Yu *et al. J Wireless Com Network* (2022) 2022:26
https://doi.org/10.1186/s13638-022-02113-7

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Heuristic approaches for the car sequencing problems with block batches

Yingjie Yu[1,3], Xiaochun Lu[2], Tao Zhao[1,3], Minjiao Cheng[2], Lin Liu[1,3*] and Wenchao Wei[3]

*Correspondence:
liulin@catarc.ac.cn
[3] China Automotive Technology and Research Center Co., Ltd., Tianjin, China
Full list of author information is available at the end of the article

Abstract

Motivated by the practical supply chain management of the automobile industry, we study the car sequencing problem (CSP) that minimizes the conflicts occur when sequentially manufacturing cars on an assembly line. The CSP is a well-established problem, subject to the paint batching constraints to decrease the energy consumption for color changeovers and production rate constraints in the assembly shop to ensure a smooth usage of car options. However, the existing solution algorithms to this problem do not take into account the block batches, which desires a consecutive production batch of cars requiring a certain option. This requirement often occurs when specialized labor time window is short in the customized car production scenario, and renders additional complexities to the traditional car sequencing problem. In this paper, we present a novel model to deal with these constraints and simultaneously generate the sequencing and replenishment decisions. Besides, we develop two math-heuristic algorithms to solve the proposed large-scale CSP. The presented heuristics are on the basis of relax-and-fix procedures, fix-and-optimize procedures and adaptive variable neighborhood search. To solve the large-sized instances (commercial solvers, i.e., Cplex, cannot obtain a feasible result within 1 h), we design and implement a reinforced parameter tuning mechanism to dynamically select the parameter values, so as to speed up the search process. The proposed models and heuristics are tested on representative instances generated from the benchmark in the literature (CSPLib), as well as large-sized instances generated from real-world cases. We report on extensive computational experiments and provide basic managerial insights into the planning process.

Keywords: Car sequencing problem, Math-heuristics, Variable neighborhood search

1 Introduction

The traditional car sequencing problem firstly proposed by [28] tackles the problem that occur when scheduling cars with various car options in a single assembly line and is addressed through converting position, time and/or technical requirements into discrete 0–1 options [27, 37]. The assembly line normally consists of series production shops, including weld shop, paint shop and assembly shop. As one of the primary sources of air emissions of regulated chemicals, the paint process generates volatile organic compounds (VOCs), hazardous air pollutants (HAPs) and others. As pointed by [12], the production process generates both solid and hazardous and solid waste, including

chemicals (i.e., chlorine bleach) to clean the paint lines and application equipments (sprayers), disposal parts and waste paint due to overspray. Hence, from an environmental and energy saving perspective, the paint batching constraints are considered to reduce the color changes in the paint shop [5]. A contradicting constraint arising in the assembly shop, however, intends to level the production of different cars, so as to maintain a proper production speed and usage of car options [25].

Particularly in the auto industry changing from mass standardization production to mass customization production, each car is composed of a different set of accessories/options, i.e., engines, tires, roofs, etc. With the advent of Internet of Things and cybernetic technologies, the fixture becomes more and more complex to assemble a specific set of options to the car bodies.As mentioned by [40], such problems often occur in the make-to-order and/or just-in-time environment. The setup time of changing from one complex fixture to another one is time-consuming and might disorganize the production rhythm. Hence, to reduce the number of setups and the corresponding total setup times, company tends to consecutively process cars with the same option, especially for options that only a few fixture can handle. Motivated by this practical requirement, we introduce block constraints, which define a subsequence with a fixed length for a specific option. If there is a changeover from other options to this option, then the following ones must be the same options as well, until the next changeover to another option.

One drawback of block production is the violation of level production requirement, which entails a smooth usage of the parts (options) in the assembly process. In the assembly shop, parts are normally delivered to the assembly line at a pre-specified frequency and speed. Most of the existing literature tackles this issue by maintaining a constant usage rate of the parts at each position of the sequence and penalize the weighted violations. However, this method neglects the processing time of different options, i.e., the installation of an engine takes around half an hour, while the installation of a sunroof only requires a few minutes. In this paper, we create a time-related evaluation approach to incorporate the processing time of each option. We argue that the new method would more accurate to describe the practical situation than the traditional one and hence be used to streamline the entire manufacturing process.

The contributions of this paper are threefold: Firstly, we extend the classical car sequencing problem to incorporate several novel modeling challenges arising in the context of the car industry, which have not yet been considered in the existing literature of CSP. A mathematical formulation is presented for the proposed scheduling problem; secondly, we develop two math-based heuristics with data-driven VNS to solve the problem; finally, we present extensive computational tests in the car industry and highlight several managerial suggestions.

The structure of this paper is planned as follows. Section 2 presents the literature reviews about relevant academic work on car sequencing problem including the solution approaches. Section 3 describes the scheduling problem and derives the formulation. Two math-heuristic algorithms that include data-driven variable neighborhood search (VNS) are presented in Sect. 4 to efficiently solve the proposed problem. Our computational experiments are reported in Sect. 5, where we illustrate the efficiency of the proposed solution approach, and describe the system implementation and its benefits to the company. The conclusion and future research directions are pointed out in Sect. 6.

2 Literature review

2.1 Solutions approaches for CSP

Among the first attempts to study the car sequencing problem (CSP), [28] describes it as generating a full sequence of cars on a single assembly line, so as to install car options (e.g., leather seats, engines, tires) on them. Several specific options are required by each of the cars assembled through the line. To avoid the potential conflicts arising from many consecutive cars requiring the same option, a maximum load ratio is adopted for some options [42]. This requirement may describe the so-called by p/q ratio constraints: Any successional q cars could consist of p cars maximum demanding a specific type of options. The objective of the car sequencing problem is to identify a complete sequence of cars that does not violate the maximum load ratio constraints for every subsequence or a full sequence with minimum constraints violation costs. The CSP has been proven to be NP-hard by [24]. A number of approaches have been developed for the CSP problem. The evaluations of these algorithms are normally based on CSPLib [13], a benchmark library for the car sequencing problem. The abovementioned approaches include the ones that searching for exact solutions and those searching for approximately optimal solutions.

[7] solved the CSP with a constraint programming language with the result is either a car sequence that assures the satisfaction all the option constraints, or a failure implying that no such sequence exists. A constraint programming approach for CSP was proposed by [4]. They consider two types of constraints: hard constraints that must be satisfied, and soft constraint that can be violated at a cost. Later, [35] suggested several heuristics impelling a pruning rule, to speed up the solution approach. [1] proposed an integer programming model for CSP, based on a binary variable indicating the car-position assignment. [17] developed an MILP formulation for CSP with soft capacity constraints, by using additional binary variable deciding if a car-position assignment satisfies the capacity constraints. A first dedicated branch and bound algorithm to solve CSP was proposed by [8]. Other exact approaches, i.e., beam search algorithms, were proposed by [2] and [15].

Apart from the complete approaches, non-exact methods, such as metaheuristics, have been proposed aiming at fast search for near optimal solutions. [20] proposed the first greedy approach for the car sequencing problem. Later, [16] evaluated six greedy heuristic approaches for the CSP. [32] suggested a local search approaches to address the car sequencing problem and suggested an inversion operator (i.e., insert a car from any position in the sequence to another position) to increase the search efficiency. Another specially relevant operator is the swap operator (i.e., change the positions of any two cars) [14]. The capabilities and potentials of large neighborhood search technique for solving CSP were firstly investigated by [29]. Among the first attempts of applying ant colony optimization algorithms to CSP, [37] proposed a dynamic sum of utilization rate and argue that ant colony algorithm is slightly superior to local search for small computation time, whereas both algorithms provide comparable solution quality for large computation time. [17] also developed an ant colony optimization (ACO) approach for CSP, which associates with a local search technique which is used to speed up the search process constructed by ants. [39] propose a large neighborhood search (LNS) based on mixed integer programming (MIP)

and substantially improved the solution qualities. For recent solution approaches of CSP, we refer to the review paper [36].

2.2 Extensions to CSP model

One of the extensions to the car sequencing problem that obtains a lot of interests by the researchers is the ROADEF'2005 challenge suggested by RENAULT [37]. Apart from capacity constraints arising in the assembly shop in the traditional CSP, the challenge introduces paint batching constraints in the paint shop and two classes of capacity constraints with different priorities in assembly shop. This extension is specially relevant to our study, as it introduces the same type of constraints that is used to incorporate usage restrictions of hazardous materials. Several contributions focus on the (meta-)heuristics approaches of this version of CSP, such as tabu search [6, 44], local search [10, 11] and variable neighborhood search [31]. The comparison of four heuristics is given in [23].

Another variation of the classical CSP is so-called extended CSP, proposed by [3, 33]. They introduced a minimum number of operations requirement into the production sequences. Hence, the extended CSP aims at the finding out the trade-off between the capacity overload and under-load. A GRASP approach is proposed by [3] to solve this problem extension. [27] extend the CSP with partial demand, and implement and compare several constructive heuristics. [38] consider restoration in the CSP and generate a look-ahead approach to solve a large scale problems. [21] study failure probabilities in the CSP and propose a sampling-based adaptive large neighborhood search heuristic. [34] tackle variable station space in the CSP and propose branch & cut algorithm to solve the proposed problem.

The abovementioned practical matters need additional modeling and computation efforts. We extend the classical car sequencing problem by introducing extra block variables and processing time computations. The block production requirement has been discussed in several literature streams, i.e., supply chain scheduling (SCS) [18], production-routing [41] and production scheduling-vehicle routing problem (PS-VRP) [26]. Based on our knowledge, this is the first attempt to model this constraint into the car sequencing problems.

Because of the NP-hardness of the proposed problem, we design and implement two math-heuristic approaches to effectively and efficiently tackle the problem. The first one is a constructive heuristic that is capable of fast generating a solution with acceptable quality based on a relax-and-fix approach; the second one is an improvement heuristic that can significantly improve the solution process. The overall technical accomplishment is embed into a decision support system and deployed in a car manufacturer producing business cars. The experimental tests show that our solution approaches are efficient in solving large sized instances and is capable of improving the company's current manufacture efficiency.

3 Problem statement and formulation

A mixed-integer linear programming (MILP) is applied to depict and analyze the proposed car sequencing problem (CSP) with block batches. Extra 0–1 decision variables are proposed to denote the starting position of each block batch. The problem involves a production stage with a single production line environment, which processes the cars

consecutively. At the production stage, the cars are sequentially processed through welding shop, paint shop and assembly shop at the same production rhythm. Each shop consists of a series of work stations, where the accessories are assembled to the car bodies according to the bill of materials (BOM). As the stop of the production process incurs a large amount of energy waste, the inventory of the accessories at each station should be enough to satisfy the production requirements. We consider cars $\mathcal{R} = \{1, 2, \cdots, N\}$ to be processed consecutively on work stations $\mathcal{M} = \{1, 2, \cdots, M\}$. Each car requires a subset of options $O_i \subset \mathcal{O}$ and each option $o \in \mathcal{O}$ should be processed on a dedicated work station. For options o with block productions, J_o denotes the number of blocks required by option o and f_o is the length of each block. The operating time for each option o relies on the number of accessories required and the technical difficulties.

The planning horizon $\mathcal{T} = \{1, 2, \cdots, T\}$ starts from the time when the first car in the sequence enters the production line, and ends at the time when the last car in the sequence leaves the production line. As the production rhythm is unchanged, the car bodies move from one station to the next at the end of each time period t. The car sequence also must comply with several capacity constraints of each shop, i.e., limited solvent consumption for cleaning paint guns and maximum number of consecutive usage of certain accessories. Thus, restrictions are represented by a parameter tuple p_o, q_o: any successional q_o cars may contain at most p_o cars requiring option o. Therefore, the production sequence must comply with the work station configurations such that conducted processing does not influence the number of cars manufactured in the same time unit, which is described by the industry popular JPH (Job per Hour) indicator.

The remaining notations of our problem are as follows:

Sets and parameters	
$\mathcal{M}$	set of work stations
$\mathcal{N}$	set of cars
$\mathcal{O}$	set of options
$p_o : q_o$	capacity constraints of option o
κ_{io}	(=1) if car $i \in N$ requires option $o \in O$
γ_{om}	operating time of option o on machine m
J_o	number blocks of option o
f_o	number of cars required in each block of option o
c_{ot}	coefficient of option o consumed at time period t
ϵ_o	average consumption rate of option o
τ_o	material quantity required by option o
λ_o	coefficient of capacity constraint violation of option o
B	an arbitrary large number

Decision variables	
x_{ij}	(=1) if car i is operated on the jth position of the production sequence
u_{osj}	(=1) if the first car of s-th block of option o is scheduled on the jth position of the production sequence
y_{ot}	consumption of accessories of option o at time period t
z_o	number of capacity constraint violations of option o
α_{ijm}	the starting time of the process of car i in position j starts on work station m

The mathematical formulation of the problem reads as follows:

$$\sum_{o} z_o \alpha_o + \sum_{o} \sum_{t} (y_{ot} c_{ot} - \epsilon_o * t) \tag{1}$$

The objective (1) is aim to minimize the total constraints violation and non-level option consumption cost.

s.t.

3.1 Capacity violation

Constraints (2) calculate the total quantity of capacity violations of option o.

$$\sum_{i} \sum_{k=j}^{j+q_o-1} x_{ik} \kappa_{io} \leq p_o - z_o, \forall o \in \mathcal{O}, j = 1, 2, \cdots, n - q_o + 1 \tag{2}$$

3.2 Car–machine assignment

Constraints (3) and (4) make sure that each position is allocated to exactly one car, and each car is produced on exactly one position.

$$\sum_{i} x_{ij} = 1, \forall j = 1, 2, \cdots, N \tag{3}$$

$$\sum_{j} x_{ij} = 1, \forall i \in \mathcal{N} \tag{4}$$

3.3 Block production

Constraints (5) imply that for each option o, there are J_o blocks.

$$\sum_{j=1}^{N-f_o+1} \sum_{s}^{J_o} u_{osj} = J_o, \forall o \in \mathcal{O} \tag{5}$$

Constraints (5) impose that the number of cars between the first cars of any two blocks should be at least $f_o + 1$, so as to separate different blocks.

$$\begin{aligned} &\mu_1 u_{o\mu_1 j_1} - \mu_2 u_{o\mu_2 j_2} \leq f_o + 1, \\ &\forall, o \in \mathcal{O}, \mu_1, \mu_2 \in \{1, 2, \cdots, N - f_o + 1\}, \mu_1 < \mu_2, j_1, j_2 \leq J_o, j_1 < j_2 \end{aligned} \tag{6}$$

Constraints (7) make sure that each car scheduled in the block has option o.

$$\sum_{i} \sum_{k=j}^{j+f_o-1} x_{ik} \kappa_{io} \geq f_o u_{osj}, \forall, o \in \mathcal{O}, s \leq J_o, j = 1, 2, \cdots, N - f_0 + 1. \tag{7}$$

3.4 Option consumption

Constraints (8) calculate the start time of producing car i on station m

$$\alpha_{ijm} \geq \alpha_{r,j-1,m} + \sum_{o} \gamma_{om} x_{r,j-1} \kappa_{ro}, \forall i, r \in \mathcal{N}, j = 2, 3, \cdots, N, m \in \mathcal{M} \tag{8}$$

Constraints (9) impose a schedule coincide with the sequence: For each car i, the start time of the process on station $m+1$ should be later than the finish time of the process on mth station .

$$\alpha_{ijm} \leq \alpha_{ij,m+1}, \forall i \in \mathcal{N}, m \in \mathcal{M}, j = 1, 2, \cdots, N \tag{9}$$

Constraints (10) present the relations between variable x and α

$$\alpha_{ijm} \leq B x_{ij}, \forall i \in \mathcal{N}, m \in \mathcal{M}, j = 1, 2, \cdots, N \tag{10}$$

Constraints (11) calculate the consumption of option o at time period t.

$$y_{ot} \geq \sum_{i,j,m:\alpha_{ijm}=t} \kappa_{io} \tau_o, \forall o \in \mathcal{O}, t \in \mathcal{T} \tag{11}$$

4 Solution approach

Solving the stand-alone CSP is known to be NP-hard [24]. Hence, the integrated car sequencing and inventory management problem is also NP-hard. Due the large computation burden, most studies adopt metaheuristics to solve the problem. For instance, tabu search (TS) and variable neighborhood search (VNS) algorithms are frequently applied to solve the problem.

Usually, commercial softwares including CPLEX or Gurobi can only generate poor-quality solutions for large-sized instances of this integrated problem, because of a great number of binary and integer variables. A more specific solution method is needed to tackle this issue. We propose one constructive heuristic and one destructive heuristic that are both capable of solving the problems. The first heuristic is a constructive heuristic based on a relax-and-fix procedure, and the second one is an fix-and-optimize destructive heuristic combining variable neighborhood search and proximity search techniques.

4.1 A constructive heuristics

Inspired by the work of [30], the structure of the formulation in Sect. 3 indicates that relax-and-fix heuristics could efficiently solve the scheduling problem. In this approach, the original problem could be iteratively decomposed into some smaller ones, which are partially relaxed subproblems. Since the size of the subproblem (the amount of variables) is reduced significantly, the computation time needed could also be reduced. Firstly, all the binary variables are divided into several subsets. During the iterative searching process, the variables of only one of these subsets are remained to be binary, and the rest are changed to be any real number. Then the acquired small-sized problem could be solved to (near) optimality and the 0-1 variables are remained at their current values. Repeat this procedure until all the 0-1 variables are fixed.

The decomposition strategies of the binary variables determine the complexities of the subproblems [9]. We adopt an index-based decomposition strategy in this study: The variables are grouped by cars index, which relates the all the binary variables.

Specifically, $\mathcal{T}^f$ denotes the subset of car indices the corresponding binary variables are fixed at their current value, $\mathcal{N}^r$ represents the subset of car indices the corresponding variables are relaxed to be any real number , and $\mathcal{N}^o$ defines the subset of car indices the corresponding variables are optimized. All the subsets are gradually updated based on the solution of each subproblem. Let $\tilde{x}_{ij}$ be the solution at each iteration. The smaller problem (Π^{s1}) to be addressed in the next step is defined by introducing the following constraints: $x_{ij} = \tilde{x}_{ij}, i \in \mathcal{N}^f$; $x_{ij} \in \mathbb{R}^+, i \in \mathcal{N}^r$; $x_{ij} \in \{0,1\}, i \in \mathcal{N}^o$. The first constraint defines the 0–1 variables that have been previously solved and fixed during the current iteration, the second constraint describes the 0–1 variables that are relaxed, and the 0–1 variables to be solved during the current iteration are defined in the last constraint.

In light of [22] and [43], we use a decomposition pattern with overlapping car index sets ($\mathcal{N}^o$ and $\mathcal{N}^f$). Specifically, only a subset of the solved solution are fixed. We use two parameters χ and ψ to achieve this strategy, where χ is the number of car indices with integrality requirements and b is the number of car indices whose variables are fixed. By imposing $\chi < \psi$, we select a car indices from the solved ψ car indices. The heuristic procedure is summarized by Algorithm 1.

Algorithm 1 A constructive heuristic

Input: χ, ψ;
$\mathcal{N}^f = \varnothing, \mathcal{N}^o = \{i_1^o, \cdots, i_\chi^o\}, \mathcal{N}^r = \{i_k^r \mid i_k^r \in \mathcal{N} \setminus \mathcal{N}^o\}$
while $\mathcal{N}^o \neq \varnothing$ **do**
  Calculate Π^{s1}
  $x_{ij} = \tilde{x}_{ij}$
  $\mathcal{N}^f = \mathcal{N}^f \cup \{i_1^o, N_2^o, \cdots, i_\psi^o\}$
  $\mathcal{N}^o = \{t_{\psi+1}^o, \cdots, i_\chi^o, i_1^r, \cdots, i_\psi^r\}$
  $\mathcal{N}^r = \mathcal{N}^r \setminus \{i_1^r, \cdots, i_\psi^r\}$
end while

4.2 An improvement heuristic

We introduce an improvement heuristic with an adaptive VNS procedure in this subsection. The VNS procedure has been widely used to solve large mixed integer linear programming problems [43]. In this procedure, firstly we generate a feasible solution and iteratively separates the binary variables from the current solution into two groups. The binary variables in the first group are fixed at their current value and the rest will be optimized in each iteration. Note that binary variables are no longer relaxed in this procedure; therefore, a feasible solution is generated at each iteration. Similar to the relax-and-fix heuristic, the decomposition strategy is crucial to the solution quality and calculation efficiency [19]. In this study, we decompose the variables according to a combination of cars $\mathcal{N}$ and periods $\mathcal{T}$ to keep sufficiently large solution space in each iteration. The subproblem Π^{s2} to be solved is acquired by imposing constraints: $x_{ij} = \tilde{x}_{ij}, y_{ot} = \tilde{y}_{ot}, i \in \mathcal{N}^f, t \in \mathcal{T}^f$; $x_{ij}, y_{ot} \in \{0,1\}, i \in \mathcal{N}^o, t \in \mathcal{T}^o$, where $\mathcal{N}^f$ and $\mathcal{T}^f$ include the variables whose values are fixed, and $\mathcal{N}^o$ and $\mathcal{T}^o$ represent the subsets of $\mathcal{N}$ and $\mathcal{T}$, indicating the variables that will be optimized.

In the VNS framework, the solution space is to systematically examined in the following. Firstly, the incumbent solution is partially fixed according to a predefined structure,

and then the neighborhood of the current solution is explored by MILP solver. If an improvement is found, VNS will repeat these two steps; otherwise, the VND continues with the next neighborhood structure and explores the corresponding solution space. Note that the moving to the next neighborhood structure is conducted only after a pre-specified number of no improvements. In case of all structures being examined without finding an improved solution, a local optimum is identified.

To speed up the search process, we propose a reinforced parameter tuning mechanism to determine the indices of cars and periods. Specifically, the probabilities of selecting any cars and periods are initialized to 1 and will be adjusted based on two parameters, namely recency and frequency. Recency represents the number of times the corresponding cars and periods has been selected, while frequency is the number of iterations since they were last used. The probability of being selected is the weighted average of these two parameters. Let λ and γbe the number of cars and periods fixed in each iteration, respectively. The proposed heuristic is described in Algorithm 2.

Algorithm 2 An improvement heuristic approach

Given: $P = \{(\lambda, \gamma)\}$
Initial solution generation: *CS*
BS = *CS*
while $P \neq \varnothing$ **do**
  Set $\mathcal{N}^o and \mathcal{T}^o$ and $\mathcal{T}^o$ according to selected (λ, γ)
  for *num*=0 to *MI* **do**
    Run Π^{s2} on solver
    Update *CS*
    if *SS* < *BS* **then**
      $x_{ij} = \tilde{x}_{ij}, y_{ot} = \tilde{y}_{ot}$
      BS = *CS*
      $num = 0$
    end if
    Update $\mathcal{N}^o and \mathcal{T}^o$
    num++
  end for
  Delete (λ, γ) from P
  if Maximum CPU time exceed **then**
    Stop the while loop
  end if
end while
Return *BS*

where BS stands for best solution upon the current iteration, CS represents the current solution, num means the maximum number of no-improvement iterations and MI is the maximum total iterations.

5 Results

The model and algorithms are coded in Visual Studio 2015 Enterprise with CPLEX (version 12.8), and the computations are conducted on a Lenvo ThinkStation P720 with six Intel Core 2.4-G processors and 16 GB RAM, equipped with macOS Catalina. All

instances are run on CPLEX with formulation for 3600 seconds (s), and the obtained solutions serve as lower bound to compare the algorithms.

5.1 Instance generation

In this section, we evaluate the efficiency of the proposed solution approaches; a series of instances are created based on the benchmark library of CSP problem CSPLib [13] and dataset from a company producing commercial vehicles. The number of cars ranges from 100 to 1000 with an increment of 200, and each class takes the value of the number of options from (3, 10). The number of work station takes value from $\{100, 200, 300\}$. In total, we have 600 instances. The values of capacity constraints p_o, q_o are randomly selected from $\{(4, 6), (5, 8), (4, 8)\}$. The unitary inventory cost and material quantity requirements by option are randomly selected from the interval [10, 50].

5.2 Parameter setting

The solution approaches consist of two stages: The first stage is initial solution generation, using a relax-and-fix constructive heuristic; the second stage improves the solutions from the first stage by either a branch-and-cut algorithm utilized in the Cplex solver or a fix-and-optimize algorithm.

For the constructive heuristic, the value of χ determines the scale of the subproblem generated in each iteration, while ψ decides the range of the search processing. Note that the solution quality is mainly determined the first iteration since only this subproblem is technically the real relaxation of the original problem. One can use a higher value of χ to achieve a better solution quality, however, at the cost of more computational expenses. The experimental results reveal that for low values of χ (*i.e.*, ≤ 3), reasonable solutions are obtained fast. However, it is still not possible to guarantee their quality, especially for instances of large-sized instances. Finally, the parameters used are as follows, $\chi = 5, \psi = 4$ and 10 s for each iteration (in total 60 s) with a trade-off between efficiency and accuracy.

For the improvement heuristic, the neighborhood structure based on (λ, γ) is important for the efficiency of the fix-and-optimize heuristic. It serves as a strategy either allows an expensive search on a large scale of the solution space, or a dedicated search within a small solution space. Preliminary experiments show that parameter λ is the major factor influencing the quality of the solution. To maintain a reasonable-sized subproblem, we consider structures with small λ values. Through experiments, the adopted neighborhood structure is by combining values $\lambda \in \{3, 5, 9, 7\}$ and $\gamma \in \{3, 4\}$.

5.3 Results analysis

The heuristic algorithms introduced above are used to solve the instances generated in Sect. 5.1. The approach comprises two stages: generating an initial solution with a limited CPU time of 5 s and improvement with time limit 60 s. B&C stands for the branch-and-cut algorithm imbedded in CPLEX, R&F represents the relax-and-fix heuristic, and F&O refers to the fix-and-optimize improvement heuristic. The solution generated by CPLEX serves as the threshold to calculate the optimality gap.

In the first stage, we utilize either B&C or R&F approach (with the parameters valued in previous sections to obtain the initial solution. The percentages of optimality gap of

each instance are presented in the first three columns of Table 1. We observe that the B&C algorithm cannot provide a feasible solution when the number of cars exceeds 300. Algorithm R&F can generate a feasible for all instances in the limited time and can provide a better solution among the solved instances.

In the second stage, both F&O and B&C are applied to improve the initial solution. For instance, in Fig. 1, "R&F–B&C" denotes that the initial result is calculated by the relax-and-fix algorithm, and then improved by the fix-and-optimize heuristic. Overall, the attribute of the initial result has a positive impact on the improvement efficiency, and R&F + F&O outperforms other algorithm combinations. R&F outperforms B&C in generating better initial solutions, with an average optimality gap 893.5 for solved instances and also leads to better final solutions. As shown in column 4 and 6, both B&C and F&O can significantly improve the initial solution generated by B&C. Note that for instances with 300, 400 and 500 cars, B&C + B&C has difficulties in finding the first feasible solution in 3 s, but is capable of providing a feasible solution within 60 s. We observe that in column 5 and 7, F&O provides better final solutions than B&C. For instances with less than 200 cars, F&O is able to provide near optimal solutions (gap less than 1%). For instances with larger number of cars, F&O can also generate affordable solutions with a short CPU time. The overall optimality gap among all instances is 8.72%.

We take the instances with 200 cars to illustrate the computational efficiency of our heuristic approaches. As we can observe in Fig. 1, F&O significantly improves the initial solution in the first 10 s, then it becomes time-consuming to search for the globally optimal solution. The B&C has a similar performance on the same set of instances, but is outperformed by F&O in the search efficiency and solution quality.

We further compare the solution quality of our approaches with other up-to-data solution approaches. The test bed is a subset of the classical car sequencing instances from [29] and [39]. Table 2 includes the characteristics of these instances. Note that these instances have more configurations compared with the instances in Sect. 5.1, and are therefore much more computationally expensive.

The comparative solution approach is a large neighborhood search (LNS) proposed in [39], consisting of mixed integer programming (MIP) and ant colony algorithm (ACO),

Table 1 Optimality gap for the heuristic approaches

Instance size	Initial solution		Initial solution & improvement			
	R&F (%)	B&C (%)	B&C–B&C (%)	R&F–B&C (%)	B&C–F&O (%)	R&F–F&O (%)
100	867	1521	5.85	4.79	2.57	0.47
200	920	1627	98.69	86.64	45.01	0.98
300	1735	–	245.13	88.11	–	2.41
400	2462	–	435.44	103.03	–	2.52
500	2678	–	696.67	134.08	–	5.41
600	3126	–	–	145.74	–	10.34
700	3633	–	–	159.06	–	15.23
800	3905	–	–	158.03	–	21.36
900	3966	–	–	211.83	–	28.44
1000	4785	–	–	250.92	–	32.09
Average	1574	2807.7	296.36	134.23	48.21	8.72

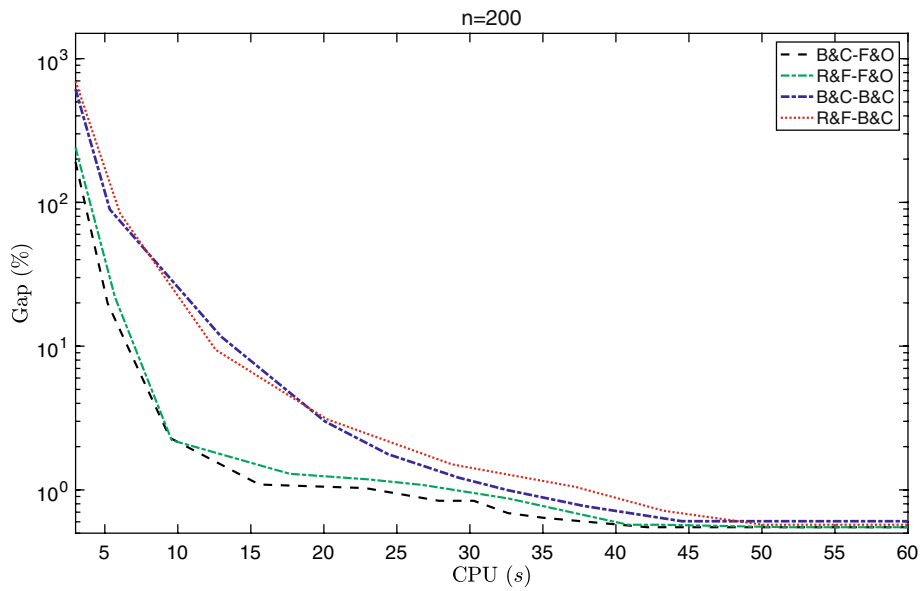

Fig. 1 Optimality gap for instances with $n = 200$ The gaps between the solutions returned by the four combination of heuristics, B&C+F&O, R&F+F&O, B&C+B&C and R&F+B&C, and the threshold value returned by CPLEX

Table 2 The classical car sequencing instances

# Cars	# Options	# Config.	*p*/*q* ratios
100	5	17–30	1/(2,3,5), 2/(3,5)
300	5	19–26	1/(2,3,5), 2/(3,5)
500	7	21–22	1/(2,3,5), 3/(4,5)

with a window size equal to the least common multiplier (LCM) of the subsequences sizes of the options. The parameter setting of this so-called LNS (MIPACO-LCM) algorithm is determined as in [39].

For the sake of fair comparison, we run the experiments 30 times to obtain statistically valid results and each experiment terminate when the solution quality does not improve any more (for 100 iterations). The results are presented in Table 3, where B-UB stands for best upper bound (feasible solution); M-UB is the mean of the value of 30 upper bounds; SD is the standard deviation associated with the mean value; Gap is calculated as $(MeanUB - MeanLB)/MeanLB$. The values of the mean of lower bound for these instances are given in [39].

The first column shows the problem instance. The next four columns show the results for LNS(MIPACO-LCM) in [39]. The last four columns summarize the results for R&F + F&O.

Comparing the mean of best solutions found, we observe that our heuristic obviously outperforms the LNS algorithms. Our approach is capable of finding a better solution in most of the instances. Regarding instance 100-35, our approach is only slightly worse than LNS, but the best solution obtained is still better (208.11 vs. 208.94). Similar observations are found for instances 300-62 and 500-88. Observing

Table 3 A comparison of LNS and R&F + F&O on 30 runs

Instance	LNS (MIPACO-LCM)				R&F + F&O			
Instance	B-UB	M-UB	SD	Gap (%)	B-UB	M-UB	SD	Gap (%)
100-22	252.04	255.71	1.88	9.59	253.66	**253.66**	0.00	8.33
100-35	208.94	**210.17**	0.78	5.80	208.11	211.30	1.33	5.93
100-64	232.84	234.65	1.03	5.35	232.15	**232.74**	0.79	4.22
100-77	166.32	169.13	1.28	11.93	167.43	**163.05**	0.00	7.64
100-82	225.03	229.18	2.07	10.71	226.06	**222.85**	0.00	7.38
100-94	168.93	173.48	2.20	13.59	167.36	**169.77**	2.04	10.95
300-08	548.57	548.57	0.00	13.14	549.22	**521.64**	1.01	9.38
300-14	712.59	**712.59**	0.00	5.58	713.47	714.93	0.56	6.22
300-53	665.57	669.34	0.70	15.30	642.64	**642.64**	0.00	5.14
300-56	467.53	467.70	0.93	11.89	451.04	**452.73**	0.45	8.41
300-62	728.48	**733.61**	2.61	8.17	719.45	741.03	2.77	9.22
300-78	517.41	**517.41**	0.00	10.63	521.69	523.64	0.45	11.95
500-14	617.25	617.25	0.00	26.04	582.37	**582.37**	0.00	21.38
500-27	1398.77	1398.77	0.00	11.51	1385.62	**1385.62**	0.00	10.64
500-65	735.80	735.80	0.00	24.78	719.32	**719.32**	0.00	20.32
500-74	672.28	672.28	0.00	17.94	648.02	**648.02**	0.00	14.04
500-79	1274.54	**1278.98**	2.03	5.63	1243.16	1284.07	1.55	6.04
500-88	1057.46	1057.46	0.00	13.33	1023.35	**1016.77**	2.01	9.38

The observed best mean value of upper bound for each instance is marked as bold

Table 4 Results of the algorithm with fifteen months of real-world data

Problem				Manual method		R&F + F&O				
n	*o*	*m*	p_o:q_o	Vio cost	Opn con	Vio cost	Opn con	Cost sv (%)	Opt gap (%)	CPU time (s)
200	18	80	4:6	876	12.12	847	12.63	9.21	2.61	1.75
200	22	80	4:6	1132	10.83	1095	11.37	10.33	7.92	1.82
200	22	80	5:7	918	11.51	901	11.88	8.74	5.77	1.93
200	22	75	4:6	1315	9.60	1274	11.61	11.35	8.01	2.07
200	20	80	4:6	916	11.29	874	11.90	5.89	5.91	1.15
400	22	80	3:6	779	12.82	771	13.12	5.74	2.62	5.20
400	22	70	4:6	824	12.25	817	12.39	6.38	3.68	5.32
400	22	80	4:6	785	12.38	761	12.56	4.81	2.57	5.22
400	16	80	2:4	741	13.52	731	13.52	5.14	2.61	5.10
400	22	80	4:6	897	10.84	852	11.97	8.67	5.73	5.35
600	19	80	4:6	892	11.90	850	12.59	7.49	5.66	11.74
600	20	65	4:8	853	12.31	842	12.41	6.32	4.75	11.10
600	22	80	4:6	922	11.32	897	12.13	6.91	6.83	11.08
600	22	80	3:5	1235	11.76	1195	11.91	9.77	8.94	11.66
600	21	80	4:6	1201	11.73	1175	12.17	9.36	9.91	11.31
Avg				952.40	11.75	925.47	12.27	7.74	5.57	6.12

the results in columns SD, we find that our approach is more stable in generating solutions with a smaller value of deviation.

It is also interesting to observe that our approach has similar performance as LNS on different instances. Both of them find high-quality results on instances 100-35,

100-64, etc., and have solutions with relatively large gaps on instances 500-14, 500-65, etc. This is due to the fact that both approaches use MIP formulations in searching for the next feasible solution. The drawback is the uncertain search efficiency for large instances because of large number of 0-1 and integer variables.

5.4 System implementation and evaluation

In this section, we describe the implementation of a decision support system with integrated heuristic approaches, and the benefits to the company using such DSS.

We started working on this problem with the car manufacturing company in 2020. At that time, the company has already utilized an enterprise resource planning (ERP) system to deal with the production and inventory planning problem. However, this system is lack of optimization-based decision-making capability to provide daily operation instructions. The production and inventory decisions were made mainly with non-optimization tools and expert experience.

The main purpose of the collaboration with the company was to streamline their decision-making process and replace their old planning procedures by automatic DSS with advanced solution approaches. The DSS was developed in Microsoft Visual C++ 2015 environment. The algorithms were coded in C++ with IBM ILOG CPLEX concert technology (version 12.8) and were run on top of their information system developed in a Java environment.

The company's previous method was developed a couple of years ago by a consulting agency. That method treats production and inventory as separate problems. The logic used to generate production planning is a greedy algorithm, which prioritizes the production requirements and adjusts the production schedules based on such priority preference. Their method is straightforward and has been used for a number of years. However, such a heuristic requires a lot of planning experiences which is difficult to be quantified and systemized. Considering the complexity of the problems, we believe that our integrated approach could outperform their manual method.

We conducted comparison experiments to evaluate the benefits of our integrated approach. The real-world instances generated in Sect. 5.1 are used as the test bed and heuristic R&F + F&O is adopted. The main parameters $(n, o, m, p_o : q_o)$ of 15 instances of real-world production and inventory data are shown in Table 4, where each row represents one specific production and inventory planning. Note that the value of m (number of work stations) is not the same across different instances, due to the ordinary maintenance requirements on the stations.

Table 4 describes the solutions of the experiments. "Vio cost" represents the constraint violations costs. "Opn con" means the non-level option consumption cost. "Cost sv" means the total cost savings acquired by the heuristic approach compared with company's current manual method. The results returned by the R&F–F&O heuristic are given in columns 8 and 9. For comparison, the improvement on total cost made by the R&F–F&O heuristic is summarized in column 10. We observe that our heuristic approach can save 7.74% of the total cost. Besides, the optimality gap between the final result and the threshold result is given in column 11, and the average percentage value is 5.57. The last column of this table is computation time (CPU) of the R&F + F&O heuristic when the solution does not improve anymore.

Form Table 4, we observe that our heuristic would improve the production planning by reducing the cost of constraint violations, from an average of 953 to 926 (2.8% decrease), and increase inventory turnovers by an average of 4.4%. Our algorithm is capable of improving the constraint violation cost and inventory turnovers in all instances, which are the key demand for the company. The results also indicate that our heuristic approach could improve the constraint violation costs in almost each of these 15 instances and reduce total cost by an average of 7.74%. Compared with the generated large instances, our algorithm can find a satisfiable solution in 12 s. Thus, using our heuristic could improve the current method.

6 Conclusion

In this paper, we extend the classical CSP by addressing block batch constraints, which has been frequently encountered in the auto industry in practice. Especially for manufactures changing from mass standardization to mass customization, the large cumulative setup times of frequently changed fixtures often disturb the normal production rhythm. Besides, the block production also deteriorates the level production requirements, which deserves a more dedicated evaluation method of non-level production cost, so as to better tune the production sequence. We propose a new mixed-integer linear programming (MILP) formulation to tackle the proposed issues. Aware of the NP-hardness, we develop two math-heuristic algorithms to handle the large-scale problems efficiently. The first one is a constructive heuristic that is capable of fast providing acceptable solutions in a limited computation time. The acquired solution can also be used as the starting point for the second improvement heuristic, which incorporates a data-driven adaptive variable neighborhood search to produce high-quality solutions to real-life-size instances within acceptable runtimes. Extensive computational experiments are reported, and several managerial insights into the planning process are given.

In the future, it would be interesting to consider other practical constraints in the car production process, i.e., mixed-line production structure, car sequencing with limited buffers. Obviously, the incorporation of extra constraints would increase the difficulties of solving the complex problem. Hence, more advanced solution algorithms are deserved to be thoroughly studied.

Abbreviations

CPU: Computation time; CSP: Car sequencing problem; ERP: Enterprise resource planning; F&O: Fix-and-optimize; HAPs: Hazardous air pollutants; MILP: Mixed integer linear programming; R&O: Relax-and-fix; VNS: Variable neighborhood search; VOCs: Volatile organic compounds.

Acknowledgements

The authors would like to thank the support of Automotive Data of China (Tianjin) Co., Ltd. Tianjin, China, for providing research data and experiment environment.

Author's contributions

Yingjie Yu was responsible for the overall design and implementation of the work. Xiaochun Lu and Tao Zhao contributed in the research design, model formulation and algorithm development. Minjiao Cheng has collected the data and the computational results. Lin Liu and Wenchao Wei have prepared the manuscript and overall review. All authors read and approved the final manuscript.

Funding

This work is partially supported by the National Science Foundation of China (71801013, 72111530098), Beijing Social Science Foundation (18GLC078), and Fundamental Research Funds for the Central Universities with Grant Number 2020JBW004.

Availability of data and materials
The data and material are available upon request.

Declarations

Competing interests
The authors declare that they have no competing interest.

Author details
[1]Automotive Data of China (Tianjin) Co., Ltd., Tianjin, China. [2]School of Economics and Management, Beijing Jiaotong University, Beijing, China. [3]China Automotive Technology and Research Center Co., Ltd., Tianjin, China.

Received: 2 November 2021 Accepted: 10 March 2022
Published online: 28 March 2022

References

1. D. Andreas, K. Alf, Sequencing JIT mixed-model assembly lines under station-load and part-usage constraints. Manag. Sci. **47**(3), 480–491 (2001)
2. J. Bautista, J. Pereira, B. Adenso-Díaz, A beam search approach for the optimization version of the car sequencing problem. Annu. Oper. Res. **159**, 233–244 (2008)
3. J. Bautista, J. Pereira, B. Adenso-Díaz, A GRASP approach for the extended car sequencing problem. J. Schedul. **11**, 3–16 (2008)
4. M.E. Bergen, P. van Beek, T. Carchrae, Constraint-based vehicle assembly line sequencing, in *Advances in Artificial Intelligence*. ed. by E. Stroulia, S. Matwin (Springer, Berlin, 2001), pp. 88–99
5. S. Bysko, J. Krystek, S. Bysko, Automotive Paint Shop 4.0. Comput. Ind. Eng. **139**, 105546 (2020)
6. J.-F. Cordeau, G. Laporte, F. Pasin, Iterated Tabu search for the car sequencing problem. Eur. J. Oper. Res. **191**(3), 945–956 (2008)
7. M. Dincbas, H. Simonis, P. Van Hentenryck, Solving the car sequencing problem in constraint logic programming, in *European Conference on Artificial Intelligence (ECAI-88* (1988)
8. A. Drexl, A. Kimms, Sequencing jit mixed-model assembly lines under station load and part-usage constraints. Manag. Sci. **47**(3), 480–491 (2001)
9. L.F. Escudero, J. Salmeron, On a fix-and-relax framework for a class of project scheduling problems. Ann. Oper. Res. **88**, 140–163 (2005)
10. B. Estellon, F. Gardi, K. Nouioua, Two local search approaches for solving real-life car sequencing problems. Eur. J. Oper. Res. **191**(3), 928–944 (2008)
11. H. Gavranović, Local search and suffix tree for car-sequencing problem with colors. Eur. J. Oper. Res. **191**(3), 972–980 (2008)
12. C.A. Geffen, S. Rothenberg, Suppliers and environmental innovation the automotive paint process. Int. J. Oper. Prod. Manag. **20**(2), 166–186 (2000)
13. I.P. Gent, T. Walsh, CSPlib: a benchmark library for constraints, in *Principles and Practice of Constraint Programming—CP'99*. ed. by J. Jaffar (Springer, Berlin, 1999), pp. 480–481
14. U. Golle, Fitness landscape analysis and design of metaheuristics for car sequencing. Car Sequenc. Probl. Anal. Solut. Methods **100**, 66 (2011)
15. U. Golle, F. Rothlauf, N. Boysen, Iterative beam search for car sequencing. Ann. Oper. Res. **226**, 239–254 (2015)
16. J. Gottlieb, M. Puchta, C. Solnon, A study of greedy, local search, and ant colony optimization approaches for car sequencing problems, in *Applications of Evolutionary Computing*. ed. by S. Cagnoni, C.G. Johnson, J.J.R. Cardalda, E. Marchiori, D.W. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G.R. Raidl, E. Hart (Springer, Berlin, 2003), pp. 246–257
17. M. Gravel, C. Gagne, W.L. Price, Review and comparison of three methods for the solution of the car sequencing problem. J. Oper. Res. Soc. **56**, 1287–1295 (2005)
18. N.G. Hall, C.N. Potts, Supply chain scheduling: batching and delivery. Oper. Res. **51**(4), 566–584 (2003)
19. S. Helber, F. Sahling, A fix-and-optimize approach for the multi-level capacitated lot sizing problem. Int. J. Prod. Econ. **123**(2), 247–256 (2010)
20. K.S. Hindi, G. Ploszajski, Formulation and solution of a selection and sequencing problem in car manufacture. Comput. Ind. Eng. **26**(1), 203–211 (1994)
21. A. Hottenrott, L. Waidner, M. Grunow, Robust car sequencing for automotive assembly. Eur. J. Oper. Res. **291**(3), 983–994 (2021)
22. R.J.W. James, B. Almada-Lobo, Single and parallel machine capacitated lotsizing and scheduling: new iterative MIP-based neighborhood search heuristics. Comput. Oper. Res. **38**, 1816–1825 (2011)
23. A. Joly, Y. Frein, Heuristics for an industrial car sequencing problem considering paint and assembly shop objectives. Comput. Ind. Eng. **55**(2), 295–310 (2008)
24. T. Kis, On the complexity of the car sequencing problem. Oper. Res. Lett. **32**(4), 331–335 (2004)
25. U.V. Manoj, J.N.D. Gupta, S.K. Gupta, C. Sriskandarajah, Supply chain scheduling: just-in-time environment. Ann. Oper. Res. **161**(1), 53–86 (2008)
26. S. Moons, K. Ramaekers, A. Caris, Y. Arda, Integrating production scheduling and vehicle routing decisions at the operational decision level: a review and discussion. Comput. Ind. Eng. **104**, 224–245 (2017)
27. I. Moya, M. Chica, J. Bautista, Constructive metaheuristics for solving the car sequencing problem under uncertain partial demand. Comput. Ind. Eng. **137**, 106048 (2019)

28. B.D. Parrello, W.C. Kabat, L. Wos, Job-shop scheduling using automated reasoning: a case study of the car-sequencing problem. J. Autom. Reason. **2**, 1–42 (1986)
29. L. Perron, P. Shaw, V. Furnon, Propagation guided large neighborhood search, in *Principles and Practice of Constraint Programming—CP 2004*. ed. by M. Wallace (Springer, Berlin, 2004), pp. 468–481
30. Y. Pochet, L.A. Wolsey, *Production Planning by Mixed Integer Programming* (Springer, Heidelberg, 2006)
31. M. Prandtstetter, G.R. Raidl, An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. Eur. J. Oper. Res. **191**(3), 1004–1022 (2008)
32. M. Puchta, J. Gottlieb, Solving car sequencing problems by local optimization, in *Applications of Evolutionary Computing*. ed. by S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G.R. Raidl (Springer, Berlin, 2002), pp. 132–142
33. C.C. Ribeiro, D. Aloise, T.F. Noronha, C. Rocha, S. Urrutia, An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. Eur. J. Oper. Res. **191**(3), 596–611 (2008)
34. N.A. Schmid, V. Limère, B. Raa, Mixed model assembly line feeding with discrete location assignments and variable station space. Omega **102**, 102286 (2021)
35. M. Siala, E. Hebrard, M.J. Huguet, A study of constraint programming heuristics for the car-sequencing problem. Eng. Appl. Artif. Intell. **38**, 34–44 (2015)
36. M. Simonetto, S. Arena, M. Peron, A methodological framework to integrate motion capture system and virtual reality for assembly system 4.0 workplace design. Saf. Sci. **146**, 105–561 (2022)
37. C. Solnon, V.D. Cung, A. Nguyen, C. Artigues, The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. Eur. J. Oper. Res. **191**, 912–927 (2008)
38. F. Taube, S. Minner, Resequencing mixed-model assembly lines with restoration to customer orders. Omega **78**, 99–111 (2018)
39. D. Thiruvady, K. Morgan, A. Amir, A.T. Ernst, Large neighbourhood search based on mixed integer programming and ant colony optimisation for car sequencing. Int. J. Prod. Res. **58**(9), 2696–2711 (2020)
40. C.A. Ullrich, Integrated machine scheduling and vehicle routing with time windows. Eur. J. Oper. Res. **227**(1), 152–165 (2013)
41. D.-Y. Wang, O. Grunder, A. El Moudni, Integrated scheduling of production and distribution operations: a review. Int. J. Ind. Syst. Eng. **19**, 94–122 (2015)
42. T. Warwick, E.P. Tsang, Tackling car sequencing problems using a generic genetic algorithm. Evol. Comput. **3**, 267–298 (1995)
43. W. Wei, L. Guimarães, P. Amorim, B. Almada-Lobo, Tactical production and distribution planning with dependency issues on the production process. Omega **67**, 99–114 (2017)
44. N. Zufferey, M. Studer, E.A. Silver, Tabu search for a car sequencing problem, in *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference* (2006). pp. 457–462

Publisher's Note