

Highlights

A multi-start metaheuristic approach for multi-objective car resequencing problem in just-in-time assembly lines: a real-world case

Aline Frazon, Mariana Kleina, Alexandre Checoli Choueiri, Henrique Cunha, João Oliveira

- Approach to the Car Resequencing Problem in an unprecedented manner in the literature, leading a practical and multi-objective approach;
- Optimization of the resequencing of cars affected by missing parts in Just-In-Time systems;
- Reduced inventory levels make production sequencing more unstable due to the increased impact of external delays;
- Compared to manual solutions, the proposed approach achieved improvements across all objectives;
- The user-friendly implementation interface facilitated adoption and acceptance by the company's planners.

A multi-start metaheuristic approach for multi-objective car resequencing problem in just-in-time assembly lines: a real-world case

Aline Frazon^{a,b,*}, Mariana Kleina^a, Alexandre Checoli Choueiri^a, Henrique Cunha^b and João Oliveira^b

^aFederal University of Parana (UFPR), Curitiba, Parana, Brazil

^bVolvo Group, Curitiba, Parana, Brazil

ARTICLE INFO

Keywords:

Multiple objective programming

Metaheuristics

Logistics

CRP Problem

ABSTRACT

This paper addresses the problem of car resequencing (CRP) in mixed automotive assembly lines, aiming to minimize the impacts of supplier delays in companies with a Just-In-Time (JIT) system. In this scenario, when there is a shortage of parts from suppliers, the cars that use these parts must be resequenced to avoid production stoppages. However, the resequencing must comply with several constraints related to production leveling, safety, operational effort, among others. Thus, using a multi-objective approach, the study proposes a multi-objective multi-start meta-heuristic that is unprecedented in the literature. The solution was implemented in Visual Basic for Applications (VBA) Excel to facilitate integration into the planning routine of the company studied. Tests demonstrated average reductions of 90% in resequencing time and up to 47% in delayed positions, significantly improving the performance of the tool compared to the manual process. The tool automates Pareto dominance analysis, offering optimized solutions and support for decision making. In addition to reducing operational costs and respecting practical limitations, the model contributes to sustainability and innovation goals, aligned with the Sustainable Development Goals (SDGs).

1. Introduction

Just-in-time (JIT) manufacturing has a significant impact on reducing waste and managing stock spaces, which in turn reduces stock costs (Choi et al., 2023; Guo et al., 2025; Yu and Oron, 2025; Tan and Fu, 2024). It can, however, also increase the production impact of supplier delays and supply crises, such as delays of imported ships and environmental crises, among other factors (Yu et al., 2024; Ramani et al., 2022). Short-term occurrences within production networks (suppliers, environment etc) negatively impacts the stability of production schedules (Moetz et al., 2019; Lahmar et al., 2003; Iliadis and Lien, 1988; Li and Zhao, 2009; Yum and Ngai, 1986).

In this scenario, when a short-term shortage of parts from a supplier cannot be compensated by express deliveries and to avoid costly rework and production stoppages, the production sequence must be resequenced (Moetz et al., 2019). However, as presented in the study by Hozak and Hill (2009), resequencing triggers a series of far-reaching reactions throughout the production network, since other cars will also be changed to avoid constraint violations in the resequencing and, as a result, other suppliers may be impacted. In addition to its importance for the productive efficiency of companies, the problem presented in this study is directly related to the UN Sustainable Development Goals (SDGs), especially SDGs 8, 9 and 12. By addressing the problem of car resequencing, it promotes industrial efficiency and innovation (SDG 9 on Industry, Innovation and Infrastructure), reduces waste and optimizes the use of resources, aligned with the JIT system, for responsible consumption and production (SDG 12, Responsible Consumption and Production). In addition, it improves working conditions by automatically performing resequencing in a fast and multi-objective manner, reducing repetitive tasks for planners and reducing operating costs, boosting sustainable economic growth (SDG 8, which deals with Decent Work and Economic Growth) (Tran et al., 2024; Santini et al., 2021; Argyris et al., 2024). The car resequencing problem (CRP) is mainly applied in the automotive sector, but it can be applied in other areas as long as production is characterized by producing different models on the same assembly lines or

*Corresponding author

 alinefrazon@gmail.com (Aline Frazon)

ORCID(s): 0000-0003-0705-6907 (Aline Frazon)

items with a high level of customization. Thus, when there is a shortage of parts, not all products are affected, and therefore there is the possibility of postponing the affected products and resequencing production. Boysen and Zenker (2013) demonstrate the CRP as NP-hard in the strong sense. Resequencing the original assembly sequence must, in addition to minimize the impacts of the alteration, consider different sequencing constraints (Jalilvand et al., 2024; Morin et al., 2009; Yilmazlar et al., 2024; Moya et al., 2019). However, even with this complexity, schedule instability in production networks and car resequencing is a topic lacking research, unlike car sequencing (Moetz et al., 2019). In mixed-model automotive assembly lines, different types or models of vehicles are produced on the same production line. According to (Boysen et al., 2009), mixed-model automotive assembly line sequencing is addressed in three major planning approaches: mixed-model sequencing, car sequencing problem (CSP) and level scheduling problem (LSP). The authors show that mixed-model sequencing focuses on minimizing sequence workload based on detailed scheduling, while Yavuz and Ergin (2018) shows that CSP focuses on ensuring that production constraints related to the allocation of car options (for example, sun-roof and air conditioning) are satisfied, and LSP help achieve the one-piece-flow, reducing the variability on assembly times in production line. Besides, car resequencing is presented in different ways in the literature.

In car resequencing applications, the use of selectivity banks stands out, which function as intermediate buffers for sequence adjustments, minimizing operational costs and penalties. Models such as those of Sun (2024) and Leng et al. (2020) integrate virtual resequencing with these banks, while Boysen and Zenker (2013) use a decomposition approach to optimize complex resequencing. Other studies, such as those of Hong et al. (2018) and Taube and Minner (2018), explore algorithms to minimize setups and restore customer orders, although without incorporating selectivity banks. Moetz et al. (2019) presented a conceptual model of a network-oriented resequencing method that solves the interdependent tasks of production resequencing and network scheduling.

This study expands these contributions by approaching the problem in a multi-objective way in the Just In Time context, focusing on different constraints, addressing the complexities of real-world production scenarios. To summarize, the objective of this work is to automate the car resequencing process, which was previously done manually, with multi-objective optimization. For this, in this work we address a multi-objective problem (CRP) with multi-start metaheuristic and two heuristics, to reschedule mixed-model automotive assembly lines, addressing the constraint satisfaction aspect of CSP, the production leveling aspect of LSP and the resequencing of the original sequence of CRP. Specifically, we aim to address the challenges posed when certain cars, originally sequenced for production, must be postponed due to, for example, supplier delay. The objectives are related to minimize the impacts caused by these postponements while ensuring that all production constraints are not violated, through multi-objective optimization. This integrated approach is crucial, as previous studies have demonstrated that combining different aspects of planning can significantly reduce production costs and improve key performance indicators (Hosseini et al., 2024). By considering both sequencing constraints and leveling objectives, our method seeks to enhance production efficiency while maintaining the flexibility necessary to adapt to unexpected disruptions, such as delays in parts delivery. Therefore, this study proposes a new approach for the car resequencing problem in a real-world case.

The remaining of the paper is organized as follows. The detailed description of the problem, their constraints and objectives is presented in Section 2. Section 3 describes the general approach proposed to handle the car resequencing problem of this study, present the metaheuristic of the problem and in 4 is the correction heuristics. Some relevant implementation issues and challenges are commented in Section 5, and the conclusion in the last section.

2. Problem description

The car resequencing problem, on its simpler form, can be stated as follows: given a set of incremental indexes S , and a set R of indexes to be relocated ($R \subset S$), define new positions to R , providing that each element $r \in R$ cannot be assigned to a given position prior to index p , $p \in P$.

The set S represents the initial sequence of cars that are to be produced, while R , subset of S , indicates the cars that for some reason cannot be processed on the current time (index), and therefore must be postponed. The minimal postponed indexes are given by P , and it reflects the time dimension (after a given period of time it is feasible to produce the car).

The notation used for the formal description of the problems is summarized bellow:

- Sets:
 - $S = \{s_1, s_2, \dots, s_a\}$: the set of original line sequence, with a cars scheduled in a positions;

- $R = \{r_1, r_2, \dots, r_b\}$: the set of cars to be postponed;
- $P = \{p_1, p_2, \dots, p_b\}$: the set of minimum p_i index (date/time) the r_i car can be postponed;
- $O = \{o_1, o_2, \dots, o_m\}$: the set of options that a car may have;
- $Z = \{z_1, z_2, \dots, z_a\}$: the set where each shift number $z_i \in Z$ corresponds to the number of the shift in which car s_i will be produced;
- Considering a day with 2 shifts, two consecutive shifts (e.g., $z_i = 1$ and $z_j = 2$) occur on the same day, with one shift representing the morning shift and the other the afternoon shift. Shifts 1 and 2 correspond to the first day, shifts 3 and 4 to the second day etc.
- For example, if $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $Z = \{1, 1, 2, 2, 3, 3, 4, 4\}$, cars s_1 and s_2 are produced in shift 1 (the first shift of day 1), cars s_3 and s_4 in shift 2 (the second shift of day 1), cars s_5 and s_6 in shift 1 (the first shift of day 2), and so on.
- Parameters:
 - $o_j = 1$, if option j is a critical option (related to high-priority constraints), $o_j = 0$ otherwise ($j \in O$);

For instance, let $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, $R = \{2, 3\}$ and $P = \{5, 7\}$. The problem consists of postponing cars 2 and 3 from positions 5 and 7, respectively. An example of this resequencing is illustrated in Figure 1.

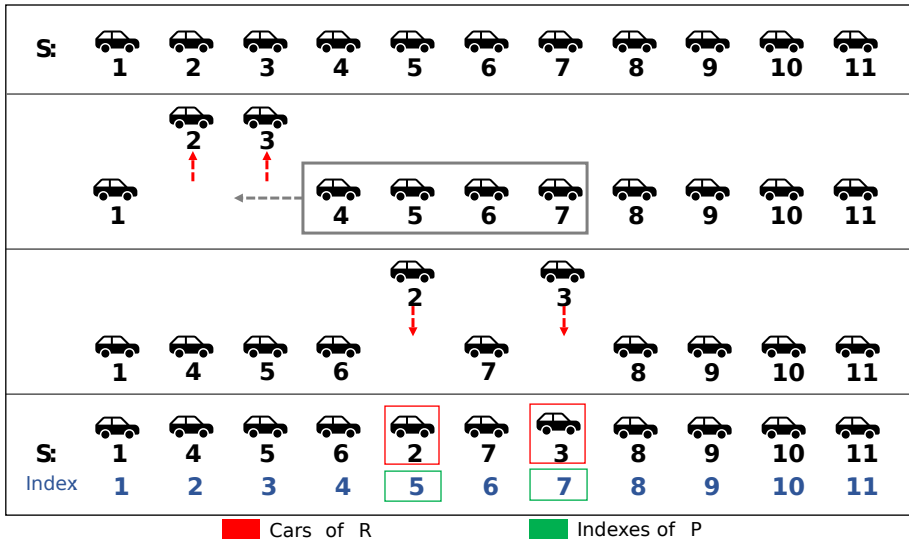


Figure 1: Example of a Car Resequencing Problem

Besides its apparent initial simplicity, there is a gamut of constraints that can be incorporated into the problem, in order to make it reflect a real manufacturing environment. Subsection 2.1 describes these constraints, considering the set of options $O = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ to exemplify them, whereas 2.2 presents the objectives of the problem.

2.1. Constraints

Performance and safety are some of the main values of the company studied, so a series of sequencing constraints are imposed on the production system to ensure safety, productivity, supplier requirements, among others. All of them are divided into five types of constraints, presented below.

2.1.1. Gap constraint

Certain combinations of car options can cause operational challenges if they are placed too closely together on a single production line due the greater difficulty of assembly, for example, sun-roof and air conditioning. The assembly line is balanced according to the flagship, but some models require more time and effort to be produced, that is, their assembly times are longer than the line's takt time.

Therefore, the *Gap constraint* is designed to ensure that specific options on the cars are spaced out appropriately, enforcing a minimum separation between cars with certain options to prevent difficulties, delays and production stops.

Consider two sets, G and GC . The set G contains subsets of car options, where each subset $G_i \subset O$ represents a group of car options that must be separated by a minimum number of positions in the production sequence. The set GC defines the corresponding minimum separation distances required between successive cars that feature these options. These sets can be mathematically expressed as follows:

- $G = \{G_1, G_2, \dots, G_h\}$: the collection of subsets, where each subset G_i represents a group of car options. The elements within each subset G_i are the specific options that need to be separated by a certain distance within the production sequence;
- $GC = \{GC_1, GC_2, \dots, GC_h\}$: the set where each GC_i specifies the minimum number of positions that must separate consecutive cars with any of the options in subset G_i .

For instance, let $G = \{\{1\}, \{2, 3\}, \{4, 7, 8\}\}$ and $GC = \{1, 6, 3\}$. Here, the first subset $G_1 = \{1\}$ indicates that cars with the option 1 should have at least $GC_1 = 1$ position between them. Similarly, the second subset $G_2 = \{2, 3\}$ means that cars with either option 2 or 3 must be separated by at least $GC_2 = 6$ positions with another car with either 2 or 3 option. The third subset $G_3 = \{4, 7, 8\}$ ensures that cars with options 4, 7, or 8 should have a minimum gap of $GC_3 = 3$ positions.

2.1.2. Consecutive cars constraint

As the *Gap constraint* presented previously, certain combinations of options can cause operational challenges. Options may require limited resources, such as special types of mounting brackets, that cannot be made available if similar cars are processed consecutively (air suspension mounting bracket, for example, are limited). Therefore, if there are only 5 brackets available and 6 cars that require these brackets are programmed consecutively, there will be no resources for the last car and, therefore, line stops will occur.

Hence, in order to reduce the impact of these options the *Consecutive cars constraint* establishes that the options of $C_i \subset O$ subset must appear at most in CC_i consecutively scheduled cars. Mathematically:

- $C = \{C_1, C_2, \dots, C_k\}$: the collection of subsets, where each $C_i \in O$ subset represents a group of car options that have consecutive occurrence limitations;
- $CC = \{CC_1, CC_2, \dots, CC_k\}$: the set where each CC_i element represent the maximum consecutively occurrences of any option for subset C_i .

For example: consider a set $C = \{\{3, 5, 9\}, \{2\}\}$ and $CC = \{4, 6\}$. Here, the first subset $C_1 = \{3, 5, 9\}$ indicates that cars with the option(s) 3 or 5 or 9 can be scheduled in at most $CC_1 = 4$ consecutively positions. Similarly, the second subset $C_2 = \{2\}$ impose that cars with option 2 must be scheduled in at most $CC_2 = 6$ consecutively positions.

2.1.3. Neighbor cars constraint

Options with advanced features, such as an advanced driver assistance system package, may require specialized handling and should not be positioned immediately adjacent to another complex option, but unlike the *Gap constraint*, it can be adjacent to itself. If two cars with these options are programmed immediately adjacent to each other, there is a risk of accidents due to operational difficulties, or line stoppages so that assembly of both models can be completed.

The *Neighbor cars constraint* addresses this issue by ensuring that a specific car option $N_i \in O$ are kept apart from a set of another options $NC_i \subset O$ by at least one position of distance. This minimizes potential assembly disruptions and maintains production efficiency. This context can be expressed as follows:

- $N = \{N_1, N_2, \dots, N_l\}$: the set of car options that are subject to *Neighbor cars constraint*;
- $NC = \{NC_1, NC_2, \dots, NC_l\}$: the collection of subsets, where each $NC_i \subset O$ subset represents a group of car options that must not be presented in a car scheduled prior or later a car with option N_i .

Let $N = \{1, 2, 3\}$ and $NC = \{\{2, 6, 7\}, \{1\}, \{5, 7\}\}$. The subset $NC_1 = \{2, 6, 7\}$ specifies that cars with option 2 or 6 or 7 should not be found directly adjacent to a car with option $N_1 = 1$, whether before or after in the sequence. Hence, $NC_2 = \{1\}$ defines that a car with option 1 must not be scheduled directly adjacent to another car with option $N_2 = 2$. Similarly, none of the options of $NC_3 = \{5, 7\}$ subset should be present in a car scheduled immediately before or after another car with option $N_3 = 3$.

2.1.4. Quantity constraint

Some options can only be produced in a certain amount per day or shift because of daily supplier constraints or equipment availability. For example, some additional vehicles pass through the painting area twice, occupying the space of a car, and therefore, if the number of scheduled cars added to these additional vehicles exceeds the maximum daily or shift capacity of the painting area, it is not possible to meet the demand. To ensure this restriction, *Quantity constraint* limits the maximum quantity of these options that can be present in cars scheduled in a same day or shift.

This constraint establishes that option $D_i \in O$ must be present in at most DC_i scheduled cars in a same **day**, namely, the sum of cars with option D_i in odd shift $z_j = x$ and even shift $z_k = x + 1$ (considering a day with 2 shifts) must be less than or equal to DC_i . Similarly, option $T_i \in O$ can be present in at most TC_i cars scheduled in a same **shift**, that is the same z_j , as follows:

- $D = \{D_1, D_2, \dots, D_u\}$: the set where each $D_i \in O$ represents the options limited by a certain maximum quantity per day;
- $DC = \{DC_1, DC_2, \dots, DC_u\}$: the set where each DC_i element represent the maximum quantity of occurrences per day of option D_i .
- $T = \{T_1, T_2, \dots, T_v\}$: the set where each $T_i \in O$ represents the options limited by a certain maximum quantity per shift;
- $TC = \{TC_1, TC_2, \dots, TC_v\}$: the set where each TC_i element represent the maximum quantity of occurrences per shift of option T_i .

Consider a set $DC = \{4, 6\}$ and $D = \{2, 5\}$, with $D \subset O$. In this case, at most 4 ($DC_1 = 4$) cars with option "2" ($D_1 = 2$) can be scheduled in a same day, just as at most 6 cars with option "5".

Thus, let $TC = \{4, 4, 11\}$ and $T = \{3, 7, 9\}$ ($T \subset O$). Similarly, at most 4 ($TC_1 = 4$) cars with option "3" ($T_1 = 3$) can be scheduled in a same shift, 4 ($TC_2 = 4$) cars with option "7" ($T_2 = 7$) and 11 cars ($TC_3 = 11$) with option "9" ($T_3 = 9$).

2.1.5. Forbiddance constraint

Additionally, a supplier-specific restriction was raised in the case study, the *Forbiddance constraint*. Due to specialized resources requirements (specialized trucks for the transport), some options cannot be produced in the same shift as others.

To meet this requirement, the restriction verifies that if a car with option $F_i \in O$ is scheduled, no option of the subset $FC_i \subset O$ can be scheduled in the same shift. The constraint sets can be mathematically expressed as:

- $F = \{F_1, F_2, \dots, F_q\}$: the set where each F_i element represents a car option with forbiddance restrictions;
- $FC = \{FC_1, FC_2, \dots, FC_q\}$: the collection of subset where each FC_i subsets specifies the options that must not be present in the same shift of option F_i .

For instance, let $F = \{4, 5, 7\}$ and $FC = \{\{1, 2\}, \{2\}, \{3, 6, 9\}\}$. In this case, option 4 can not be present in a shift that already have a car scheduled with option 1 neither 2. Similarly, option 5 and option 2 must not be scheduled in the same shift, and a car with option 7 can't be scheduled in the same shift of cars with options 3 or 6 or 9.

2.2. Objective functions

The goal of the problem is to reschedule affected cars minimizing five different objectives:

- The number of high-priority constraints violations;
- The number of low-priority constraints violations;
- The total number of cars postponed by the sequence change;
- The overall cost associated with rescheduling;
- The sum of positions by all postponed cars that are delayed;
- The maximum number of positions that a car was postponed. Each objective function is modeled below.

In equation 1, the first objective is modeled. High-priority constraints are related to critical options. If car s_i causes a high-priority constraint violation, then $high = 1$; 0 otherwise. Thus, the objective is to minimize the occurrence of high-priority constraints violations for all cars in the sequence.

$$\min Obj_i = \sum_{i=1}^a high_i \quad (1)$$

On the other hand, non-critical options are part of the low-priority constraints. Objective 2, modeled in equation 2, consists of minimizing the occurrence of low-priority constraints violations in all cars in the sequence. Similar to equation 1, if car s_i presents a low-priority constraint violation, then $low = 1$; 0 otherwise. The objective function seeks to minimize the sum of low .

$$\min Obj_{ii} = \sum_{i=1}^a low_i \quad (2)$$

Objective 3 is modeled in equation 3. If car s_i was postponed, then $\delta_i = 1$; 0 otherwise. Thus, the function seeks to minimize the sum of cars that had to be postponed by user input or by constraints violations corrections.

$$\min Obj_{iii} = \sum_{i=1}^a \delta_i \quad (3)$$

The cost function of objective 4 is presented mathematically in 4. The cost of a cars sequencing or resequencing depends on the initial positions of each car that was postponed, given that the removal of a car from its initial position and advancement of subsequent ones generates rework and waste costs proportional to how close this vehicle was to the current moment before being postponed. Therefore, the cost function of equation 4, if car s_i was postponed, will be considered the cost of the initial position occupied by car s_i .

$$\min Obj_{iv} = \sum_{i=1}^a \delta_i \cdot Cost_i \quad (4)$$

Similar to the previous equation, the objective function of 5, presented in equation 5, seeks to minimize the sum of positions by all postponed cars that are delayed, therefore, if car s_i was postponed, the distance between its initial and final position will be considered in the sum that seeks to be minimized.

$$\min Obj_v = \sum_{i=1}^a \delta_i \cdot Distance_i \quad (5)$$

Moreover, objective function 6 described in equation 6 seeks to minimize, instead sum, the maximum distance between the initial and final position of a car s_i that was delayed, that is, $\delta_i = 1$.

$$\min Obj_{vi} = \{ \max \delta_i \cdot Distance_i \} \quad (6)$$

Some of these objectives are often conflicting. Therefore, for a multi-objective optimization there is a set of trade-off optimal solution, called Pareto Front. With this, the decision makers can choose the best solution according to post-analysis preference information (Bui and Alam, 2008).

Section 3 presents the optimization approach and algorithm to search the Pareto Front.

3. Solution approach

As previously presented, some of the objectives are often conflicting, so the optimization algorithm must result in a set of tradeoff solutions to balance all objectives, called the Pareto Front. Because of this conflict, it is difficult to compare these solutions to determine which is the best solution, so they are compared in the form of Pareto dominance

(Ning et al., 2018). This means that two dominant solutions, which are part of the Pareto Front, cannot be directly compared to each other because one is better at objective i, for example, and the other is better at objective iii.

In the company studied, assembly sequence rescheduling is made with significant frequency, weekly or often daily. Because of this, the solution approach had to be fast and user-friendly. Therefore, the algorithm developed was written in Visual Basic for Applications (VBA), a programming language integrated with Excel, given the ease of use of the tool and its usability by the company's employees.

First, the user inputs the S assembly sequence and Z shifts numbers (this step can be done manually or automatically, through system integration), the K number of iterations, as well as the R cars that should be postponed and their respective P minimum positions. Then, the algorithm removes the R cars from the sequence S and advances the subsequent ones, in order to follow the two main optimization steps: the correction of the constraints that were violated with the removal of these cars, and the construction of the Pareto Front.

A characteristic of the problem is that, due to inventory limitations linked with JIT system, cars scheduled to start production up to 5 hours after the current moment (approximately the first 30 positions) that are not included in set R cannot be changed, even if they are causing constraint violations. Thus, after removing the R cars and bringing forward the subsequent ones, the algorithm checks and corrects the constraints by postponing cars that are scheduled after the first 30 positions, pointing out, if any, the violations within this interval without correcting them. The correction of each constraint follow correction heuristics presented in the section 4. After correcting all constraint violations, the algorithm inserts the R cars that were removed in the sequence again, starting from their respective P minimum positions so as not to generate new constraints violations. Then, it updates the dominance analysis between the solutions. The process is repeated until the number of solutions defined by the user (K) is generated, according to the metaheuristic presented in section 3.1. In each iteration, a new solution is generated to update the Pareto dominance analysis given the randomness present in the correction process. Thus, the metaheuristic is classified as multi-start because it generates new solutions in each iteration, and the greater the number of iterations, the more solutions will be generated. Finally, the solutions of the Pareto Front are presented to the user, highlighting the value sof the objective functions, decision variables and highlighting the remaining constraint violations and the cars that had their positions changed. The structure of the algorithm is presented in Figure 2.

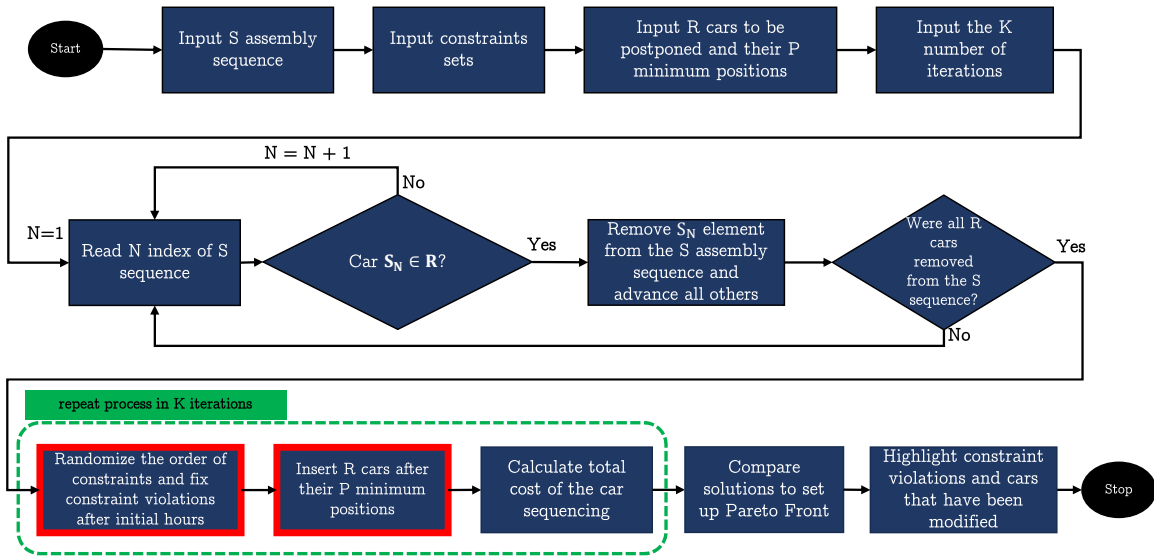


Figure 2: Algorithm structure

3.1. Car resequencing problem metaheuristic

The metaheuristic of the car resequencing problem proposed in this study was named CRP. The metaheuristic procedure is presented in the pseudocode 1, then, are presented the heuristic for generating new solutions in 2 and the heuristic for correcting constraints in 3.

First, *Par* receives the set of all inputs made by the user, including the sets described in the 2 section, number of iterations, among others, that is: $Par = \{S, R, P, K, Limit, o_j, \}$. The number of iterations performed is provided by the user, and in each iteration a solution is found. The fewer iterations, the faster the algorithm will run, but the lower the chance of finding the dominant solutions that are closest to the optimum, which consists of a 6-dimensional graphic, where each dimension refers to the value of each objective function. Moreover, *Limit* is the maximum number of positions that each car can be postponed (except those in set *R*). Then, the heuristic presented in the algorithm 1 generates the initial solution (Sol_{best}), which is Pareto Front. Finally, the multi-start metaheuristic runs during *K* iterations randomly changing the order of constraint correction, generating new solutions with CRP heuristic presented in 2 and updating the Pareto Front in each iteration.

Dominance analysis of the *UpdatePareto* function consists of comparing the current solution with each solution fixed in the Pareto Front until then through the value of their objective functions. If the solution found is not dominated by any previous solution, it enters the Pareto Front as follows: if the current solution is better than some previous solution(s) of the Pareto Front in at least one objective and equal in the other objectives, the current solution dominates the other in question and replaces it in the Pareto Front; If there is no other solution that the current solution can replace (i.e. it does not dominate any previous solution) but no previous solution dominates the current one, it enters the Pareto Front because it is not dominated by any other solution. Thus, the Pareto Front is formed by all non-dominated solutions. After updating the dominance analysis between the solutions, the algorithm generates another solution by changing the order of correction of the constraints in line 7 and, thus, generating different solutions, since the order of correction of the constraints changes the final assembly sequence.

Algorithm 1 Multi-Start Metaheuristic

```

1: Par = UserInput()
2: k = Par.K
3: Const = {1, 2, 3, 4, 5}
4:  $Sol_{best} = CRP(Const, Par)$ 
5: Pareto = UpdatePareto ( $Sol_{best}$ )
6: for i = 1, ..., K do
7:   C = Randomize (Const)
8:    $Sol_{current} = CRP(C, Par)$ 
9:   Pareto = UpdatePareto ( $Sol_{current}$ )
10: end for
```

As presented previously, new solutions (i.e., new resequencings) are generated in each iteration by the CRP heuristic presented in 2, which uses as inputs the order of the constraints generated in pseudocode 1 and the user inputs *Par*. First, the original assembly sequence *S* is stored, with the cars $\in P$ already removed, the cardinality of *S*, that is, the size of the assembly sequence, the *R* and *P* sets, as well as the number of cars to be postponed (cardinality of *P*). The high-priority constraints, inputed by user, are included in *HPC* in line 6. In lines 7-11, the constraints are corrected in the required order through the CH correction heuristic presented in 3. After correcting the constraints, in lines 12-18 all the *b* cars of *R* (set of cars to be postponed) are inserted in the line sequence in the first positions after their respective *P* minimum positions, so as not to generate new violations of the constraints, and then the other cars positions are changed to give space for them. Thus, the main strategies for generating new solutions are changing the order of constraint correction and the randomness present in the correction heuristic, which increases the number of solutions explored in the search space.

Algorithm 2 CRP(Const, Par)

```

1:  $S = Par.S$ 
2:  $a = |S|$ 
3:  $R = Par.R$ 
4:  $P = Par.P$ 
5:  $b = |Par.P|$ 
6:  $HPC = Par.HPC$ 
7: for  $Constraint = 1, \dots, |Const|$  do
8:   if  $Constraint$  is violated then
9:      $S = CH(Constraint, S, Par)$ 
10:   end if
11: end for
12: for  $i = 1, \dots, b$  do
13:    $j = \text{FirstPosition}(r_i, p_i)$ 
14:   for  $y = (j + 1), \dots, a$  do
15:      $s_y = s_{y-1}$ 
16:   end for
17:    $s_j = r_i$ 
18: end for

```

4. Correction heuristic

The constraints violations correction heuristic of the problem was implemented with different algorithms for each constraint. The logic of the correction heuristic is presented in the algorithm 3. A characteristic of the problem is that, since the system is JIT, the cars to be produced are already sold, so they cannot be postponed for a long time due to customer demand, with a limit on the positions that each car can be postponed (except those in set R), this limit is included in *Limit* from the user input. The analysis of each constraint is performed on all cars in the sequence S , which contains a cars (line 2), and according to line 5 of the pseudocode, the cars that are not part of the set of unfeasible cars are checked, that is, that cannot be allocated without causing constraint violations. At the end of the process, these cars are highlighted to the user so that there can be negotiation about the conditions of their production. Thus, when identifying a car s_x that causes a violations of this constraint, the possible positions subsequent to its original position are checked to relocate it within this limit, as per line 6 to line 10. If there are no feasible positions within this range, as per line 11, this car is considered infeasible and is disregarded in the correction of the other restrictions. Otherwise, to contribute to new solutions and avoid the Pareto Front search focusing on local searches, a random position is chosen among the set of possible positions to relocate the car s_x in line 14, and thus all other cars between the previous and future positions of this car are brought forward.

Algorithm 3 CH(Constraint, S, Par)

```

1:  $Limit = Par.Limit$ 
2:  $a = |S|$ 
3:  $Unf = \emptyset$ 
4: for  $x = 1, \dots, a$  do
5:   if  $s_x \notin Unf$  car is causing violation of Constraint then
6:     for  $Pos\_after = x + 1, \dots, x + Limit$  do
7:       if  $Pos\_after$  car is a feasible position for  $s_x$  car then
8:          $Feasible = Pos\_after$ 
9:       end if
10:    end for
11:    if  $Feasible == \emptyset$  then
12:       $Unf = s_x$ 
13:    else
14:       $e =$  randomly chosen element of  $Feasible$ 
15:       $car = s_x$ 
16:      for  $y = x, \dots, e - 1$  do
17:         $s_y = s_{y+1}$ 
18:      end for
19:       $s_e = car$ 
20:    end if
21:  end if
22: end for

```

4.1. Gap constraint violation correction

The specific algorithm for the *Gap constraint* of the correction heuristic consists of checking, for each set of constraints G_i , whether there is any violation. If there is, the algorithm selects the first car w that is causing the constraint violation at element i and checks the set of previously unanalyzed positions before and after GC_i consecutive cars with options not included in G_i . Following the logic of the heuristic, a position j called from this set is chosen to insert car w . If there are no possible positions, the car is considered infeasible. Then, the algorithm checks the other cars and options of the sets G . Figure 3 presents the process in the form of a flowchart.

4.2. Consecutive cars constraint violation correction

Similar to the previous algorithm, the constraint violation correction of the "Consecutive cars" constraint also checks each set C_i . When searching for the possible positions to relocate the selected car for removal, the algorithm checks those that have at most CC_i consecutive cars with option(s) C_i immediately before or after, since when inserting car w in this position there will be 1 more occurrence of these options. The logic can be analyzed in the Figure 4.

4.3. Neighbor cars constraint violation correction

To correct violations in the "Neighbor cars" constraint, unlike the previous ones, instead of selecting the first car in the sequence that causes a violation, the following procedure is followed: in the first violation, either the car that has options from the set N_i or the car with options belonging to NC_i is randomly selected. Thus, when reallocating the selected car, the positions that do not have cars with options from the forbidden (opposite) set programmed immediately before or after are mapped. Then, as in the other correction algorithms, a position is chosen randomly among these. The procedure is shown in Figure 5.

4.4. Quantity constraint violation correction

Similar to the previous algorithm, when correcting violations of the *Quantity constraint*, the car to be removed is randomly chosen among those that are part of the first violation of the restriction. Then, the procedure is repeated as previously presented, as shown in Figure 6.

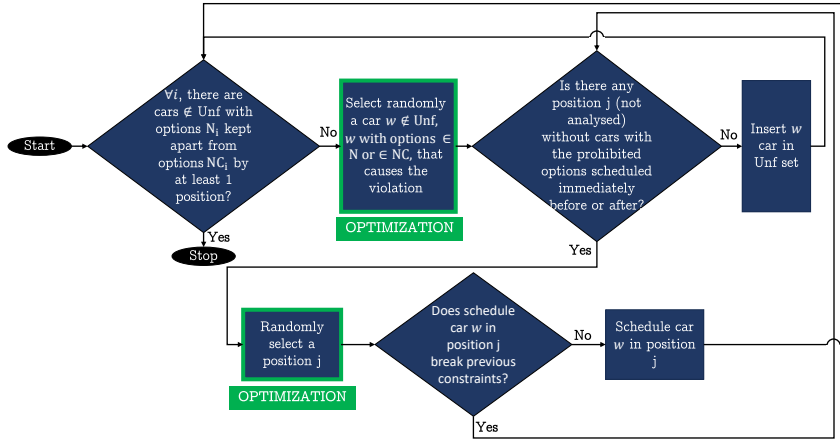


Figure 5: Neighbor cars constraint correction (Constraint=3 in Algorithm 3)

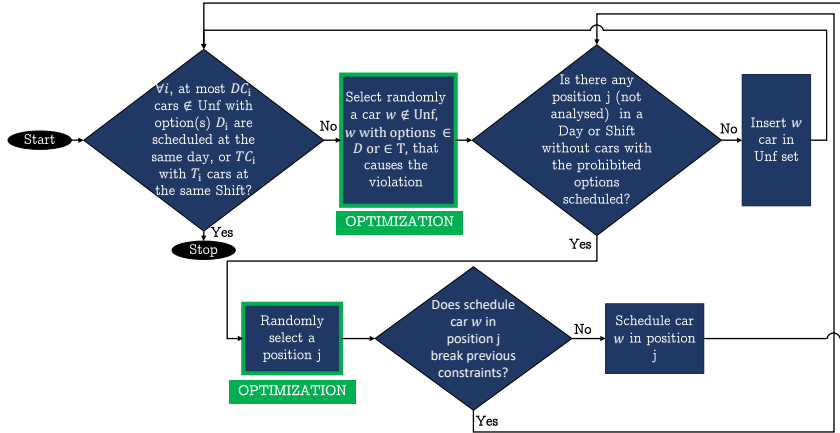


Figure 6: Quantity constraint correction (Constraint=4 in Algorithm 3)

and the number of cars that have options belonging to FC_i . If the second quantity is greater than the first, the cars with options from the set F_i are chosen for removal, and vice-versa, to ensure that the smallest possible number of cars will be removed from that turn, contributing to objective functions and a better result of this solution. Then, the possible positions for insertion of each removed car are raised and chosen randomly. The logic of this correction is shown in Figure 7.

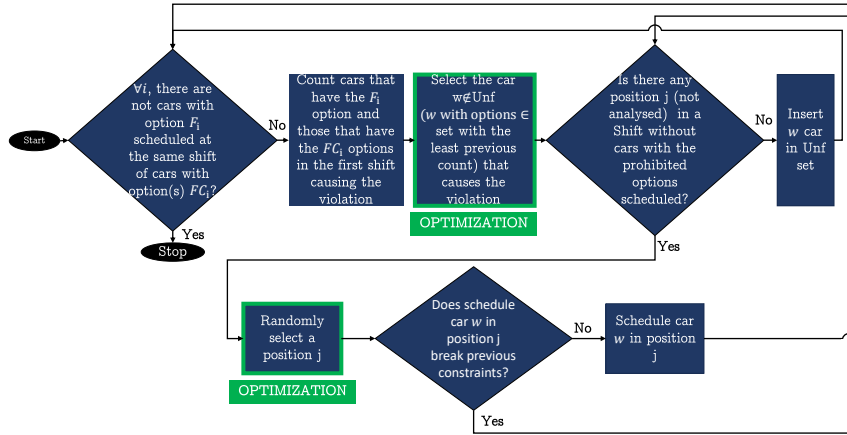


Figure 7: Forbiddance constraint correction (Constraint=5 in Algorithm 3)

5. Implementation

The proposed solution was implemented in VBA and linked to a Workbook developed in Excel, a tool chosen due to its wide application in the company where the study was conducted and its usability by planners. As a result, the tool, in one click, automatically resequences the cars and returns to the planner the sequences, among those in the Pareto Front, that have the best values of each objective function, as shown in Figure 8.

Click to reschedule

Clear data

best values:
i) 3
ii) 0
iii) R\$ 2.663,31
iv) 544
v) 184

Best value on objective i) Number of high-priority constraints violations						Best value on objective ii) Number of low-priority constraints violations								
Sequence of pareto front						Sequence of pareto front								
i) ii) iii) iv) v)						i) ii) iii) iv) v)								
A 3 4 R\$ 2.663,31 132 90						A 3 0 R\$ 4.678,90 115 80								
Assembly sequence						Assembly sequence								
Options			Car	Date/time	High priority constraint	Number of violations	Options			Car	Date/time	High priority constraint	Number of violations	
			895892	05/08/2024 06:50	2	3				894190	05/08/2024 06:50	2	0	
			893558	05/08/2024 07:00	4	1				892818	05/08/2024 07:00	4	0	
1,5,8,			896665	05/08/2024 07:11	5	0	1,5,8,			895756	05/08/2024 07:11	5	0	
			895232	05/08/2024 07:22							896388	05/08/2024 07:22		
			898147	05/08/2024 07:32							893673	05/08/2024 07:32		
			898563	05/08/2024 07:43							893563	05/08/2024 07:43		
1,4,7,9,			897423	05/08/2024 07:54	Cars that were postponed						893145	05/08/2024 07:54	Cars that were postponed	
			895496	05/08/2024 08:04	Car: Previous date/time Future date/time						894278	05/08/2024 08:04	Car: Previous date/time Future date/time	
			896615	05/08/2024 08:15	893395 05/08/2024 07:00 05/08/2024 08:40						895938	05/08/2024 08:15	897750 05/08/2024 09:51 05/08/2024 09:51	
			893791	05/08/2024 08:26	897917 05/08/2024 07:11 05/08/2024 09:51						894773	05/08/2024 08:26	897658 05/08/2024 10:12 05/08/2024 10:12	
3,			898465	05/08/2024 08:36	895808 05/08/2024 07:22 05/08/2024 10:12						893716	05/08/2024 08:36	893608 05/08/2024 10:34 05/08/2024 10:34	
8,			892994	05/08/2024 08:47							895325	05/08/2024 08:47		
			892647	05/08/2024 08:58							894773	05/08/2024 08:58		
			896297	05/08/2024 09:08							895016	05/08/2024 09:08		
			895552	05/08/2024 09:19							895346	05/08/2024 09:19		
			898556	05/08/2024 09:30							894069	05/08/2024 09:30		
5,7,8,			893395	05/08/2024 09:40							894969	05/08/2024 09:40		
			897917	05/08/2024 09:51							897750	05/08/2024 09:51		
			895494	05/08/2024 10:02							893067	05/08/2024 10:02		
			895808	05/08/2024 10:12							897658	05/08/2024 10:12		

Figure 8: Final tool to car resequencing problem

To evaluate the performance of the final tool, computational tests were performed. Tests were performed based on real scenarios by an experienced professional in the field, and a comparison was made between the sequence generated by manual and automatic resequencing. The data consisted of sequences from 3 different production lines: the assembly sequence of line 1, with an average of 1300 cars, line 2, with 700 cars, and line 3, with 60 cars. The tests began with line 3, then line 2, and finally line 1, given the volume of cars present in the assembly sequence. The tests were run with 20 instances, totaling 60 test cases.

5.1. Execution time

The results of the computational time (performed with $K = 100$), in minutes, calculated on all test instances are summarized at Table 1:

Instance	Cars Postponed	Line 1 (s)	Line 2 (s)	Line 3 (s)
1	3	123	73	8
2	3	126	75	7
3	3	121	75	10
4	3	124	72	7
5	3	128	73	9
6	4	128	70	9
7	4	126	70	10
8	4	122	72	8
9	4	123	72	9
10	4	127	71	8
11	5	124	69	8
12	5	124	73	9
13	5	128	71	8
14	5	126	72	8
15	5	122	75	9
16	6	120	73	9
17	6	128	70	10
18	6	126	73	10
19	6	128	74	9
20	6	120	74	8
21	7	125	76	11
22	7	123	69	10
23	7	121	70	11
24	7	121	71	11
25	7	120	72	10
26	8	121	74	9
27	8	121	74	9
28	8	127	73	11
29	8	126	69	11
30	8	121	69	10
31	9	122	76	11
32	9	127	69	10
33	9	122	76	9
34	9	124	71	10
35	9	120	74	10
36	10	131	71	10
37	10	123	74	10
38	10	124	70	11
39	10	132	76	11
40	10	121	74	10
41	11	124	70	11
42	11	129	69	10
43	11	132	74	11
44	11	131	71	10
45	11	120	71	10
46	12	123	74	11
47	12	122	74	11

Instance	Nº of cars postponed	Line 1 (sec)	Line 2 (sec)	Line 3 (sec)
48	12	134	75	10
49	12	125	75	11
50	12	123	72	10
51	13	129	71	11
52	13	124	74	12
53	13	133	75	11
54	13	127	74	12
55	13	129	73	11
56	14	131	75	11
57	14	129	71	11
58	14	129	71	12
59	14	122	71	12
60	14	125	75	11

Table 1: Computational times

Therefore, the average execution time can be presented as:

- Line 1 (1300 cars): average of 2 minutes per instance;
- Line 2 (700 cars): average of 1 minute and 10 seconds per instance;
- Line 3 (60 cars): average of 9 seconds per instance.

Compared to the manual process, which took approximately 10 minutes per adjustment, considering the average time of the most frequent cases (line 1, with 5 to 10 cars to be changed), the solution created achieved an average reduction of 90% in resequencing time.

5.2. Objective Function Performance

Given the multi-objective nature of the CRP problem, the values of each objective function were recorded in all instances in the manual and automatic rescheduling using the developed tool. The following results were observed with the use of the tool compared to the manual process:

- Number of high-priority constraints violations: average reduction of 23%;
- Number of low-priority constraints violations: average reduction of 40%;
- Total number of cars postponed: average reduction of 25%;
- Cost of the rescheduling: average reduction of 38%;
- Sum of positions by all postponed cars: average reduction of 47%;
- Maximum number of positions that a car was postponed: average reduction of 33%.

5.3. Comparison with Manual Solutions

A comparative analysis was performed between the solutions found by manual rescheduling, performed by experienced planners, and the automatic one by the created tool, solving the same test cases. The results showed that:

- The solutions generated by the automatic tool outperformed or remained equal to the manual solutions in all six objectives;
- Manually generated solutions often failed to reach the Pareto Front, resulting in more constraint violations, postponements and, therefore, costs.

5.4. Financial Impact

To measure the financial impact of the implementation, the annual cost was used through the value of the employees' hours used to calculate the benefits of improvement projects, by company standard. Before the implementation of the tool, planners took about 10 minutes per adjustment. Considering an average of 3 sequence adjustments per week on the 3 lines, a total of 9 resequencings per week, over a year with 50 productive weeks, totaled 4,500 minutes per year. Considering a cost of R\$220.00/h, there was an annual resequencing cost of R\$16,500.00 before implementation.

With the tool developed in the study, adjustments take an average of 1 minute, reducing the annual time to 450 minutes (a 90% reduction in time) and thus, the cost to R\$3,300.00 limited to the Pareto Front analysis time of 2 minutes per adjustment (80% reduction in cost).

6. Conclusions

This study addressed the Car Resequencing Problem in a way that was unprecedented in the literature, seeking to optimize different real-world objectives and reconciling aspects of non-violation of CSP constraints with the leveling of the LSP workforce through different constraints. For this purpose, a multi-start metaheuristic was proposed, and two heuristics were developed, one for generating new solutions and the other for correcting constraint violations, for resequencing cars that must be postponed due to missing parts, unavailability of machines, etc.

The algorithms were applied in VBA through a tool developed in Excel, given the usability of the tool in the production scheduling areas of different companies. One click is all it takes for the developed tool to search for the Pareto Front and show the user the best options to aid decision-making, outperforming the manual alternatives in all objectives.

The implementation of the study through the tool achieved significant improvements in the different problem objectives, as well as in the resequencing time and reliability of the information. The tool developed automated the sequence change process, making the process faster and more reliable, through a robust method, involving multi-start metaheuristics and CRP heuristics, through a user-friendly software with wide application, Excel. Thus, all computational results showed improvements in different aspects, such as resequencing time, indicators, quality of the Pareto Front to support decision making, and cost. The user-friendly interface of the implementation, when carried out in Excel, facilitated adoption and acceptance by the company's planners, who were able to make adjustments efficiently while evaluating the Pareto Front at the time of decision-making. The perceived financial benefits further highlight the relevance of the solution. A limitation of the study is the network integration with supplier sequencing to further increase the reliability of resequencing.

For future research, alternative metaheuristic approaches and advanced software, in addition to Excel, as applications, are suggested to further optimize implementation performance.

7.

References

- Argyris, N., Østerdal, L.P., Hussain, M.A., 2024. Value-driven multidimensional welfare analysis: A dominance approach with application to comparisons of european populations. *European Journal of Operational Research* URL: <https://www.sciencedirect.com/science/article/pii/S0377221724009342>, doi:<https://doi.org/10.1016/j.ejor.2024.11.043>.
- Boysen, N., Fliedner, M., Scholl, A., 2009. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research* 192, 349–373. URL: <https://www.sciencedirect.com/science/article/pii/S0377221707009332>, doi:<https://doi.org/10.1016/j.ejor.2007.09.013>.
- Boysen, N., Zenker, M., 2013. A decomposition approach for the car resequencing problem with selectivity banks. *Computers and Operations Research* 40, 98–108. URL: <https://www.sciencedirect.com/science/article/pii/S0305054812001153>, doi:<https://doi.org/10.1016/j.cor.2012.05.012>.
- Bui, L.T., Alam, S., 2008. *Multi-Objective Optimization in Computational Intelligence: Theory and Practice* (Premier Reference Source). 1 ed., IGI Global, USA.
- Choi, T.Y., Netland, T.H., Sanders, N., Sodhi, M.S., Wagner, S.M., 2023. Just-in-time for supply chains in turbulent times. *Production and Operations Management* 32, 2331–2340. URL: <https://doi.org/10.1111/poms.13979>, doi:10.1111/poms.13979, arXiv:<https://doi.org/10.1111/poms.13979>.
- Guo, Z., Wei, L., Zhang, J., Hu, Z., Sun, H., Li, X., 2025. Multi-objective flexible job shop scheduling based on feature information optimization algorithm. *Computers and Operations Research* 179, 107027. URL: <https://www.sciencedirect.com/science/article/pii/S0305054825000553>, doi:<https://doi.org/10.1016/j.cor.2025.107027>.
- Hong, S., Han, J., Choi, J.Y., Lee, K., 2018. Accelerated dynamic programming algorithms for a car resequencing problem in automotive paint shops. *Applied Mathematical Modelling* 64, 285–297. URL: <https://www.sciencedirect.com/science/article/pii/S0307904X18303512>, doi:<https://doi.org/10.1016/j.apm.2018.07.035>.
- Hosseini, A., Otto, A., Pesch, E., 2024. Scheduling in manufacturing with transportation: Classification and solution techniques. *European Journal of Operational Research* 315, 821–843. URL: <https://www.sciencedirect.com/science/article/pii/S0377221723007701>, doi:<https://doi.org/10.1016/j.ejor.2023.10.013>.
- Hozak, K., Hill, J.A., 2009. Issues and opportunities regarding replanning and rescheduling frequencies. *International Journal of Production Research* 47, 4955–4970. URL: <https://doi.org/10.1080/00207540802047106>, doi:10.1080/00207540802047106, arXiv:<https://doi.org/10.1080/00207540802047106>.
- Iliadis, I., Lien, L.C., 1988. Resequencing delay for a queueing system with two heterogeneous servers under a threshold-type scheduling. *IEEE Transactions on Communications* 36, 692–702. doi:10.1109/26.2789.
- Jalilvand, S., Mahmoodjanloo, M., Baboli, A., 2024. Flexible programming model for efficient workload control in the car sequencing problem. *IFAC-PapersOnLine* 58, 1294–1299. URL: <https://www.sciencedirect.com/science/article/pii/S2405896324015660>, doi:<https://doi.org/10.1016/j.ifacol.2024.09.154>. 18th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2024.
- Lahmar, M., Ergun, H., Benjaafar, S., 2003. Resequencing and feature assignment on an automated assembly line. *Robotics and Automation, IEEE Transactions on* 19, 89 – 102. doi:10.1109/TRA.2002.807556.
- Leng, J., Jin, C., Vogl, A., Liu, H., 2020. Deep reinforcement learning for a color-batching resequencing problem. *Journal of Manufacturing Systems* 56, 175–187. URL: <https://www.sciencedirect.com/science/article/pii/S0278612520300911>, doi:<https://doi.org/10.1016/j.jmsy.2020.06.001>.
- Li, J., Zhao, Y.Q., 2009. Resequencing analysis of stop-and-wait arq for parallel multichannel communications. *IEEE/ACM Transactions on Networking* 17, 817–830. doi:10.1109/TNET.2009.2020820.
- Moetz, A., Stylos-Duesmann, P., Otto, B., 2019. Schedule instability in automotive production networks: The development of a network-oriented resequencing method. *IFAC-PapersOnLine* 52, 2810–2815. URL: <https://www.sciencedirect.com/science/article/pii/S2405896319316234>, doi:<https://doi.org/10.1016/j.ifacol.2019.11.634>. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.
- Morin, S., Gagné, C., Gavel, M., 2009. Ant colony optimization with a specialized pheromone trail for the car-sequencing problem. *European Journal of Operational Research* 197, 1185–1191. URL: <https://www.sciencedirect.com/science/article/pii/S037722170800310X>, doi:<https://doi.org/10.1016/j.ejor.2008.03.033>.
- Moya, I., Chica, M., Bautista, J., 2019. Constructive metaheuristics for solving the car sequencing problem under uncertain partial demand. *Computers and Industrial Engineering* 137, 106048. URL: <https://www.sciencedirect.com/science/article/pii/S0360835219305078>, doi:<https://doi.org/10.1016/j.cie.2019.106048>.
- Ning, X., Qi, J., Wu, C., Wang, W., 2018. A tri-objective ant colony optimization based model for planning safe construction site layout. *Automation in Construction* 89, 1–12. URL: <https://www.sciencedirect.com/science/article/pii/S0926580517309329>, doi:<https://doi.org/10.1016/j.autcon.2018.01.007>.
- Ramani, V., Ghosh, D., Sodhi, M.S., 2022. Understanding systemic disruption from the covid-19-induced semiconductor shortage for the auto industry. *Omega* 113, 102720. URL: <https://www.sciencedirect.com/science/article/pii/S030504832200127X>, doi:<https://doi.org/10.1016/j.omega.2022.102720>.
- Santini, A., Bartolini, E., Schneider, M., Greco de Lemos, V., 2021. The crop growth planning problem in vertical farming. *European Journal of Operational Research* 294, 377–390. URL: <https://www.sciencedirect.com/science/article/pii/S0377221721000643>, doi:<https://doi.org/10.1016/j.ejor.2021.01.034>.
- Sun, H., 2024. Integrating virtual resequencing with car resequencing via selectivity banks for mixed-model assembly lines. *Computers and Industrial Engineering* 189, 109990. URL: <https://www.sciencedirect.com/science/article/pii/S0360835224001116>, doi:<https://doi.org/10.1016/j.cie.2024.109990>.

[//doi.org/10.1016/j.cie.2024.109990](https://doi.org/10.1016/j.cie.2024.109990).

- Tan, Z., Fu, G., 2024. Just-in-time scheduling problem with affine idleness cost. *European Journal of Operational Research* 313, 954–976. URL: <https://www.sciencedirect.com/science/article/pii/S0377221723006690>, doi:<https://doi.org/10.1016/j.ejor.2023.08.038>.
- Taube, F., Minner, S., 2018. Resequencing mixed-model assembly lines with restoration to customer orders. *Omega* 78, 99–111. URL: <https://www.sciencedirect.com/science/article/pii/S0305048316309124>, doi:<https://doi.org/10.1016/j.omega.2017.11.006>.
- Tran, T.H., Nguyen, T.B.T., Le, H.S.T., Phung, D.C., 2024. Formulation and solution technique for agricultural waste collection and transport network design. *European Journal of Operational Research* 313, 1152–1169. URL: <https://www.sciencedirect.com/science/article/pii/S0377221723006811>, doi:<https://doi.org/10.1016/j.ejor.2023.08.052>.
- Wu, J., Ding, Y., Shi, L., 2021. Mathematical modeling and heuristic approaches for a multi-stage car sequencing problem. *Computers and Industrial Engineering* 152, 107008. URL: <https://www.sciencedirect.com/science/article/pii/S0360835220306781>, doi:<https://doi.org/10.1016/j.cie.2020.107008>.
- Yavuz, M., Ergin, H., 2018. Advanced constraint propagation for the combined car sequencing and level scheduling problem. *Computers and Operations Research* 100, 128–139. URL: <https://www.sciencedirect.com/science/article/pii/S0305054818302028>, doi:<https://doi.org/10.1016/j.cor.2018.07.018>.
- Yilmazlar, I.O., Kurz, M.E., Rahimian, H., 2024. Mixed-model sequencing with stochastic failures: A case study for automobile industry. *European Journal of Operational Research* 319, 206–221. URL: <https://www.sciencedirect.com/science/article/pii/S0377221724004673>, doi:<https://doi.org/10.1016/j.ejor.2024.06.019>.
- Yu, R., Oron, D., 2025. Single-machine scheduling with fixed energy recharging times to minimize the number of late jobs and the number of just-in-time jobs: A parameterized complexity analysis. *European Journal of Operational Research* URL: <https://www.sciencedirect.com/science/article/pii/S0377221725000323>, doi:<https://doi.org/10.1016/j.ejor.2025.01.007>.
- Yu, W., Wong, C.Y., Jacobs, M.A., Chavez, R., 2024. What are the right configurations of just-in-time and just-in-case when supply chain shocks increase? *International Journal of Production Economics* 276, 109352. URL: <https://www.sciencedirect.com/science/article/pii/S0925527324002093>, doi:<https://doi.org/10.1016/j.ijpe.2024.109352>.
- Yum, T.S., Ngai, T.Y., 1986. Resequencing of messages in communication networks. *IEEE Transactions on Communications* 34, 143–149. doi:10.1109/TCOM.1986.1096505.