CIRRELT

CIRRELT-2025-04

A Logic-Based Benders Decomposition for the Car Resequencing Problem with a Painted Body Storage

Xinyi Guo
Jean-François Côté
Canrong Zhang
Lixin Miao

January 2025

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2025-003

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

A Logic-Based Benders Decomposition for the Car Resequencing Problem with a Painted Body Storage

Xinyi Guo[1], Jean-François Côté[1,2,*], Canrong Zhang[1,2], Lixin Miao

1. Division of Logistics and Transportation, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, FSA, Université Laval

Abstract. The cost of producing diverse cars depends on the sequence in which they are arranged in the body shop, paint shop, and assembly shop. Before entering the downstream assembly shop, the upstream car sequence shared by the body shop and paint shop is readjusted via the painted body storage, which consists of several first-in-first-out lanes. The car resequencing problem addressed in this paper requires determining the upstream and downstream sequences and the car-to-lane assignment to minimize the total cost of the three shops. We propose a nested logic-based Benders decomposition approach with three levels, where each car is assigned a body and a color in the first level to determine the upstream sequence. In the second level, cars are rearranged by determining their configurations and downstream positions. A feasible assignment of cars to lanes is sought in the third level to respect this sequence change. We provide a mathematical formulation for each level and propose two shortest-path problem reformulations for the first level, where solving the first reformulation is equivalent to a *k*-shortest path problem. The second reformulation is a shortest-path model restricted by demand constraints. A lower bound, valid inequalities, and a heuristic method are also proposed as enhancements. In our CRSP, the number of cars to be resequenced is not limited by the PBS size. Computational results show that our approach can handle instances of up to 120 cars, about ten times more than previous studies. A sensitivity analysis is conducted to provide some managerial insights.

Keywords: Car resequencing problem, Nested logic-based Benders, decomposition, k-shortest path problem

Acknowledgements. We thank the Digital Research Alliance of Canada for providing high-performance computing facilities.

---

* Corresponding author: jean-francois.cote@fsa.ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2025

1 Introduction

A car model is made up of three attributes: the body (e.g., hatchback, SUV), the color, and the configuration (e.g., flagship version, deluxe version), which are processed in three consecutive shops. First, metal parts are welded together to frame the overall body of one car in the body shop. This car then moves to the paint shop to be sprayed with a color. Finally, in the assembly shop, a group of options (e.g., sunroof, power seats) are assembled to complete the configuration of the car.

To adapt to the high customization nowadays, various car models are arranged in one sequence to be processed on a mixed-model production line (MMPL) that runs through the above three shops.

The car sequence arranged on the MMPL has a crucial impact on the cost of each shop. To reduce expenses related to switching welding tools and cleaning spray nozzles, the body shop and the paint shop look for a sequence with the least number of body and color changes, respectively. Meanwhile, a production requirement known as the color batch limit restricts the number of consecutive cars with the same color within a constant to maintain spray quality. The assembly shop prefers a sequence that minimizes workstation overload or ensures a steady consumption of options. Several methods are available to create a car sequence that minimizes the cost of the assembly shop or the total cost of the paint shop and assembly shop. So far, the body shop has not been considered.

In real-world manufacturing, a sequence of cars with preset attributes can be changed to a new sequence via a buffer. The buffer is a physical structure placed between two adjacent shops. For clarity, we refer to the two shops as the upstream and downstream shops. A car leaving the upstream shop can be temporarily stored in the buffer so that other cars behind it can enter the downstream shop first. This sequence change benefits the reduction in the total cost of both shops simultaneously.

The resulting *car resequencing problem* (CRSP) with a buffer studies how to change an upstream car sequence via the buffer to a downstream one. The *painted body storage* (PBS) between the paint shop and the assembly shop is the most widely used buffer. It consists of multiple linear lanes where cars enter and exit in a First-In-First-Out (FIFO) manner. Another way to change the car sequence is *virtual resequencing* (VRS), which works by swapping the preset attributes of cars. Applying VRS in the CRSP with a buffer contributes to further cost reduction [3], but it increases complexity because pure VRS is also NP-hard [13].

This work addresses a CRSP involving a PBS and VRS faced by a Chinese carmaker. For cars to be produced in one shift, their upstream sequence shared by the body shop and paint shop is changed via the PBS to the downstream one in the assembly shop. Our goal is to minimize the total cost of the three shops by determining the body, color, configuration, downstream position of each car, and which lane of the PBS it enters. Unlike the carmaker making decisions sequentially, we optimize the objectives of the three shops jointly for the first time.

In related studies, the number of cars is assumed not to exceed the PBS size. This assumption yields a static car-to-lane assignment problem and may cause unnecessary PBS construction costs. Moreover, the largest instance that can be solved so far contains only 18 cars [27], which is insufficient for a real-world production plan (e.g., 50 cars are produced per one-hour shift).

In this work, the number of cars to be rearranged is not restricted by the PBS size. We develop a *nested logic-based Benders decomposition* (NLBBD) method with three levels to solve real-world instances. The optimal upstream sequence is obtained in the first level by determining the body and color of each car. It is then rearranged in the second level by assigning each car a configuration and a downstream position. To achieve the sequence change, we seek a feasible car-to-lane assignment in the third level.

We propose two shortest-path problem reformulations for the first level to improve our method. Other algorithmic enhancements include a lower bound to tighten the first-level formulation, three valid inequalities to guide the second level, and a heuristic method to construct initial solutions. Besides experiments to show the efficiency of our approach, a sensitivity analysis is also conducted to derive managerial insights.

Regarding the contributions, our approach is the first exact method for the CRSP that involves physical and virtual resequencing, to our knowledge. The two shortest-path problem reformulations are applicable to general sequencing problems that aim to minimize the number of changeovers. With the first reformulation, repeatedly solving the master problem in the logic-based Benders decomposition (LBBD) can be treated as solving a k-shortest path problem (kSPP). We prove the stopping criteria and verify the optimal solution. The second reformulation is based on an aggregated graph and can provide higher-quality solutions.

In the remainder of this paper, Section 2 reviews related studies. Section 3 details the CRSP in this work. The NLBBD and enhancements are proposed in Section 4. Section 5 describes two shortest-path problem reformulations. Section 6 states how to perform our approach overall. Computational results and the sensitivity analysis are shown in Section 7. Section 8 gives the concluding remarks.

2 Literature Review

This section reviews related studies on the car sequencing problems, the CRSP, and the LBBD, as well as some inspirations for our work.

2.1 Sequencing Problems in Car Manufacturing

Car sequencing problems initially address how to arrange cars to be assembled in the assembly shop, where each station serves to install one specific option having a limited supply. Three approaches are discussed to determine a desired car sequence: *mixed-model sequencing* (MMS), *car sequencing problem* (CSP), and *level scheduling* (LS). The first two aim to minimize the station work overload, whereas the third aims to keep options consumed steadily from one period to the next.

Work overload is minimized explicitly in the MMS via various objective functions [14]. The CSP works implicitly by using a set of $H_w : N_w$ ratio rules defined for options. The ratio means the number of cars with option w in any subsequence of length N_w cannot exceed H_w. The CSP aims to find a sequence that respects all rules or minimizes rule violations [24].

LS is acknowledged as a priority in the Just-In-Time production system. Our carmaker applies it to maintain a constant consumption of options. Like Wu et al. [27], we minimize

the absolute deviation of the actual consumption from the ideal consumption of all options to achieve the goal.

Boysen et al. [8] review these approaches. Besides heuristics, some graph representations are proposed to obtain exact solutions or lower bounds [7, 15].

Subsequently, in the 2005 ROADEF challenge, car sequencing problems are extended to include the paint shop. Participants needed to determine a car sequence that minimizes the number of color changes and ratio-rule violations, while respecting the color batch limit. Benoist [5] proves the optimality of 54% of the best-known solutions of the benchmark in the challenge. Using a new formulation and column generation, Jahren and Achá [18] obtain better lower and upper bounds. Exact and heuristic methods developed for this challenge are reviewed by Solnon et al. [23].

This work jointly optimizes the objectives of all three shops for the first time. We enrich the graph representations for car sequencing problems, which are adaptable to other sequencing problems.

2.2 Car Resequencing Problem

According to the review of Boysen et al. [9], the reason for readjusting a car sequence divides the CRSP into reactive and proactive resequencing. The former handles unforeseen production disturbances, while the latter aims to reconstruct a sequence that benefits the shop where cars are ready to enter. This work focuses on the latter, where cars are rearranged for the assembly shop.

A car sequence can be readjusted virtually without moving cars (i.e., VRS) or physically via a buffer. The key in VRS is to decouple cars from their preset attributes. As for *physical* resequencing, four types of buffers are available: the automated storage and retrieval system, mix bank, pull-off table, and insert buffer. The mix bank comprising several FIFO lanes is the most widely used one [25]. Our addressed PBS is a specific mix bank in front of the assembly shop.

Considerable heuristics are developed for the CRSP with a mix bank, which is NP-hard in the strong sense. These methods often decompose the complete CRSP into a release subproblem that guides cars to leave the buffer, and a fill subproblem that assigns cars to lanes [6]. Meanwhile, such methods can only manage no more cars than the size of the mix bank. For more cars, the entire car sequence is split into segments to apply these methods in a rolling horizon.

Few efforts are made for exact methods. Ko et al. [19] treats the CRSP with a mix bank having two lanes as a traveling salesman problem. By dynamic programming (DP), they get the optimal solution for 20 cars within one hour. This number is then extended to 56 using stronger bounds [16]. The CRSP in Wu et al. [27], which is closely similar to our CRSP, is formulated as a mixed integer programming (MIP) model and solved for 18 cars within one hour using Gurobi.

In response to the challenges outlined by Boysen and Zenker [6] about the CRSP, such as the need for more efficient exact procedures and the impact of some parameters, we propose an exact method for real-world instances and conduct a sensitivity analysis on two types of parameters.

2.3 Logic-based Benders Decomposition

The LBBD was formally proposed by Hooker and Ottosson [17] as an extension of Benders decomposition (BD) [4]. It decomposes an optimization problem with a complex formulation into a master problem (MP) and one or several independent subproblems (SPs). In each iteration, the MP is solved first to assign values to its complicating variables. Then, given these values, the SPs are solved to optimality. Based on the results of the SPs, the MP is amended by adding *optimality* or *feasibility* cuts. This process continues until the optimal solution is proven to be found. Unlike BD, the SP in LBBD can take any form. The *optimality* and *feasibility* cuts are problem-specific and devised using the inference dual of the SP.

The LBBD has succeeded in various combinatorial optimization problems, especially when the MP is an assignment or a planning problem, and the SP is a scheduling problem [12]. It is the first time it is being used to tackle the CRSP.

There are two ways to implement the LBBD. In the standard one, the MP is repeatedly solved to optimality in the iterative process. In the other one, known as *branch-and-check* (BAC), the MP is solved only once by *branch-and-cut*. The SPs are solved for each feasible solution to the MP to check whether to add cuts in a callback way [20].

Regarding techniques to improve the LBBD, the MP is usually tightened by incorporating the relaxation of the SP [20]. This incorporation is regarded as a remedy for losing the SP-related information in the MP. Codato and Fischetti [10] attempt to enhance the *feasibility* cut by finding a minimal infeasible set of the MP variables. Before adding *optimality* cuts, Angulo et al. [2] add subgradient Benders cuts in BD to the MP. Some problem-specific cuts constructed using the lower bounds on the SPs can also work well [11, 12]. A complex MP or SP can be decomposed again, which results in a multi-level LBBD where each level is easy to solve [26, 22].

3 Problem Definition

Consider a total of $d = \sum_{b\in B}\sum_{m\in M}\sum_{j\in J} d_{bmj}$ cars to be produced in a one-hour shift, where d_{bmj} is the demand for the car model with body $b \in B = \{1, 2, ..., \bar{b}\}$, color $m \in M = \{1, 2, ..., \bar{m}\}$, and configuration $j \in J = \{1, 2, ..., \bar{j}\}$.

All cars are first arranged in an upstream sequence to be processed successively in the body shop and paint shop. The two shops aim to minimize the number of body and color changes, respectively. Meanwhile, the color batch limit requires the number of consecutive cars with the same color not to exceed $\bar{c}$. Upon leaving the paint shop, this upstream car sequence is readjusted by the PBS.

The PBS comprises $\bar{l}$ lanes, each lane $l \in L = \{1, 2, ..., \bar{l}\}$ with a maximum capacity of $\bar{q}$. As Figure 1 shows, the transfer car at the PBS entrance (exit) carries one car from the paint shop (PBS) to one lane of the PBS (the assembly shop) each time. The two transfer cars operate independently, i.e., when one car enters the PBS, another can leave the PBS at the same time. Cars in the same lane move forward via a conveyor belt. Whenever a car enters the PBS, the number of cars stored in each lane cannot exceed $\bar{q}$. Using the PBS, each car $i \in V = \{1, 2, ..., d\}$ is moved to a new position $p \in P = \{1, 2, ..., d\}$ in the assembly shop, which aims to keep all options consumed steadily.

The goal of the CRSP is to minimize the total cost of the body shop, paint shop, and assembly shop, whose objectives are weighted as α, β, and γ, respectively. Decisions include the body, color, configuration, lane of the PBS, and downstream position assigned to each car.

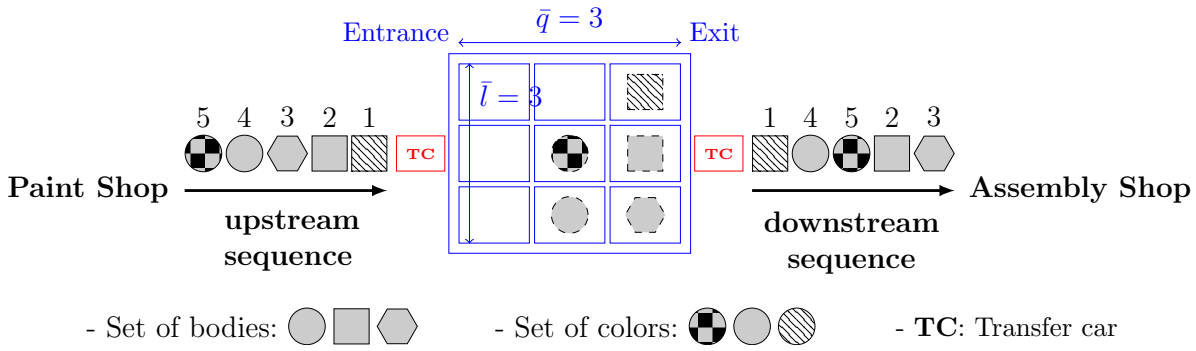


Figure 1: Illustration of a PBS with $\bar{l} = 3$ lanes and capacity $\bar{q} = 3$ of each lane.

We provide a complete formulation (CF) to model the CRSP, where most variables are binary. The total cost depends on the upstream and downstream sequences. It follows that we apply the standard LBBD to decompose the CF into an MP that determines the upstream sequence and an SP that determines the downstream sequence and assigns cars to lanes. Besides, the car-to-lane assignment decisions do not directly affect the cost of the assembly shop. We thus decompose the SP again, leading to a three-level LBBD, i.e., the NLBBD presented below. The CF and the standard LBBD can be found in our online supplement.

4 A Nested Logic-Based Benders Decomposition

In our NLBBD method, the first and the second levels are responsible for determining the upstream and the downstream sequences, respectively. The third level involves assigning cars to lanes. This section presents the formulation for each level, along with a lower bound, three valid inequalities, and a heuristic method as technical enhancements.

4.1 First Level: Determine the Upstream Sequence

The upstream sequence is defined by the body and color of each car. Decisions on the two attributes and the associated costs can be expressed by the following decision variables:

- $x^i_{bm} = 1$ if car at upstream position i has body b and color m, 0 otherwise, $i \in V, b \in B, m \in M$;
- $f_{i,i+1} = 1$ if cars at upstream positions i and $i+1$ have different bodies, 0 otherwise, $i \in V \setminus \{d\}$;
- $g_{i,i+1} = 1$ if cars at upstream positions i and $i+1$ have different colors, 0 otherwise, $i \in V \setminus \{d\}$;
- Θ represents the cost of the assembly shop, $\Theta \geqslant 0, \Theta \in \mathbb{R}$.

The upstream sequence is then determined by solving the first-level formulation $\mathcal{L}_1$:

$$(\mathcal{L}_1) \quad \min \quad \alpha \cdot \sum_{i \in V \setminus \{d\}} f_{i,i+1} + \beta \cdot \sum_{i \in V \setminus \{d\}} g_{i,i+1} + \Theta \tag{1}$$

$$s.t. \quad \sum_{b \in B} \sum_{m \in M} x^i_{bm} = 1, \qquad i \in V, \tag{2}$$

$$\sum_{i \in V} x^i_{bm} = \sum_{j \in J} d_{bmj}, \qquad b \in B, m \in M, \tag{3}$$

$$\sum_{i=p}^{p+\bar{c}} \sum_{b \in B} x^i_{bm} \leqslant \bar{c}, \qquad p \in \{1, ..., d - \bar{c}\}, m \in M, \tag{4}$$

$$f_{i,i+1} \geqslant \sum_{m \in M} x^i_{bm} - \sum_{m \in M} x^{i+1}_{bm}, \qquad i \in V \setminus \{d\}, b \in B, \tag{5}$$

$$g_{i,i+1} \geqslant \sum_{b \in B} x^i_{bm} - \sum_{b \in B} x^{i+1}_{bm}, \qquad i \in V \setminus \{d\}, m \in M, \tag{6}$$

$$[\textit{Optimality} \text{ Cuts}], \tag{7}$$

$$\Theta \geqslant \Delta, \tag{8}$$

$$x^i_{bm} \in \{0, 1\}, \qquad i \in V, b \in B, m \in M, \tag{9}$$

$$f_{i,i+1}, g_{i,i+1}, \Theta \in \mathbb{R}, \qquad i \in V \setminus \{d\}. \tag{10}$$

The objective function (1) minimizes the total cost of the three shops. Constraints (2) and (3) ensure that each car gets only one body and one color, and that each body-color demand is met. Constraints (4) represent the color batch limit. Constraints (5) and (6) define the variables used to indicate body and color changes. Since $\mathcal{L}_1$ is a relaxation of the CRSP, variable Θ may underestimate the actual cost of the assembly shop. If it does, one *optimality* cut in constraints (7) is added to $\mathcal{L}_1$. The cut links the actual cost with the $\mathcal{L}_1$ variables, and we introduce it in Section 4.2 after obtaining the actual cost from the second level. By progressively incorporating these cuts, Θ approximates and eventually equals the actual cost. We tighten $\mathcal{L}_1$ by a bounding constraint (8). Δ is a lower bound on the cost of the assembly shop. We compute it in Section 4.3 by a relaxation of the second level.

4.2 Second Level: Determine the Downstream Sequence

After solving $\mathcal{L}_1$, we denote its optimal solution as $\hat{x}^i_{bm}$. All $\hat{x}^i_{bm} = 1$ indicate the optimal upstream sequence $\mathcal{S}^u$. By assigning each car a new position and a configuration, $\mathcal{S}^u$ is altered in the second level to a downstream sequence $\mathcal{S}^d$ that minimizes absolute deviations between the actual and ideal consumption of all options. Assembling configuration j requires r^w_j of option $w \in W = \{1, 2, ..., \bar{w}\}$. The ideal consumption ratio of option w is $\sigma_w = \sum_{j \in J} r^w_j \cdot \left(\sum_{b \in B} \sum_{m \in M} d_{bmj}\right)/d$ [21]. Let binary variable $y^i_{pj} = 1$ if the car at upstream position i requires configuration j and goes to downstream position p. The second-level formulation $\mathcal{L}_2$ to find $\mathcal{S}^d$ is written as:

$$(\mathcal{L}_2) \quad F_2(\hat{x}^i_{bm}) = \min \ \gamma \cdot \sum_{p \in P} \sum_{w \in W} \left| \sum_{j \in J} r^w_j \cdot \left(\sum_{p'=1}^{p} \sum_{i \in V} y^i_{p'j} \right) - p \cdot \sigma_w \right| \tag{11}$$

$$
\begin{aligned}
s.t. \quad & \sum_{j\in J}\sum_{p\in P} y^i_{pj} = 1, && i \in V, && (12)\\
& \sum_{i\in V}\sum_{j\in J} y^i_{pj} = 1, && p \in P, && (13)\\
& \sum_{i\in \mathcal{T}_{bm}}\sum_{p\in P} y^i_{pj} = d_{bmj}, && b \in B, m \in M, j \in J, && (14)\\
& [\textit{Feasibility}\ \text{Cuts}], && && (15)\\
& y^i_{pj} \in \{0,1\}, && i \in V, p \in P, j \in J. && (16)
\end{aligned}
$$

The objective function (11) minimizes the sum of absolute deviations of the actual options consumed from the ideal options consumed and can be easily linearized. Constraints (12) ensure that each car is assigned only one configuration and one downstream position. Constraints (13) limit each downstream position to be occupied by a unique car. Each set $\mathcal{T}_{bm}$ in constraints (14) contains all cars with body b and color m, i.e., $\mathcal{T}_{bm} = \{i \in V | \hat{x}^i_{bm} = 1\}$. Constraints (14) guarantee that the number of cars with configuration j in $\mathcal{T}_{bm}$ exactly matches the demand d_{bmj}. After obtaining a solution to $\mathcal{L}_2$, i.e., a downstream sequence, we need to find a car-to-lane assignment to achieve the sequence change. However, this change may violate the PBS capacity or FIFO rules. As a result, one *feasibility* cut in constraints (15) can be added to $\mathcal{L}_2$ to remove this downstream sequence. We introduce this cut in Section 4.4.

If $d > \bar{l}\times\bar{q}$, then, each car $i \in \{(\bar{l}-1)\times\bar{q}+2, \ldots, d\}$ cannot move to any downstream position between 1 and $p_i - 1$, where $p_i = i - (\bar{l}-1)\times\bar{q}$. We can hence set the related y^i_{pj} variables to 0 before solving $\mathcal{L}_2$, that is, $y^i_{1j} = y^i_{2j} = \cdots = y^i_{p_i-1,j} = 0$, $j \in J$, $i \in \{(\bar{l}-1)\times\bar{q}+2, \ldots, d\}$, $p_i = i - (\bar{l}-1)\times\bar{q}$.

Given $\hat{x}^i_{bm}$, the minimum cost $F_2(\hat{x}^i_{bm})$ of the assembly shop is obtained by solving $\mathcal{L}_2$. We now use it to define the *optimality* cut in constraints (7). As stated previously, when $F_2(\hat{x}^i_{bm})$ is underestimated by the value of Θ, i.e., $\hat{\Theta} < F_2(\hat{x}^i_{bm})$, we add the following *optimality* cut to $\mathcal{L}_1$:

$$
\textit{Optimality}\ \text{Cut}: \qquad \Theta \geqslant F_2(\hat{x}^i_{bm}) \cdot \left(\sum_{(i,b,m)\in\mathcal{X}} x^i_{bm} - |\mathcal{X}| + 1 \right) \tag{17}
$$

where $\mathcal{X} = \{(i,b,m) | \hat{x}^i_{bm} = 1, i \in V, b \in B, m \in M\}$. Given $\hat{x}^i_{bm}$, this cut provides Θ with the exact cost of the assembly shop. Once the same $\hat{x}^i_{bm}$ is encountered again in a subsequent iteration, cut (17) makes Θ equal to $F_2(\hat{x}^i_{bm})$, and the global optimal solution is proven to be found.

To reduce the cuts (17) added, we use the bounding constraint (8) in $\mathcal{L}_1$ to raise the value of Θ to a lower bound Δ on $\mathcal{L}_2$. In the next section, we show how to get Δ by solving a relaxation $\mathcal{R}$ of $\mathcal{L}_2$.

4.3 Computing A Lower Bound on $\mathcal{L}_2$

The relaxation $\mathcal{R}$ is a *level scheduling* problem. It has the same objective as $\mathcal{L}_2$, but omits the body shop, paint shop, and PBS. The decision to be made is which configuration of the car should be placed at each downstream position. Let integer variable ξ^j_p be the number of cars

with configuration j from downstream position 1 to p, $\mathcal{R}$ is formulated as:

$$\mathcal{R}: \quad \Delta = \min \quad \gamma \cdot \sum_{p\in P}\sum_{w\in W}\left|\sum_{j\in J} r_j^w \cdot \xi_p^j - p\cdot\sigma_w\right| \tag{18}$$

$$s.t. \quad \sum_{j\in J}\xi_p^j = p, \qquad p\in P, \tag{19}$$

$$\xi_p^j \leqslant \xi_{p+1}^j \leqslant \xi_p^j + 1, \qquad p\in P\backslash\{d\}, j\in J, \tag{20}$$

$$\xi_d^j = \sum_{b\in B}\sum_{m\in M} d_{bmj}, \qquad j\in J, \tag{21}$$

$$\xi_p^j \in \mathbb{Z},\ 0\leqslant \xi_p^j \leqslant \max_{j\in J}\left\{\sum_{b\in B}\sum_{m\in M} d_{bmj}\right\}, \qquad p\in P, j\in J. \tag{22}$$

The objective function (18) minimizes absolute deviations between the actual and ideal consumptions of all options. Constraints (19) ensure that the number of cars to be assembled from position 1 to p is exactly p. Constraints (20) indicate that the number of cars with each configuration from position p to $p+1$ either remains unchanged or increases by one. The demand for each configuration is guaranteed by constraints (21). In $\mathcal{R}$, the difference $\xi_p^j - \xi_{p-1}^j = 1$ indicates that the car at position p has configuration j. Although $\mathcal{R}$ is a MIP, our tests on all instances show that it is easy to obtain its optimal solution within 0.2 seconds.

4.4 Third Level: Determine the Car-to-Lane Assignment

In the third level, each car is assigned to one lane of the PBS, while respecting the capacity constraints and FIFO rules. We denote a feasible solution of $\mathcal{L}_2$ as $\hat{y}_{pj}^i$. The downstream sequence is given by set $\mathcal{Y} = \{(i,p)|\sum_{j\in J}\hat{y}_{pj}^i = 1, i\in V, p\in P\}$, where pair (i,p) indicates that car i is moved to downstream position p. We then define the concept of two (i,p) pairs that conflict.

Definition 4.1. Pair (i_1,p_1) is in conflict with pair (i_2,p_2) if and only if $i_1 < i_2$, $p_1 > p_2$ or $i_1 > i_2$, $p_1 < p_2$, for $i_1, i_2 \in V, i_1 \neq i_2, p_1, p_2 \in P, p_1 \neq p_2$.

In accordance with the FIFO rules, cars i_1 and i_2 cannot be assigned to the same lane of the PBS if pair (i_1,p_1) conflicts with pair (i_2,p_2). This allows us to construct the FIFO constraints using a set $\mathcal{I} = \{(i_1,i_2)|(i_1,p_1)$ conflicts with $(i_2,p_2),\ (i_1,p_1),(i_2,p_2)\in\mathcal{Y}\}$ defined for the downstream sequence $\mathcal{Y}$. Let binary variable $h_l^i = 1$ if car i is assigned to lane l, the car-to-lane assignment decisions are made through the following feasibility problem:

$$(\mathcal{L}_3) \quad \sum_{l\in L} h_l^i = 1, \qquad i\in V, \tag{23}$$

$$h_l^{i_1} + h_l^{i_2} \leqslant 1, \qquad l\in L, (i_1,i_2)\in\mathcal{I}, \tag{24}$$

$$\sum_{k=1}^{i} \rho_k^i \cdot h_l^k \leqslant \bar{q}, \qquad i\in V, l\in L, \tag{25}$$

$$h_l^i \in \{0,1\}, \qquad i\in V, l\in L. \tag{26}$$

In $\mathcal{L}_3$, the binary parameter $\rho_k^i = 1$ indicates that when car i enters the PBS, car k remains stored in the PBS. The values of all ρ_k^i, $i, k \in V, k \leqslant i$ can be derived from $\hat{y}_{pj}^i$, whose calculation procedure is provided in our online supplement. Constraints (23) ensure that each car enters exactly one lane. Constraints (24) require cars to comply with the FIFO rules. When car i enters lane l, one constraint in (25) ensures that the number of cars still in lane l cannot exceed capacity $\bar{q}$.

As mentioned before, $\hat{y}_{pj}^i$ is obtained without regard to PBS capacity and FIFO rules. When its $\mathcal{Y}$ is infeasible for $\mathcal{L}_3$, the following *feasibility* cut is added to $\mathcal{L}_2$ as one of the constraints (15):

$$\textit{Feasibility Cut}: \quad \sum_{(i,p)\in\mathcal{Y}} \sum_{j\in J} y_{pj}^i \leqslant |\mathcal{Y}| - 1 \tag{27}$$

We sum y_{pj}^i over on J for each (i,p) in $\mathcal{Y}$ to form the left-hand side of the cut (27). We can do this because the feasibility of $\mathcal{L}_3$ is independent of the decisions on configuration. This cut only eliminates one $\mathcal{Y}$ that makes $\mathcal{L}_3$ infeasible. Next, we propose three valid inequalities to guide the second-level search toward a feasible solution of $\mathcal{L}_2$ that makes $\mathcal{L}_3$ as feasible as possible.

4.5 Valid Inequalities

The three valid inequalities serve to eliminate the feasible solutions of $\mathcal{L}_2$, whose downstream sequences make $\mathcal{L}_3$ infeasible. The first two are capacity inequalities used to remove the solutions that require more spaces than the available PBS capacity. The third, named the clique inequality, targets the solutions that require more lanes than the total lanes of the PBS.

4.5.1 Capacity Inequality I

We introduce the set $\mathcal{K}_k = \{(i_1,p_1),(i_2,p_2),...,(i_k,p_k)\}$, without loss of generality, $i_1 < i_2 < ... < i_k \in V$ and $p_1 > p_2 > ... > p_k \in P$, to represent that k distinct lanes are used by these k cars. Then, for each pair in $\mathcal{Y}^k = \{(i,p)|(i,p) \in \mathcal{Y}, i < i_k\}$, we check if it conflicts with all pairs in $\mathcal{K}_k$ to construct a set $\mathcal{V}_k = \big\{(i,p)|(i,p) \in \mathcal{Y}^k, (i,p) \text{ conflicts with all pairs in } \mathcal{K}_k\big\}$.

Clearly, all cars whose (i,p) pairs are in $\mathcal{V}_k$ should be stored in the other $\bar{l} - k$ lanes ($\bar{l} > k$). When the number of these cars is greater than the remaining capacity of the PBS, i.e., $|\mathcal{V}_k| > (\bar{l} - k) \times \bar{q}$, we add the first capacity inequality (28) to the search tree of $\mathcal{L}_2$:

$$\text{Capacity Inequality I}: \quad \sum_{(i,p)\in\mathcal{K}_k\cup\mathcal{V}_k} \sum_{j\in J} y_{pj}^i \leqslant |\mathcal{K}_k| + |\mathcal{V}_k| - 1 \tag{28}$$

It is possible to check if cut (28) for each $k \in \{2,...,\bar{l}-1\}$ is violated. In this work, we only check for $k = 2$, as preliminary tests show that checking for $k > 2$ takes more computational time.

4.5.2 Capacity Inequality II

For each pair (i_k,p_k) in $\mathcal{Y}$, there are two groups of cars that cannot enter the same lane as car i_k. The first consists of cars with smaller upstream indexes than i_k and bigger downstream positions than p_k. In contrast, each car in the second group has a bigger upstream index than i_k,

but a smaller downstream position than p_k. We divide all pairs in $V \times P = \{(i,p)|i \in V, p \in P\}$ into two parts to represent the two groups, then get the following sets:

$$\mathcal{Y}_1 = \{(i,p) \in V \times P | i < i_k, p > p_k\}, \quad \overline{\mathcal{Y}}_1 = \{(i,p)|(i,p) \in \mathcal{Y}_1 \cap \mathcal{Y}\} \tag{29}$$

$$\mathcal{Y}_2 = \{(i,p) \in V \times P | i > i_k, p < p_k\}, \quad \overline{\mathcal{Y}}_2 = \{(i,p)|(i,p) \in \mathcal{Y}_2 \cap \mathcal{Y}\} \tag{30}$$

If $|\overline{\mathcal{Y}}_2| > 0$, then at least one lane is set aside for the second group of cars to leave the PBS first. Except for the other lane assigned to car i_k, at most $(\bar{l}-2) \times \bar{q}$ spaces are available to store all cars in the first group. When the number of cars in the first group is greater than the available capacity, i.e., $|\overline{\mathcal{Y}}_2| > 0$ and $|\overline{\mathcal{Y}}_1| > (\bar{l}-2) \times \bar{q}$, we add the second capacity inequality (31):

$$\text{Capacity Inequality II}: \quad \sum_{j \in J} y^{i_k}_{p_k,j} + \sum_{(i,p) \in \mathcal{Y}_1} \sum_{j \in J} y^i_{pj} + \sum_{(i,p) \in \mathcal{Y}_2} \sum_{j \in J} y^i_{pj} \leqslant |\overline{\mathcal{Y}}_1| + |\overline{\mathcal{Y}}_2| \tag{31}$$

4.5.3 Clique Inequality

We first transform a dowsntream sequence $\mathcal{Y}$ into an undirected graph $G_{\mathcal{Y}} = (\mathcal{V}_{\mathcal{Y}}, \mathcal{E}_{\mathcal{Y}})$. Each (i,p) pair in $\mathcal{Y}$ is considered as a vertex of $G_{\mathcal{Y}}$. An edge is created between two vertices if they conflict. As a result, two cars cannot enter the same lane if their (i,p) pairs are two endpoints of an edge of $G_{\mathcal{Y}}$.

Then, all cars whose (i,p) pairs form a *clique* $\mathcal{C}$ of $G_{\mathcal{Y}}$ must enter distinct lanes. This is because a *clique* is a subset of vertices of an undirected graph, where every pair of vertices is connected. Therefore, at least $|\mathcal{C}|$ lanes are needed to rearrange all cars whose (i,p) pairs belong to $\mathcal{C}$ to their respective downstream positions. If there exists a *clique* $\mathcal{C}$ whose size is greater than the total number of lanes, i.e., $|\mathcal{C}| > \bar{l}$, we generate and add a clique inequality (32) as follows:

$$\text{Clique Inequality}: \quad \sum_{(i,p) \in \mathcal{C}} \sum_{j \in J} y^i_{pj} \leqslant \bar{l} \tag{32}$$

In this work, we only find the *maximal cliques* for a $\mathcal{Y}$ using the *Bron-Kerbosch* algorithm with pivoting. The reason is that a *maximal clique* is a *clique* that cannot be extended into a larger one by adding an additional vertex, thus, its inequality (32) is the strongest. The online supplement provides the illustration of the $G_{\mathcal{Y}}$ built for a $\mathcal{Y}$ with ten cars and its *maximal cliques* found.

4.6 A Heuristic Solution for $\mathcal{L}_2$ and $\mathcal{L}_3$

Once $\mathcal{R}$ from Section 4.3 is solved, we get the configuration sequence that incurs the lowest cost to the assembly shop. We denote this sequence as $\mathcal{S}^c = \{c_1, ..., c_p, ..., c_d\}$, where c_p is the configuration of the car placed at downstream position p. In a specific iteration, if $\hat{\Theta}$ of $\mathcal{L}_1$ equals Δ, it is possible to construct a heuristic solution for $\mathcal{L}_2$ and $\mathcal{L}_3$ before directly solving them.

Let $\mathcal{S}^u = \{(1,b_1,m_1), ..., (i,b_i,m_i), .., (d,b_d,m_d)\}$ denote an upstream sequence obtained from the first level, where (i,b_i,m_i) indicates that car i requires body b_i and color m_i. Guided

by $\mathcal{S}^c$, we first attempt to alter $\mathcal{S}^u$ to a downstream sequence for $\mathcal{L}_2$.

- *Step 1.* Include all cars that have body b and color m in set $\mathcal{O}_{bm} = \{i|(i, b_i, m_i) \in \mathcal{S}^u, i \in V, b_i = b, m_i = m\}$, and sort them in ascending order of the car index, $b \in B, m \in M$;
- *Step 2.* For each configuration $j \in J$, count all body-color combinations with positive demands to define a map $\mathcal{Q}_j = \{(b, m) : \mathcal{O}_{bm}|d_{bmj} > 0, b \in B, m \in M\}$;
- *Step 3.* Take configuration c_p from $\mathcal{S}^c$. Select the car with the lowest index i from all $\mathcal{O}_{bm}$ sets of map $\mathcal{Q}_{c_p}$. Place car i at position p and assemble it in configuration c_p, i.e., set $\hat{y}^i_{p,c_p} = 1$. Then, delete car i from all $\mathcal{O}_{bm}$ sets of map $\mathcal{Q}_j$, $j \in J$. Repeat for p from 1 to d.

The reason for choosing the car with the lowest index is to reduce resequencing effort. Once *Step 3* is finished, we obtain a downstream sequence $\mathcal{Y}$ that makes the assembly shop cost Δ. Then, we continue *Step 4* to verify the feasibility of $\mathcal{Y}$.

- *Step 4.* Check if $\mathcal{Y}$ violates any valid inequality. If it does, $\mathcal{Y}$ is infeasible for $\mathcal{L}_3$, and ***stop***. Otherwise, try *Step 5* to construct a heuristic car-to-lane assignment for $\mathcal{L}_3$.
- *Step 5.* Assign car i to the lane with the lowest index l, ensuring the cars already assigned to it do not conflict with car i. Set $\hat{h}^i_l = 1$. Repeat for i from 1 to d.
- *Step 6.* Check if $\mathcal{Y}$ and $\hat{h}^i_l$ respect the capacity constraints (25). If so, they are the optimal solution of $\mathcal{L}_2$ and $\mathcal{L}_3$, and ***stop***. If not, go to *Step 7*.
- *Step 7.* Solve $\mathcal{L}_3$ using $\mathcal{Y}$. If $\mathcal{L}_3$ is feasible, we get the optimal solution of $\mathcal{L}_2$ and $\mathcal{L}_3$. Otherwise, the above steps fail to construct a heuristic solution for $\mathcal{L}_2$ and $\mathcal{L}_3$.

We illustrate our heuristic method with an example in the online supplement.

5 First-Level Reformulations

In this section, we model the determination of the upstream sequence on two kinds of acyclic-directed graphs and propose two shortest-path problem reformulations $\mathcal{M}_1$ and $\mathcal{M}_2$ for $\mathcal{L}_1$ in order to solve larger instances.

5.1 A Shortest-Path Problem Reformulation $\mathcal{M}_1$

We start by building the first graph $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{A}_1)$, where $\mathcal{L}_1$ is reformulated as a classical shortest-path model $\mathcal{M}_1$. In $\mathcal{G}_1$, each node is labeled as $(b, m \,|\, \eta \,|\, a^1_1, ..., a^1_{\bar{m}}, ..., a^{b'}_{m'}, ..., a^{\bar{b}}_1, ..., a^{\bar{b}}_{\bar{m}})$, where $a^1_1 + ... + a^1_{\bar{m}} + ... + a^{b'}_{m'} + ... + a^{\bar{b}}_1 + ... + a^{\bar{b}}_{\bar{m}} = p$. It represents the decision to produce a car with body b and color m at upstream position p. In its label, $a^{b'}_{m'}$ is the number of cars with body b' and color m' from upstream position 1 to p. η denotes the number of consecutive cars with color m from the last color change position to the current position p. The use of $a^{b'}_{m'}$ and η allows the demand and the color batch limit constraints to be respected in the building

process of $\mathcal{G}_1$. Any node with the label in which $\eta > \bar{c}$ or $a^{b'}_{m'} > \sum_{j \in J} d_{b'm'j}$ is infeasible and is not created.

The new nodes are spawned by their parent nodes. For a node n_1 with label $(b, m \,|\, \eta \,|\, a^1_1, ..., a^{\bar{b}}_{\bar{m}})$, we try to attach it to a car with body $b^* \in B$ and color $m^* \in M$ to create a potential child node n_2 labeled $(b^*, m^* \,|\, \eta' \,|\, a^{1'}_1, ..., a^{\bar{b}'}_{\bar{m}})$. The value of $a^{b^*}_{m^*}$ of n_1 is incremented by 1 for n_2. If $a^{b^*}_{m^*} + 1$ does not exceed the demand $\sum_{j \in J} d_{b^* m^* j}$, we continue to update η for n_2. If the two cars represented by n_1 and n_2 have the same color, i.e., $m = m^*$, we set $\eta' = \eta + 1$. For $m \neq m^*$, η' is renewed to 1. If η' respects the color batch limit, i.e., $\eta' \leqslant \bar{c}$, node n_2 is created.

We introduce a source node n_s with label $(0, 0 \,|\, 0 \,|\, 0, ..., 0)$ as the first node to start the creation process. For each node, at most $\bar{b} \times \bar{m}$ potential child nodes can be created, where each one represents a car with a certain body-color combination. Once a node with a label in which $a^1_1 + ... + a^{b'}_{m'} + ... + a^{\bar{b}}_{\bar{m}} = d$ is created, it becomes a terminal node because all demands are met at this point. Regarding the labels of terminal nodes, the values of $a^1_1, ..., a^{b'}_{m'}, ..., a^{\bar{b}}_{\bar{m}}$ equal the demand for each car model, respectively. Hence, we can merge all the terminal nodes into a single terminal node n_t with label $(\sum_{j \in J} d_{11j}, ..., \sum_{j \in J} d_{b'm'j}, ..., \sum_{j \in J} d_{\bar{b}\bar{m}j})$.

The arcs of $\mathcal{G}_1$ are created along with the creation of new nodes. When node n_2 is spawned by node n_1, we create an arc $(n_1, n_2) \in \mathcal{A}_1$ to connect them. This arc is associated with a cost c_{n_1,n_2}. If n_2 is created by extending a car with the same body and color as the car represented by n_1, or n_1 is the source node, $c_{n_1,n_2} = 0$. Extending a car with only a different body or only a different color makes $c_{n_1,n_2} = \alpha$ and $c_{n_1,n_2} = \beta$, respectively, while changing both body and color makes $c_{n_1,n_2} = \alpha + \beta$. Figure 2 illustrates the $\mathcal{G}_1$ built for an example with four cars.

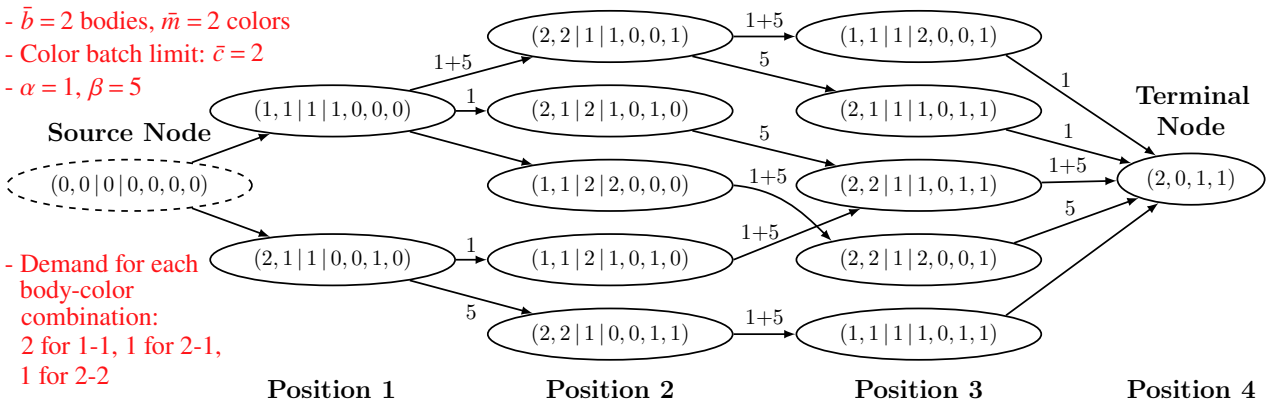


Figure 2: Illustration of the First Acyclic-Directed Graph $\mathcal{G}_1$.

In $\mathcal{G}_1$, each path from the source node to the terminal node corresponds to a feasible solution of $\mathcal{L}_1$, as any node that violates the demand or the color batch limit constraint is not created. Among such paths, the kth shortest one is equivalent to the optimal upstream sequence in the kth iteration. As a result, solving the master problem (MP) $\mathcal{L}_1$ repeatedly, as done in the standard LBBD, can be treated as a k-shortest path problem (kSPP), which aims to get the top-k shortest paths from a source to a terminal in a digraph [1]. If the MP solution in a specific iteration does not reach the stopping criteria, we start a new iteration to update the MP decision by finding the next shortest path instead of adding cuts and solving the MP again. The new path found is the next best MP solution. Even if its cost is the same or higher than

the current MP solution, it may help to reduce the optimal objective value of the subproblem. The implementation of the NLBBD with $\mathcal{G}_1$ is detailed in Section 6. We do not provide model $\mathcal{M}_1$ because we do not solve it.

In Section 7.2, it is shown that the size of $\mathcal{G}_1$ surges as the number of cars increases. Consequently, we attempt to compress $\mathcal{G}_1$ by simplifying the labels of its nodes. The resulting aggregated graph $\mathcal{G}_2$ and the second shortest-path problem reformulation $\mathcal{M}_2$ for $\mathcal{L}_1$ are presented in what follows.

5.2 An Aggregated Shortest-Path Problem Reformulation $\mathcal{M}_2$

In the aggregated graph $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{A}_2)$, the label of each node is simplified to $(p \,|\, b, m \,|\, \eta)$. It signifies placing a car with body b and color m at upstream position p. η still counts the number of consecutive cars with color m from the last color change position to the current position p.

The construction of $\mathcal{G}_2$ is similar to that of $\mathcal{G}_1$. We start with a source node n_s labeled $(0 \,|\, 0, 0 \,|\, 0)$. For each node $n_1 \in \mathcal{N}_2$ with label $(p \,|\, b, m \,|\, \eta)$, we try to create a child node n_2 by placing a car with body $b^* \in B$ and color $m^* \in M$ at position $p+1$. If m and m^* are the same color and $\eta + 1 \leqslant \bar{c}$, n_2 with label $(p+1 \,|\, b^*, m^* \,|\, \eta+1)$ is created. If $m \neq m^*$, n_2 is also created but labeled as $(p+1 \,|\, b^*, m^* \,|\, 1)$. Once n_2 is created, it is linked to n_1 using an arc $(n_1, n_2) \in \mathcal{A}_2$. The arc cost c_{n_1,n_2} of $\mathcal{G}_2$ is computed in the same way as of $\mathcal{G}_1$. When a node with the label in which $p = d$ is created, it becomes a terminal node. We denote $\mathcal{N}_t$ as the set of all the terminal nodes of $\mathcal{G}_2$.

Likewise, each path in $\mathcal{G}_2$ from the source node to one terminal node maps an upstream sequence. Such a path is feasible for $\mathcal{L}_1$ only if it meets the production demands. Therefore, with $\mathcal{G}_2$, we reformulate $\mathcal{L}_1$ as a shortest-path model $\mathcal{M}_2$ restricted by demand constraints. Let binary variable $x_{n_1,n_2} = 1$ denote that arc $(n_1, n_2) \in \mathcal{A}_2$ is selected. $\mathcal{M}_2$ initially consists of an objective function that minimizes the sum of the path cost and variable Θ, as well as the flow conservation and demand constraints. We denote the optimal solution of $\mathcal{M}_2$ as $\hat{x}_{n_1,n_2}$ and $\hat{\Theta}$. When $\hat{\Theta}$ underestimates the cost $F_2(\hat{x}_{n_1,n_2})$ of the assembly shop, we add an *optimality* cut (33), where $\mathcal{X}' = \{(n_1, n_2) \in \mathcal{A}_2 | \hat{x}_{n_1,n_2} = 1\}$, to $\mathcal{M}_2$ and begin a new iteration.

$$\textit{Optimality Cut}: \qquad \Theta \geqslant F_2(\hat{x}_{n_1,n_2}) \cdot \left(\sum_{(n_1,n_2) \in \mathcal{X}'} x_{n_1,n_2} - \left| \mathcal{X}' \right| + 1 \right) \tag{33}$$

The online supplement provides the formulation of $\mathcal{M}_2$ and the $\mathcal{G}_2$ built for the example in Figure 2. With simplified labels, $\mathcal{G}_2$ can be smaller and take less time to build. However, $\mathcal{M}_2$ is a MIP and requires more time to be solved than directly finding one shortest path in $\mathcal{G}_1$. The comparisons between $\mathcal{M}_1$ and $\mathcal{M}_2$ and between $\mathcal{G}_1$ and $\mathcal{G}_2$ are presented in Section 7.2.

6 Overall Implementation

This section explains how we implement the NLBBD method with $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. The algorithmic flowchart for each can be found in the online supplement. Both implementations start by solving $\mathcal{R}$ to obtain the lower bound Δ and the best configuration sequence $\mathcal{S}^c$. After $\mathcal{G}_1$

or $\mathcal{G}_2$ is built, the iterative process begins. In the kth iteration, we obtain the optimal upstream sequence $\mathcal{S}_k^u$ from the first level. Then, we use $\mathcal{S}_k^u$ to get the optimal downstream sequence $\mathbf{y}_k$ along with the cost $F_2(\mathcal{S}_k^u)$ of the assembly shop, and a feasible car-to-lane assignment $\mathbf{h}_k$. We first execute the heuristic method to get $\mathbf{y}_k$ and $\mathbf{h}_k$. If the heuristic method succeeds, then $F_2(\mathcal{S}_k^u) = \Delta$, and there is no need to solve $\mathcal{L}_2$. If not, we solve $\mathcal{L}_2$. For each incumbent integer solution of $\mathcal{L}_2$, we check if it violates any valid inequality. If so, it gets pruned. Otherwise, we solve $\mathcal{L}_3$. If $\mathcal{L}_3$ is infeasible, a *feasibility* cut (27) is added to the $\mathcal{L}_2$ search tree via a callback routine. After obtaining $F_2(\mathcal{S}_k^u)$, we stop or proceed to the next iteration according to the stopping criteria.

Specifically, using $\mathcal{G}_1$, in the kth iteration, we find the kth shortest path p_k from the source to the terminal by using the kSPP algorithm of Al Zoobi et al. [1] to obtain $\mathcal{S}_k^u$. The cost of p_k is $F_1(p_k)$, or equivalently, $F_1(\mathcal{S}_k^u)$. It is equal to the body shop and paint shop costs incurred by $\mathcal{S}_k^u$. Then, we use the heuristic method to find solution $\mathbf{y}_k$ and $\mathbf{h}_k$ of the second and third levels. If the method fails to identify feasible $\mathbf{y}_k$ and $\mathbf{h}_k$, we solve $\mathcal{L}_2$ to compute $F_2(\mathcal{S}_k^u)$. Let UB be the global upper bound initialized to a big constant MAX. If UB$> F_1(\mathcal{S}_k^u) + F_2(\mathcal{S}_k^u)$, we set UB$= F_1(\mathcal{S}_k^u)+F_2(\mathcal{S}_k^u)$ and replace the current best solution $(p^*, \mathcal{S}^{u*}, \mathbf{y}^*, \mathbf{h}^*)$ with $(p_k, \mathcal{S}_k^u, \mathbf{y}_k, \mathbf{h}_k)$. If the stopping criterion UB $\leqslant F_1(\mathcal{S}_k^u) + \Delta$ or $F_2(\mathcal{S}_k^u) = \Delta$ is met, we stop and return the global optimal solution $(p^*, \mathcal{S}^{u*}, \mathbf{y}^*, \mathbf{h}^*)$. Otherwise, we continue to the next iteration.

Next, we demonstrate the correctness of the stopping criteria using the following proposition.

Proposition 1 *If UB $\leqslant F_1(\mathcal{S}_k^u) + \Delta$ or $F_2(\mathcal{S}_k^u) = \Delta$, where $\mathcal{S}_k^u$ is the first-level solution in the kth iteration, then the NLBBD procedure can be stopped, and the optimal solution is $(p^*, \mathcal{S}^{u*}, \mathbf{y}^*, \mathbf{h}^*)$.*

Proof. First, we prove the condition UB $\leqslant F_1(\mathcal{S}_k^u) + \Delta$. Let UB be the best objective value and $(p^*, \mathcal{S}^{u*}, \mathbf{y}^*, \mathbf{h}^*)$ be the best solution seen so far. Since $F_1(\mathcal{S}_k^u)$ is non-decreasing as k increases, we have $F_1(\mathcal{S}_k^u) \leqslant F_1(\mathcal{S}_{k'}^u)$ for $k' > k$. We know $F_1(\mathcal{S}_{k'}^u) + \Delta \leqslant F_1(\mathcal{S}_{k'}^u) + F_2(\mathcal{S}_{k'}^u)$ as Δ is the lower bound on $F_2(\mathcal{S}_k^u)$ for any $k \geqslant 1$. After the kth iteration, if UB $\leqslant F_1(\mathcal{S}_k^u) + \Delta$, then UB$= F_1(\mathcal{S}^{u*}) + F_2(\mathcal{S}^{u*}) \leqslant F_1(\mathcal{S}_k^u) + \Delta \leqslant F_1(\mathcal{S}_{k'}^u) + \Delta \leqslant F_1(\mathcal{S}_{k'}^u) + F_2(\mathcal{S}_{k'}^u)$ holds. It means that in the subsequent iterations, there will be no $\mathcal{S}_{k'}^u$ with a smaller objective value than UB, so we can stop.

Second, we prove the condition $F_2(\mathcal{S}_k^u) = \Delta$. If $F_2(\mathcal{S}_k^u) = \Delta$, then we have $F_1(\mathcal{S}^{u*}) + F_2(\mathcal{S}^{u*}) \leqslant F_1(\mathcal{S}_k^u) + F_2(\mathcal{S}_k^u) = F_1(\mathcal{S}_k^u) + \Delta \leqslant F_1(\mathcal{S}_{k'}^u) + \Delta \leqslant F_1(\mathcal{S}_{k'}^u) + F_2(\mathcal{S}_{k'}^u)$ for any $k' > k$. It also means that after the kth iteration, there is no $\mathcal{S}_{k'}^u$ whose objective value is strictly smaller than $F_1(\mathcal{S}^{u*}) + F_2(\mathcal{S}^{u*})$, thus we can stop. □

As a side note, in our implementation, we avoid building $\mathcal{G}_1$ completely in the first iteration. Instead, we build the set of nodes only and add an additional label that is used to compute the first shortest path p_1. This way allows us to save a significant amount of memory.

In contrast, the NLBBD using $\mathcal{G}_2$ follows the standard LBBD scheme. By solving $\mathcal{M}_2$, $\mathcal{S}_k^u$ is derived from all $\hat{x}_{n_1,n_2} = 1$ of the kth iteration. If $\hat{\Theta}_k > \Delta$, it is unnecessary to use the heuristic method for $\mathcal{S}_k^u$, instead we directly solve $\mathcal{L}_2$. Once $F_2(\mathcal{S}_k^u)$ is obtained, we check if $\hat{\Theta}_k < F_2(\mathcal{S}_k^u)$. If so, the decision $\mathcal{S}_k^u$ is modified by adding an *optimality* cut (33) to $\mathcal{M}_2$ and solving $\mathcal{M}_2$ again. Otherwise, if $\hat{\Theta}_k = F_2(\mathcal{S}_k^u)$, we stop and get the optimal solution $(\hat{\mathbf{x}}_k, \mathcal{S}_k^u, \mathbf{y}_k, \mathbf{h}_k)$.

7 Computational Results

This section presents a computational analysis of our approaches and enhancements. Section 7.1 introduces all the instances used. Section 7.2 compares the performance of the NLBBD with that of standard LBBD and Gurobi. Section 7.3 provides managerial insights from a sensitivity analysis. We coded all algorithms in C++ and solved all MIPs with Gurobi 10.0.3. All tests were performed within one hour and on a single 2.40 GHz AMD Rome 7532 processor with 220 GB memory. All instances, codes, and detailed computational results are available at https://sites.google.com/view/xinyiguo.

7.1 Instance Generation

Our experiments use two datasets: a basic one for algorithm comparison and an extended one for sensitivity analysis. The basic dataset consists of 30 real-world instances provided by a Chinese carmaker, who works eight one-hour shifts daily to produce $d = 50$ cars per shift, and 330 artificial instances, of which every 30 are generated for one d in $\{10, 20, 30, 40, 60, 70, 80, 90, 100, 110, 120\}$. In the 30 real-world instances, there are $\bar{b} = 2$ bodies, $\bar{m} = 3$ colors, $\bar{j} = 3$ configurations, and $\bar{w} = 10$ options. The demand distribution of each attribute and the options required by each configuration are given in the online supplement. Following these distributions, we generate the production demand quotas for all artificial instances, which is also described in the online supplement.

Spray nozzle cleaning is the most costly process. Referring to the objective-weight ratio $10^6 : 1$ between the paint shop and the assembly shop in [27], we assign weights to the three shops as $\alpha = 1$, $\beta = 10000$, and $\gamma = 100$. The color batch limit $\bar{c}$ is 5 for $d \leqslant 30$ and 10 for $d \geqslant 40$. Finally, the number of lanes ($\bar{l}$) and the capacity per lane ($\bar{q}$) are $\bar{l} = 3$, $\bar{q} = 3$ for $d \in \{10, 20, 30\}$, $\bar{l} = 5$, $\bar{q} = 5$ for $d \in \{40, 50, 60\}$, and $\bar{l} = 6$, $\bar{q} = 6$ for $d \in \{70, 80, 90, 100, 110, 120\}$.

The extended dataset consists of several B-M-J sub-datasets. Each sub-dataset contains 150 artificial instances, with every 30 instances corresponding to one $d \in \{10, 15, 20, 25, 30\}$. All instances in the sub-dataset B-M-J have B bodies, M colors, and J configurations. For each car, its body, color, and configuration are uniformly generated from their respective sets B, M, and J. For each configuration in J, the binary demand for each of the ten options is generated with uniform distribution. We fix the PBS size as 3×3 and keep three weights unchanged as in the basic dataset.

7.2 Algorithm Comparison

All tests in this section are performed on the basic dataset. We first evaluate the three valid inequalities. Then, we compare the performance of our approaches with standard LBBD and Gurobi in solving the CRSP. Finally, we present the difference in the size between $\mathcal{G}_1$ and $\mathcal{G}_2$.

We first compare the efficiency of the three valid inequalities used to solve $\mathcal{L}_2$ in Table 1. For this, we ran our NLBBD with $\mathcal{G}_1$ on small and medium instances, without using the heuristic method. Firstly, all three valid inequalities were allowed to be added. Then, we tested the impact of removing each one by not checking the condition for adding it.

We see that using all three valid inequalities ("cut1-2-3") performs best, as it solves the most instances. The clique inequality (32) is the strongest, while the first capacity inequality

Table 1: Performance of the Three Valid Inequalities Used to Solve $\mathcal{L}_2$.

d	#Instances Solved				Avg. Computational Time (sec.)				#Feasibility Cut Added			
	cut1-2	cut1-3	cut2-3	cut1-2-3	cut1-2	cut1-3	cut2-3	cut1-2-3	cut1-2	cut1-3	cut2-3	cut1-2-3
10	30	30	30	30	0.1	0.1	0.1	0.1	4.0	0.2	**0.1**	**0.1**
20	30	30	30	30	3.3	1.2	**0.7**	1.0	66.1	3.4	**2.7**	4.0
30	30	30	30	30	71.1	11.4	**7.6**	8.1	555.7	20.6	17.2	**11.6**
40	5	**30**	29	**30**	1,118.1	235.7	**90.4**	156.3	2,121.6	119.6	**63.3**	67.0
50*	1	27	**28**	**28**	1,455.9	1,081.8	566.0	**173.2**	1,222.0	164.8	185.9	**68.3**
60	0	11	19	**30**		1,379.4	1,274.8	**270.9**	—	194.3	213.1	**72.8**
70	0	0	10	**23**		$> 3,600$	2,040.5	**848.7**	—	—	**33.3**	63.5
Sum	96	158	176	**201**								

(28) is the weakest. Compared with "cut1-2-3", removing (28), i.e., "cut2-3", leads to the loss of 25 optimal solutions, while removing (32), i.e., "cut1-2", 105 solutions are lost. Meanwhile, using the second capacity inequality (31) together with (32) yields more promising feasible solutions because fewer feasibility cuts are added. Thus, the three valid inequalities are kept in the rest of our tests.

Table 2 compares the performance of all the algorithms we used to solve the CRSP. We ran the NLBBD with $\mathcal{L}_1$ ("NL") and the two first-level reformulations $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, to solve all basic instances. The use of the heuristics before solving $\mathcal{L}_2$ is denoted by "H". For comparison, we also solved a complete formulation using Gurobi ("CF") and standard LBBD ("SL"). As described in Section 3, SL is a two-level decomposition method. Its MP is the same as $\mathcal{L}_1$, but its SP is a MIP used to determine the downstream sequence and car-to-lane assignment simultaneously.

Table 2: Comparison of all Algorithms.

d	#Instances Solved (#Ins. Exceed Memory)								Avg. Total Computational Time (sec.)							
	CF	SL	NL	$\mathcal{M}_1$	$\mathcal{M}_2$	NL+H	$\mathcal{M}_1$+H	$\mathcal{M}_2$+H	CF	SL	NL	$\mathcal{M}_1$	$\mathcal{M}_2$	NL+H	$\mathcal{M}_1$+H	$\mathcal{M}_2$+H
10	30	30	30	30	30	30	30	30	25.3	2.6	0.4	0.1	0.3	0.4	**0.0**	0.2
20	10	29	30	30	30	30	30	30	2,789.4	221.1	5.0	1.0	4.3	4.5	**0.3**	3.3
30	0	26	**30**	**30**	28	**30**	**30**	28		695.7	130.2	8.1	33.5	76.6	**1.3**	26.5
40	0	3	29	**30**	28	**30**	**30**	28		1,905.8	312.7	156.3	300.2	146.4	**94.7**	213.0
50*	0	0	23	28	24	28	**30**	29			596.3	173.2	1,195.5	293.0	**53.6**	582.9
60	0	0	23	**30**	13	24	**30**	27			445.0	270.9	1,275.1	307.3	**22.0**	114.0
70	0	0	15	23	7	20	**29**	17			1,270.3	848.7	1,674.9	316.9	**121.5**	375.4
80	0	0	18	17	1	24	**29**	20			1,614.7	1,671.0	3,112.2	586.1	**347.2**	954.2
90	0	0	9	15 (1)	1	22	**30**	18	$> 3,600$		2,215.8	1,917.3	2,194.4	694.5	**534.8**	850.4
100	0	0	3	8 (3)	0	15	**28** (1)	10			2,352.4	2,474.1		1,409.7	1,165.2	**576.9**
110	0	0	1	2 (6)	0	17	**23** (6)	12			2,518.6	1,904.8	$> 3,600$	1,363.5	1,732.8	**1,080.8**
120	0	0	0	2 (15)	0	13	**17** (13)	14			$> 3,600$	2,484.5		1,870.6	1,352.0	**1,109.6**
Sum	40	88	211	245	162	283	**336**	263								

The results in Table 2 highlight the performance of our NLBBD approaches. SL performs close to CF because its SP is intractable. In comparison, NL, $\mathcal{M}_1$, and $\mathcal{M}_2$ consume less time and solve the instances with 120 cars, which is far better than 18 cars in Wu et al. [27]. Although NL and $\mathcal{M}_2$ are more efficient than SL, they cannot solve all real-world instances. In contrast, $\mathcal{M}_1$ outperforms significantly. It solves many more instances but requires more computing resources. The numbers in the parentheses show how many instances are not solved

due to the memory limit. Meanwhile, the heuristic method succeeds in constructing the optimal solution for $\mathcal{L}_2$ and $\mathcal{L}_3$. With it, more instances with up to 336 are solved, and less time is consumed.

For the small and medium instances that are solved, Table 3 reports the total time taken by different methods based on the LBBD to find upstream sequences ("Avg. Time. $\mathcal{S}^u$") and to obtain downstream sequences along with car-to-lane assignments ("Avg. Time. $\mathcal{S}^d$ & $\mathbf{h}$"), respectively. The columns of "#Iter" compare the average number of iterations required for these methods.

Table 3: Comparison of all LBBD-based Methods.

d	Avg. Time. $\mathcal{S}^u$ (sec.)				Avg. Time. $\mathcal{S}^d$ & $\mathbf{h}$ (sec.)							#Iter			
	SL	NL	$\mathcal{M}_1$	$\mathcal{M}_2$	SL	NL	$\mathcal{M}_1$	$\mathcal{M}_2$	NL+H	$\mathcal{M}_1$+H	$\mathcal{M}_2$+H	SL	NL	$\mathcal{M}_1$	$\mathcal{M}_2$
10	0.5	0.3	**0.0**	0.2	2.1	0.1	0.1	0.1	0.0	0.0	0.0	1.7	1	1	1
20	29.1	4.0	**0.0**	2.9	192.0	0.9	0.9	1.4	0.3	0.3	0.4	6.9	1.1	1.0	1
30	54.8	124.3	**0.3**	28.2	640.8	5.8	7.7	5.3	0.4	1.0	0.5	2.5	1	1.0	1
40	26.1	108.9	**30.0**	178.6	1,879.6	203.8	124.4	121.5	42.8	60.4	0.0	1	2.0	1.4	1
50*	523.2	386.0	**8.8**	499.0	—	210.2	160.0	696.4	26.7	42.5	0.2	—	1.3	1.4	1
60	306.3	253.2	**18.6**	159.1	—	191.8	251.8	1,116.0	10.2	3.0	0.0	—	1	1.1	1
70	337.6	346.6	**68.3**	241.8	—	923.7	755.2	1,433.0	0.2	49.8	0.2	—	1.1	1.3	1

The four columns under "Avg. Time. $\mathcal{S}^u$" show that $\mathcal{M}_1$ and $\mathcal{M}_2$ outperform SL and NL to find the optimal upstream sequence. Especially, $\mathcal{M}_1$ shows a remarkable advantage. This is because running the kSPP algorithm is faster than solving MIPs. The columns for "Avg. Time. $\mathcal{S}^d$ & $\mathbf{h}$" comparisons show that decomposing the SP of SL into $\mathcal{L}_2$ and $\mathcal{L}_3$ is another boost to save time. With NL, $\mathcal{M}_1$, and $\mathcal{M}_2$, the average reduction in the time to find the optimal downstream sequence and a feasible car-to-lane assignment is from 95.4% to 92.0% as the number of cars increases from 10 to 40. When incorporating the heuristics in $\mathcal{M}_2$, this time is reduced even to 0. The comparison in "#Iter" columns shows that the $\mathcal{S}^u$ obtained by solving $\mathcal{M}_2$ seems to be of high quality. $\mathcal{M}_2$ requires only one iteration, whereas other methods need more, or do not completely solve the instances in the first iteration ("—").

Table 4: Comparison of $\mathcal{G}_1$ and $\mathcal{G}_2$.

d	#Nodes		#Arcs		Time.Build (sec.)	
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_1$	$\mathcal{G}_2$
10	1,335.0	192.8	3,769.1	806.6	0.0	0.0
20	28,487.5	460.0	102,404.1	2,099.1	0.0	0.0
30	375,454.1	789.7	1,632,756.8	4,057.6	0.6	0.0
40	2,432,088.5	1,686.3	11,760,209.8	9,276.9	5.1	0.0
50*	6,936,812.1	2,125.1	34,204,292.7	11,629.9	16.2	0.0
60	15,014,016.1	1,736.2	73,682,858.3	9,184.2	37.1	0.0
70	41,075,423.6	3,058.1	210,771,549.5	16,948.4	348.6	0.0

Table 4 gives details on the size of $\mathcal{G}_1$ and $\mathcal{G}_2$. As the number of cars increases, the size of $\mathcal{G}_1$, i.e., the number of nodes ("#Nodes") and arcs ("#Arcs"), soars significantly. In comparison, $\mathcal{G}_2$ is much smaller and is built in 0.1s, which may be helpful if the computing resources are limited.

7.3 Sensitivity Analysis and Managerial Insights

In this section, we change the demand pattern and the PBS size to analyze their impact on the CRSP and the NLBBD approach so as to derive managerial insights. All experiments were performed on the extended dataset using the NLBBD based on $\mathcal{G}_1$ with the heuristic method.

Impact of Demand Pattern

We performed experiments on sub-datasets B-3-3, 2-M-3, and 2-3-J, where B increased from 2 to 5, M from 3 to 6, and J from 3 to 6, and regard the results of sub-dataset 2-3-3 as the baseline to analyze the effects of changing the number of bodies ($\bar{b}$), colors ($\bar{m}$), and configurations ($\bar{j}$).

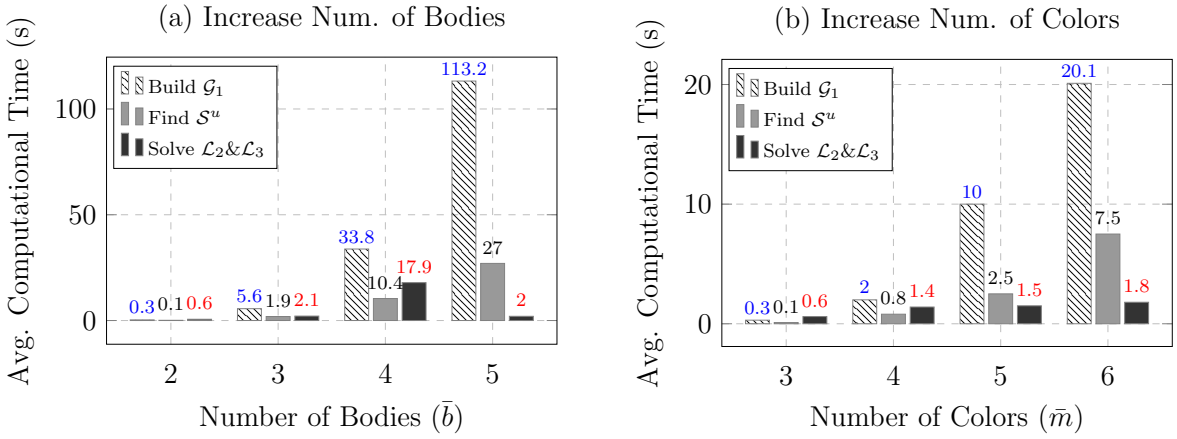


Figure 3: The Average Time Spent on Each Part as the Number of Bodies and the Number of Colors Increase.

Figure 3 shows that the total computational time increases as $\bar{b}$ and $\bar{m}$ increases. More bodies and colors enlarge $\mathcal{G}_1$, and more effort is required to find the upstream sequence. Initially, the most time-consuming part is to solve $\mathcal{L}_2$ and $\mathcal{L}_3$, accounting for an average of 60%. When $\bar{b}$ is 3 ($\bar{m}$ is 4), building $\mathcal{G}_1$ takes over by 58.3% (47.6%) and proceeds to consume the most time. Moreover, an increase in $\bar{b}$ causes more difficulties than in $\bar{m}$. This is because, with the color batch limit, the extension of $\mathcal{G}_1$ resulting from increasing $\bar{m}$ is less than that resulting from increasing $\bar{b}$.

For the case of increasing $\bar{j}$, since $\bar{j}$ is independent of $\mathcal{G}_1$, we only present its effect on solving $\mathcal{L}_2$ and $\mathcal{L}_3$. Figure 4 depicts the computational time taken to solve $\mathcal{L}_2$ and $\mathcal{L}_3$, as well as the number of iterations ("#Iter") between the two levels and the first level. These results are averaged over the number of cars and the number of configurations. According to the sharp increase in time and the more iterations, increasing either $\bar{j}$ or d makes $\mathcal{L}_2$ and $\mathcal{L}_3$ harder to solve.

Impact of PBS Size

The experiments to analyze the effects of changing the PBS size, including the number of lanes ($\bar{l}$) and the capacity of each lane ($\bar{q}$), were performed on sub-dataset 2-3-3.

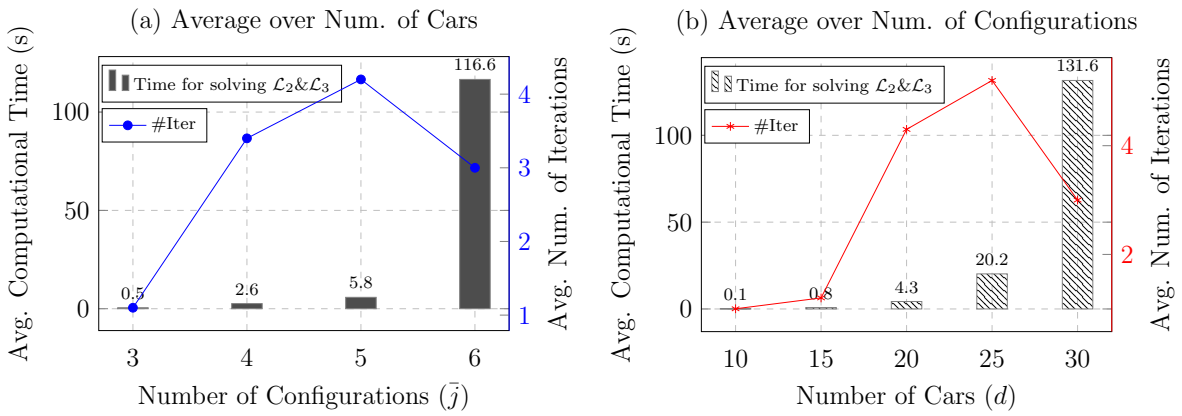


Figure 4: The Average Time to Solve $\mathcal{L}_2$ and $\mathcal{L}_3$ and the Average Number of Iterations Between the Two Levels and the First Level when Increasing the Number of Configurations.

We first analyze the total cost saved by our integrated approach over the sequential decision method. The former jointly minimizes the cost of the three shops. In the latter, we first determine the sequence that minimizes the cost of the two upstream shops. This sequence is then readjusted to a downstream one that minimizes the cost of the assembly shop. The total cost is the sum of these two costs, which is essentially that of the first iteration in the NLBBD. Figure 5(a) depicts the cost differences between the two methods as the PBS size varies. We see that the cost savings are \$569.9 for $\bar{l} = 2$, $\bar{q} = 1$, \$219.5 for $\bar{l} = 2$, $\bar{q} = 2$, and \$215.4 for $\bar{l} = 3$, $\bar{q} = 1$. The smaller the PBS is, the more cost our integrated method saves. It is more profitable to invest in more lanes than capacity.

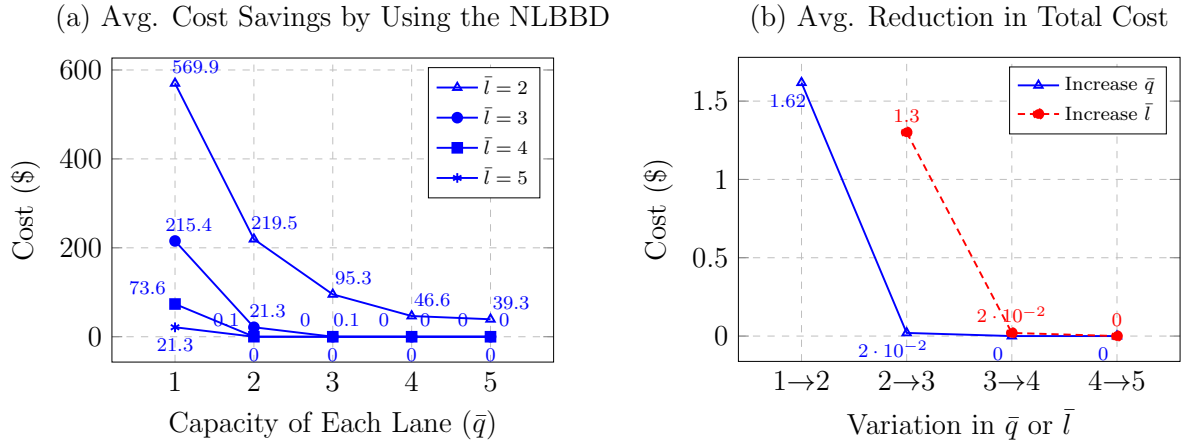


Figure 5: The Average Cost Savings by the NLBBD Approach over the Sequential Decision Method and the Average Reduction in Total Cost as the PBS Size is Changed.

Figure 5(b) shows the effect of varying $\bar{l}$ and $\bar{q}$ on the reduction in total cost, where "1→2" represents $\bar{l}$ (or $\bar{q}$) is increased from 1 to 2. We see that enlarging the PBS has a negligible effect. This is because, with the NLBBD, even a small PBS can achieve comparable low cost as a large PBS. Once reaching a certain size, the CRSP can be solved as two independent

sequencing problems.

Table 5: Detailed Computational Results of Changing the Number of Lanes ($\bar{l}$) and Per-Lane Capacity ($\bar{q}$).

	Δ-F_2. Gap (%)					Avg. Time (s)					#Iter					#Opt. H Sol.			
$\bar{q}\setminus\bar{l}$	2	3	4	5	$\bar{q}\setminus\bar{l}$	2	3	4	5	$\bar{q}\setminus\bar{l}$	2	3	4	5	$\bar{q}\setminus\bar{l}$	2	3	4	5
1	0.33	0	0	0	1	58.6	1.7	1.3	1.3	1	272.0	6.4	4.9	4.3	1	10	13	19	21
2	0	0	0	0	2	2.0	2.9	0.1	0.0	2	6.7	4.3	1	1	2	12	20	26	29
3	0	0	0	0	3	2.6	0.1	0.0	0.0	3	5.0	1	1	1	3	16	24	29	30
4	0	0	0	0	4	3.5	0.1	0.0	0.0	4	4.7	1	1	1	4	16	28	30	30
5	0	0	0	0	5	0.2	0.1	0.0	0.0	5	1.2	1	1	1	5	16	29	30	30

Table 5 details the effect of enlarging the PBS on our NLBBD approach. We present the average computational results for instances with 10, 15, and 20 cars. In the "Δ-F_2. Gap (%)" columns, each value is the gap between the lower bound Δ and the optimal objective value of $\mathcal{L}_2$. The minor gaps suggest that the Δ provided by $\mathcal{R}$ is tight. The results in the "Avg. Time (s)" columns contradict the existing conclusion that a larger PBS leads to increased complexity and time [25]. With our approach, the computational time is significantly reduced as the PBS is enlarged. Two trends can explain this reduction: the decrease in the number of iterations ("#Iter"), and the increase in the number of instances solved by our heuristic method ("#Opt. H Sol."). With the extension of the PBS, two sequences of cars can be determined in stages. For managers, a profitable PBS should have a minimum size ($\bar{l} \times \bar{q}$) that can make a hold of "#Iter"=1.

8 Conclusion

This paper addresses a car resequencing problem, where a sequence of cars is constructed for a body shop, a paint shop, and an assembly shop. To minimize the total cost, the upstream sequence in the first two shops is readjusted to a downstream one for the last shop via a painted body storage (PBS) buffer. We decompose this problem into three levels and solve it with a nested LBBD method. The upstream sequence is constructed in the first level. Then, it is altered in the second level to the downstream sequence. In the third level, we seek a car-to-lane assignment that respects the sequence change. To enhance our method, we propose a lower bound, three valid inequalities, a heuristic method, and two shortest-path problem reformulations for the first-level sequencing problem: one is based on a graph ($\mathcal{G}_1$) with no infeasible solutions, and the other is based on an aggregated graph ($\mathcal{G}_2$) and restricted by demand constraints. The first reformulation transforms the LBBD iterative procedure into solving a k-shortest path problem (kSPP), while the second is solved repeatedly.

References

[1] A. Al Zoobi, D. Coudert, and N. Nisse. Finding the k shortest simple paths: Time and space trade-offs. *ACM Journal of Experimental Algorithmics*, 28(1):1–23, 2023.

[2] G. Angulo, S. Ahmed, and S. S. Dey. Improving the integer l-shaped method. *INFORMS Journal on Computing*, 28(3):483–499, 2016.

[3] Z. Atan, T. Ahmadi, C. Stegehuis, T. de Kok, and I. Adan. Assemble-to-order systems: A review. *European Journal of Operational Research*, 261(3):866–879, 2017.

[4] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math*, 4(1):238–252, 1962.

[5] T. Benoist. Soft car sequencing with colors: Lower bounds and optimality proofs. *European Journal of Operational Research*, 191(3):957–971, 2008.

[6] N. Boysen and M. Zenker. A decomposition approach for the car resequencing problem with selectivity banks. *Computers & operations research*, 40(1):98–108, 2013.

[7] N. Boysen, M. Fliedner, and A. Scholl. Level scheduling of mixed-model assembly lines under storage constraints. *International Journal of Production Research*, 47(10):2669–2684, 2009.

[8] N. Boysen, M. Fliedner, and A. Scholl. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349–373, 2009.

[9] N. Boysen, A. Scholl, and N. Wopperer. Resequencing of mixed-model assembly lines: Survey and research agenda. *European Journal of Operational Research*, 216(3):594–604, 2012.

[10] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[11] J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial benders' cuts for the strip packing problem. *Operations Research*, 62(3):643–661, 2014.

[12] Ö. Elçi and J. Hooker. Stochastic planning and scheduling with logic-based benders decomposition. *INFORMS Journal on Computing*, 34(5):2428–2442, 2022.

[13] T. Epping, W. Hochstättler, and P. Oertel. Complexity results on a paint shop problem. *Discrete Applied Mathematics*, 136(2-3):217–226, 2004.

[14] U. Golle, F. Rothlauf, and N. Boysen. Car sequencing versus mixed-model sequencing: A computational study. *European Journal of Operational Research*, 237(1):50–61, 2014.

[15] U. Golle, F. Rothlauf, and N. Boysen. Iterative beam search for car sequencing. *Annals of Operations Research*, 226:239–254, 2015.

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

[17] J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[18] E. Jahren and R. A. Achá. A column generation approach and new bounds for the car sequencing problem. *Annals of Operations Research*, 264:193–211, 2018.

[19] S.-S. Ko, Y.-H. Han, and J. Y. Choi. Paint batching problem on m-to-1 conveyor systems. *Computers & Operations Research*, 74:118–126, 2016.

[20] K. P. Martínez, Y. Adulyasak, and R. Jans. Logic-based benders decomposition for integrated process configuration and production planning problems. *INFORMS Journal on Computing*, 34(4):2177–2191, 2022.

[21] J. Miltenburg. Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science*, 35(2):192–207, 1989.

[22] A. Riise, C. Mannino, and L. Lamorgese. Recursive logic-based benders' decomposition for multi-mode outpatient scheduling. *European Journal of Operational Research*, 255(3): 719–728, 2016.

[23] C. Solnon, A. Nguyen, C. Artigues, et al. 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, 2008.

[24] Y. Sun, S. Esler, D. Thiruvady, A. T. Ernst, X. Li, and K. Morgan. Instance space analysis for the car sequencing problem. *Annals of Operations Research*, pages 1–29, 2022.

[25] F. Taube and S. Minner. Resequencing mixed-model assembly lines with restoration to customer orders. *Omega*, 78:99–111, 2018.

[26] D. Wheatley, F. Gzara, and E. Jewkes. Logic-based benders decomposition for an inventory-location problem with service constraints. *Omega*, 55:10–23, 2015.

[27] J. Wu, Y. Ding, and L. Shi. Mathematical modeling and heuristic approaches for a multi-stage car sequencing problem. *Computers & Industrial Engineering*, 152:107008, 2021.

Supplement

1 Complete Formulation

The decision variables of the complete formulation are as follows:

- x^i_{bmj} equals 1 if car i has body b, color m and configuration j, 0 otherwise, $i \in V, b \in B, m \in M, j \in J$;
- $f_{i,i+1}$ equals 1 if car i and $i+1$ have different bodies, 0 otherwise, $i \in V \setminus \{d\}$;
- $g_{i,i+1}$ equals 1 if car i and $i+1$ have different colors, 0 otherwise, $i \in V \setminus \{d\}$;
- y^i_p equals 1 if car i goes to downstream position p in the assembly shop, 0 otherwise, $i \in V$, $p \in P$;
- h^i_l equals 1 if car i is assigned to lane l, 0 otherwise, $i \in V, l \in L$;
- z^i_{vl} equals 1 if car i enters the PBS and car v simultaneously leaves the PBS from lane l, 0 otherwise, $i, v \in V, v \leqslant i, l \in L$;
- u_{pw} is the absolute difference between the actual and ideal consumption of option w from downstream position 1 to p, $u_{pw} \geqslant 0, p \in P, w \in W$;

Then, the complete formulation of the CRSP is written as:

$$\min \quad \alpha \cdot \sum_{i \in V \setminus \{d\}} f_{i,i+1} + \beta \cdot \sum_{i \in V \setminus \{d\}} g_{i,i+1} + \gamma \cdot \sum_{p \in P} \sum_{w \in W} u_{pw} \tag{1}$$

$$s.t. \quad \sum_{b \in B} \sum_{m \in M} \sum_{j \in J} x^i_{bmj} = 1, \qquad i \in V, \tag{2}$$

$$\sum_{i \in V} x^i_{bmj} = d_{bmj}, \qquad b \in B, m \in M, j \in J, \tag{3}$$

$$\sum_{i=p}^{p+\bar{c}} \sum_{b \in B} \sum_{j \in J} x^i_{bmj} \leqslant \bar{c}, \qquad p \in \{1, ..., d-\bar{c}\}, m \in M, \tag{4}$$

$$f_{i,i+1} \geqslant \sum_{m \in M} \sum_{j \in J} x^i_{bmj} - \sum_{m \in M} \sum_{j \in J} x^{i+1}_{bmj}, \qquad i \in V \setminus \{d\}, b \in B, \tag{5}$$

$$g_{i,i+1} \geqslant \sum_{b \in B} \sum_{j \in J} x^i_{bmj} - \sum_{b \in B} \sum_{j \in J} x^{i+1}_{bmj}, \qquad i \in V \setminus \{d\}, m \in M, \tag{6}$$

$$\sum_{l \in L} h^i_l = 1, \qquad i \in V, \tag{7}$$

$$\sum_{p \in P} y^i_p = 1, \qquad i \in V, \tag{8}$$

$$\sum_{i \in V} y^i_p = 1, \qquad p \in P, \tag{9}$$

$$\sum_{p=p_1}^{d} y^{i_1}_p + \sum_{p=1}^{p_1} y^{i_2}_p + h^{i_1}_l + h^{i_2}_l \leqslant 3, \qquad \begin{array}{l} i_1 \in V \setminus \{d\}, p_1 \in P, \\ i_2 \in \{i_1+1, ..., d\}, l \in L, \end{array} \tag{10}$$

$$h_l^k + h_l^v + \sum_{l' \in L} z_{vl'}^i - 2 \leqslant \sum_{i'=k}^{i-1} \sum_{l' \in L} z_{kl'}^{i'}, \qquad \begin{array}{l} v \in V \setminus \{1\}, l \in L, \\ k \in \{1, 2, ..., v-1\}, \\ i \in \{v, v+1, ..., d\}, \end{array} \tag{11}$$

$$\sum_{v=1}^{i} \sum_{l \in L} z_{vl}^i \leqslant 1, \qquad i \in V, \tag{12}$$

$$\sum_{i=v}^{d} \sum_{l \in L} z_{vl}^i \leqslant 1, \qquad v \in V, \tag{13}$$

$$\sum_{p \in P} y_p^i \cdot p \geqslant \left(1 - \sum_{l \in L} z_{il}^i\right) \cdot i + 1, \qquad i \in V, \tag{14}$$

$$\sum_{p \in P} y_p^i \cdot p \geqslant 1 + \sum_{i'=1}^{i-1} \sum_{v=1}^{i'} \sum_{l \in L} z_{vl}^{i'} - d \cdot \left(1 - \sum_{l \in L} z_{il}^i\right), \qquad i \in V \setminus \{1\}, \tag{15}$$

$$\sum_{p \in P} y_p^i \cdot p \leqslant 1 + \sum_{i'=1}^{i-1} \sum_{v=1}^{i'} \sum_{l \in L} z_{vl}^{i'} + d \cdot \left(1 - \sum_{l \in L} z_{il}^i\right), \qquad i \in V \setminus \{1\}, \tag{16}$$

$$\sum_{p \in P} y_p^i \cdot p - \sum_{p \in P} y_p^v \cdot p \geqslant 1 - d \cdot \left(1 - \sum_{l \in L} z_{vl}^i\right), \qquad \begin{array}{l} i \in V \setminus \{1\}, \\ v \in \{1, 2, .., i-1\}, \end{array} \tag{17}$$

$$\sum_{i'=1}^{i} h_l^{i'} - \sum_{i'=1}^{i} \sum_{v=1}^{i'} z_{vl}^{i'} \leqslant \bar{q}, \qquad i \in V, l \in L, \tag{18}$$

$$\sum_{i=v}^{d} z_{vl}^i \leqslant h_l^v, \qquad v \in V, l \in L, \tag{19}$$

$$u_{pw} \geqslant \sum_{j \in J} r_j^w \cdot \left(\sum_{p'=1}^{p} \sum_{i \in V} \sum_{b \in B} \sum_{m \in M} x_{bmj}^i \cdot y_{p'}^i\right) - p \cdot \sigma_w, \quad p \in P, w \in W, \tag{20}$$

$$u_{pw} \geqslant p \cdot \sigma_w - \sum_{j \in J} r_j^w \cdot \left(\sum_{p'=1}^{p} \sum_{i \in V} \sum_{b \in B} \sum_{m \in M} x_{bmj}^i \cdot y_{p'}^i\right), \quad p \in P, w \in W, \tag{21}$$

$$x_{bmj}^i, h_l^i, y_p^i, z_{vl}^i \in \{0, 1\}, \qquad \begin{array}{l} i, v \in V, v \leqslant i, b \in B, \\ m \in M, j \in J, p \in P, l \in L, \end{array} \tag{22}$$

$$f_{i,i+1}, g_{i,i+1}, u_{pw} \in \mathbb{R}, \qquad i \in V \setminus \{d\}, p \in P, w \in W. \tag{23}$$

The objective function (1) minimizes the total cost associated with body and color changes, as well as the absolute differences between the actual and ideal consumption for all options. Constraints (2) and (3) ensure that each car is processed to exactly one car model and that the demand for each model is met. Constraints (4) work as the color batch limit. Constraints (5) and (6) define the variables used to indicate whether there is a change in body and color between two consecutive cars, respectively. Constraints (7), (8), and (9) ensure that each upstream car is uniquely positioned in the downstream sequence via one lane, and that each downstream position is occupied by one car. The adherence to the FIFO rules for cars assigned to the same

lane is guaranteed by constraints (10) and (11). For (10), if car i_2 leaves the PBS before car i_1 ($i_1 < i_2$), they cannot be assigned to the same lane. For (11), if car k and car v ($k < v$) are assigned to the same lane, car k has already entered the assembly shop when car v leaves the PBS. Constraints (12) (Constraints (13)) ensure that at any time, at most one car can enter (leave) the PBS. Constraints (14), (15), and (16) state that if car i does not leave the PBS immediately upon its arrival, its downstream position must be greater than i. Otherwise, its downstream position should be one plus the number of cars in the assembly shop. Constraints (17) ensure that if car v leaves the PBS as car i enters, the downstream position of car v must be smaller than that of car i. Constraints (18) limit the number of cars in each lane within the capacity. Constraints (19) guarantee that if car v is not assigned to lane l, it cannot leave the PBS via lane l when any other car enters the PBS. The absolute difference of option w consumed from downstream position 1 to p is linearized by constraints (20) and (21). Finally, Constraints (22) and (23) restrict the value range of each decision variable.

2 Standard Logic-based Benders Decomposition

Because of the PBS, there are two car sequences in the CRSP. It is possible to decompose the CRSP into two levels: one is a *master problem* (MP) for the upstream-sequence decision, and the other is a *subproblem* (SP) for the remaining decisions. Since all decision variables except u_{pw} are binary, we apply the logic-based Benders decomposition (LBBD) to decompose the above complete formulation.

The upstream sequence is determined in the MP by assigning each car one body and one color. We transform the original four-index variable x^i_{bmj} into a three-index one x^i_{bm}. Let $x^i_{bm} = 1$ if the car at upstream position i has body b and color m, 0 otherwise. Then, the MP has exactly the same formulation as the $\mathcal{L}_1$ of the nested LBBD method.

Given the optimal solution $\hat{\mathbf{x}}$ to the MP, the decisions to be made in the SP are the configuration and downstream position of each car and the car-to-lane assignment. We retain variables h^i_l, z^i_{vl}, and u_{pw} of the complete formulation in the SP. Let binary variable $y^i_{pj} = 1$ if car i requires configuration j and goes to downstream position p. The objective function of the SP is $F_{SP}(\hat{\mathbf{x}}) = \min_{(\mathbf{y},\mathbf{h},\mathbf{z})}\{\gamma \cdot \sum_{p\in P}\sum_{w\in W} u_{pw}\}$. The constraints of the SP consist of constraints (7)–(21) and the value-range constraints of the decision variables. The only changes are that term $\sum_{j\in J} y^i_{pj}$ replaces the original variable y^i_p, and constraints (24), (25), and (26) replace constraints (3), (20), and (21), respectively, where $\mathcal{T}_{bm} = \{i \in V | \hat{x}^i_{bm} = 1\}$.

$$\sum_{i\in\mathcal{T}_{bm}}\sum_{p\in P} y^i_{pj} = d_{bmj}, \qquad b \in B, m \in M, j \in J, \tag{24}$$

$$u_{pw} \geqslant \sum_{j\in J} r^w_j \cdot \left(\sum_{p'=1}^{p}\sum_{i\in V} y^i_{p'j}\right) - p\cdot\sigma_w, \qquad p \in P, w \in W, \tag{25}$$

$$u_{pw} \geqslant p\cdot\sigma_w - \sum_{j\in J} r^w_j \cdot \left(\sum_{p'=1}^{p}\sum_{i\in V} y^i_{p'j}\right), \qquad p \in P, w \in W. \tag{26}$$

After solving the SP, if the minimum cost of the assembly shop is underestimated by the

value of Θ, i.e., $\hat{\Theta} < F_{SP}(\hat{\mathbf{x}})$, we add the following *optimality* cut (27) to the MP:

$$\textit{Optimality Cut}: \quad \Theta \geqslant F_{SP}(\hat{\mathbf{x}}) \cdot \left(\sum_{(i,b,m)\in\bar{\mathcal{X}}} x^i_{bm} - \left|\bar{\mathcal{X}}\right| + 1 \right) \tag{27}$$

where $\bar{\mathcal{X}} = \{(i,b,m) | \hat{x}^i_{bm} = 1, i \in V, b \in B, m \in B\}$. During the iterative process, we generate and add this cut to the MP progressively until $\hat{\Theta}_k$ equals $F_{SP}(\hat{\mathbf{x}}_k)$ in a specific kth iteration.

3 Compute Parameter ρ^i_k of $\mathcal{L}_3$

Algorithm 1: Algorithm to Compute the Value of ρ^i_k of $\mathcal{L}_3$

Input: V: The set of cars; $\hat{y}^i_{pj}$: A feasible solution of $\mathcal{L}_2$
Output: The value of ρ^i_k of $\mathcal{L}_3$

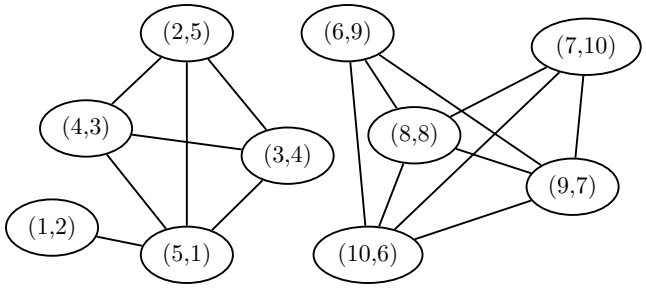
```
int p ← 1, r ← 0; for i, k ∈ V, k ⩽ i do boolean ψ^i_k ← 0 ;
for i = 1 to |V| do
    if Σ_{i'=1}^{i} Σ_{j∈J} ŷ^{i'}_{pj} = 0 then
        if r = 0 then ψ^i_1, ψ^i_2, ..., ψ^i_i ← 0 ;
        else ψ^i_1, ψ^i_2, ..., ψ^i_r ← 1; ψ^i_{r+1}, ..., ψ^i_i ← 0 ;
    else
        p ← p + 1; r ← r + 1;
        for k = 1 to r do ψ^i_k ← 1;
        for k = r + 1 to i do ψ^i_k ← 0;
    end
end
for i, k ∈ V, k ⩽ i do ρ^i_k ← 1 − Σ_{p=1}^{i} (ψ^i_p · Σ_{j∈J} y^k_{pj}) ;
return ρ^i_k
```

4 Undirected Graph Representation for a Downstream sequence

Figure 1 illustrates the undirected graph $G_{\mathcal{Y}}$ built for a downstream sequence $\mathcal{Y}^e$ with 10 cars. This $G_{\mathcal{Y}}$ has three *maximal cliques*: $\mathcal{C}^*_1$, $\mathcal{C}^*_2$, and $\mathcal{C}^*_3$, each with 4 vertices.

5 An Example to Illustrate the Heuristic Method

We consider a production schedule with $d = 4$ cars, $\bar{b} = 2$ bodies, $\bar{m} = 2$ colors, and $\bar{j} = 2$ configurations. The demand for each car model (b, m, j) is: 1 for $(1,1,1)$, 1 for $(1,1,2)$, 1 for $(2,1,1)$, and 1 for $(2,2,2)$. The PBS has $\bar{l} = 2$ lanes, each with capacity $\bar{q} = 2$. Suppose the configuration sequence obtained by solving $\mathcal{R}$ is $\mathcal{S}^c = \{2,1,2,1\}$ and the upstream sequence obtained from the first level is $\mathcal{S}^u = \{(1,1,1),(2,1,1),(3,2,1),(4,2,2)\}$. We now explain how



$$\mathcal{Y}^e = \{(1,2),(2,5),(3,4),(4,3),(5,1),(6,9),(7,10),(8,8),(9,7),(10,6)\}$$
$$\mathcal{C}_1^* = \{(2,5),(3,4),(4,3),(5,1)\}$$
$$\mathcal{C}_2^* = \{(6,9),(8,8),(9,7),(10,6)\}$$
$$\mathcal{C}_3^* = \{(7,10),(8,8),(9,7),(10,6)\}$$

Figure 1: Representation of the undirected graph $G_{\mathcal{Y}}$ built for the downstream sequence $\mathcal{Y}^e$.

to perform steps 1 to 5 of the heuristic method to construct a downstream sequence $\mathbf{y}$ for $\mathcal{L}_2$ and a car-to-lane assignment $\mathbf{h}$ for $\mathcal{L}_3$.

- *Step 1.* Include all cars that have body b and color m in set $\mathcal{O}_{bm} = \{i | (i, b_i, m_i) \in \mathcal{S}^u, i \in V, b_i = b, m_i = m\}$, and sort them in ascending order of the car index, $b \in B, m \in M$;

$$\mathcal{O}_{11} = \{1,2\}, \quad \mathcal{O}_{21} = \{3\}, \quad \mathcal{O}_{22} = \{4\}$$

- *Step 2.* For each configuration $j \in J$, count all body-color combinations with positive demands to define a map $\mathcal{Q}_j = \{(b,m) : \mathcal{O}_{bm} | d_{bmj} > 0, b \in B, m \in M\}$;

$$\mathcal{Q}_1 = \{(1,1):\{1,2\},(2,1):\{3\}\}, \quad \mathcal{Q}_2 = \{(1,1):\{1,2\},(2,2):\{4\}\}$$

- *Step 3.* Take configuration c_p from $\mathcal{S}^c$. Select the car with the lowest index i from all $\mathcal{O}_{bm}$ sets of map $\mathcal{Q}_{c_p}$. Place car i at position p and assemble it in configuration c_p, i.e., set $\hat{y}^i_{p,c_p} = 1$. Then, delete car i from all $\mathcal{O}_{bm}$ sets of map $\mathcal{Q}_j$, $j \in J$. Repeat for p from 1 to d.

Table 1: The Process of Performing *Step 3*.

Downstream Position p	**Configuration** c_p	**Change in** $\mathcal{Q}_j$	**Selected Upstream Car** i	**Reconstruct Downstream Sequence y**
1	2	$\mathcal{Q}_1 = \{(1,1):\{1,2\},(2,1):\{3\}\}$ $\mathcal{Q}_2 = \{(1,1):\{1,2\},(2,2):\{4\}\}$	1	$\hat{y}^1_{12} = 1$
2	1	$\mathcal{Q}_1 = \{(1,1):\{2\},(2,1):\{3\}\}$ $\mathcal{Q}_2 = \{(1,1):\{2\},(2,2):\{4\}\}$	2	$\hat{y}^2_{21} = 1$
3	2	$\mathcal{Q}_1 = \{(1,1):\{\ \},(2,1):\{3\}\}$ $\mathcal{Q}_2 = \{(1,1):\{\ \},(2,2):\{4\}\}$	4	$\hat{y}^4_{32} = 1$
4	1	$\mathcal{Q}_1 = \{(1,1):\{\ \},(2,1):\{3\}\}$ $\mathcal{Q}_2 = \{(1,1):\{\ \},(2,2):\{\ \}\}$	3	$\hat{y}^3_{41} = 1$

Table 1 illustrates the process of *Step 3*. Once *Step 3* is completed, we get a downstream sequence $\mathcal{Y} = \{(1,1),(2,2),(3,4),(4,3)\}$ that costs the assembly shop Δ.

- *Step 4.* Check if $\mathcal{Y}$ violates any valid inequality. If it does, $\mathcal{Y}$ is infeasible for $\mathcal{L}_3$, and ***stop***. Otherwise, try *Step 5* to construct a heuristic car-to-lane assignment $\mathbf{h}$ for $\mathcal{L}_3$.

  For the PBS in our example, the $\mathcal{Y}$ constructed in *Step 3* violates no valid inequalities. Therefore, we proceed to *Step 5*.

- *Step 5.* Assign car i to the lane with the lowest index l, ensuring the cars already assigned to it do not conflict with car i. Set $\hat{h}_l^i = 1$. Repeat for i from 1 to d.

  We first assign car 1 to lane 1 and set $\hat{h}_1^1 = 1$. Then, since the (i, p) pair of car 2 does not conflict with that of car 1, we can also assign car 2 to lane 1 and set $\hat{h}_1^2 = 1$. Similarly, car 3 can enter lane 1. For car 4, its (i, p) pair conflicts with that of car 3, so we assign car 4 to lane 2 and set $\hat{h}_2^4 = 1$. The car-to-lane assignment $\mathbf{h}$ constructed is: $\hat{h}_1^1 = 1, \hat{h}_1^2 = 1, \hat{h}_1^3 = 1, \hat{h}_2^4 = 1$.

6 Illustration of the Aggregated Acyclic-Directed Graph $\mathcal{G}_2$

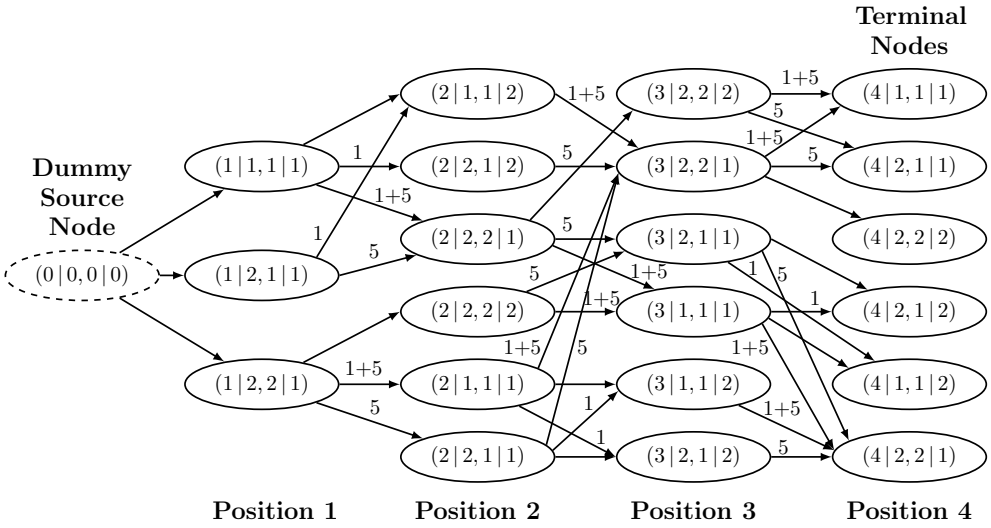


Figure 2: Illustration of the Aggregated Acyclic-Directed Graph $\mathcal{G}_2$.

7 An Aggregated Shortest-Path Problem Reformulation $\mathcal{M}_2$

To construct the demand constraints of $\mathcal{M}_2$, we introduce a binary parameter ζ_{n_1,n_2}^{bm} for each arc (n_1, n_2) of $\mathcal{G}_2$. Let $\zeta_{n_1,n_2}^{bm} = 1$ if arc (n_1, n_2) indicates extending a car with body b and color m. Then, with decision variables $x_{n_1,n_2} \in \{0, 1\}$, $(n_1, n_2) \in \mathcal{A}_2$, $\mathcal{M}_2$ is written as:

$$(\mathcal{M}_2) \quad \min \sum_{(n_1,n_2)\in\mathcal{A}_2} c_{n_1,n_2} \cdot x_{n_1,n_2} + \Theta \tag{28}$$

$$s.t. \sum_{n_2:(n_s,n_2)\in\mathcal{A}_2} x_{n_s,n_2} = 1, \tag{29}$$

$$\sum_{n_t\in\mathcal{N}_t} \sum_{n_1:(n_1,n_t)\in\mathcal{A}_2} x_{n_1,n_t} = 1, \tag{30}$$

$$\sum_{n_2:(n_1,n_2)\in\mathcal{A}_2} x_{n_1,n_2} - \sum_{n_2:(n_2,n_1)\in\mathcal{A}_2} x_{n_2,n_1} = 0, \qquad n_1 \in \mathcal{N}_2 \backslash (\mathcal{N}_t \cup \{n_s\}), \quad (31)$$

$$\sum_{(n_1,n_2)\in\mathcal{A}_2} \zeta^{bm}_{n_1,n_2} \cdot x_{n_1,n_2} \geqslant \sum_{j\in J} d_{bmj}, \qquad b \in B, m \in M, \quad (32)$$

$$[\textit{Optimality Cuts}], \quad (33)$$

$$\Theta \geqslant \Delta, \Theta \in \mathbb{R}, x_{n_1,n_2} \in \{0,1\}, \qquad (n_1,n_2) \in \mathcal{A}_2. \quad (34)$$

Constraints (29) and (30) ensure that a feasible path starts from the source node and ends at only one terminal node. The flow balance is guaranteed by constraints (31). Constraints (32) ensure that a feasible path meets the demand for all body-color combinations.

8 Demand Distribution of Each Attribute in Real-world Instances & Generation of Production Demand Quotas for Artificial Instances

Table 2: Demand Distribution of Each Attribute in 30 Real-world Instances.

Body	Proportion	Color	Proportion	Body	Proportion	Color	Proportion	Distribution of configuration
1	50%-70%	1	20%-50%	2	30%-50%	1	30%-70%	Uniform Distribution
		2	25%-40%			2	15%-35%	1 (33.3%), 2 (33.3%), 3 (33.3%)
		3	25%-40%			3	15%-35%	

Configuration (j)	Amount of option w needed (r^w_j)
1	r^1_1: 1, r^2_1: 0, r^3_1: 1, r^4_1: 1, r^5_1: 1, r^6_1: 1, r^7_1: 1, r^8_1: 0, r^9_1: 1, r^{10}_1: 1
2	r^1_2: 0, r^2_2: 1, r^3_2: 1, r^4_2: 1, r^5_2: 0, r^6_2: 0, r^7_2: 1, r^8_2: 1, r^9_2: 1, r^{10}_2: 1
3	r^1_3: 0, r^2_3: 0, r^3_3: 0, r^4_3: 1, r^5_3: 0, r^6_3: 1, r^7_3: 1, r^8_3: 0, r^9_3: 1, r^{10}_3: 1

For each artificial instance that needs to produce d cars, we follow Table 2 to generate its production demand quota. We first generate an integer τ with uniform distribution from $[\lfloor 0.5 \cdot d \rfloor, \lfloor 0.7 \cdot d \rfloor + 1]$ as the number of cars with body 1. Then, let τ_1, τ_2, and τ_3 denote the number of cars with body 1 and each of the three colors (color 1, color 2, and color 3). τ_1 and τ_2 are generated from $[\lfloor 0.2 \cdot \tau \rfloor, \lfloor 0.5 \cdot \tau \rfloor + 1]$ and $[1, \tau - \tau_1]$ with uniform distribution, respectively. τ_3 equals the remainder $\tau - \tau_1 - \tau_2$. For the other $d - \tau$ cars with body 2, we replace $0.2 \cdot \tau$ and $0.5 \cdot \tau$ with $0.3 \cdot (d - \tau)$ and $0.7 \cdot (d - \tau)$ to determine the number of cars with each color similarly. The color batch limit $\bar{c}$ is 5 for $d \leqslant 30$ and 10 for $d \geqslant 40$. After that, we get the color m^* with the highest demand q_{m^*}, and the minimum number of color changes is $\lceil q_{m^*} \setminus \bar{c} \rceil - 1$. If $\lceil q_{m^*} \setminus \bar{c} \rceil - 1 > d - q_{m^*}$, we repeat the color generation procedure until $\lceil q_{m^*} \setminus \bar{c} \rceil - 1 \leqslant d - q_{m^*}$, which ensures that each instance has a feasible solution respecting the color batch limit [5]. The configuration of each car is generated from the set $J = \{1, 2, 3\}$ with uniform distribution.

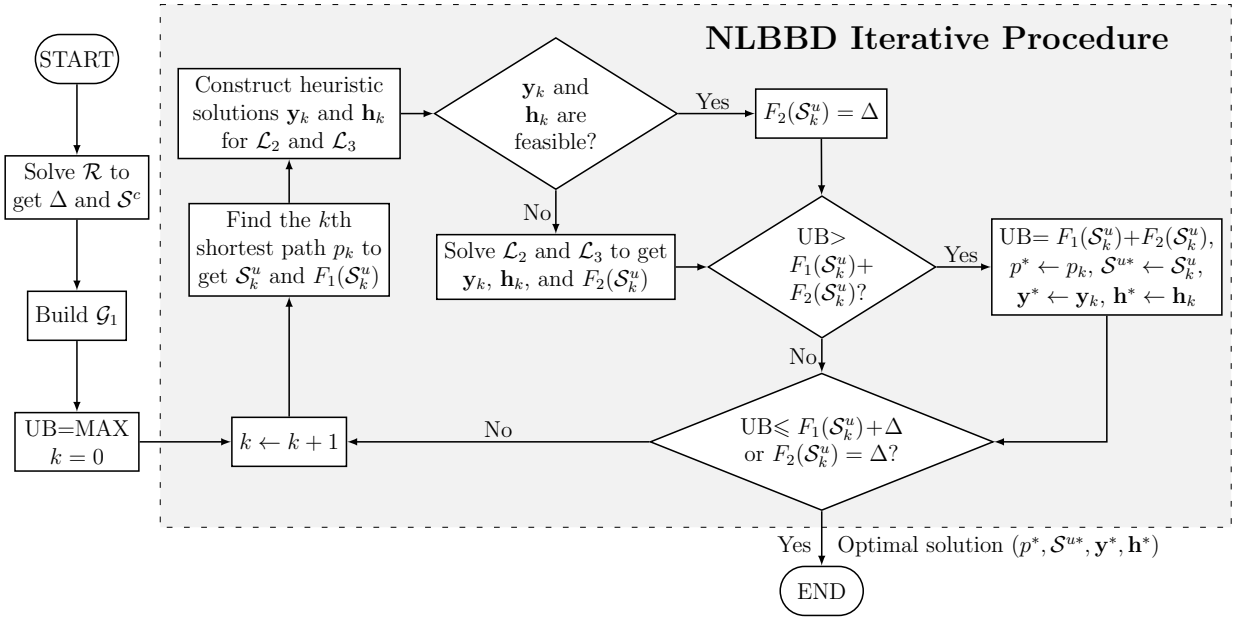


Figure 3: Flowchart of the NLBBD with $\mathcal{G}_1$.

9 Flowcharts of the NLBBD with $\mathcal{G}_1$ and $\mathcal{G}_2$

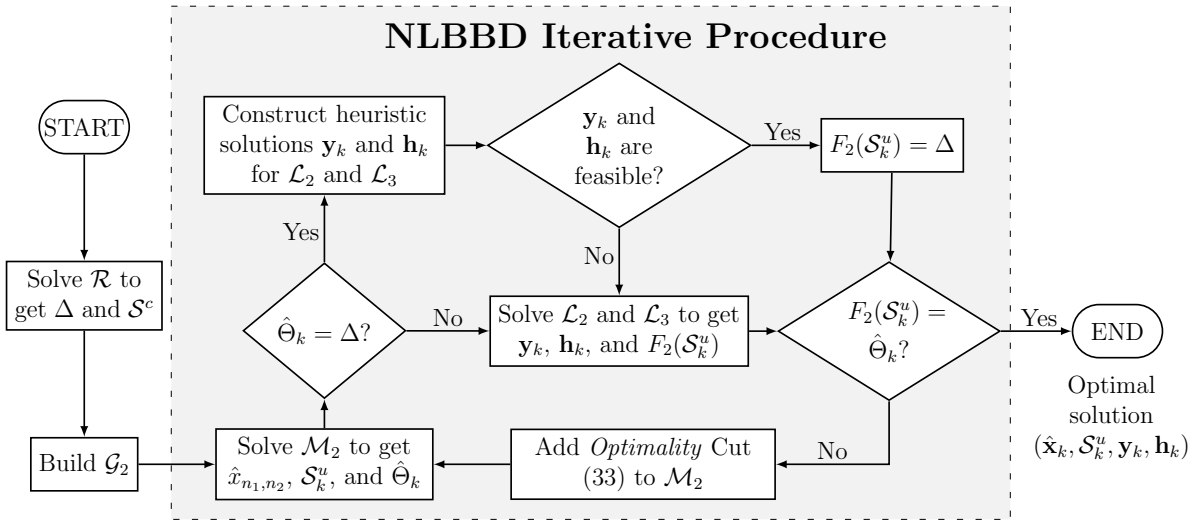


Figure 4: Flowchart of the NLBBD with $\mathcal{G}_2$.