimization criterion is concerned with an efficient utilization of the carrying devices that are used to transport the raw material items through the paint shop. To understand the details behind the second optimization objective, the reader needs to know that all items scheduled for painting have to be placed on custom carrier devices that will move through the paint shop's painting cabins. In each cabin, several painting robots will then apply paint on the raw material pieces. Due to the fact that there are many different carrier types available, each being able to transport certain configurations of demanded materials, it will be necessary to use a variety of different carrier device types during production. Although combinations of different raw material items may be transported by a single carrier, it is never possible to schedule products with different colors on a single carrying device. Figure 1 shows a schematic of two carrier types and three possible material configurations.

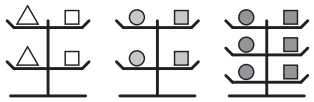

Figure 1: Schematic showing three carriers of two different carrier types. The carriers shown on the left and in the middle have the same type, while the type of the carrier shown on the right side is different. The left carrier uses a material configuration that transports two triangular and two square raw material pieces while the middle carrier transports two circular and two square raw material pieces and the carrier on the right side transports three circular pieces and three squared pieces. The figure exemplifies how the same carrier 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, lightgray or gray).

The paint shops of the automotive supply industry are designed to support an almost fully automated production process. Therefore, any scheduled carrying devices will be automatically moved through the paint shop on a circular conveyor belt system. Carriers can be inserted and removed from and onto 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 will be moved through the cyclic paint shop system where it repeatedly will pass by the painting cabins, the carrier gates, and a material gate, until the schedule will select the carrier for ejection at the output gate. At the material gate unpainted raw materials may be placed on any empty carrying device by paint shop employees. A loaded carrier will then move to the painting cabins, where the scheduled color will be applied on all carried items. Whenever a loaded carrier arrives at the material gate after having completed a full round, another employee will take off the colored material pieces and may place new uncolored raw materials onto the carrier that will then be painted in the following round. Figure 2 shows a schematic of the paint shop's layout and visualizes the movement of carriers through the paint shop.

Because of the circular layout of the paint shop, the painting schedule is organized in rounds. Within each painting round, several carrier units will be painted one after the other in a sequence that is predetermined by the schedule. However, the number of processed carriers per round as well as

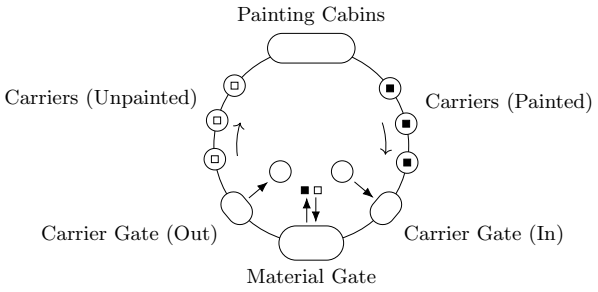

Figure 2: Schematic showing a paint shop layout that is commonly used in the automotive supply industry. During production a number of carrying devices will be inserted onto a circular system of conveyor belts. These will then transport unpainted material pieces to the painting cabins and later return with the colored pieces to the material gate where finished products can be unloaded.

	R1	*R2*	*R3*	...
1	A1	A2	C1	...
2	A1	A2	C2	...
3	A2	C1	C3	...
4	B1	B2	B1	...
5	B2	B3	B2	...

Figure 3: An Example painting schedule for three rounds. Each column represents the scheduled carrier sequences scheduled within a single round. Each cell contains information about the associated carrier type (letter), carrier configuration (number) and scheduled color (background color).

the exact sequence do not necessarily have to be equal for each round. A schedule will therefore plan the painting sequences for multiple rounds and determine the raw material and color configurations for each sequenced carrying device. One can represent a candidate solution to the paint shop scheduling problem as a table, where each column represents the scheduling sequence for a single round. Each table cell will then assign the carrier type, material configuration, and color that should be scheduled in the associated round sequence. Figure 3 pictures an example of a small painting schedule.

Considering all carrier configurations and colors that can be scheduled for production, a tremendous number of different schedules can be created, however many constraints impose restrictions regarding due dates and allowed carrier sequences have to be fulfilled on feasible schedules.

A multi-objective minimization function further includes two optimization criteria. As already mentioned, the first optimization goal is to minimize color changes in the scheduling sequence, while the second optimization goal is concerned with an efficient utilization of carrying devices. In the following we will further explain the second minimization goal.

Since a paint shop schedule will usually not use the same

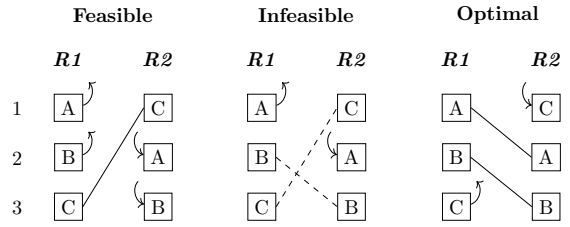

Figure 4: Three possible options to reuse carriers between two consecutive rounds. The feasible option on the left side of the figure will reuse only a single carrier of type C and requires a total of two carrier insertions and two carrier removals. The infeasible option shown in the middle of the figure suggests to keep carriers of type B and C between two consecutive rounds. However, this is technically not possible as C can not 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 removal for this example.

carrier type sequence in each round, it is often required to remove and insert the carriers from the conveyor belt system between rounds. However, if carriers of the same type are scheduled in two consecutive rounds it may be possible to reuse some of them as long as the sequence of kept carriers is compatible with the scheduled carrier sequence in the succeeding round. Since the insertion and removal of carrier units from the circular track might lead to delays and can in general not be done in parallel, it is desired to keep the number of such operations as low as possible. Note that for any given two consecutive rounds, the minimal amount of required carrier insertions and removals can be calculated by determining the LCS of the two carrier type sequences. One can also think of the carrier type sequences as two strings, where the minimal number of required carrier changes corresponds to the edit distance (Wagner and Fischer 1974) with only insertion and deletion operations. Figure 4 visualizes three alternative ways how carriers between consecutive rounds may be reused.

Formal Problem Description

In this section we provide a formal specification of the paint shop scheduling problem described in this paper.

Input parameters

The following parameters describe instances of the problem:

Set of carrier types: T

Set of colors: C

Set of materials: M

Set of carrier configurations: K
A carrier configuration is always associated to a single carrier type and provides information about the materials that are placed on this carrier.

Number of rounds to schedule: n

Set of all rounds to schedule: $R = \{1, \ldots, n\}$

Maximum number of carrier slots per round: s

Set of carrier slots per round: $S = \{1, \ldots, s\}$

Minimum number of carriers that have to be scheduled in each round: q

Number of available carriers of type t in round r: $a_{r,t}, \forall r \in R, t \in T$
The number of available carriers are input parameters because in practice some carriers will be scheduled for cleaning and maintenance from time to time (independently of the production schedule).

Set of demands: $D \subseteq \{(a, m, r, c) | a \in \mathbb{N}_{>0}, m \in M, r \in \mathbb{N}_{>0}, c \in C\}$
Each demand will ask for 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 lying outside the scheduling horizon.

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

Carrier type of each carrier configuration:

$$v_k \in T, \forall k \in K$$

Number of carriers scheduled in the round previous to the scheduling horizon (history round): p

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: $pc_i \in C, \forall i \in \{1, \ldots, p\}$

Set of forbidden carrier type sequences. All elements in F define forbidden carrier type sequences of length two that may not appear anywhere in the schedule: $F \subset \{(t_1, t_2) | t_1, t_2 \in T, t_1 \neq t_2\}$

Minimum block length for carrier type t: $b_t^{\min}, \forall t \in T$
Whenever a carrier of type t is scheduled, the same carrier type has to be used for the next consecutive carriers until the given minimum block length is reached. (For example let $b_{t_1}^{\min} = 3$ and the previously scheduled carrier type sequence be $\langle t_3, t_3, t_2, t_1 \rangle$, then to satisfy the minimum block length at least the next two carriers in the sequence have to be t_1).

Maximum block length for carrier type t: $b_t^{\max}, \forall t \in T$

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 in the scheduled sequence: $o_{c_1,c_2} \in \mathbb{N}, \forall c_1, c_2 \in C$
For example let $o_{v,w} = 3$ for colors v and 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 $y \neq w$).

Function that assigns color transition costs for all pairs of colors: $f_c : \{C \times C\} \to \mathbb{N}$

Decision variables

We define the following decision variables for the paint shop scheduling problem:

The carrier configuration scheduled in round i and position j[1]:

$$x_{i,j} \in K \cup \{\epsilon\}, \forall i \in \{0, \ldots, r\}, j \in S$$
$$x_{0,j_1} = \lambda \quad (\text{where } v_\lambda = pt_{j_1}), \forall j_1 \in \{1, \ldots, p\}$$
$$x_{0,j_2} = \epsilon, \forall j \in \{p+1, \ldots, s\}$$

If the value ϵ is assigned, the position is empty and no carrier will be scheduled at the position.

The color that is used in round i at position j:

$$c_{i,j} \in C \cup \{\epsilon\}, \forall i \in \{0, \ldots, r\}, j \in S$$
$$c_{0,j_1} = pc_{j_1}, \forall j_1 \in \{1, \ldots, p\}$$
$$c_{0,j_2} = \epsilon, \forall j \in \{p+1, \ldots, s\}$$

If the value ϵ is assigned, the position is empty and will not be painted.

Helper Variables for Hard Constraints

To formulate the problem's hard constraints, we introduce the following helper variables and functions:

The number of carriers that are scheduled in round i: $p_i \in \{0, \ldots, s\}, \forall i \in \{0, \ldots, n\}$
p_0 refers to the number of carriers scheduled in the history round.

The total number of carriers scheduled in the entire schedule, excluding the history round: $pt \in \{0, \ldots, n \cdot s\}$

Sequence coordinate helper function: $f_s(i, j) = p + \sum_{r \in \{2 \ldots i\}} p_{r-1} + j$
This helper function converts the two-indexed scheduling coordinates (round and position within rounds) into a one-indexed scheduling coordinate. For example let exactly 100 carriers be scheduled in round 1, then $f_2(2, 3)$ will be set to the value 103.

The carrier configuration that is scheduled at the one-indexed position coordinate i:

$$seqx_i \in K \cup \{\epsilon\}, \forall i \in \{1, \ldots, p + n \cdot s\}$$

The color that is scheduled at the one-indexed position coordinate i: $seqc_i \in C \cup \{\epsilon\}, \forall i \in \{1, \ldots, p + n \cdot s\}$

Hard Constraints

1. Unplanned carrier positions should always be scheduled last in a round:

$$(x_{i,j} = \epsilon) \Rightarrow (x_{i,j+1} = \epsilon), \forall i \in R, j \in \{1, \ldots, s-1\} \quad (1)$$

[1]The input parameters do not specify any information about the configurations used in the history round. For simplicity we fix the corresponding decision variables for the history round to any configuration λ that is compatible with the used carrier type in the history round.

2. Any scheduled carrier position must also assign a color and any unscheduled position must not assign a color:

$$(x_{i,j} \neq \epsilon) \Leftrightarrow (c_{i,j} \neq \epsilon), \forall i \in R, j \in S \tag{2}$$

3. Force the correct number of scheduled carriers to the associated helper variables:

$$\begin{gathered} p_0 = p \\ p_r = |\{j \in \{1, \ldots, s\} | x_{r,j} \neq \epsilon\}|, \forall r \in R \\ pt = \sum_{r \in R} p_r \end{gathered} \tag{3}$$

4. Bind the values of the decision variables to the associated one indexed sequence helper variables:

$$\begin{gathered} seqx_j = x_{0,j} \wedge seqc_j = pc_j, \forall j \in \{1, \ldots, p\} \\ x_{i,j} \neq \epsilon \Rightarrow \\ (seqx_{(fs(i,j))} = x_{i,j} \wedge seqc_{(fs(i,j))} = c_{i,j}), \\ \forall i \in R, j \in S \\ (k > p + pt) \Leftrightarrow seqx_k = \epsilon, \\ \forall k \in \{p+1, \ldots, p + n \cdot s\} \end{gathered} \tag{4}$$

5. All demands must be satisfied in time (overproduction is allowed):

$$\begin{gathered} \sum_{\{(d_a, d_m, d_r, d_c) \in D | \ d_m = m \wedge d_r <= r \wedge d_c = c\}} d_a \leq \\ \sum_{\{i \in \{1, \ldots, r\}, j \in \{1, \ldots, s\} | c_{i,j} = c\}} u_{(x_{i,j}), m} \\ \forall r \in R, m \in M, c \in C \end{gathered} \tag{5}$$

6. Carrier availabilities must be respected in each round (X here refers to the set of all $x_{i,j}$ variables):

$$|\{x_{i,j} \in X | i = r \wedge v_{(x_{i,j})} = t\}| \leq a_{r,t}, \forall r \in R, t \in T \tag{6}$$

7. The minimum round capacity must be fulfilled in each round:

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

8. Forbidden carrier type sequences must not appear in the schedule:

$$\begin{gathered} v_{(seqx_i)} \neq t_1 \vee v_{(seqx_{i+1})} \neq t_2, \\ \forall (t_1, t_2) \in F, i \in \{p, \ldots, (p + n \cdot s - 1)\} \end{gathered} \tag{8}$$

9. Minimum carrier block length restrictions must be fulfilled:

$$\begin{gathered} (v_{(seqx_i)} \neq t \wedge v_{(seqx_{i+1})} = t) \Rightarrow \bigwedge_{j \in \{2, \ldots, b_t^{\min}\}} (v_{(seqx_{i+j})} = t), \\ \forall t \in T, i \in \{1, \ldots, (p + n \cdot s - b_t^{\min} - 1)\} \end{gathered} \tag{9}$$

$$\begin{gathered} \neg(v_{(seqx_{p+n \cdot s - b_t^{\min}+1})} \neq t \wedge v_{(seqx_{p+n \cdot s - b_t^{\min}+2})} = t) \\ \forall t \in T \end{gathered} \tag{10}$$

10. Maximum carrier block length restrictions must be fulfilled:

$$\begin{gathered} \bigvee_{j \in \{0, \ldots, b_t^{\max}\}} (v_{(seqx_{(i+j)})} \neq t), \\ \forall t \in T, i \in \{1, \ldots, (p + n \cdot s - b_t^{\max})\} \end{gathered} \tag{11}$$

11. No forbidden color sequences should occur in the schedule:

$$\begin{gathered} (seqc_i = c_1) \Rightarrow \bigwedge_{j \in \{1, \ldots, o_{(c_1, c_2)}\}} (seqc_{(i+j)} \neq c_2), \\ \forall c_1, c_2 \in C, i \in \{1, \ldots, (p + n \cdot s - o_{(c_1, c_2)})\} \end{gathered} \tag{12}$$

Helper Variables and Constraints for the Objective Function

To formulate the problem's minimization function, we introduce the following helper variables:

The amount of color change costs occurring in round r of the schedule: $cc_r, \forall r \in R$

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

$$sc_r, \forall r \in \{0, \ldots, n-1\}$$

The number of carriers that will not be changed after round r and reused in round $r+1$: $sk_r, \forall r \in \{0, \ldots, n-1\}$

Edge helper variables:

$$e_{r,k,l} \in \{0,1\}, \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S$$

Edge variables that are set to true whenever a carrier from round r at position k is reused in round $r+1$ at position l.

The following hard constraints are used to assign values to the helper variables:

1. Sum up the color change costs per round in the associated helper variables. The value includes a potential color change cost that occurs between the last position of the previous round to the first position of the target round (We assume here that if the value ϵ is assigned to any parameter of f_c, the function will return 0):

$$\begin{gathered} cc_r = \sum_{j \in \{1, \ldots, s-1\}} f_c(c_{r,j}, c_{r,j+1}) + \\ \sum_{\{j \in \{p_{r-1}\}\}} f_c(c_{r-1,j}, c_{r,1}), \forall r \in R \end{gathered} \tag{13}$$

2. The number of necessary carrier changes between two given rounds are calculated with the helper variables sk_r that determine how many carriers can be kept after each round.

$$sc_r = p_r - sk_r + p_{r+1} - sk_r, \forall r \in \{0, \ldots, n-1\} \tag{14}$$

3. The sk_r variables are assigned by summing up the number of associated edge variables that are set to 1. Note that each edge variable set to 1 will represent a carrier that is kept between two consecutive rounds:

$$sk_r = \sum_{k,l \in S} e_{r,k,l}, \forall r \in \{0, \ldots, r-1\} \tag{15}$$

4. The following constraints enforce that edges between carriers of consecutive rounds (carriers connected by an edge will be reused) are only allowed if the carrier types at both positions are equal and not set to ϵ:

$$\begin{gathered} (e_{r,k,l} = 0) \Leftarrow (x_{r,k} = \epsilon \vee x_{r+1,l} = \epsilon), \\ \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S \end{gathered} \tag{16}$$

$$\begin{gathered} (e_{r,k,l} = 1) \Rightarrow (v_{(x_{r,k})} = v_{(x_{r+1,l})}), \\ \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S \end{gathered} \tag{17}$$

5. The following constraint forbids crossings between selected edges of two consecutive rounds. These crossings have to be forbidden to enforce the correct order of kept carriers.:

$$(e_{r,k,l} = 1) \Rightarrow$$
$$\left(\bigwedge_{\substack{m \in \{1,\ldots,k-1\},\\ n \in \{l,\ldots,s\}}} (e_{r,m,n} = 0) \wedge \bigwedge_{\substack{m \in \{k,\ldots,s\},\\ n \in \{1,\ldots,l-1\}}} (e_{r,m,n} = 0) \right) \quad (18)$$
$$\forall r \in \{0,\ldots,n-1\}, k \in S, l \in S$$

Objective function

The objective function aims to minimize the number of carrier changes (sc) and color change costs (cc). The sums are squared, since it is preferable to distribute the required changes over the scheduling horizon and to avoid peaks of many changes within a single round.

$$minimize \sum_{r \in \{0,\ldots,n-1\}} sc_r^2 + \sum_{r \in R} cc_r^2 \quad (19)$$

A Greedy Algorithm for Paint Shop Scheduling

One of the requirements stated by the automotive company who requested an automatic scheduling system was to provide practical solutions to large scale problems within few minutes of running time, and therefore we chose to at first approach the problem using a greedy algorithm.

Phase 1: Constructing a round layout

Although the method cannot guarantee to always produce feasible solutions, in practice the few remaining violations can quickly be repaired by a human planner.

Phase 1: Constructing a round layout

A challenging property of the paint shop scheduling problem is that demanded materials and associated carrier configurations have to be distributed over the rounds of the scheduling horizon. One strategy to keep the number of required carrier and color changes low in each round, is to minimize the number of scheduled colors per round while trying to reuse as many carrier types as possible between rounds. The first phase of our greedy algorithm follows this idea, while assigning carrier configurations and colors to each round without considering an exact round sequence at first.

Therefore, phase 1 of the greedy heuristic will, given a customer demand, insert a carrier device with the configuration that maximizes the number of pieces for that demand. The color is specified by the demand. Insertion is done greedily, i.e. minimize violations with respect to the current state. Customer demands are processed one after the other: demands with earliest due dates first.

Even without knowing the exact carrier sequence, phase 1 can still consider hard constraints that do not depend on the sequence and can calculate a lower bound of the objective.

Phase 2: Determining the carrier sequence for each round

After the execution of Phase 1, the heuristic has decided which carrier configurations and colors should be scheduled within each round. Furthermore, Phase 1 has already considered hard constraints that are not sequence dependent (due round, carrier availability, and round capacity). Phase 2 will therefore only determine the exact carrier sequence within each round while trying to fulfill sequence dependent hard constraints and aiming for a low number of color and carrier type changes.

The main idea behind the second phase is to determine the carrier sequence one round at a time. It keeps the sequence from the previous round as closely as possible, i.e. carriers not used in the current round are removed. The remaining carriers are inserted greedily. Therefore, the algorithm will start with the first round and determine its sequence based on the scheduling sequence from the history round which is part of the input parameters.

After the sequence has been determined for round 1 the algorithm will continue to sequence round 2 and so on.

A Local Search Based Approach for Paint Shop Scheduling

In this section we introduce a local search based approach to solve the paint shop scheduling problem. We propose three different types of neighborhood moves, and several metaheuristic techniques to escape local optima.

Cost Function

We extend the objective function described in Equation 19 to also include a sum of all hard constraint violations hv that will be multiplied with a constant M that is guaranteed to be larger than the largest possible objective value. The sum of hard constraint violations will be calculated independently for each constraint in a way that captures the distance to a feasible solution (E.g. If the minimum round capacity constraint is violated for any round, the minimal number of missing carriers will be included in the sum of all hard constraint violations). Equation 20 defines the extended objective function.

$$\begin{aligned} minimize & \sum_{r \in \{0,\ldots,n-1\}} sc_r^2 + \sum_{r \in R} cc_r^2 + hv \cdot M \\ M &= s^2 \cdot r + (maxColorCost)^2 \cdot r + 1 \\ maxColorCost &= \max\{f_c(c_1, c_2) | c_1, c_2 \in C\} \end{aligned} \quad (20)$$

Search Neighborhoods

We propose the following three neighborhood moves for local search:

1. *Carrier removal*: Any carrier assignment that is placed in the schedule can be simply removed. Whenever a carrier is removed from a round, all carriers that have been planned after the removed position in the same round will be shifted down by one position.
2. *Carrier insertion*: A new carrier assignment can be inserted in any round that has not reached its full capacity. Carriers that have been previously planned at or after the newly inserted carrier's position will be shifted upwards by one position.

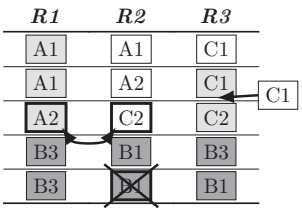

Figure 5: This figure shows a visualization of how the three neighborhood move types (swapping positions, delete positions, and insert positions) can make modifications to a paint shop schedule.

3. *Carrier swap*: Any two carrier assignments in the schedule can be swapped. In this case both the selected carrier configurations and colors are exchanged.

The local search approach we propose will also consider block moves where multiple consecutively scheduled carrier assignments may be inserted, deleted or swapped at once. In our experiments we used a maximum block move size that corresponds to the largest input parameter given with the minimum block length constraint ($\max\{b_t^{\min} | t \in T\}$), since block moves will be especially effective for repairing minimum block length violations. Figure 5 shows a visualization of the three neighborhood move types. In the following, whenever we say search move we refer to a single neighborhood move that can be either a carrier removal, insertion or swap.

Neighborhood Generation

Since generating the complete search neighborhood usually cannot be done within reasonable time for large instances, we propose to incorporate elements of a min-conflicts heuristic into our local search approach to focus on promising parts of the search neighborhood. Our algorithm will therefore track any carrier assignment in a candidate solution that is causing a constraint violation, a carrier-, or a color change. Furthermore, for constraints that require additional carrier configurations to be inserted into the schedule we track which carrier configurations are still missing. Our neighborhood generation routine will therefore consider two types of conflicts:

1. Position Conflicts: All positions in the schedule that are involved in at least one constraint violation will be considered to be in conflict.
2. Insertion Conflicts: Some constraints can be violated because of a number of missing carrier assignments in the schedule (e.g. demand constraint, min capacity). For those constraints we track information about what carrier configuration needs to be inserted to repair any violation. For some insertion conflicts it is irrelevant which color and configuration is inserted to repair the constraint violation (e.g. minimum round capacity constraint). Our algorithm will randomly select a configuration and color for insertion in such a case.

Algorithm 1 further describes our neighborhood generation routine.

Algorithm 1 Generate Neighborhood Moves

```
allMoves ← []
Calculate position and insertion conflicts
ic ← select random insertion conflict
pc ← select random position from position conflicts
sp ← generate random swap position
for i ← 1 to maxBlockMoveSize do
    Add insertion of size i based on ic to allMoves
    Add deletion of size i at pc to allMoves
    Add swap of size i for pc and sp to allMoves
end for
Add randomly generated search move to allMoves
return allMoves
```

After Algorithm 1 has generated a collection of potential search moves, we propose two alternative methods to select the best neighborhood move. The first option will always select the neighborhood move that will lead to the lowest cost value, while the second option will use a tabu list to prevent the repeated selection of recently performed moves. Note however, that the second option will always select a move that will lead to a new unknown best solution even if it is contained in the tabu list.

Neighborhood Move Acceptance

We further propose to use an innovative simulated annealing based acceptance function that will decide whether or not a selected move should be accepted during a search iteration (if not accepted, no move will be performed in the current iteration). In addition to the standard homogeneous simulated annealing temperature cooling scheme (Kirkpatrick, Gelatt, and Vecchi 1983), we incorporate a problem specific factor t' that will adjust the acceptance probability P based on the search progress (see Equation 21, where e and e' are the current- and the neighbor solution cost, and T is the current temperature).

$$P(e, e', T) = \exp(-(e' - e)/(T \cdot t')) \tag{21}$$

We included the factor t' into our algorithm due to the observation that the impact of unit improvements on the objective function (such as reducing the number of hard constraints violations by one or lowering the number of required color or carrier changes by one) will depend on the current objective value. The idea is to set t' to a value that roughly estimates the cost improvement that would occur to the current solution if the number of violations or required color and carrier changes is reduced by one. To calculate t' we do the following in each iteration: As long as the current solution violates any hard constraint, we simply set $t' = M$. However, if the current solution is feasible we instead calculate t' based on the current solution's cost as described in Algorithm 2 (The values $colorCosts$ and $carrierCosts$ store the sum of color- or carrier costs from the objective function.)

The rationale behind algorithm 2 is to normalize the acceptance rate of moves that decrease the number of carrier or color changes during the overall search progress. This is done by calculating the average cost improvements that would occur to the current solution if the total number of color and carrier changes in the schedule is lowered by one. The calculated value depends on the quality of the current solution and significantly changes during the search progress.

Algorithm 2 Calculating t' if the current solution does not violate any hard constraints.

$a \leftarrow \frac{colorCosts}{numberOfRounds}$
$b \leftarrow \frac{carrierCosts}{numberOfRounds}$
$c \leftarrow (\sqrt{a} + maxColorCost)^2 - a$
$d \leftarrow (\sqrt{b} + 1)^2 - b$
$t' \leftarrow \min(c, d)$
return t'

Empirical Evaluation

We generated 24 instances for the paint shop scheduling problem based on actual planning scenarios from the automotive industry[2]. Instances 13–24 have been generated by processing scheduling scenarios as they have recently appeared at a real life production site of our industrial partner.

Those instances describe six different planning horizons of 7, 20, 50, 70, 100 and 200 rounds (we generated two instances for each horizon). Early experiments with the instances showed that exact methods could not provide any solutions to these instances (the solvers ran out of memory on a machine with 48GB RAM), and we therefore decided to manually scale down Instances 13–24 by randomly selecting roughly 5% of the materials, colors, configurations and demands to create the smaller Instances 1–12. We conducted all benchmark experiments using an Intel Xeon E5345 2.33 GHz CPU with 48 GB RAM, and limited the running time to 60 minutes.

In first experiments we evaluated our search neighborhoods with a simple random walk move generation and standard simulated annealing techniques, however this approach could not produce feasible solutions for the larger instances.

After we implemented the adaptive simulated annealing scheme as well as the conflict based neighborhood generation technique we propose in this paper, we were able to produce feasible solutions for all but the four largest instances within the time limit. Although the greedy algorithm was not able to find any feasible solutions on its own, we could utilize greedily constructed initial solutions together with the metaheuristic approach to produce feasible solutions for all of our benchmark instances. In our final experiments we evaluated two variants of the combined greedy and local search approach: One variant that will always select the best move from the generated neighborhood and a second variant that will use a tabu list to prevent the repeated selection of recently performed moves. Based on our initial experiments we set the following parameters: Initial temperature $t_1 = 0.25$, tabu list length $tl = 0.001$ (relative to the instance size), cooling rate $\alpha = 0.95$. In addition to the approaches that are discussed in this paper, we also compare to the exact method from (Winter and Musliu 2019) using the solvers chuffed (Chu et al. 2018) and gurobi (Gurobi Optimization 2018). Table 1 gives an overview of our final experimental results.

Columns 2 and 3 of Table 1 show the results achieved with standard simulated annealing compared to the methods proposed in this paper that make use of an adaptive simulated annealing acceptance without the greedy algorithm (in this case local search will start from an empty schedule). The results show that the local search methods proposed in this paper produced better results for the majority of the smaller instances (1-12) and most of the larger instances (13-20). Although the standard simulated annealing approach can process search iterations much faster and produced better results for two of the small instances, the results show that the proposed local search methods were more robust in our experiments especially when it comes to solving larger instances.

Exact methods could produce optimal results for seven of the smaller instances and could provide good solutions for three additional instances. The methods we have proposed in this paper were able to produce feasible solutions for all instances and could provide the best results for all of the large practical sized instances. Starting from a greedily generated solution did not always have positive effects on the results for instances 1–12, however for the larger instances 13–24 methods incorporating greedily constructed solutions produced the best results. Adding a tabu list mechanism to our metaheuristic approach did not lead to improved results for most of the instances, although this technique could produce the best results for instances 15, 20 and 23.

Conclusion

In this paper we have introduced a novel paint shop scheduling problem that appears in the automotive industry. We additionally have provided a formal specification of the problem and generated 24 different problem instances that are closely based on real-life planning scenarios. Furthermore, we proposed a greedy algorithm as well as a local search based approach that includes novel neighborhood operators and innovative techniques to escape local optima.

Through a series of benchmark experiments we have shown that the methods described in this work can provide promising upper bounds for all instances within a running time of one hour, and are able to produce better results than existing exact methods for all of the large practical sized instance. The experimental results further show that our solution approaches provide promising results even for the largest instances which are based on large scale real-life planning scenarios.

Our solution approaches are currently in the deployment phase at the production site of an automotive supply company. First tests with real-life data at the production site

[2]https://www.dbai.tuwien.ac.at/staff/winter/ps_instances.zip

	SA	LS	LS/G	LS/G/T	EM
I1	NA	1028	844	882	**775***
I2	896	868	932	927	**842***
I3	1011	990	992	994	**961***
I4	NA	1016	975	1050	**918***
I5	618	616	593	599	**530***
I6	913	887	891	895	**842***
I7	1120	1084	1088	1137	**1046**
I8	NA	1871	1834	2553	**1237***
I9	NA	1767	1735	2421	**1006**
I10	1134	1262	1243	1269	**973**
I11	**5236**	6298	5476	6439	NA
I12	6753	**5723**	7916	8274	NA
I13	NA	2097235	**116235**	123830	NA
I14	NA	1985513	**118628**	130552	NA
I15	NA	8159361	180863	**172679**	NA
I16	NA	8621490	**262252**	262897	NA
I17	NA	23320626	**421777**	455321	NA
I18	NA	23947097	**581021**	606917	NA
I19	NA	34294393	**555829**	576225	NA
I20	NA	34713814	930564	**927822**	NA
I21	NA	NA	**917955**	957854	NA
I22	NA	NA	**1128716**	1142530	NA
I23	NA	NA	1889804	**1884125**	NA
I24	NA	NA	**2086450**	NA	NA

Table 1: The final results (total solution cost as defined in Equation 19, NA if no feasible solution could be achieved) for all instances produced with the standard simulated annealing (SA), the metaheuristic methods and adaptive simulated annealing acceptance proposed in this paper (LS), the combined approach using the proposed local search methods and the greedy algorithm (LS/G), the combined approach that also uses a tabu list (LS/G/T), and the best results produced by exact methods (EM). The best result within each line is formatted in bold face. Results marked with a * denote proven optimal solutions.

have shown that our methods can produce results that significantly improve the solutions that are generated by human life planners. Our implementation furthermore allows an interaction between the human planner and the local search algorithm, where the user can manually adjust results produced by the algorithm and trigger a restart of local search if desired.

In future work, we plan to consider the hybridization of exact techniques with our local search based approach 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

Chu, G.; Stuckey, P. J.; Schutt, A.; Ehlers, T.; Gange, G.; and Francis, K. 2018. Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed.

Epping, T.; Hochstättler, W.; and Oertel, P. 2004. Complexity results on a paint shop problem. *Discrete Applied Mathematics* 136(2):217 – 226.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & OR* 13(5):533–549.

Gurobi Optimization, L. 2018. Gurobi optimizer reference manual.

Hirschberg, D. S. 1977. Algorithms for the longest common subsequence problem. *J. ACM* 24(4):664–675.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by simulated annealing. *science* 220(4598):671–680.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1990. Solving large-scale constraint-satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, USA, July 29 - August 3, 1990, 2 Volumes.*, 17–24.

Prandtstetter, M., and Raidl, G. R. 2008. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research* 191(3):1004 – 1022.

Solnon, C.; Cung, V. D.; Nguyen, A.; and Artigues, C. 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):912 – 927.

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

Sugimori, Y.; Kusunoki, K.; Cho, F.; and Uchikawa, S. 1977. Toyota production system and kanban system materialization of just-in-time and respect-for-human system. *International Journal of Production Research* 15(6):553–564.

Wagner, R. A., and Fischer, M. J. 1974. The string-to-string correction problem. *J. ACM* 21(1):168–173.

Winter, F., and Musliu, N. 2019. Constraint based modeling for scheduling paint shops in the automotive supply industry. Technical report, TU Wien, CD-TR, 2019/1.