Journal of Scheduling (2022) 25:453–475
https://doi.org/10.1007/s10951-021-00713-7

A large neighborhood search approach for the paint shop scheduling problem

Felix Winter[1] · **Nysret Musliu**[1]

Accepted: 9 October 2021 / Published online: 2 December 2021

Abstract
Minimizing the setup costs caused by color changes is one of the main concerns for paint shop scheduling in the automotive industry. Yet, finding an optimized color sequence is a very challenging task, as a large number of exterior systems for car manufacturing need to be painted in a variety of different colors. Therefore, there is a strong need for efficient automated scheduling solutions in this area. Previously, exact and metaheuristic approaches for creating efficient paint shop schedules in the automotive supply industry have been proposed and evaluated on a publicly available set of real-life benchmark instances. However, optimal solutions are still unknown for many of the benchmark instances, and there is still a potential of reducing color change costs for large instances. In this paper, we propose a novel large neighborhood search approach for the paint shop scheduling problem. We introduce innovative exact and heuristic solution methods that are utilized within the large neighborhood search and show that our approach leads to improved results for large real-life problem instances compared to existing techniques. Furthermore, we provide previously unknown upper bounds for 14 benchmark instances using the proposed method.

Keywords Paint shop scheduling · Large neighborhood search · Local search

1 Introduction

Modern-day paint shops of the automotive supply industry need to paint a large number of exterior systems every day. As manual planning approaches often cannot cope with the growing demands of car manufacturers that are raised by the recent trend toward full automation, automated scheduling techniques are becoming more and more important.

When finding optimized paint shop schedules, one of the main goals is to minimize the color changes that appear in the production sequence. Since the painting equipment has to be cleaned after each color change, any change leads to a loss of valuable resources and can further cause problematic delays. Minimizing the required color changes is not an easy task on its own, as very large numbers of different colors and products have to be considered in practical scheduling scenarios. However, several constraints that restrict technically feasible color sequences make it even more challenging to automatically create efficient schedules in practice.

In the literature, several publications have investigated the minimization of color changes in automotive scheduling problems (e.g. Spieckermann et al. (2004); Solnon et al. (2008); Prandtstetter and Raidl (2008)). Nonetheless, most of the previous publications consider problems that appear in car manufacturing, which although similar, include very different constraints and objectives than the ones that occur in paint shops of the automotive supply industry.

Recently, we have introduced a paint shop scheduling problem (PSSP) from the automotive supply industry together with a set of 24 real-life problem instances in Winter et al. (2019). Furthermore, we have investigated exact approaches based on constraint programming for the PSSP and provided a NP-hardness proof Winter and Musliu (2021); Winter et al. (2020). Experimental results with the practical benchmark instances revealed that exact approaches could prove several optimal results for small- to medium-sized instances; however, optimal solutions for many large real-life instances are still unknown. The empirical evaluation further

✉ Felix Winter
winter@dbai.tuwien.ac.at

Nysret Musliu
musliu@dbai.tuwien.ac.at

[1] Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, DBAI, Institute of Logic and Computation, TU Wien, Vienna, Austria

showed that a simulated annealing approach together with a construction heuristic from Winter et al. (2019) can successfully produce feasible solutions for all real-life instances within one hour. However, the number of required color changes in the produced solutions was still very high for most of the instances, resulting in schedules that have a large potential for further cost improvements in practice.

In this work, we propose for the first time a large neighborhood search approach to efficiently solve large problem instances of the PSSP. To efficiently generate large neighborhood moves, we analyze a new sub-problem of the PSSP which we call the Paint Shop Color Change Problem (PSCCP). Furthermore, we investigate how optimized solutions to the PSCCP can be used to efficiently improve candidate solutions to the PSSP and propose a constraint programming model as well as an innovative heuristic approach to efficiently find optimized solutions to the PSCCP. The introduced solution techniques are especially effective in reducing the color cost objective of the PSSP instances, as the complex constraints regarding materials and production devices that appear in the original PSSP do not have to be considered while solving corresponding PSCCP instances.

Additionally, we propose a novel construction heuristic for the PSSP which also utilizes solution methods to the PSCCP to produce high-quality initial solutions that can serve as a starting point for local search and hence improve solutions for large-scale real-life instances.

The following list summarizes the main contributions and newly developed algorithms that we propose in this paper:

- To efficiently solve large instances of the PSSP, we identify the PSCCP as an important sub-problem of the PSSP that we utilize to minimize the required color changes in paint shop schedules.
- We provide a constraint programming model as well as an innovative heuristic approach that can be used to efficiently solve the PSCCP
- Using the solution methods for the PSCCP, we propose a novel large neighborhood search operator that incorporates the introduced solution methods to tackle real-life instances of the PSSP.
- We propose an innovative construction heuristic for the PSSP which is able to speed up the solution process for very large-scale instances.
- We systematically evaluate our methods using benchmark instances from the literature and show that the proposed methods can improve state-of-the-art results for many real-life planning scenarios.
- We provide 14 previously unknown upper bounds for real-life benchmark instances of the PSSP.

The remainder of this paper is organized as follows: In Sect. 2, we provide an overview of related literature for the PSSP and other problems that consider color change minimization in paint shops of the automotive industry. Afterward, in Sect. 3 we provide a detailed description of the PSCCP and further introduce heuristic and exact solution methods in Sect. 4. In Sect. 5, we describe how instances of the PSCCP can be generated from candidate solutions to the PSSP and further show how solutions to the PSCCP can be used to minimize costs for instances of the PSSP within the framework of large neighborhood search. In Sect. 6, we propose a novel construction heuristic for the PSSP to quickly generate initial solutions for local search. We provide an overview of conducted computational results in Sect. 7, before we finally give concluding remarks at the end of the paper.

2 Literature review

In this paper, we propose an innovative large neighborhood search operator for the PSSP which appears in paint shops of the automotive supply industry and has been recently introduced in Winter et al. (2019). In addition to a set of 24 real-life-based benchmark instances, a local search method based on simulated annealing and a construction heuristic have further been proposed in Winter et al. (2019). The simulated annealing-based metaheuristic method uses swap-, insertion- and deletion neighborhood operators that are used to iteratively improve candidate solutions. Experiments showed that this approach was able to produce feasible solutions for all of the 24 benchmark instances within reasonable run-time.

In Winter and Musliu (2021), the decision variant of the PSSP has been proven to be NP-complete and an exact approach based on constraint programming (CP) has been proposed. Using state-of-the-art CP solvers, the exact approach could be successfully used to find optimal solutions for 7 of the smaller-sized benchmark instances of the PSSP. The CP approach was later improved in Winter et al. (2020) with the introduction of a string edit distance global constraint, which could be used to efficiently model parts of the objective function of the PSSP leading to much faster optimality proofs for the smaller instances. However, experiments further showed that the exact approach is not able to produce solutions for instances 11–24, and therefore, the simulated annealing-based approach currently represents the state-of-the-art solution method for larger instances. Due to its unique objective function and challenging constraints to the best of our knowledge, no other solution methods for the PSSP that are able to provide high-quality solutions to large real-life instances have been proposed in the recent literature.

In this paper, we introduce an innovative large neighborhood search operator that can be utilized to minimize color changes in the solution schedule. The operator essentially

solves a sub-problem of the PSSP which we call the PSCCP. The main goal of this problem is to assign colors to the production sequence in the paint shop in such a way that the number of required color changes is minimized and due dates are fulfilled.

A color change sequencing problem that is similar to the PSCCP has been studied under the name of the paint shop problem (PSP) in the literature Epping et al. (2004). One can view the PSCCP which we investigate in this paper later on in Sect. 3 as an extension of the PSP that additionally includes due date constraints. However, as the PSP does not consider due date violations, solution methods for the PSP can in general not be used to find feasible solutions for instances of the PSCCP.

The PSP essentially asks to find an optimal assignment of a given set of colored letters to a predetermined word in such a way that color changes are minimized. In Epping et al. (2004), the authors provide a dynamic programming algorithm that can solve the PSP in polynomial time if the number of letters and colors is bounded and show that the decision variant of the problem is NP-complete otherwise. A local search approach using a swap neighborhood and an exact method based on linear programming for the PSP has been proposed in Meunier and Neveu (2012). The authors randomly generate 15 benchmark instances to experimentally evaluate their methods. They conclude that the local search approach overall produced better results than the linear programming approach in their experiments.

In Spieckermann et al. (2004), a sequential ordering problem that appears in car manufacturing paint shops is investigated. This variant aims to minimize the color changes in the production sequence. However, this ordering problem deals with efficient utilization of selectivity banks in which multiple cars are grouped together in banks and therefore imposes substantially different constraints than the problems studied in this paper.

Another problem that originates from the automotive industry and includes the minimization of color changes in its objective function is called the car sequencing problem Solnon et al. (2008). This problem is, however, very different from the problems studied in this paper due to its consideration of capacity and color batch size constraints. An extensive overview of solution methods for the car sequencing problem which includes exact- as well as heuristic methods is given in Solnon et al. (2008).

As the minimization objectives of the PSSCP and the PSSP aim to reduce the setup costs during production by minimizing color changes in the schedule, both problems can be compared to machine scheduling problems with sequence-dependent setup costs. Machine scheduling problems that consider the minimization of setup costs between consecutively scheduled jobs have been thoroughly studied in the past. Comprehensive surveys on the topic can be found in Allahverdi et al. (1999, 2008); Allahverdi (2015), and real-life applications in the automotive industry continue to be the topic of intensive study. For example, recently a parallel machine scheduling problem considering setup times that originates from the automotive industry was tackled using iterative job splitting heuristics in Lee et al. (2021). Furthermore, another recent scheduling application from automobile component manufacturing that considers setup costs was investigated in Van and Hop (2021), where the authors propose a genetic algorithm as a solution method.

Note that both mentioned machine scheduling applications are given as an input a predetermined set of jobs that need to be scheduled on multiple parallel machines, which is a common property of machine scheduling problems with setup times. However, an instance of the PSSP and the PSCCP does not provide jobs, but rather defines a set of demands that need to be fulfilled. Therefore, finding an efficient production sequence, in this case, includes an additional complex decision-making process regarding the selection of carrier devices and associated color assignments. This causes the problems investigated in this paper to be substantially different from traditional machine scheduling problems with setup costs.

3 The paint shop color change problem

In this section, we describe the PSCCP, which appears as a sub-problem within the PSSP. Later in Sect. 5, we will give a full specification of the PSSP and describe how solution methods to the PSCCP can be utilized within a large-neighborhood search approach for the PSSP.

In the automotive supply industry, a very large number of synthetic materials that are demanded by car manufacturers need to be painted each day in highly automated paint shops. Therefore, sets of multiple raw material pieces are put on custom-made carrier devices which are moving on a cyclic conveyor belt system through the paint shops of the automotive supply industry. The loaded carriers then automatically move to the painting cabins where painting robots apply paint to the carried items.

The main aim of the PSCCP is to find an optimized coloring of a carrier production sequence. Thus, a predetermined sequence of different carrier device types that are used to transport materials in the paint shop is given as input to the problem. In the paint shop, the materials that are transported on a single carrier have to be painted using one unique color, and therefore, the aim of the PSCCP is to assign a single color to each individual carrier in such a way that the number of color changes in the production sequence is minimized.

A feasible coloring sequence has to ensure that all color demands are fulfilled, where color demands are given as part

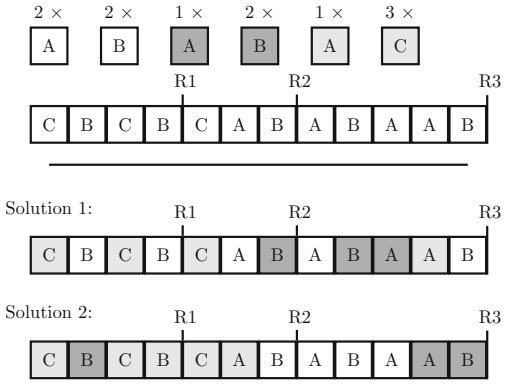

Fig. 1 Illustration of a simple instance for the PSCCP together with two example solutions

of the input in terms of carrier type quantities. To model a notion of time, the given carrier sequence which is part of the input is further grouped into several scheduling periods, which are referred to as rounds since in industrial paint shops of the automotive supply industry carriers are usually moving on a cyclic conveyor belt system. In practice, processing of a single round is done within a fixed amount of time depending on the size of the paint shop, even though each round may schedule a different number of carriers. All color demands which are given as part of a problem instance set a due date (which is specified in terms of rounds) that needs to be fulfilled.

As an example, consider a simple instance of the PSCCP which is illustrated at the top of Fig. 1.

The carrier sequence that is presented in Fig. 1 contains three consecutive rounds called R1, R2, and R3, where the end of each round is visualized by tick marks. The letter in each of the cells denotes the carrier type in the sequence so that within the first round R1 a sequence of four carriers is scheduled: C,B,C,B.

In addition to the predetermined carrier sequence for each scheduling round, an instance of the PSCCP also defines color demands which are illustrated at the very top of Fig. 1. We can see that for this example two carriers of type A, as well as two carriers of type B, need to be painted using a white color, whereas one carrier of type A and two carriers of type B need to be painted in a dark gray color and so on. Note that each color demand usually also has a due date in terms of a scheduling round. For reasons of simplicity in the illustrated example, all color demands are due until the end of the round R3 and therefore in this case a solution is feasible as long as all color demands are scheduled in the sequence.

The bottom of Fig. 1 further illustrates two feasible solutions to the example instance. Solution 1 has been assigned using a naive approach where each carrier has been colored greedily from left to right and causes 10 color changes in total. Solution 2 on the other hand shows an improved solution that causes only four color changes. Note that the schedule actually contains a total of 12 carriers, whereas only 11 carriers are required to be painted. However, this is still a valid instance of the problem, as in such a case one of the carriers may be painted arbitrarily to minimize the overall number of color changes. A scenario like in this example where the number of carriers appearing in the production sequence is larger than the total number of demanded carriers can also occur in practice as technical requirements of the paint shop conveyor belt systems might not allow ejecting all unnecessary carrier devices at once between rounds.

In the following, we provide a formal definition of PSCCP, where we make use of the Iverson bracket notation[1] for reasons of simplicity.

3.1 Input parameters

An instance of the PSCCP specifies several parameters including a predetermined production sequence, information about carrier types, colors, and demands. In the following, we describe all parameters in detail:

A set of carrier types T is given as part of the input, which includes all the different types of carrier devices that appear in the production sequence.

The set of colors C specifies all colors that can be assigned to the carriers in the schedule.

The demands for an instance are given as a set D that consists of 4-tuples, where each tuple defines the demand quantity $a_d, d \in D$, the carrier type of the demand $u_d, d \in D$, the demanded color $v_d, d \in D$ and the demand's due date $w_d, d \in D$. For example, consider a demand d_1 where $a_{d_1} = 10$, $u_{d_1} = t_1$, $v_{d_1} = c_1$, and $w_d = 5$. Then, this demand d_1 would require 10 carrier devices of type t_1 in the schedule to be painted using color c_1 until at latest round 5 (note that the due date is given in terms of paint shop rounds).

A history color h is given as part of the input which denotes the last color used in the previous production schedule (before the production sequence of the instance). The previous schedule is not part of the current problem; however, the color assigned to the first carrier of the solution sequence might cause a color change regarding the history color.

The number of scheduling rounds n, $(R = \{1, \ldots, n\})$ denotes the number of rounds that are processed in the given production sequence. For each of these rounds, the number of scheduled carriers is given by a parameter $s_r, r \in R$. Finally, the detailed predetermined production sequence for the instance is determined by a list consisting of the scheduled carrier type sequence for each round $(l_{i,j}, \forall i \in R, j \in \{1, \ldots, s_i\})$.

The full list of formal input parameters is summarized in Table 1.

[1] $[P] = 1$, if $P = true$ and $[P] = 0$ if $P = false$.

Table 1 The input parameters of the PSCCP

Set of carrier types:	T
Set of colors:	C
Set of demands:	D
Quantity of demand:	$a_d \in \mathbb{N}_{>0}, \forall d \in D$
Carrier type of demand:	$u_d \in T, \forall d \in D$
Color of demand:	$v_d \in C, \forall d \in D$
Due date of demand:	$w_d \in \mathbb{N}_{>0}, \forall d \in D$
History Color:	$h \in C$
Number of scheduling rounds:	n $(R = \{1, \ldots, n\})$
Number of carriers per round:	$s_r \in \mathbb{N}_{>0}, \forall r \in R$
List of scheduled carriers:	$l_{i,j} \in T,$ $\forall i \in R, j \in \{1, \ldots, s_i\}$

3.2 Decision variables

The set of decision variables for the PSCCP decide which color should be used for each scheduled carrier position in the production sequence:
Scheduled color in round i and position j

$$x_{i,j} \in C, \forall i \in R, j \in \{1, \ldots, s_i\} \tag{1}$$

3.3 Hard constraints

In feasible solutions to the PSCCP, all color demands need to be fulfilled in time:

$$\sum_{d \in D} a_d \leq \sum_{i \in \{1,\ldots,w_d\}} \sum_{j \in \{1,\ldots,s_i\}} [x_{i,j} = v_d \wedge l_{i,j} = u_d] \quad \forall d \in D \tag{2}$$

3.4 Objective function

The objective function builds a sum of all color changes in the production sequence including round overlapping changes, and a possible change from the last color used in the history schedule (history color):

$$\textit{minimize } [h \neq x_{1,1}] + \sum_{i \in R} \sum_{j \in \{1,\ldots,s_i - 1\}} [x_{i,j} \neq x_{i,j+1}], + \sum_{i \in \{1,\ldots,r-1\}} [x_{i,s_i} \neq x_{i+1,1}] \tag{3}$$

3.5 Complexity analysis

If we consider instances of the PSCCP with only a single round in the sequence (i.e., the due date constraint is not violated as long as all demands are fulfilled at any time in the sequence) and further ignore the history color of the problem, the PSCCP is equivalent to another problem from the literature Epping et al. (2004) called the paint shop problem (PSP). Thus, the PSCCP is a generalization of the PSP. As the decision variant of the PSP has been shown to be NP-complete in Epping et al. (2004) as long as the number of colors and different types in the sequence are not bounded, we can argue that the decision variant of the PSCCP also is NP-hard under the same assumptions, as we can solve the PSP with the PSCCP by simply creating an instance for the PSCCP with a single round in the sequence and no due date constraints.

4 Solution methods

In this section, we propose two solution approaches to the PSCCP: an innovative heuristic approach and an exact approach based on constraint modeling.

4.1 A heuristic solution approach for the PSCCP

The main idea behind this heuristic approach is to greedily determine the coloring of a single carrier in the given sequence one step at a time and thereby constructing a solution. To find out which carrier should be colored next during each iteration of the algorithm, the heuristic essentially evaluates all possible single color assignments on the current partially colored carrier sequence.

Figure 2 illustrates the main execution steps of the construction heuristic we propose for the PSCCP.

At first, the algorithm starts with a fully uncolored production sequence and creates an ordered list of all color demands ordered by their due dates. As long as this list is not empty, the algorithm then iteratively processes a series of execution steps. These steps essentially go over all demands still contained in the list and calculate how much it would cost to color a single carrier in the production sequence to fulfill the associated demand. The single carrier color assignment that causes the lowest cost increase with the current state of the production sequence is selected and applied to the partial solution. Afterward, the quantity of the demand affected by this color assignment is updated and in case the demand quantity is lowered to zero, the demand is completely removed from the demand list.

Algorithm 1 provides pseudo-code to describe the proposed construction heuristic algorithm in further detail.

The function PSCCPHeuristic is the main entry point of the heuristic and takes in addition to the instance parameters a single positive integer parameter which is called $demandLookAhead$. In a first step, the list of all demands is sorted by their due dates. If there are multiple demands with the same due date, rare colors (which are determined by the total requested quantity for a color over all demands)

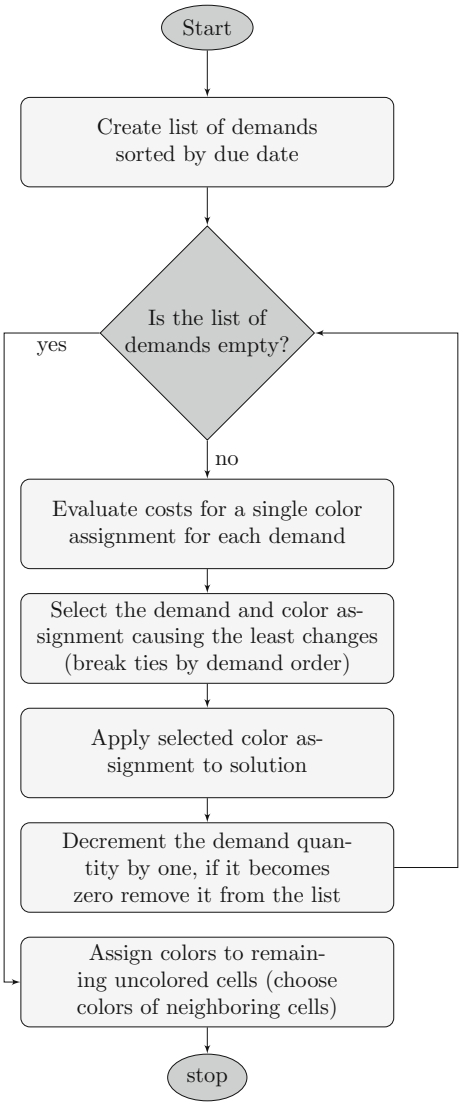

Fig. 2 Main execution steps of the heuristic solution method for the PSCCP

Algorithm 1: PSCCP Heuristic

```
fn GetBestColorPosition (demand d)
    bestCost = −1
    bestPos = null
    seq = reversed carrier sequence starting from w_d
    for position in seq do
        Color position with v_d
        cost = Evaluate costs for current sequence
        Uncolor position
        if bestCost == −1 or cost < bestCost then
            bestCost = cost
            bestPos = position
    return ⟨bestCost, bestPos⟩
fn PSCCPHeuristic (demandLookAhead)
    unfulfilledDemands = sort demands by due round
    while |unfulfilledDemands| > 0 do
        counter = 0
        bestDemand = null
        bestCost = ∞
        bestPosition = null
        for d in unfulfilledDemands do
            if counter == demandLookAhead then
                break
            ⟨cost, position⟩ = GetBestColorPosition(d)
            if bestCost > cost then
                bestCost = cost
                bestPosition = position
                bestDemand = d
            counter = counter + 1
        if bestDemand == null then
            break
        Apply color to bestPosition
        Update unfulfilledDemands
    for position in Sequence do
        Apply color of previously colored position
```

are selected first. The idea here is that rare colors are potentially harder to group in the schedule and therefore should be scheduled earlier on.

After all demands have been ordered, the algorithm then enters its main loop that continuously applies single color assignments to the carrier sequence until all demands are fulfilled. Within the loop, the heuristic goes over the next $demandLookAhead$ demands that are not already fulfilled by the current partially colored schedule. For each of these demands, the costs caused for any possible single color assignment that can potentially fulfill the demand are calculated. Eventually, in each iteration the algorithm performs the single color assignment that causes the lowest costs (Ties are broken by selecting the best cost assignment that was encountered first).

For some instances, it can be the case that some carriers remain uncolored even after all demands have been fulfilled. This situation can occur in some real-life instances, for example when additional carriers are required in the schedule to fulfill minimum carrier capacity requirements. In such a case, the heuristic simply goes over the sequence and tries to color those remaining uncolored carriers based on what colors have been assigned to neighboring carriers to keep the number of color changes as low as possible.

Although the heuristic can find candidate solutions quickly even for large instances (for realistic instances with 100 or more demands the $demandLookAhead$ parameter can be lowered if necessary), in general it does not guarantee to find a feasible solution as due round constraint violations might occur in the resulting schedule.

4.2 An exact approach for the PSCCP

To approach the PSCCP with state-of-the-art exact constraint programming (CP) and mixed-integer programming (MIP) solvers, we propose to model the problem with the use of the high-level constraint modeling language MiniZinc Nethercote et al. (2007). This allows us to implement the problem definition from Sect. 3 in a declarative way and utilize the model with state-of-the-art CP and MIP solvers as an exact solution approach to the PSCCP.

In our constraint model, we define all the input parameters from Table 1 using integer value ids in the ranges from 1 to $|T|$ for the carrier types, from 1 to $|C|$ for the colors, and from 1 to $|D|$ for the demands. We use an additional input $s = \max\{s_r | r \in R\}$, that is set to the maximum round length. The value s is used to define the input carrier sequence as a two-dimensional integer array of dimensions $|R| \times s$. Each position to the array is either set to the scheduled carrier id, or to 0 if the position is unused.

We model the decision variables from Eq. (1) using a two-dimensional integer array and set the variable domain to $\{0, \ldots, c\}$, where 0 will be used to mark unused positions.

The due date constraint from Eq. (2) is modeled with the use of a counting predicate that counts all occurrences of the associated color and carrier type combination for each relevant due date in the schedule. The resulting value is then constrained to be greater than or equal to the required quantity until the due date.

Another constraint sets all unused positions in the decision variable array to 0.

Finally, the solution objective uses counting predicates to model the conditional sums from Eq. (3).

Listing 1 displays the detailed MiniZinc model for the PSCCP. For details on the syntax of the MiniZinc language, please refer to a recent version of the MiniZinc Handbook[2].

```
% INPUT
% carrier types
int: t;
set of int: T = 1..t;

% colors
int: c;
set of int: C = 1..c;

% history color
int: h;

% rounds to schedule
int: r;
set of int: R = 1..r;

% carrier sequence
int: s;
set of int: S = 1..s;
array[R,S] of 0..t: carriers;
array[R] of S: s_r;

% carrier demands
int: num_demands;
set of int: D = 1..num_demands;
array[D] of int: d_t;
array[D] of int: d_c;
array[D] of int: d_qty;
array[D] of int: d_r;

% VARIABLES
array[R, S] of var 0..c: x;

% CONSTRAINTS
% All demands must be satisfied
  in time
constraint forall(d in D where
  d_r[d] <= r) (
   d_qty[d]
   <=
   count(i in 1..d_r[d], j in
     1..s_r[i])(x[i,j] = d_c[d]
     /\ carriers[i,j] = d_t[d])
);

% set unused positions to zero
constraint forall(i in R, j in
  s_r[i]+1..s) (
   x[i,j] = 0
);

% OBJECTIVE
solve minimize bool2int(h !=
  x[1,1]) +
   count(i in 1..r, j in
     1..s_r[i]-1)(x[i,j] !=
     x[i,j+1]) +
   count(i in 1..r-1)(x[i,s_r[i]]
     != x[i+1,1]);
```

Listing 1 MiniZinc model code for the PSCCP

[2] https://www.minizinc.org/.

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

Fig. 3 A simple paint shop schedule, which illustrates a candidate solution to the PSSP

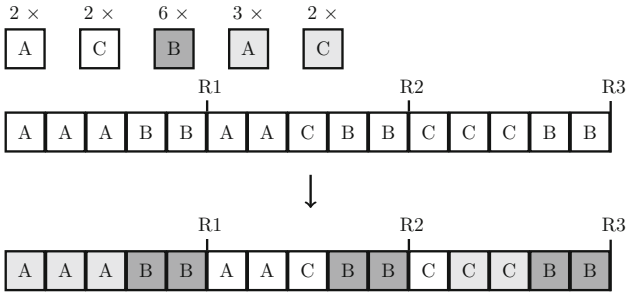

Fig. 4 Visualization of a simple PSCCP instance that would be associated with the candidate schedule for the PSSP as it is shown in Fig. 3

5 A large neighborhood search approach for the paint shop scheduling problem

In this section, we propose an innovative large neighborhood search operator for the PSSP that utilizes the solution methods for the PSCCP that have been introduced in the previous section. First, we provide a short recap of the problem description as well as the main objective goals of the PSSP in Sect. 5.1 and provide the full formal specification in Sect. 5.2. Afterward, we describe how solutions to the PSCCP can be utilized to improve candidate schedules for the PSSP in Sect. 5.3. Later, in Sects. 5.4 and 5.5, we introduce the large neighborhood search operator and describe how it can be used to improve local search for the PSSP.

5.1 The paint shop scheduling problem

The PSSP is a practical problem that appears in paint shops of the automotive supply industry, which we have previously introduced together with a metaheuristic solution approach in Winter et al. (2019). The problem appears in real-life paint shops of the automotive supply industry, where a large variety of different products need to be painted. Products that should be painted in similar colors are grouped into configurations, where each configuration is then placed on a carrier device. Therefore, the main goal of the problem is to determine efficient configurations as well as an optimal carrier production sequence.

Figure 3 visualizes a candidate solution schedule to the PSSP.

Similar to the examples for the PSCCP shown in previous sections, the carrier sequence shown in Fig. 3 is structured into scheduling rounds $R1$–$R3$, which are referred to as rounds in the PSSP. Figure 3 presents the carrier sequence of each round as a column of the solution table. In contrast to the PSCCP, for the PSSP the carrier sequence is not predetermined. Thus, solution methods need to produce an optimized carrier sequence and assign configurations to the carriers. In Fig. 3, carrier configurations are illustrated as numbers next to the letters, so that for example the first carrier in round 1 uses configuration number 1 of a type A carrier, and the third carrier in round 2 uses configuration number 2 of a type C carrier. We can view the PSCCP as a sub-problem that appears within the PSSP, as once a carrier sequence has been determined, we can solve an associated instance of the PSCCP to find an optimized coloring for the associated carrier sequence. Figure 4 visualizes a solution to the PSCCP instance that corresponds to the PSSP example schedule shown in Fig. 3. We can see in this figure that the carrier type sequence of the PSCCP example instance is equal to the sequence used in the associated PSSP example. However, the configuration numbers are omitted for the PSCCP instance as carrier configurations do not need to be considered for the PSCCP.

5.2 Formal specification of the PSSP

In this section, we further provide the full problem specification of the PSSP.

Note that some of the instance parameters of the PSSP (like e.g. colors, carrier types, rounds, demands) are identical or similar to the input parameters of the PSCCP and therefore use the same identifiers. These parameters indeed model exactly the same objects from the practical real-life application in paint shops of the automotive supply industry. However, an instance of the PSSP is still substantially different from an instance of the PSCCP and cannot be directly used as an input to solution methods for the PSCCP. Thus, in the following the input parameters are defined from scratch without reusing the definitions from Sect. 3. Later on, in Sect. 5.3 we describe in detail how instances of the PSSP can be translated to associated instances of the PSCCP.

For further information on background and solution methods to the PSSP, please refer to Winter et al. (2019); Winter and Musliu (2021).

5.2.1 Input parameters

The following parameters describe instances of the PSSP:
Set of carrier types: T

Set of colors: C
Set of materials: M
Set of carrier configurations: K
A carrier configuration is always associated with 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:

$$f_c : \{C \times C\} \to \mathbb{N}$$

5.2.2 Decision variables

The following decision variables are defined for the paint shop scheduling problem:
The carrier configuration scheduled in round i ($i = 0$ refers to the history round) and position j:[3]

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

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:

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

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

5.2.3 Helper variables for hard constraints

To formulate the problem's hard constraints, the following helper variables and functions are defined:
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:

$$pt \in \{0, \ldots, n \cdot s\}$$

[3] 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.

Sequence coordinate helper function:

$$f_s(i, j) = p + \sum_{r \in \{2...i\}} p_{r-1} + j$$

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:

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

5.2.4 Hard constraints

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

$$(x_{i,j} = \epsilon) \Rightarrow (x_{i,j+1} = \epsilon), \quad \forall i \in R, j \in \{1, \ldots, s-1\} \tag{4}$$

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

$$(x_{i,j} \neq \epsilon) \Leftrightarrow (c_{i,j} \neq \epsilon), \forall i \in R, j \in S \tag{5}$$

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

$$\begin{aligned} &p_0 = p \\ &p_r = |\{j \in \{1, \ldots, s\} | x_{r,j} \neq \epsilon\}|, \forall r \in R \\ &pt = \sum_{r \in R} p_r \end{aligned} \tag{6}$$

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

$$\begin{aligned} &seqx_j = x_{0,j} \wedge seqc_j = pc_j, \forall j \in \{1, \ldots, p\} \\ &x_{i,j} \neq \epsilon \Rightarrow \\ &(seqx_{(f_s(i,j))} = x_{i,j} \wedge seqc_{(f_s(i,j))} = c_{i,j}), \\ &\forall i \in R, j \in S \\ &(k > p + pt) \Leftrightarrow seqx_k = \epsilon, \\ &\forall k \in \{p+1, \ldots, p + n \cdot s\} \end{aligned} \tag{7}$$

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

$$\sum_{\{(d_a,d_m,d_r,d_c) \in D | \ d_m = m \wedge d_r <= r \wedge d_c = c\}} d_a \leq \sum_{\{i \in \{1,\ldots,r\}, j \in \{1,\ldots,s\} | c_{i,j} = c\}} u_{(x_{i,j}),m} \quad \forall r \in R, m \in M, c \in C \tag{8}$$

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

$$|\{x_{i,j} \in X | i = r \wedge v_{(x_{i,j})} = t\}| \leq a_{r,t}, \forall r \in R, t \in T \tag{9}$$

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

$$p_r >= q, \forall r \in R \tag{10}$$

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

$$v_{(seqx_i)} \neq t_1 \vee v_{(seqx_{i+1})} \neq t_2, \quad \forall (t_1, t_2) \in F, i \in \{p, \ldots, (p + n \cdot s - 1)\} \tag{11}$$

9. Minimum carrier block length restrictions must be fulfilled:

$$\begin{aligned} &(v_{(seqx_i)} \neq t \wedge v_{(seqx_{i+1})} = t) \\ &\Rightarrow \bigwedge_{j \in \{2,\ldots,b_t^{\min}\}} (v_{(seqx_{i+j})} = t), \\ &\forall t \in T, i \in \{1, \ldots, (p + n \cdot s - b_t^{\min} - 1)\} \end{aligned} \tag{12}$$

$$\neg(v_{(seqx_{p+n \cdot s - b_t^{\min}+1})} \neq t \wedge v_{(seqx_{p+n \cdot s - b_t^{\min}+2})} = t) \quad \forall t \in T \tag{13}$$

10. Maximum carrier block length restrictions must be fulfilled:

$$\bigvee_{j \in \{0,\ldots,b_t^{\max}\}} (v_{(seqx_{(i+j)})} \neq t), \quad \forall t \in T, i \in \{1, \ldots, (p + n \cdot s - b_t^{\max})\} \tag{14}$$

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

$$(seqc_i = c_1) \Rightarrow \bigwedge_{j \in \{1,\ldots,o_{(c_1,c_2)}\}} (seqc_{(i+j)} \neq c_2), \quad \forall c_1, c_2 \in C, i \in \{1, \ldots, (p + n \cdot s - o_{(c_1,c_2)})\} \tag{15}$$

5.2.5 Helper variables and constraints for the objective function

To formulate the problem's minimization function, the following helper variables are defined:
The number 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 and 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):

$$cc_r = \sum_{j \in \{1,\ldots,s-1\}} f_c(c_{r,j}, c_{r,j+1}) + \sum_{\{j \in \{p_{r-1}\}\}} f_c(c_{r-1,j}, c_{r,1}), \forall r \in R \quad (16)$$

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

$$sc_r = p_r - sk_r + p_{r+1} - sk_r, \forall r \in \{0, \ldots, n-1\} \quad (17)$$

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:

$$sk_r = \sum_{k,l \in S} e_{r,k,l}, \forall r \in \{0, \ldots, n-1\} \quad (18)$$

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:

$$(e_{r,k,l} = 0) \Leftarrow (x_{r,k} = \epsilon \vee x_{r+1,l} = \epsilon), \quad \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S \quad (19)$$

$$(e_{r,k,l} = 1) \Rightarrow (v_{(x_{r,k})} = v_{(x_{r+1,l})}), \quad \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S \quad (20)$$

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.:

$$(e_{r,k,l} = 1) \Rightarrow \left(\bigwedge_{m \in \{1,\ldots,k-1\}, n \in \{l,\ldots,s\}} (e_{r,m,n} = 0) \wedge \bigwedge_{m \in \{k,\ldots,s\}, n \in \{1,\ldots,l-1\}} (e_{r,m,n} = 0) \right) \quad \forall r \in \{0, \ldots, n-1\}, k \in S, l \in S \quad (21)$$

5.2.6 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.

$$minimize \sum_{r \in \{0,\ldots,n-1\}} sc_r^2 + \sum_{r \in R} cc_r^2 \quad (22)$$

5.3 Utilizing the PSCCP to improve PSSP solutions

In the following, we describe how solution methods to the PSCCP can be utilized to improve given candidate solutions to the PSSP without changing the predetermined carrier sequence. This technique has the benefit that the number of required color changes in the production sequence can be improved without the need to consider the complex constraints of the PSSP that are related to the creation of feasible carrier type sequences. For this purpose, we take any candidate solution to the PSSP and create an instance for the PSCCP by simply removing the color assignments from all carriers in the sequence. We further generate demands for the PSCCP instance by looking at each individual carrier and by analyzing which paint shop demands are fulfilled by this carrier. The due round of the earliest demand that is processed by each individual carrier serves as the due round of the associated color demand in the PSCCP instance. Similarly, a solution to the generated PSCCP instance can be applied to the original candidate solution of the PSSP by applying the produced color sequence to the predetermined carrier sequence.

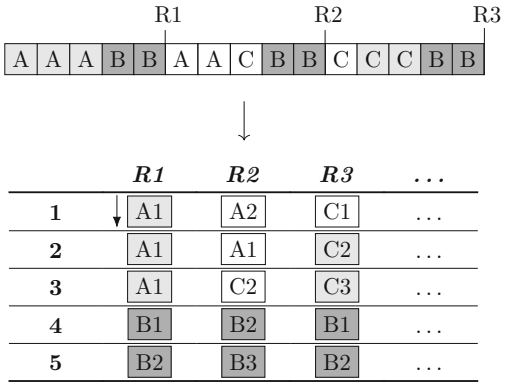

Fig. 5 An example PSCCP solution is mapped to the paint shop schedule for the original PSSP solution from Fig. 3

Note that in the original PSSP each carrier type actually can be used in a number of different configurations. Selecting a configuration can affect which and how many product types are placed on the carrier. When we create a PSCCP instance, we may ignore these configurations as long as we fulfill all the color demands, since we can safely remap the previously used configurations to the associated carrier type color pairs of the PSCCP solution later when we apply it to the PSSP schedule. Figure 5 illustrates this process by displaying a solution to the PSCCP example that was previously shown in Fig. 4 as well as a remapping to the PSSP schedule that is originally shown in Fig. 3.

The top of Fig. 5 visualizes the solution to the PSCCP, whereas the bottom shows the application of the solution to the PSSP schedule. Note that carrier types (which are represented by letters) have not changed compared to the original schedule shown in Fig. 3. However, a carrier of type A with configuration number 2 (A2) that was previously scheduled in R1 at position 3 now is scheduled at position 1 of R2. Furthermore, position 3 in R1 now schedules a type A carrier using configuration number 1 (A1). The reason why the configuration numbers need to change in this example is that the demands for the associated PSSP instance require a white carrier of configuration A2 and a light gray carrier of configuration A1. This remapping of the configurations can be easily done algorithmically by going over all changed carrier positions and reassigning the configurations to fulfill any missing demands of the PSSP with an earliest demands first strategy.

Up to now, we have seen that the PSCCP solution methods cannot directly change the configurations used in the corresponding PSSP schedule, but only indirectly affect their positioning in the schedule. However, if we pass additional information about the configurations (i.e., what product types are associated with each configuration) to an instance of the PSCCP and specify the demands in a way to request product types instead of carrier types, we could give the solution methods more possibilities to fulfill the demands by not only reassigning colors but also reassigning configurations in the carrier type sequence. We initially experimented with extended variants of the PSCCP that support the reassignment of configurations, but these variants turned out to be impractical for most benchmark instances with exact methods due to the largely increased size of the search space. However, we could successfully adapt the heuristic approach to support a reassignment of configurations without a notable loss in performance and therefore implemented this variant for our experimental evaluation of the PSCCP heuristic. The procedure shown in Algorithm 1 is still used with the only difference that when a position is evaluated to be colored for a given demand, the heuristic tries to assign the best configuration (i.e., the one which produces the most pieces for the demand) to this position. After execution has finished, the assigned colors and configurations are then transferred back to the PSSP solution.

5.4 A large neighborhood search operator for the PSSP

We now describe how the state-of-the-art metaheuristic technique for the PSSP which we previously proposed in Winter et al. (2019) can utilize solution methods for the PSCCP to improve candidate schedules during local search. Therefore, we propose a large neighborhood search (LNS) operator Φ_{LNS} for the PSSP that takes a candidate solution, solves the corresponding PSCCP problem to find an optimal coloring for the schedule, and then applies the optimal coloring to the PSSP candidate solution. Although Φ_{LNS} cannot be used to solve the complete PSSP problem, as it is unable to conduct any changes on the carrier sequence, it can effectively improve color change costs of the given schedule.

To implement Φ_{LNS}, we can directly use the solution methods we proposed in Sects. 4.2 and 4.1. However, in practice we need an additional time limit parameter that causes the operator to stop if solving the associated PSCCP takes too long. In the case of an exact solver, we can still use any intermediate solution when the time is running out, whereas for the heuristic approach we simply exit the main loop early and apply colors of adjacent positions for uncolored positions in case of a timeout.

In initial experiments with Φ_{LNS}, we discovered that for many of the realistically sized instances of the PSSP the operator usually ran out of time before any improvement could be achieved. The main reason for this is that the corresponding PSCCP (which is an NP-hard sub-problem on its own) has a search space that is too large to be effectively used within a local search neighborhood for the original PSSP.

Therefore, we further propose another LNS operator Φ_{LNS*} that destroys only parts of the colored carrier sequence and therefore reduces the search space for the asso-

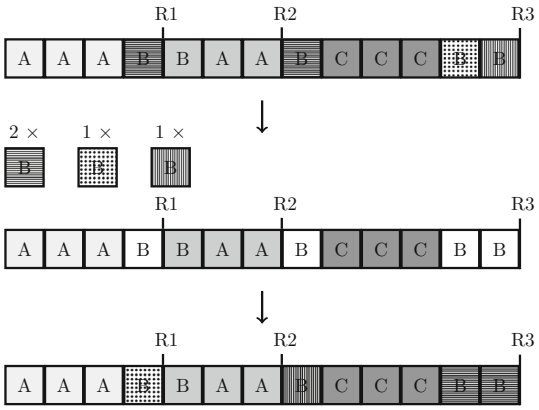

Fig. 6 Example how Φ_{LNS*} can reduce the number of color changes by rearranging small color blocks in the overall sequence

ciated PSCCP. The main idea behind Φ_{LNS*} is to leave large areas in the sequence that use only a single color intact, as they do not cause any color changes. In consequence, only color assignments from the remaining areas will be reassigned during the application of Φ_{LNS*}. Thus, many color assignments are predetermined in the corresponding PSCCP instances and usually only a small number of carriers need to be colored which speeds up the solution process of the Φ_{LNS*}. The intuition behind Φ_{LNS*} is further illustrated in Fig. 6.

At the top of the figure, we see a sequence that uses six different colors. The carriers colored in shades of gray are already arranged into "color blocks" of length 3, whereas the carriers using different pattern colors (horizontal stripes, dots, and vertical stripes) are not yet well arranged. The idea behind Φ_{LNS*} is to remove only small color block assignments from the sequence (which in this example are the pattern colors), while leaving larger color block assignments intact (which in this example are the gray colors). This is visualized in the middle of Fig. 6.

Finally, solving the corresponding PSCCP instance in this example only needs to reassign four colors and can actually find an improved solution that reduces the number of required color changes in total (the corresponding sequence is illustrated at the bottom of Fig. 6).

In addition to a time limit, we introduce a second parameter $k \in \mathbb{N}_{>0}$ for the operator Φ_{LNS*} to configure which areas of the sequence should have their colors reassigned. The parameter works in a way that any consecutive block of carriers that use a single color with a length that is greater than or equal to k will be left intact. In the example shown in Fig. 6 k is set to $k = 3$ as single color blocks of consecutive carriers with length ≥ 3 are not reassigned.

5.5 Integrating the LNS operator into local search for the PSSP

The previously introduced simulated annealing approach uses three local search neighborhoods to solve the PSSP: *carrier insertion*, *carrier removal*, and *carrier swap*. These existing neighborhoods consider the insertion, removal, or swapping of carriers in the schedule and can affect either single carriers or blocks of consecutive carriers. To incorporate the Φ_{LNS*} operator into the local search approach, we use an additional parameter $\alpha \in [0, 1]$ which defines the probability to call the LNS operator instead of a standard neighborhood move during a local search iteration. Furthermore, we only call the Φ_{LNS*} operator if the current solution has no hard constraint violations as the LNS operator can only improve the color change objective.

Figure 7 illustrates how the LNS operator is called within a local search iteration.

The candidate solution produced by the application of the LNS operator is accepted by local search (i.e., the result will be used as the current solution for the next search iteration) if the costs of the candidate solution are either improved, or the result leads to a new minimum color change cost. The latter is the case if a new upper bound for color change costs has been achieved by LNS, but overall solution costs are not improved (e.g., because a hard constraint has been violated). If this is the case, the LNS result is still used for the next search iteration and local search may try to improve the overall best-known solution from this point within a limited number of iterations (which is determined by an additional parameter β). If no overall best cost solution can be achieved within β iterations, local search resets its current solution back to the priorly known best solution. The rationale behind this fallback and the β parameter is to accept low color cost solutions that include some hard constraint violations (e.g., unfulfilled demands) to give local search a chance to repair these violations quickly and thereby to potentially find a new best solution.

Algorithm 2 presents the pseudo-code with further details on how the LNS operator is called during local search.

6 A novel construction heuristic for the PSSP

Previously, we proposed a construction heuristic (CH) for the PSSP in Winter et al. (2019). Experimental results showed that the use of a construction heuristic was important to produce feasible solutions for large problem instances within reasonable run-time. In this section, we propose a novel construction heuristic (CH*) that utilizes the solution methods for the PSCCP to color partially uncolored schedules during the creation of an initial schedule.

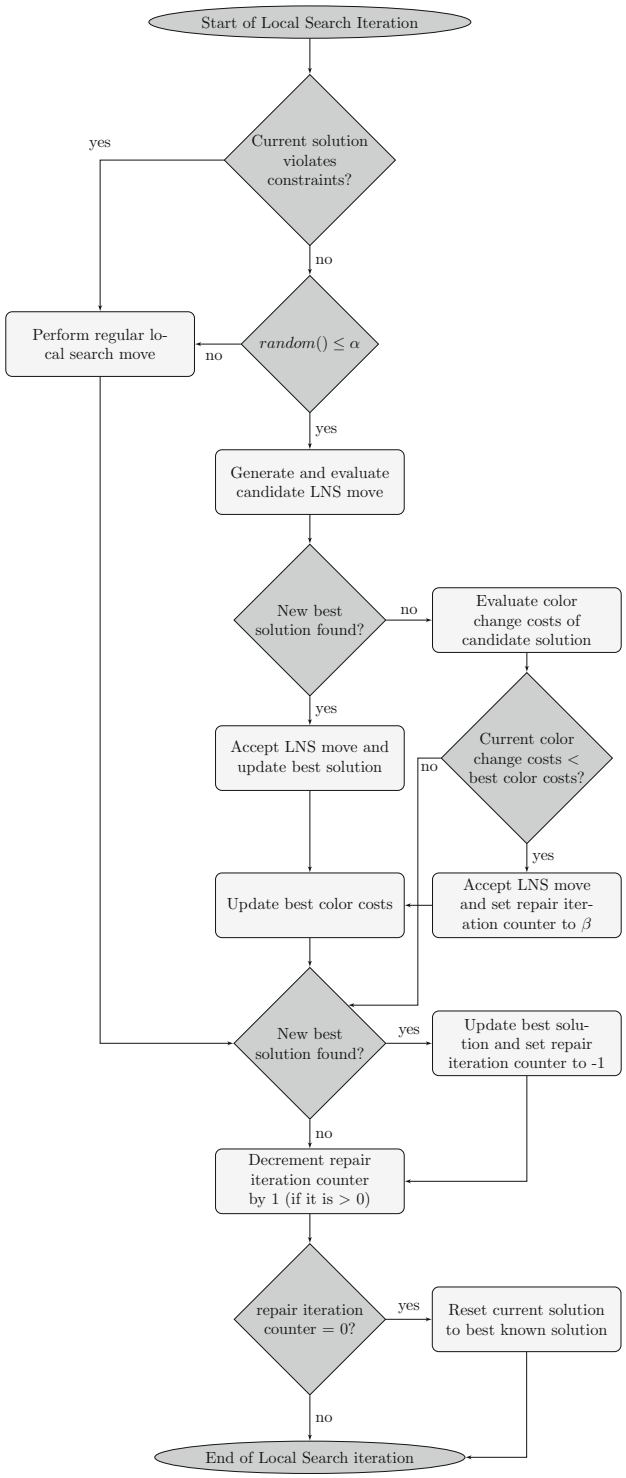

Fig. 7 Main execution steps during a local search iteration for the proposed PSSP algorithm that utilizes the LNS operator

Algorithm 2: Incorporating Φ_{LNS*} within local search

```
lnsRepairIterationCounter = 0
while main local search loop do
    if
     currentSolution.Violations == 0 ∧ random.Next() ≤ α
     then
        lnsSolution = perform Φ_LNS*(k)
        if lnsSolution.Cost < bestSolution.Cost then
            currentSolution = lnsSolution
            bestSolution = currentSolution
        else
            if lnsSolution.Cost < currentSolution.Cost then
                currentSolution = lnsSolution
            else
                colorChgCount =
                 CountColorChanges(lnsSolution)
                if colorChgCount < bestColorChgCount then
                    currentSolution = lnsSolution
                    lnsRepairIterationCounter = β
        bestColorChgCount =
         min {colorChgCount, bestColorChgCount}
    else
        perform standard local search iteration
    if currentSolution.Cost < bestSolution.Cost then
        bestSolution = currentSolution
        lnsRepairIterationCounter = 0
    if lnsRepairIterationCounter > 0 then
        lnsRepairIterationCounter =
         lnsRepairIterationCounter − 1
        if lnsRepairIterationCounter == 0 then
            currentSolution = bestSolution
```

To explain the details about CH*, we first need to mention that besides the minimization of the color changes in the schedule, a solution to the PSSP should also minimize the necessary carrier sequence changes between each pair of consecutive rounds in the schedule. Without going further into the details of the complex objective function of the PSSP, it is sufficient to know that the main aim is to keep the carrier type sequence of consecutive rounds as similar as possible. Furthermore, note that there is a history carrier sequence that is part of the input in any instance to the PSSP. This history round denotes the carrier sequence that was processed last before the current schedule. Therefore, the first round of the solution schedule should use a carrier sequence that is as similar as possible to the given history carrier sequence.

The existing construction heuristic CH creates an initial schedule by trying to first estimate an even distribution of the carrier demands over all scheduling rounds without determining the detailed carrier sequences. In a second step, CH looks at each of the rounds individually and tries to find a feasible colored carrier sequence. At the same time, CH also aims at keeping the number of color changes as well as carrier type changes from the previous round as low as possible.

The pseudo-code in Algorithm 3 presents the details about the novel construction heuristic CH* that we propose in this paper.

Algorithm 3: A novel construction heuristic for the PSSP

fn *CreateInitialSchedule*
- 1. copy history sequence to all rounds
- 2. Solve PSCCP problem to find coloring
- **while** ∃*unfulfilled demands* ∨ *carrier limits reached* **do**
  - 3. insert new carrier in all rounds
  - 4. Solve PSCCP problem to find coloring

return *colored schedule*

The main idea behind CH* is to first copy the uncolored carrier type sequence of the given history round to all rounds in the schedule, essentially keeping the number of carrier changes between each of the sequences at 0. In a second step, the construction heuristic solves an associated instance of the PSCCP to find an optimized coloring to the candidate schedule. After this coloring has been determined, there might still be remaining unfulfilled demands as not all required carrier types are necessarily included in the copied history sequence. If this is the case, the heuristic iteratively tries to insert a new carrier into each round of the schedule to handle remaining unfulfilled demands and again solves a partial instance of the PSCCP, where any colors that have been previously assigned are fixed. The process is repeated until either no new carriers can be inserted due to resource limits or if all demands are fulfilled. Figure 8 illustrates the main steps of the construction heuristic.

The rationale behind CH* is to keep the number of carrier changes between the rounds as low as possible by copying the history round, while at the same time the number of color changes is minimized by solving corresponding instances of the PSCCP. CH* does not guarantee to produce a feasible schedule (note that the existing heuristic CH also does not guarantee this); however, it is able to provide an initial schedule that is usually very low in costs compared to initial schedules produced by CH.

Algorithm 3 and Fig 8 describe and illustrate the main processing steps and the core idea of the construction heuristic for the PSSP that we propose in this paper. Additionally, we included in our implementation of the construction heuristic a consideration of minimum- and maximum-consecutive carrier block constraint violations when generating the initial carrier sequence. These constraints affect feasible carrier sequences such that numbers of consecutively scheduled carriers of the same type are restricted. We simply try to fix potential constraint violations by going over the carrier sequence and insert or remove single carriers. This is done twice in our implementation of CH*: First after the history round has been copied, and a second time before returning the initial schedule. Please refer to Winter et al. (2019) for details about the minimum- and maximum-consecutive carrier block constraints.

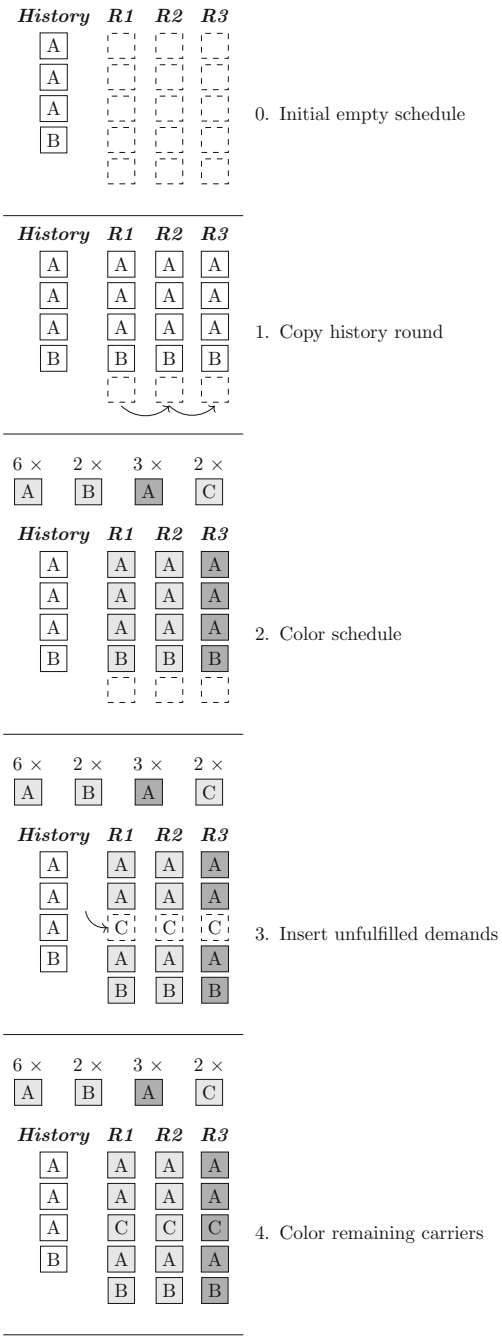

Fig. 8 Illustration of the main processing steps of the proposed construction heuristic (CH*). Steps 3 and 4 are repeated until all demands are fulfilled or no more carriers can be inserted due to resource limits

7 Empirical evaluation

In this section, we provide an extensive experimental evaluation of the large neighborhood search operator and construction heuristic we propose in this paper. First, we describe the setup and computational environment we have used to conduct our benchmark experiments in Sect. 7.1. Afterward, we elaborate on how parameters for the heuristic algorithms have been selected using state-of-the-art automated parameter tuning software in Sect. 7.2. Finally, the results of all our experiments are presented and discussed in Sect. 7.3.

7.1 Experimental environment

To evaluate the performance of the proposed LNS operator Φ_{LNS*} and the novel construction heuristic CH* for the PSSP, we extended the code for the simulated annealing-based metaheuristic approach from Winter et al. (2019) (we chose the variant that was able to find solutions for all benchmark instances) to implement these methods. All experiments discussed in this section were performed using the 24 publicly available benchmark instances from the literature.[4] Instances 1–12 are very small to medium-size and instances 13–24 contain very la

rge real-life paint shop scheduling scenarios.

Table 2 summarizes the size parameters for the 24 benchmark instances, where every row contains parameters for a single instance.

Columns 2 and 3 include information about the total number of rounds (Rounds) as well as the maximum capacity of carriers per round (Round Capacity), whereas Columns 4–6 display the total number of colors (Colors), carrier types (Carrier Types), and demands (Demands) that are specified by the instances. Finally, Column 7 indicates whether forbidden carrier and color sequence hard constraints are imposed by the instance.

Note that instances 1–12 are considered to be small instances, as the round capacity as well as the number of colors, carrier types, and demands are much smaller than for instances 13–24. Many of these small instances could be previously solved to optimality using exact methods in Winter and Musliu (2021); Winter et al. (2020). Instances 13–24 on the other hand represent real-life scheduling scenarios from a large-scale industrial paint shop; therefore, they all use the same round capacity, colors, and carrier types. As the parameters of these instances lead to a very large search space, none of these instances could be efficiently tackled using existing exact methods, but the simulated annealing approach could provide practical solutions in a reasonable time. As some of the benchmark instances to the PSSP define hard constraints that impose forbidden color sequences, we adapted our implementation of the PSCCP heuristic (see Algorithm 1) to include violations to this constraint in its cost evaluation function (i.e., the best color position is the one which introduces the lowest number of forbidden color violations, ties are broken by color change costs). Furthermore, we used a performance efficient implementation of the cost evaluation function that utilizes incremental evaluation (i.e., only areas in the schedule that have been modified since the last evaluation call will be reevaluated).

To incorporate the forbidden color constraint in the MiniZinc model (Listing 1), we initially experimented with the deterministic finite automaton encoding we previously proposed in Winter and Musliu (2021) for this constraint. However, early experiments showed that it was more effective to not include the forbidden color constraint in the model so that the solution time needed by the large neighborhood search operator is reduced. If forbidden color violations are caused by the operator, the local search process is usually able to quickly repair any forbidden color violations. We further used an up-to-date version of the MiniZinc software Nethercote et al. (2007) to solve instances of the PSCCP with recent versions of the CP solver chuffed Chu (2011) and the MIP solver Gurobi (2020).

As the PSSP actually uses squared color change costs per round in its objective function instead of a simple summation of the changes, we further incorporated such a solution objective in both the exact modeling approach and the heuristic approach for the PSCCP by slightly changing the solution objective to consider the squared color changes per round. Additionally, in some instances for the PSSP the costs for a single color change may in rare cases vary depending on the specific pair of colors which are involved. We considered these specific costs in the implementation of the heuristic PSCCP approach; however, we decided to not include it in the MiniZinc model as it drastically slowed down the model compilation and the overall solution process in early experiments.

Initial experiments further showed that the PSCCP solution process of the exact solvers was too time-intensive for the repeated call in CH*, which caused an impractical run time of the heuristic. Therefore, we evaluated only an implementation of CH* that uses the PSCCP heuristic in our experiments. The LNS operator on the other hand was still evaluated using both the heuristic and exact solution methods for the PSCCP.

To compare the existing simulated annealing-based approach as well as the existing PSSP construction heuristic with the proposed methods in this paper, we evaluated a variety of different configurations of the LNS operator and the novel construction heuristic in our experiments. Table 3 gives an overview of all the different evaluated solution methods.

Column 1 of the table displays an abbreviation that will be used to refer to the respective method later in this section, whereas Column 2 describes the configuration of the respec-

[4] https://www.dbai.tuwien.ac.at/staff/winter/.

Table 2 Overview of instance size parameters for the 24 publicly available benchmark instances for the PSSP

Instance	Rounds	Round capacity	Colors	Carrier types	Demands	Forbidden Seq.
I 1	7	19	6	2	2	no
I 2	7	19	7	2	0	yes
I 3	20	19	7	2	0	no
I 4	20	19	4	2	4	yes
I 5	50	19	4	3	22	no
I 6	50	19	6	2	0	yes
I 7	70	19	4	4	55	no
I 8	70	19	8	2	5	yes
I 9	100	19	6	2	39	no
I 10	100	19	6	2	35	yes
I 11	200	19	6	3	118	no
I 12	200	19	7	4	384	yes
I 13	7	480	20	46	108	no
I 14	7	480	20	46	108	yes
I 15	20	480	20	46	238	no
I 16	20	480	20	46	238	yes
I 17	50	480	20	46	1055	no
I 18	50	480	20	46	1055	yes
I 19	70	480	20	46	1743	no
I 20	70	480	20	46	1743	yes
I 21	100	480	20	46	2469	no
I 22	100	480	20	46	2469	yes
I 23	200	480	20	46	5907	no
I 24	200	480	20	46	6057	yes

tive method. All evaluated approaches use local search and thereby randomly selected moves; therefore, we conducted 10 repeated runs for each instance and used the arithmetic mean solution costs for our evaluations if not stated otherwise.

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 @ 2.20GHz and 252 GB RAM.

7.2 Parameter configuration

To configure the heuristic methods we propose in this paper, we need to select a number of parameters. In a first step, we selected reasonable defaults for each parameter based on some manual tuning runs with a few realistically sized instances. Afterward, we used a recent version of the state-of-the-art automated parameter configuration software SMAC Lindauer et al. (2017), which we used to tune the parameters. Table 4 provides an overview of all configured parameters.

The first column of Table 4 displays the parameter name, while the second column shows a brief description of the parameter. Column three presents the allowed parameter range for the tuning process, and column four displays the default value given to SMAC. (We selected the ranges manually so that they include a reasonable range near the default value.)

The approach from Winter et al. (2019) used two manually tuned parameters: The initial temperature and cooling rate for simulated annealing (see the default values in Table 4). To encourage a fair comparison, we decided to tune these two parameters in a first tuning process using the simulated annealing approach that uses the existing construction heuristic (LS/C). In a second tuning process, we tuned the parameters for the LNS method and novel construction heuristic proposed in this paper. Therefore, we handed the solution method that uses the novel construction heuristic and the heuristic PSCCP solution method (LNS-H/C*) to the second tuning process.

We executed both tuning processes with SMAC using the following settings: Instances 1–24 were used as the training set, and we set the cutoff time per instance to 30 minutes. The overall tuning process was given a wall clock time limit of 4 full days.

The tuned results for all parameters are shown in Column 5 of Table 4. We used these parameter settings for all of our final experiments and set the runtime limit to 1 hour.

Table 3 Overview of the evaluated solution methods

Solver ID	Description
LS	Existing local search approach using simulated annealing from Winter et al. (2019), starting from an empty initial solution
LS/C	As LS, but starting from an initial solution that was created by the existing construction heuristic from Winter et al. (2019)
LS/C*	As LS/C, but using the novel construction heuristic to create an initial solution
LNS-H	As LS, but including the proposed LNS operator that uses the heuristic solution method for the PSCCP, starting from an empty solution
LNS-H/C*	As LNS-H but starting from a solution that is created from the novel construction heuristic
LNS-CP	As LNS-H but using the exact MiniZinc approach with the chuffed CP solver
LNS-CP/C*	As LNS-CP but starting from a solution created by the novel construction heuristic
LNS-IP	As LNS-H but using the exact MiniZinc approach with the gurobi MIP solver
LNS-IP/C*	As LNS-IP but starting from a solution created by the novel construction heuristic

Table 4 Overview of the configured parameters and tuning results

Parameter	Description	**Range**	**Default Value**	**Tuning Result**
Initial temperature	The initial temperature used by simulated annealing	[0.1, 0.5]	0.25	0.1297
Cooling rate	The cooling rate used by simulated annealing	[0.9, 0.99]	0.95	0.9462
k	Configures which color assignments are destroyed and repaired during a lns move	{2, 5, 10, 20, 40}	5	2
α	The probability to conduct a LNS operator move in a local search iteration	[0.00001, 0.001]	0.0001	0.000044
β	Configures how many iterations can be used by local search to find a new best result after a LNS move	{100, 200, 500, 1000, 10000}	1000	10000
Demand look ahead	Configures how many demands are considered when searching for a color assignment in the novel construction heuristic	{1, 20, 50, 100, 150}	50	20
Time limit	Configures the time limit (in seconds) for the PSCCP solution method within LNS	{60, 90, 120, 180}	60	120

7.3 Computational results

An overview of the final results for the 24 paint shop scheduling benchmark instances is presented in Table 5.

Columns 2–10 show the relative per instance results for each of the evaluated solvers (see Table 3 for an explanation about the solver IDs). To calculate relative per instance results, the mean solution costs (produced by 10 repeated experimental runs) were divided by the overall best mean solution costs per row. In other words, this means that a value of 1 indicates an overall best mean cost result for an instance, and values greater than 1 display the mean costs relative to the best mean costs in our experiments.

We can see in the results shown in Table 5, that approaches which are starting the search from a heuristically generated solution are able to find feasible solutions for all 24 benchmark instances, whereas approaches that start from an empty initial solution cannot produce solutions for instances 23 and 24 within the time limit. Furthermore, the results show that starting from a heuristically constructed schedule leads to better results for the larger instances 13–24. However, for the small- to medium-sized instances 1–12, for some instances best results were achieved by solution methods that started from an empty initial schedule. This indicates that generating a good solution quickly is beneficial especially for large-scale instances, where starting from an empty schedule would be too time-intensive.

We can see that results produced by the existing methods are improved for all of the instances by at least one of the new solution methods that use the novel construction heuristic or the LNS operator. This shows the strength of the new techniques, especially for the large instances, where many results are improved by factors larger than 4.

If we look at the results produced with the LNS operator, we see differences in the quality of the results depending on the utilized PSCCP solution approach. Overall, the approach that uses the heuristic PSCCP solution method produces the best results for the majority of the instances; however, the exact PSCCP solution approach using the CP solver chuffed can reach the best results for many of the smaller instances. The approach using the gurobi integer programming solver for the PSCCP produces similar results and reaches the best results for 3 of the smaller instances and the largest instance.

Table 6 presents the absolute best results for all evaluated approaches. The displayed costs are the best costs out of the 10 repeated experimental runs for each method and instance.

The best results per row are formatted in boldface. Overall, we can see that the best cost results show a similar outcome as the relative mean results. However, as this table presents the single best result out of ten repetitive runs, we can see some outliers that do not match the best relative mean results. For example, we can see that the traditional local search approach is able to produce the best result for instance 8, although it did not achieve the best score in the mean cost comparison. Furthermore, the best bound for the largest instance in this case is produced by the LNS-CP/C* approach, although the best relative mean cost results were achieved using the LNS-IP/C* approach.

Figure 9 shows a comparison of all approaches that start from an initial solution that was generated by a construction heuristic.

Table 5 Overview of the relative results for each of the evaluated solution methods

Instance	LS/C	LS/C*	LS	LNS-H/C*	LNS-H	LNS-CP/C*	LNS-CP	LNS-IP/C*	LNS-IP
I 1	1.14	1.11	1.13	1.06	1.06	1	1.05	1.07	1.06
I 2	1.12	1.05	1.03	1.08	1.03	1	1.02	1.1	1.04
I 3	1.05	1	1.05	1	1.07	1	1.04	1	1.06
I 4	1.08	1.06	1.1	1.07	1.06	1.03	1	1.03	1.07
I 5	1.11	1	1.11	1	1.19	1	1.18	1	1.19
I 6	1.03	1	1.03	1.01	1.05	1.02	1.04	1	1.05
I 7	1.22	1.01	1.22	1.01	1.28	1	1.26	1.02	1.26
I 8	1.09	1.05	1.02	1.05	1.04	1.03	1	1.2	1.06
I 9	1.13	1.12	1.09	1.1	1.06	1.08	1	1.38	1.15
I 10	1.03	1	1.03	1.03	1.07	1.01	1.05	1.04	1.04
I 11	1.4	1.08	1.38	1.09	1.49	1	1.38	1.17	1.43
I 12	1.38	1	1.11	1.08	1.24	1.02	1.2	1.21	1.2
I 13	10.82	1.07	255.51	1	205.52	1.05	283.54	1.37	261.09
I 14	9.13	1.19	145.91	1	122.31	1.04	162.4	1.67	131.98
I 15	5.25	1.54	252.21	1	193.19	1.35	254.73	1.43	248.85
I 16	2.97	1.06	107.54	1	84.6	1.22	111.05	1.16	102.74
I 17	3.16	1.32	214.34	1	171.9	1.26	219.45	1.38	218.38
I 18	1.63	1.06	47.53	1	37.9	1.23	47.83	1.01	48.17
I 19	4.19	1.57	270.05	1	222.06	1.74	272.74	1.72	271.31
I 20	1.51	1.16	53.08	1	42.62	1.29	53.54	1.25	53.61
I 21	5.65	1.07	284.26	1	226.61	1.2	285.26	1.23	284.18
I 22	2.07	1.15	60.56	1	56.32	1.1	60.73	1.27	60.89
I 23	4.04	1.23		1		1.35		1.33	
I 24	1.58	1.02		1.12		1.08		1	

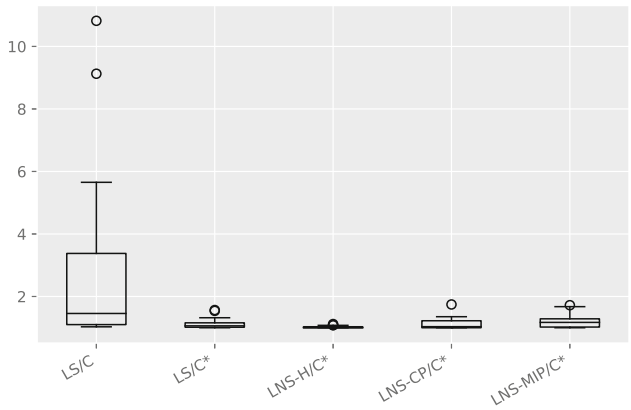

Fig. 9 Box plots comparing the relative mean cost results produced by approaches that use a construction heuristic to generate an initial solution

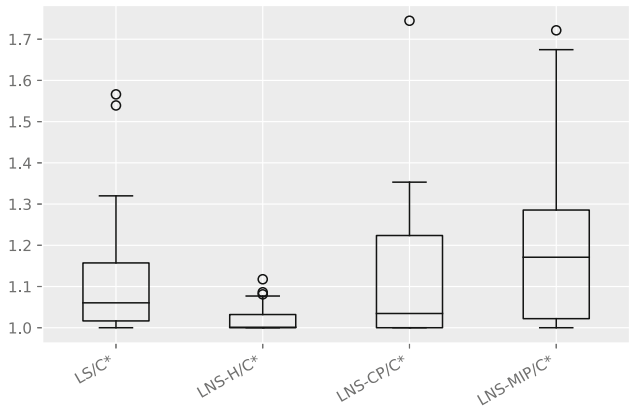

Fig. 10 Box plots comparing the relative mean best results produced by approaches that use the novel construction heuristic to generate an initial solution

The figure visualizes the relative mean cost results for all 24 instances as box plots. We can see that approaches using the novel construction heuristic overall lead to the best results in our experiments with a median value close to 1, whereas the existing construction heuristic has a median value at roughly 1.5 and in general a much wider range with some outliers even lying above the value 8.

Figure 10 shows box plots only for approaches that use the novel construction heuristic.

We can see that overall the approach using a large neighborhood search operator that utilizes the heuristic PSCCP

Table 6 Overview of the best results produced by the evaluated methods

Instance	LS/C	LS/C*	LS	LNS-H/C*	LNS-H	LNS-CP/C*	LNS-CP	LNS-IP/C*	LNS-IP
I 1	822	808	849	802	825	795	798	834	794
I 2	889	865	876	929	880	844	847	937	871
I 3	988	961	990	961	1009	961	971	961	1007
I 4	966	956	994	956	953	930	943	974	951
I 5	574	530	577	530	600	531	599	530	598
I 6	875	845	863	850	888	853	872	849	879
I 7	1032	867	1033	856	1089	858	1056	882	1063
I 8	1496	1480	1426	1489	1437	1459	1485	1770	1434
I 9	1345	1321	1240	1332	1240	1308	1272	1561	1316
I 10	1077	1058	1088	1085	1121	1053	1111	1082	1099
I 11	4318	3346	4357	3268	4608	3030	4353	3595	4407
I 12	5238	3463	4168	3699	4463	3697	4625	4501	4476
I 13	74236	7379	1683619	6592	1406142	6815	1577682	9023	1788255
I 14	110714	10341	1710055	9374	1503198	9108	1923650	12460	1329706
I 15	153728	28385	7291013	25427	5590121	29240	7481155	32572	7332157
I 16	213003	47097	7690901	42765	5995974	51953	8005671	61039	5898146
I 17	323829	105397	21716866	68841	17103946	107991	22135745	95567	21884939
I 18	597419	315377	22684889	285572	17709696	381400	22764282	311956	22865284
I 19	497486	108953	32447139	96825	25871047	144226	32631108	130235	32544683
I 20	913110	491845	33061150	476506	26323904	609803	33287708	568279	33573545
I 21	937094	126750	48199455	135757	37984334	146352	48381085	157073	47520813
I 22	1674595	615530	49354821	535846	40215243	486698	49219231	690072	49633330
I 23	1714000	495635		357051		506831		522690	
I 24	2609884	1209816		1290190		1141429		1327593	

solution method produces the best results, having a median value close to 1 and the smallest interquartile range. When comparing the approaches using exact PSCCP solution methods with the approach using no LNS operator, we can see that the LNS approach using chuffed has the lowest median, followed by the existing local search approach which has the second smallest interquartile range.

In Fig. 11, we can see box plots for the mean costs results for instances 1–22, which were produced by the approaches that start from an empty initial schedule.

Again the LNS approach that uses the heuristic PSCCP solution method has the smallest median value, followed by the LNS approach using the CP solver to solve the PSCCP before the MIP-based LNS approach and the existing local search approach. This indicates that the novel techniques can improve results even when they are used without an initial construction heuristic.

Table 7 displays the relative mean costs produced by the local search together with the existing construction heuristic approach in direct comparison with the LNS variant that uses the heuristic PSCCP solution method together with the novel construction heuristic. We selected these two methods for a direct comparison as the former was the overall best performing existing approach from the literature in our experiments, and the latter is the overall best performing novel approach that we propose in this paper.

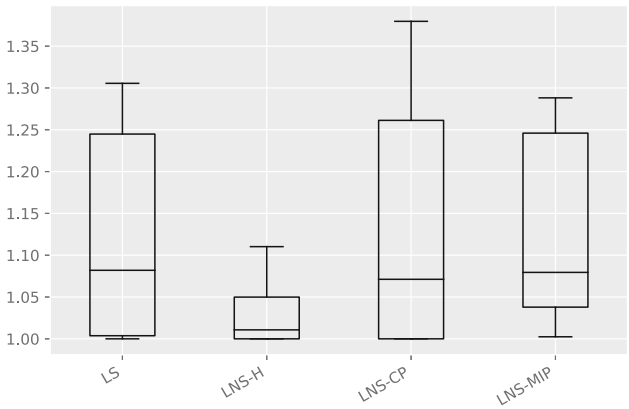

Fig. 11 Box plots comparing the relative mean best results produced by approaches that start from an empty initial solution

The results in Table 7 show that the novel approach produced the best results for all instances in this comparison and was even better than the existing approach in 23 out of 24 cases.

Table 7 Direct comparison of the relative mean cost results produced by the overall best existing method with the overall best novel method proposed in this paper

Instance	LS/C	LNS-H/C*
I 1	1.14	1.06
I 2	1.12	1.08
I 3	1.05	1
I 4	1.08	1.07
I 5	1.11	1
I 6	1.03	1.01
I 7	1.22	1.01
I 8	1.09	1.05
I 9	1.13	1.1
I 10	1.03	1.03
I 11	1.4	1.09
I 12	1.38	1.08
I 13	10.82	1
I 14	9.13	1
I 15	5.25	1
I 16	2.97	1
I 17	3.16	1
I 18	1.63	1
I 19	4.19	1
I 20	1.51	1
I 21	5.65	1
I 22	2.07	1
I 23	4.04	1
I 24	1.58	1.12

Finally, we compare the best upper bounds on the solution costs for each of the 24 PSSP benchmark instances which were produced by the novel methods proposed in this paper with previously published upper bounds in Table 8.

Column 2 of the table displays the best upper bounds for each instance that were achieved with the existing metaheuristic proposed in Winter et al. (2019) (LS), and Column 3 displays the best solutions produced with exact solution methods for the PSSP from Winter and Musliu (2021). Note that the exact methods could further prove that the best results for instances 1–9 are optimal. The best results produced by the solution methods proposed in this paper are shown in Column 4. Best results per instance are formatted in boldface.

The results in Table 8 show that the LNS approach is able to improve results for all 24 instances compared to the existing metaheuristic results. Furthermore, we can see that the methods proposed in this paper are able to provide previously unknown upper bounds for instances 11–24. Exact methods still produce the best results for instances 1–10, but the LNS approaches are able to reach optimal results for two of the instances and several additional nearly optimal results for small instances.

Table 8 Overview on the best upper bounds for all 24 PSSP benchmark instances that were produced by existing heuristic methods, existing exact methods, and the methods proposed in this paper

Instance	LS	EM	LNS
I 1	806	775	794
I 2	868	842	844
I 3	990	961	961
I 4	975	918	930
I 5	593	530	530
I 6	887	842	845
I 7	1084	844	856
I 8	1834	1237	1434
I 9	1735	975	1272
I 10	1134	964	1053
I 11	5236		3030
I 12	5723		3463
I 13	116235		6592
I 14	118628		9108
I 15	172679		25427
I 16	262252		42765
I 17	421777		68841
I 18	581021		285572
I 19	555829		96825
I 20	927822		476506
I 21	917955		126750
I 22	1128716		486698
I 23	1884125		357051
I 24	2086450		1141429

8 Conclusion

In this work, we introduced a novel paint shop coloring problem (PSCCP) that appears as a sub-problem in a real-life paint shop scheduling problem. We further proposed a heuristic solution approach as well as an exact technique based on constraint modeling to solve instances of the PSCCP.

Additionally, we proposed an innovative LNS operator that can be utilized within metaheuristic approaches based on local search for a recently introduced paint shop scheduling problem (PSSP). This novel LNS operator utilizes the proposed solution methods for the PSCCP to find optimized color sequences, which can be used to improve candidate schedules for the PSSP by applying the resulting colors to PSSP solutions.

We evaluated all proposed techniques by performing a series of experiments using a set of benchmark instances for the PSSP from the recent literature. The results showed that the proposed methods lead to a clear performance improvement as results for every benchmark instance could be improved when compared with existing metaheuristic

approaches. The LNS operator which used the heuristic solution method for the PSCCP overall produced the best results for the majority of the instances, followed by variants that utilized exact methods for the PSCCP using state-of-the-art constraint programming and mixed integer programming solvers. Furthermore, we were able to provide previously unknown upper bounds to 14 benchmark instances using the novel techniques.

In future work, it could be interesting to investigate metaheuristic solution methods for the PSCCP.

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

Funding Open access funding provided by TU Wien (TUW).

References

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378.

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, 27*(2), 219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*(3), 985–1032.

Chu, G.G. (2011). Improving combinatorial optimization

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

Gurobi Optimization, L.: Gurobi Optimizer Reference Manual (2020)

Lee, J. H., Jang, H., & Kim, H. J. (2021). Iterative job splitting algorithms for parallel machine scheduling with job splitting and setup resource constraints. *Journal of the Operational Research Society, 72*(4), 780–799.

Lindauer, M., Eggensperger, K., Feurer, M., Falkner, S., Biedenkapp, A., Hutter, F. (2017). SMAC v3: Algorithm Configuration in Python. GitHub

Meunier, F., & Neveu, B. (2012). Computing solutions of the paintshop-necklace problem. *Computers & Operations Research, 39*(11), 2666–2678.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., & Tack, G. (2007). MiniZinc: Towards a Standard CP Modelling Language. Lecture Notes in Computer ScienceIn C. Bessière (Ed.), *Principles and Practice of Constraint Programming – CP 2007* (pp. 529–543). Berlin, Heidelberg: Springer.

Prandtstetter, M., & 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., & 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., & Voß, S. (2004). A sequential ordering problem in automotive paint shops. *International Journal of Production Research, 42*(9), 1865–1878.

Van, B. K., & Hop, N. V. (2021). Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness- tardiness. *Journal of Industrial and Production Engineering, 38*(1), 18–28.

Winter, F., Musliu, N. (2021). Constraint based modeling for scheduling paint shops in the automotive supply industry. ACM Transactions on Intelligent Systems and Technology.

Winter, F., Musliu, N., Demirovic, E., Mrkvicka, C. (2019). Solution approaches for an automotive paint shop scheduling problem. In: ICAPS, pp. 573–581. AAAI Press.

Winter, F., Musliu, N., Stuckey, P.J. (2020). Explaining propagators for string edit distance constraints. In: AAAI. AAAI Press.