Online resequencing of buffers for automotive assembly lines

Malte Lübben, Sven Pries, Celso Gustavo Stall Sikora

Institute for Operations Research, University of Hamburg

Moorweidenstraße 18, 20148, Hamburg

Abstract

Mixed-model assembly lines are state of the art in automotive production systems. Because of the high number of customizable options which can be ordered in a vehicle, there is a huge variety of possible products. An important problem in this context is the sequencing of such products. Inevitably, there will be deviations from the intended production sequence in the course of production, as disruptions occur. The products must then be resequenced to ensure an optimal sequence. In this work, we consider the usage of a buffer (in form of an automated storage and retrieval system) between the paint shop and the final assembly to resequence the orders. We consider a high number of variants and, with this, a random input sequence for the buffer. Additionally to the physical resequencing in the buffer, the options get decoupled from the products. That allows virtual resequencing, in which parts and materials are interchanged. The dispatching selection must be made without full information in an online problem. To solve this problem, different heuristics and a lookahead algorithm are applied to minimize the amount of utility work in a paced automotive assembly line.

Keywords: Online resequencing, automotive buffer, mixed-model assembly lines, utility work

Computers & Industrial Engineering 2022
DOI: 10.1016/j.cie.2022.108857

1. Introduction

When the mass production of cars with assembly lines was introduced by Henry Ford in 1913, there was only one model that could be produced at a time (Ford, 2013). In the following century, as cars got more and more affordable and widespread, the customers became more demanding. The car manufacturers had to adapt and had to be more responsive to customer needs, which led to more variants being offered. This adaption process resulted in mass customization where the low-cost processes of mass production and the flexibility of individual customization were brought together (Boysen et al., 2012).

*Corresponding author
Email address: celso.sikora@uni-hamburg.de (Celso Gustavo Stall Sikora)

When looking at today's automotive industry, the amount of customization options is enormous. Due to the combinatorial nature of customized options, some car models are available in up to 10^{32} variants (Meyr, 2004). To produce such a huge variety efficiently, automotive production usually consists of a series of flow-shop production systems such as the press shop, the paint shop, and the final assembly (FA), as shown in figure 1. Each section may have very different optimal production sequences: for the paint shop, the color is of most importance, while the final assembly is affected by the products' different processing times of each assembly operation. The production sequence is, however, restricted due to the organization in a flow-shop system. A degree of flexibility can be implemented with buffers between the sections, giving some resequencing capability. Furthermore, not only the objectives of each department may differ, but also the pre-defined sequence can get disturbed during the process. For instance, the paint shop has very high rework rates (65% to 85% are reported by Boysen et al. (2012)). This way, the operation of buffers is important to resequence the products in a way that the resulting production sequence is feasible.

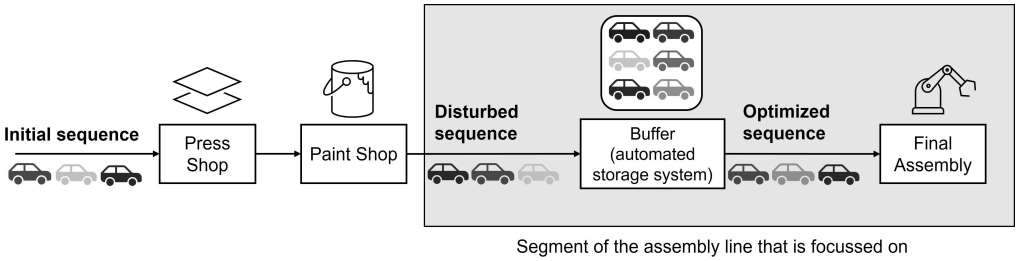

Figure 1: Schematic structure of the production in the automotive industry and portrayal of the buffer location.

Considering a constant conveyor belt speed and different processing times for every option, the challenge for the final assembly is to find a sequence in which the changing processing times do not exceed the stations' planned working time. A long sequence of products requiring long operations may cause work overload. Therefore, the production sequence after the paint shop must be altered, respectively resequenced, to reduce costs in the final assembly. For this purpose, we consider a buffer in between these two sections to reorder the products and to optimize the sequence for the final assembly. The final assembly consists of a series of stations with predetermined tasks being performed at each station. The design of an assembly line is known in the literature as the assembly-line balancing problem (ALBP) (Boysen et al., 2009a). Given a set of tasks with certain processing times and precedence relations, the best possible assignment of tasks to stations must be found. The basic version of the problem considers only one product and is named the simple assembly-line balancing problem (SALBP) (Boysen et al., 2009a). In this version, tasks are assigned in a way that the stations are not allowed to exceed the cycle time, while the precedence relations between tasks must be followed. In an assembly line with multiple products, some products may exceed the cycle time if they are compensated with simpler products in an adequate production sequence.

If the assembly line balancing is given, the following problem is the sequencing of products. As we consider a buffer in a mixed-model assembly line, an important problem hereby is the selection

decision, which product from the buffer must be dispatched in every cycle. In this paper, we investigate this problem for a high number of different models and aim at finding a good selection rule for the buffer control. The input sequence is modeled as random resulting in an online resequencing problem. The main contribution of this paper is a method for simultaneous physical and virtual resequencing of products. While in physical resequencing the vehicles themselves are swapped, virtual resequencing changes the customer an order is assigned to. The reassignment of an order allows for further sequence flexibility and potential better sequences as shown in an example in Section 3 and the results on Section 5. The best product sequences are only achievable when both forms of resequencing are available.

Although the resequencing of the production can improve the utilization of the assembly lines, the reordering of products is not always desired in large automotive manufacturers, since the planning of the sequence has to be coordinated with the suppliers (Boysen et al., 2012). In fact, two automotive manufacturers contacted to discuss about resequencing after the paint shop disclosed that their main objective is to maintain the planned sequence. However, long delays may require that the sequence is altered when another vehicle must be selected. For such cases, the manufacturers could resequence the products based on the costs of the final assembly. For the cases in which a sequence change is unavoidable, the proposed method can be used to select the best sequence in respect to production costs from the available products.

This article is an extended version of a PhD thesis chapter published in (Sikora, 2022, p. 133-159). In the following, the relevant literature is reviewed in section 2 and then the problem description is presented in section 3. This is followed by the presentation of the solution approaches (section 4), which are compared in a computational study in section 5. Finally follows an evaluation of the results and a conclusion (sections 6 and 7).

2. Literature review

The problem describing the adjustment of the sequence is called the resequencing problem (Boysen et al., 2009a). Most papers published in the field of resequencing mixed-model assembly lines can be systematically classified by the framework presented by Boysen et al. (2012). According to this classification, resequencing research can be distinguished by their characteristics in the following criteria: resequencing objective, planning horizon, trigger, resequencing object, and solution approach.

Boysen et al. (2009b) name three existing objectives for sequencing mixed-model assembly lines. The objectives for resequencing for the final assembly can be derived from them and are based on the same structures: level scheduling (LS), car sequencing (CS), and mixed-model sequencing (MMS). Level scheduling aims at maintaining the material supply at a steady level. In contrast, MMS and CS are workload-oriented approaches that minimize work overload. They differ in the level of detail, MMS is very detailed and considers operation times, worker movements, station borders, and other available characteristics. The car sequencing, on the other hand, does not model the process times directly. In such an approach, the principle is to formulate sequencing rules of

type $H_o : N_o$, which means that out of a sequence of N_o positions, a maximum of H_o may contain option o. Work overload can then be avoided when a sequence, which does not violate any of these rules, is found (Boysen et al., 2012). One advantage of this approach is that it requires less effort to obtain data. Car sequencing at multi-model assembly lines was recently investigated by Günay & Kula (2020) and Kampker et al. (2019), whereas level scheduling was dealt with by Cao & Sun (2019) and Taube & Minner (2018). The mixed-model sequencing approach for resequencing prior the FA was researched so far by Gujjula & Günther (2009), Boysen et al. (2011), Franz et al. (2014), Franz et al. (2015) and Mosadegh et al. (2017).

The second classification characteristic is the planning horizon, including the information availability and the design of the decision variable. Since a product is dispatched in every product cycle, many single decisions within a resequencing problem must be taken. Thus, the problem can either be considered as dynamic or as static in a rolling planning environment. When the sequence of products entering the buffer is unforeseeable, it even becomes an online problem (Boysen et al., 2012). The dynamic mixed-model resequencing problem for the final assembly has rarely been analyzed so far. Still, some research can be found at Choi & Shin (1997), Inman & Schmeling (2003), Franz et al. (2015) and more recently at Bock & Boysen (2021).

The trigger of resequencing can either be proactive or reactive. Resequencing is reactive when it is triggered by unforeseen disturbances like machine breakdowns, material failure, or urgent orders. In contrast, proactive resequencing aims at shuffling a sequence by the needs of the FA. Schumacher et al. (2018) propose a supplement to the framework of Boysen et al. (2012) by distinguishing between proactive sequencing for multiple departments and proactive sequencing within the final assembly.

In terms of the resequencing object, it can be distinguished between physical and virtual resequencing. For physical resequencing, a buffer is needed in which the products can be stored and reshuffled. A general distinction can be made between four different types: automated storage and retrieval systems (ASRS), mix banks, pull-off tables, and insert buffers (Boysen et al., 2012). In ASRS, each place in the buffer can be individually controlled for placement or removal of products. They are widely used in the automotive industry (Günay & Kula, 2020). In the context of resequencing, they have been investigated so far by Inman & Schmeling (2003), Franz et al. (2014), Franz et al. (2015) as well as Bock & Boysen (2021). A mix bank consists of multiple parallel lines, which makes it possible to store products in multiple queues. Another option is to use pull-off tables to directly remove or reinsert products from the assembly line. This buffer consists of unitary spaces beside the conveyor belt that can be used to store a single product. Boysen et al. (2012) also name a fourth buffer type, the insert buffer, which has not been dealt with in resequencing literature so far.

In contrast to physical resequencing, the customer orders are shuffled in virtual resequencing instead of physically reordered. It also contains the idea of decoupling orders from vehicles: Products which are not yet assembled can be assigned to a different order as previously planned. Virtual resequencing is first investigated by Inman & Schmeling (2003) and later supplemented by Xu &

Zhou (2016) and Cao & Sun (2019). It has the advantage that there are no setup or operation costs for buffers. As a disadvantage, it requires more planning effort. None of the researchers so far investigate the combined performance of physical and virtual resequencing.

In the literature, the solution approaches for the resequencing problem are either classified as exact, heuristic, or simulative. Considering practically relevant instance sizes and the nature of an online problem, an exact procedure which guarantees a solution cannot be found in a reasonable decision period (Gujjula et al., 2011). Tsai (1995) shows that the sequencing problem for mixed-model assembly lines to minimize overload situations is NP-hard.

Table 1 contains a summary of the articles mentioned above. As before, they are classified by the characteristics introduced by Boysen et al. (2012).

Table 1: Literature overview for the resequencing of mixed-model assembly lines.●: the respective paper explicitly deals with resequencing, ○: the respective paper deals with sequencing.

Author(s) (Year)	(Re-) sequencing objective			Information availability		Buffer type			Resequencing trigger		Resequencing object		Solution method		
	MMS	CS	LS	static	dynamic	ASRS	mix-bank	pull-off table	proactive	reactive	virtual	physical	exact	heuristic	simulation
Bock & Boysen (2021)	●			●		●			●			●	●	●	●
Boysen et al. (2011)	○			●				●					●	●	●
Cao & Sun (2019)			○	●					●		●		●	●	●
Choi & Shin (1997)			○		●		●		●			●		●	●
Franz et al. (2014)	●			●		●				●		●	●	●	●
Franz et al. (2015)	●				●	●				●		●	●	●	●
Gujjula & Günther (2009)	●			●				●		●		●	●	●	●
Gujjula & Günther (2010)	○			●										●	●
Gujjula et al. (2011)	○			●										●	●
Günay & Kula (2020)		●		●			●			●		●	●		●
Inman & Schmeling (2003)		○	○		●	●					●		●	●	●
Kampker et al. (2019)		●						●	●			●	●	●	●
Mosadegh et al. (2017)	○			●									●	●	●
Taube & Minner (2018)			●	●			●		●			●	●	●	●
Xu & Zhou (2016)				●					●		●			●	●
Proposed method (2022)	●				●	●			●		●	●		●	●

Of the articles which discuss the MMS, only Bock & Boysen (2021), Franz et al. (2015), Franz et al. (2014), Gujjula & Günther (2009), Günay & Kula (2020), Kampker et al. (2019) and Taube & Minner (2018) have a resequencing object (instead of a sequencing one). Gujjula & Günther (2009) present a mixed-integer program (MIP) and a local search algorithm to resequence/postpone blocked orders with pull-of-tables. A blocked order hereby describes the case when an important part of an order is missing. Then the order gets deferred from the planned sequence and has to be inserted later when the corresponding parts are available. Likewise, Franz et al. (2014) and Franz et al. (2015) look at blocked orders and insertion strategies with ASRS buffer types. Hence, all three articles discuss reactive resequencing.

Boysen et al. (2011) propose a problem formulation by introducing the skip policy instead of the side-by-side policy for processing overload. The skip policy means, that an overloaded cycle is skipped by the regular workers and gets completely assigned to a utility worker. Such workers are trained to perform any task and are therefore more costly than regular workers. The side-by-side variant, on the other hand, represents the version that the regular worker processes the overloaded cycle together with the utility worker. Gujjula & Günther (2010) and Gujjula et al. (2011) focus on heuristics to handle large problem instances in a static environment. Mosadegh et al. (2017) survey the MMS problem by taking into account stochastic processing times.

Choi & Shin (1997) and Inman & Schmeling (2003) investigate dynamic resequencing for level scheduling and car sequencing. Franz et al. (2015) contribute by proposing a mathematical model for this problem with the mixed-model sequencing approach. All these works deal with a dynamic problem. The online problem on the other hand was recently investigated by Bock & Boysen (2021), where they propose a real-time control system for a resequencing buffer. Additionally to the workflow along the assembly line, they also consider the part feeding processes. The main difference between the Bock & Boysen (2021) and the proposed approach lays on how the logistic part of the process is modeled. In Bock & Boysen (2021), the production of the required parts in the feeder lines is considered in the resequencing, for which the set-up costs are minimized. This consideration goes along with the Just-in-Sequence concept, in which the parts are delivered to the stations in the same sequence as the product orders. In contrast to this work, we model the reorganization of parts within the stations of the final assembly (see section 3). The products are modeled as a selection of multiple options, whose parts are assumed to be available at the workstations. This approach is less adequate to be implemented in a Just-in-Sequence environment, however, also brings advantages such as the possibility of virtual resequencing.

In this paper, we focus on proactive mixed-model sequencing by employing an ASRS buffer. Our problem is investigated in a proactive setting with a mixture of physical and virtual resequencing. As further described in section 3, the input sequence in the buffer is considered to be random, so that the problem is defined as dynamic and online. The proposed solution approaches are based on heuristics and incomplete enumeration combined with a simulator, which are described in section 4.

3. Problem description

The proposed problem is related to the operation of a buffer before the beginning of the final assembly. In the automotive industry, this buffer is generally located between the paint shop and the final assembly (Boysen et al., 2009b). According to Boysen et al. (2012), the paint shop is the most unreliable part of the process, since even small imperfections require repainting iterations. Furthermore, the sequencing of the models to be produced is a multi-objective problem that must consider multiple departments in the industry. Therefore, it is assumed that the sequence before the buffer is defined based on another objective function and can be prone to errors. Concerning the final assembly objective, the entry sequence of the buffer is herein modeled as random, while the products can be reordered to better fit the aims of the next assembly phase.

The selected buffer for the application is an automated storage and retrieval system, illustrated in figure 2. This buffer presents the highest degree of flexibility since the products can be stored and received from any position in any order. Other buffer types could also be modeled within the problem framework.

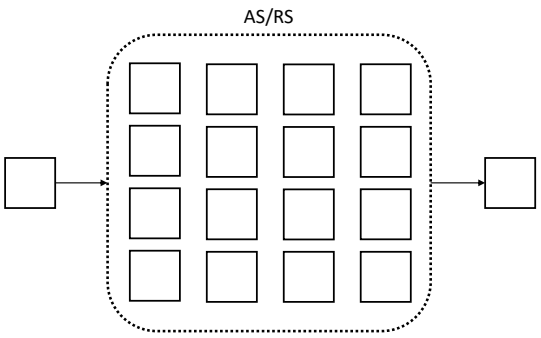

Figure 2: Illustration of an automated storage and retrieval system (AS/RS) buffer.

The final assembly line structure is considered to be given, that is, every task is already assigned to a workstation. For the instance generation described further in section 5, each task may present one or multiple options. An option is a workpiece that is mounted or an activity that must be performed on the product and can vary based on the respective order. The mounting of a sunroof, for instance, may depend on its size. The task 'mount sunroof' can then be modeled based on multiple options: 'normal sunroof', 'panoramic sunroof', or even 'no sunroof'. The processing time of each task is defined by the respective option and can also be zero (as in the case of 'no sunroof'). A product variant is defined by selecting one option per task. Therefore, the total number of variants produced in the line is given by the product of the number of options of all tasks. The entry sequence of products is created based on a random choice for each of the options.

The cycle time (CT, or the interval between the production of two vehicles) of automotive production nowadays is often less than one minute (Emde & Gendreau, 2017). In such an environment, the selection of the next model within the stored products in the buffer must be made within this time limitation. In every cycle time, a product is removed from the buffer, while the

next random variant enters. As the future input is unknown at the moment of the decision, the selection problem is said to be online.

The assembly line after the buffer is assumed to be continuous with fixed start and end borders. The workers move along with the products on a conveyor belt, returning to their initial position after completing a product to start the next one in the next cycle time (CT). Since multiple variants are present, the station load $L_{j,s}$ (sum of the tasks' processing times of product j in station s) varies according to the product variant. The workings of such a station are illustrated for an example in figure 3. Each row corresponds to the assembly of a product. When the worker finishes the assembly, he or she can start working on the next piece, which is phased CT-equivalent length units to the left. Note that the station length (l_s) is usually longer than the cycle time equivalent so that the products requiring longer than the cycle time can be processed as well. This results in a starting position of the next cycle different from the left border. With this extra length, the products can be scheduled in a way that the processing times of more complex variants are partly compensated with less complex ones. If a variant requires such a processing time that the worker would surpass the stations' border, a utility worker can be used to aid in the production, as is illustrated for the fourth product of figure 3. The objective of the resequencing problem is to dynamically select a production sequence that minimizes the use of such utility workers.

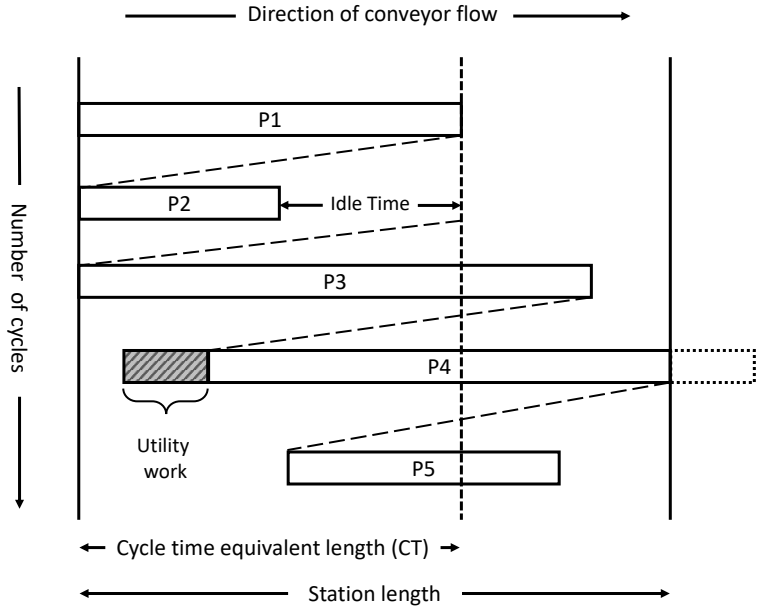

Figure 3: Example of a production sequence in a paced line using utility work.

Formally, the position of the worker in station s after the processing of the j^{th} product is given by

$$Pos_{s,j} = \max\{0, \min\{Pos_{s,j-1} + L_{j,s} - CT, l_s\}\}, \tag{1}$$

while the required utility work for the j^{th} product in station s is defined as

$$UW_{s,j} = \max\{0, Pos_{s,j-1} + L_{j,s} - l_s\}. \quad (2)$$

Although any product in an ASRS buffer can be selected at any time, logistical restrictions are considered in the decision. It is assumed that by entering the buffer, the logistic department prepares and delivers the required pieces and materials that will be built into the product. These materials are stored at the side of the stations of the assembly line and are different for each option required for a task. To limit the storage amount, a due date is defined for each material. Restrictions on the selection of the products are therefore applied to preserve the due dates.

As the materials are defined based on options, it is assumed that they can be mounted in any product requiring the same option. That is, the window of a sunroof that is transported to the station by the entry of product 1 could be mounted in product 2 if they both require the same sunroof option.

The option-based due dates allow the virtual resequencing of models, in which their parts and materials are interchanged. Furthermore, by using the entry buffer, physical resequencing can take place. As already stated, the objective of such resequencing is to minimize the utility work required for the assembly.

A simplified example of the problem is illustrated in figure 4. In such an example, only one station is considered, in which two tasks are performed. Each of the tasks are related to two components each: A or B for task 1; C or D for task 2. The processing times of mounting each component and the resulting processing load for all possible four products (AC, AD, BC, and BD) are given in figure 4. Assume a cycle time of 6 time units and a station length of the equivalent of 8 time units. In a selected moment in period or cycle 1, the worker finishes the operations at position 7 (top right part of figure 4) and the contents of the buffer are, in order, BD, BC, AD, and AC (Buffer content, figure 4). The required parts for these products are stored along the station and each part is associated with a due date given in periods (Station information, figure 4). For this example, very tight due dates are selected to show the influence of the virtual resequencing. The components of product BD, for instance, must be mounted up until the period 2, that is, either now (period 1) or in the very next cycle.

In the given condition, assembly BD as the next product would require utility work, since the worker is almost at the right border of the station and BD requires more than the cycle time. If no virtual resequencing is allowed, the due dates of the parts in the storage are necessarily associated with the products of the buffer. One optimal solution for this case is the sequence AD - BD - BC - AC requiring one unity of utility work. Due to the tight due dates, either starting with BD or sequencing BD and BC after each other is unavoidable.

By considering virtual resequencing, it is possible to assemble all four products without requiring utility work. In the sequence BC - AD - BD - AC, the first product uses the part B with due date 2 and part C with due date 3 without requiring utility work. As the second product, AD uses part D with the due date of 2 and part A with the due date of 4, restoring the worker position

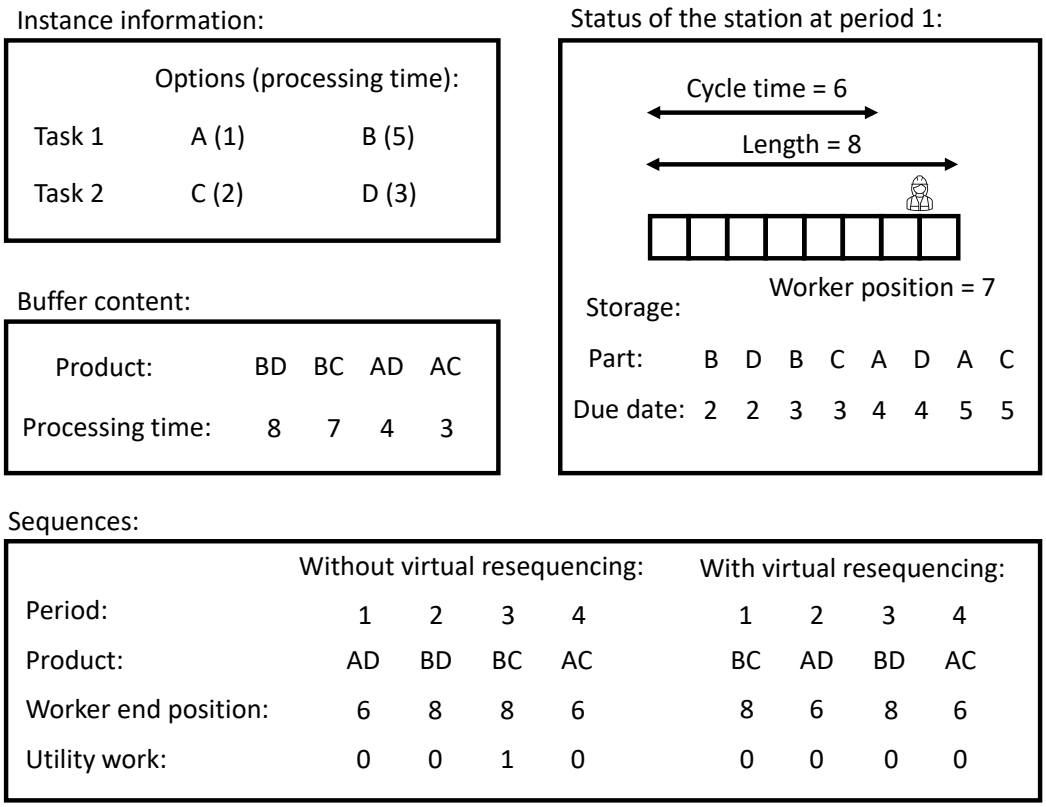

Figure 4: Example of the resequencing problem with and without virtual resequencing for a line with a single station.

to 6. This assignment would be impossible without the virtual resequencing. After that, products BD and AC are sequenced with the remaining parts within the due dates.

Note that the example is a simplification of the problem. According to the given definition, after each product selection, a new product arrives at the buffer, which is not considered in this example. Furthermore, only one station is depicted.

4. Solution approaches

To solve the problem described in section 3, some heuristics based on priority rules are proposed. The rules are used for product selection and as a benchmark for other procedures such as the incomplete enumeration approach that is presented later.

4.1. Heuristics

Priority rules consist of calculating a score $Score_j$ for each possible job $j \in J$, which is then used in the decision of the next product to select. In MMS Problems, a score function can be built from the available information of the models in the buffer and the state of the FA. In this case, the station load $L_{j,s}$ (sum of the processing times of all tasks within a station) of job j at station s, the cycle time CT, positions of workers at the respective station $Pos_{s,j-1}$, and lengths of stations l_s for each station $s \in S$ are considered. Furthermore, the stock of workpieces or materials by the

assembly line and the buffer content can be used to decide whether the composition of a selected job is appropriate for the materials waiting at the stations or not.

The selected priority rules are described in table 2.

Table 2: Proposed priority rules

Name	Formulation	Description
FiFo		Select the job that was placed earliest in the buffer.
Shortest PT	$Score_j = \sum_{s \in S} L_{j,s}$	Select the job with the smallest sum of processing time.
Shortest specific PT	$Score_j = \sum_{s \in \tilde{S}} L_{j,s}$	Select the job with the smallest processing time of a subset stations. Only those stations $s \in \tilde{S}$ are considered where the worker is not at the start of the station at the beginning of the cycle: $\tilde{S} \subseteq S\{s \mid Pos_s > 0\}$.
Alternating	$Score_j = \sum_{s \in S} L_{j,s}$ and $Score_j = \sum_{s \in S} -L_{j,s}$	Select alternately the job with the highest and the one with the lowest total processing time.
Min UW	$Score_j = \sum_{s \in S} max\{0, Pos_s + L_{j,s} - l_s\}$	Select the job, that would cause the smallest amount of utility work based on the final situation of the prior cycle.

The **FiFo** rule and the **shortest PT** rule can be found in Corsten (2016). The **alternating** rule is derived from the approach of Yano & Rachamadugu (1991) and the **Min UW** rule comes from the work of Thomopoulos (1967). In addition to the score values, a penalty term f_j is introduced for each job, which comes into effect when the selected job does not meet the options' due date. The reason for this is, that the just-in-time provision of options at the stations must not be delayed. The required parts are delivered to the workstations depending on the planned sequence. The space along the workstations is very limited, so that the delivered pieces must be used within the due date.

$$f_j = \begin{cases} 1 & \text{if any option due date would be exceeded by the selection of job } j \\ 0 & \text{otherwise} \end{cases}$$

In general, all priority rules can be described as denoted below.

$$j^* = \arg \min_{j \in J} \{Score_j + f_j \cdot M\} \tag{3}$$

Here, M is a sufficiently large number to lower the selection probability of an unsuitable job. Hereby $M \gg Score_j$ applies, so that a job with a penalty is not selected as long as at least one other job without penalty exists. In the case of equal scores, the FiFo rule serves as a tie-breaker.

4.2. Lookahead search

The lookahead approach can be described as an incomplete enumeration of all sequence combinations. As stated in the section above, the caused utility work can be calculated with the knowledge about the setup (station lengths and cycle time), the state of the assembly line (worker positions), and the processing times of the job. As a result, a new state (new worker positions) is reached, with whom the next state can be calculated by taking into account the processing times of the remaining jobs. An illustration of the integration of the lookahead procedure in the simulation is shown in fig. 5. For each given iteration k, the (partial) sequences are enumerated. For the selection, a combination of jobs can then be considered and the sequence with the lowest sum of

utility work is selected. Because only one job per cycle can be selected, the job that is placed first in the sequence gets chosen and the procedure restarts in the next cycle in the next simulation iteration. This forecast can be done up to any depth lower than or equal to the buffer size.

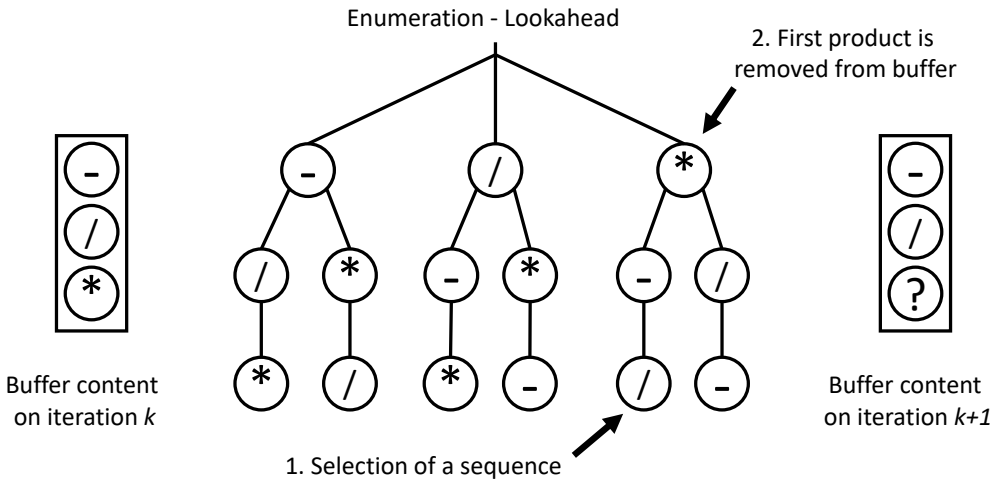

Figure 5: Schematic decision sequence for the simulation with the looakhead procedure.

To meet the options' due dates, the selection cannot be done solely based on utility work. We can either consider only the sequences that do not cause due date violations (variant A) or introduce a rule similar to (3) in order to minimize occurring delays (variant B), where we penalize the sequences which would cause due-date violations.
Although the selection rules give priority to critical components (components for which the due dates have almost been reached), depending on the previous assignments it may be impossible to meet all deadlines. As the heuristics and the lookahead procedure only select one model per cycle, the future sequence with the remaining products may be infeasible. In order to verify the feasibility of the sequencing of the other vehicles in the buffer, a matching problem can be defined for the remaining products and options (see 4.2.2).

In the lookahead approach, no error is encountered when the matching problem is applied. However, the solution time increases, as multiple matching problems must be solved.

4.2.1. Variant A - Lookahead search with error tolerance

Considering the decoupling of the options from the orders, the selection of some combinations can make this problem unsolvable in the context of not getting any delay on the materials. In this variant, the deterioration of the score of a sequence if an option due date cannot be met is considered instead of a matching problem. This can be prevented by looking at as many cycles in advance as the buffer has places or by solving a matching problem (see variant B). The procedure is the same as in section 4.1. With this, there is hope to get better computing times while keeping the solution quality at a good level.

4.2.2. Variant B - Lookahead search with matching problem

To completely avoid delays, a matching problem, in which the remaining options and the remaining orders are brought together, must be solved. The notation for the following model is given in Table 3.

Table 3: Nomenclature of the variables and parameters of the model for the matching problem.

description	domain	type	meaning
O		index	set of options o
P		index	set of products p in the buffer
G		index	set of option types g
$R_{o,g}$	(O,G)	data	1, if option o is from type g, 0 otherwise
$R^{max}_{g,p}$	(G,P)	data	1, if product p requires an option from type g, 0 otherwise
T_o	(O)	data	remaining time of option o until due date is met
B		data	buffer size
$y_{o,p}$	(O,P)	binary	1, if option o is assigned to product p, 0 otherwise
c_p	(P)	integer	the cycle in which product p should be sent to the line

The matching problem (MP) can be defined as a feasibility problem subject to:

$$\sum_{p\in P} y_{o,p} = 1 \qquad \forall\ o \in O \tag{4}$$

$$\sum_{o\in O} R_{o,g}\cdot y_{o,p} = R^{max}_{g,p} \qquad \forall\ p \in P,\ g \in G \tag{5}$$

$$c_p \le T_o + (1 - y_{o,p})\cdot B \qquad \forall\ o \in O,\ p \in P \tag{6}$$

$$c_{p+1} = c_p + 1 \qquad \forall\ p \in P \setminus |P| \tag{7}$$

$$c_{|P|} \le B - 1 \tag{8}$$

$$y_{o,p} \in \{0,1\} \qquad \forall\ o \in O,\ p \in P \tag{9}$$

$$c_p \ge 0 \qquad \forall\ p \in P. \tag{10}$$

The aim of this model is to solve the sequencing problem keeping the due dates without caring for utility work. The restrictions are given by expressions (4) to (8). Every option can only be assigned once to a production cycle (4). By restriction (5) it is ensured that the correct option types are assigned to each product. Restriction (6) defines the cycle number in which the order must be produced with the assigned options. All options must be assigned to a cycle time before their due date. Equation (7) indicates that only one product can leave the buffer in every cycle. As a consequence, the cycle number of the last product $c_{|P|}$ must be smaller than the size of the buffer minus one to generate a feasible solution (see restriction (8)).

While the matching itself is always possible, only sequences which allow all options to meet their due dates are considered further in the lookahead enumeration. A product cannot be selected in case the matching problem with the remaining options and jobs is infeasible. This assignment problem must be solved multiple times in a cycle. In our setting, this is done with an enumera-

tion procedure that stops when a feasible solution is found. Therefore, the computation time is increased, but because the problem can be solved in a short time period, it can be used for an online application.

4.2.3. Variant C - Lookahead search with only physical sequencing

To measure the effects of virtual sequencing in contrast to the relatively inflexible physical sequencing, a third variant is proposed. Note that the appropriate options for an order are delivered to the stations as soon as the order enters the buffer. For variants A and B, the options are then decoupled from the order, which allows mounting the options on other products, respectively to virtually resequence the products. In contrast to that, for variant C the options are at all times connected to their original product and cannot be mounted on other products.

5. Numerical study

To evaluate the quality of the proposed selection rules and the lookahead procedures, a dataset containing 1,050 instances is generated. The construction of the dataset and all simulation parameters are given.

5.1. Dataset

A resequencing instance is based on a given buffer and assembly line structure. Therefore, a solution of an assembly line balancing problem instance is used in the definition of each resequencing instance. The dataset is based on the medium instances (containing 50 tasks) introduced by Otto et al. (2013), who propose a systematic generation procedure for the SALBP. The 525 base instances are adapted to consider multiple models and are solved considering two different objective functions, resulting in the 1050 instances used for this numerical study. The generation procedure and the utilized parameters for the dataset are described in the following subsections.

The 525 proposed instances of Otto et al. (2013) are generated considering three parameters: precedence relations graph Ordering Strength (OS), precedence relations Graph Structure (GS), and processing times Task Distributions (TD). The ordering strength is a value from 0 to 1 which measures how many precedence relations are present in the graph – 0 represents no precedence relations, while 1 allows only a single feasible task sequence. The graph structure can contain bottleneck tasks (BN) – tasks with multiple predecessors, chains of tasks (CH) – sequences of tasks with only one predecessor and successor, or mixed (MX) – containing both bottlenecks and chains. Finally, for the selected time distributions it is assumed, that the processing times are either small in relation to the cycle time with a peak at the bottom (PB), representing about 50% of the cycle time (peak in the middle, (PM)), or be a combination of both (bi-modal, (BM)). All instances and results are available in the supplementary material of the paper.

5.1.1. Adaptation of instances with a single product to multiple products

To obtain instances containing multiple models, the processing times of the tasks of all instances in the dataset of Otto et al. (2013) are extended to consider multiple options. Each task is

randomly assigned a number of options between 1 and 3 with a uniform distribution. For the medium instances of Otto et al. (2013) containing 50 tasks, the number of possible products in the adapted instances varies between 1 (all tasks have only one option) to almost 10^{24} (when all tasks present 3 options). In the average case, when one-third of the tasks present each 1, 2, or 3 options, respectively, the expected number of unique products is in the order of 10^{13}.

The processing times are computed by considering the SALBP given task duration PT_t for task t as the first option duration. For the other options (second and third), a random processing time is drawn from a uniform distribution over the interval $[0.5 \cdot PT_t; 1.5 \cdot PT_t]$. The resulting value is rounded to the nearest integer. In practice, there are cases in which one product does not require the realization of a task. This is considered in the instance by introducing the probability of a 'non-required' option. For the tasks containing 2 or 3 options, the processing time of the second option is set to 0 with a probability of 30%. Furthermore, all randomly generated processing times are capped to 1,000 time units. This modification is used to avoid very large processing times that would greatly surpass the planned cycle time of the assembly line.

To simulate production sequences, the probability of each option must also be given. For the instances of the dataset, a random value ($U[0, 1]$) is generated for each option. These values are used as weights and are adjusted so that the sum of options' probabilities is equal to 1.

The precedence relations, as well as the number of stations, are kept identical to the instances of Otto et al. (2013). For the definition of the cycle time (and as a consequence the speed of the conveyor belt), an SALBP instance is generated considering an equivalent single model based on the expected processing times of each task. The expected processing time of a task in the SALBP instance is denoted with $PT_t^{avg} = \sum_{o \in O} p_{t,o} \cdot PT_{to}$. The SALBP equivalent instances are solved using the algorithm SALOME (Scholl & Klein, 1997) (available at `https://assembly-line-balancing.de/`) with a laptop (Intel Core i5-8250U with 16 GB RAM) using a time limit of 1.000 seconds. Out of the 525 base instances of Otto et al. (2013), 500 instances are solved to optimally, and the other 25 present small gaps. The optimal cycle time of the SALBP (CT^{SALBP}) based on the average processing times can be seen as a lower bound for the cycle time in a mixed-model assembly line. Since the equivalent SALBP considers the expected processing time of each task, all product models are aggregated into one without the need for sequencing. The cycle time of the mixed-model assembly-line instance is defined as in Sikora (2021) and is given as $CT = CT^{SALBP}/0.95$. The selected value for the cycle time assures that the assembly line balancing solution has high productivity while it allows for other balancing solutions which are not the optimal setting of the SALBP equivalent problem.

5.1.2. ALBP objective functions used for the instance generation

As already stated, two objective functions are used in the definition of the 525 base instances and their solutions are used as a production system in the resequencing instances. The selected objective functions are adaptations of the horizontal and vertical balancing objective functions listed in Merengo et al. (1999). In horizontal balancing, the objective is to minimize the differences between

the multiple models within stations. Differently, vertical balancing aims at the minimization of deviations between stations, for which the average station loads are used.

The nomenclature used in both models is given in table 4. The vertical-balancing approach uses

$$Obj_{Ver} : \sum_{s \in S} (L_{max}^{avg} - L_s^{avg}) \tag{11}$$

as the objective function, while

$$Obj_{Hor} : L_{max} \tag{12}$$

is used for the horizontal balancing. The proposed horizontal balancing objective function in Merengo et al. (1999) is a non-linear function. Hence, for simplicity, the maximal station load of the most complex product is chosen as the objective.

Table 4: Nomenclature of the variables and parameters of the model used to generate assembly line balancing solutions for the instances.

Variable	**Type**	**Meaning**
$x_{t,s}$	Binary	1, if task t is assigned to station s, 0, otherwise
L_s^{avg}	Continuous	The average load in station s
L_{max}^{avg}	Continuous	Maximal average load in the line
L_{max}	Continuous	Maximal station load in the line
Parameter	**Domain**	**Meaning**
$Prec$	$(t_1, t_2) \in T^2$	Precedence relations between two tasks t_1 and t_2
$PT_{t,o}$	$t \in T, o \in O_t$	Processing time of option o of task t
$p_{t,o}$	$t \in T, o \in O_t$	Relative demand of option o of task t ([0;1])

Then both models minimize their objective function (11) or (12), respectively, subject to:

$$\sum_{s \in S} x_{t,s} = 1 \qquad \forall\, t \in T \tag{13}$$

$$\sum_{k \in S : k \leq s} x_{t_1,k} \geq \sum_{k \in S : k \leq s} x_{t_2,k} \qquad \forall\, (t_1, t_2) \in Prec,\ s \in S \tag{14}$$

$$L_s^{avg} = \sum_{t \in T} \sum_{o \in O_t} PT_{t,o} \cdot p_{t,o} \cdot x_{t,s} \qquad \forall\, s \in S \tag{15}$$

$$L_{max}^{avg} \geq L_s^{avg} \qquad \forall\, s \in S \tag{16}$$

$$L_{max} \geq \sum_{t \in T} \max_{o \in O_t}(PT_{t,o}) \cdot x_{t,s} \qquad \forall\, s \in S \tag{17}$$

$$x_{t,s} \in \{0, 1\} \qquad \forall\, t \in T, s \in S \tag{18}$$

$$L_s^{avg}, L_{max}^{avg}, L_{max} \geq 0 \qquad \forall s \in S \tag{19}$$

The restrictions are given by expressions (13) to (17). Restriction (13) is the occurrence restriction, each task must be assigned to a station. The precedence relations are modeled in inequality (14). These formulations are used to solve the assembly line balancing problem by minimizing the differences between products within or between stations. Expressions (15) and (16) model the average load of each station s and the maximal average station load of the line, respectively. These restrictions are used to define the variables used in the objective function of vertical balancing. The variable of the objective function of horizontal balancing is defined by inequality (17).

The assembly-line balancing problem instance for both objectives is solved using Gurobi 8.0.1 with a laptop (Intel Core i5-8250U with 16 GB RAM) using a time limit of 3.600 seconds. The proposed instances are based on the best solution obtained within this solution time.

5.1.3. Resequencing parameters of the dataset

To generate resequencing instances, some parameters of the buffer and the reordering constraints must be defined. As already defined in section 3, an ASRS buffer is used in between the painting and final assembly sections. It is assumed that the buffer contains 10 positions, in which one product can be stored. The results of other buffer sizes are presented in section 6.3. For each product entering the buffer, a due date for each piece necessary for the assembly is defined. The due date is defined as double the size of the buffer in cycle times ($2 \cdot B$). This means that either the planned product or an equivalent one (in respect to a given option) must be selected within the due date of the given option. Note that the due date based on options allows virtual resequencing since the pieces can be mounted on other products.

5.2. Simulation settings

To evaluate the approaches from section 4, a simulation is set up. One simulation run includes 10,000 cycles, in which a random product is placed in the buffer each time and a product is selected for final production. The simulation is dynamic and between 50 and 500 experiment runs are executed until the confidence interval is small enough to allow a comparison between approaches. After a minimal of 50 simulation runs, the simulation is aborted if the relative width (the total width divided by the mean value) of the confidence interval is less than 1% on a 95% confidence level. The lookahead search is only performed for a depth of up to 3 cycles due to time limits for the simulation. In practice, more depth could be used within one minute.

6. Evaluation

The results of the numerical study are discussed below. Firstly, the priority rules are evaluated and compared to the lookahead approach. In the second subsection, the results of the variants of the lookahead algorithm are compared to each other. All stated computing times refer to simulations carried out with a PC with an Intel Core i5-4590 processor and 16 GB RAM.

6.1. Comparison of the priority rules and between the horizontal and vertical balancing

The results of the vertical and the horizontal balancing approach are displayed in table 5 and table 6, respectively. Every data point is the mean value of 25 different instances that all share the same parameters (GS, OS, and TD).

Table 5: Results for vertical instances.

GS: Graph structure, OS: Order strength, TD: Time distribution, BN: Bottleneck, CH: Chain, MX: Mixed, PB: Peak at the bottom, PM: Peak in the middle, BM: Bimodal.

Parameters			FiFo		Min PT		Alternating		Specific PT		Min UW		Lookahead Depth2		Lookahead Depth3	
GS	OS	TD	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value
BN	0.2	PB	21.33	0.00%	20.25	1.34%	16.82	0.94%	16.58	0.53%	8.47	0.16%	5.89	0.07%	4.74	0.00%
BN	0.2	PM	267.22	0.00%	263.57	1.45%	254.98	0.79%	251.46	0.13%	219.21	0.86%	205.64	0.87%	191.87	0.05%
BN	0.2	BM	84.38	0.00%	82.35	1.37%	75.30	0.82%	75.08	0.35%	56.60	0.41%	50.91	0.33%	45.97	0.02%
BN	0.6	PB	25.51	0.00%	24.43	1.36%	20.32	0.95%	19.19	0.55%	11.85	0.24%	8.70	0.10%	7.36	0.00%
BN	0.6	PM	239.46	0.00%	236.10	1.41%	228.18	0.77%	221.14	0.11%	193.37	0.74%	181.53	0.74%	168.49	0.03%
BN	0.6	BM	104.92	0.00%	102.39	1.31%	92.86	0.81%	91.78	0.29%	77.42	0.61%	67.83	0.45%	60.83	0.02%
CH	0.2	PB	20.71	0.00%	19.64	1.36%	16.15	0.94%	15.40	0.55%	7.78	0.21%	5.26	0.09%	4.22	0.00%
CH	0.2	PM	300.53	0.00%	296.51	1.38%	286.87	0.75%	282.58	0.12%	250.71	0.88%	236.27	0.85%	220.87	0.04%
CH	0.2	BM	99.25	0.00%	96.91	1.27%	88.52	0.80%	89.18	0.33%	69.21	0.53%	63.15	0.43%	57.06	0.02%
CH	0.6	PB	21.78	0.00%	20.74	1.31%	17.24	0.90%	16.83	0.53%	8.92	0.18%	6.26	0.08%	5.21	0.00%
CH	0.6	PM	211.87	0.00%	208.69	1.38%	201.42	0.75%	194.66	0.13%	168.33	0.73%	156.51	0.68%	144.39	0.04%
CH	0.6	BM	102.08	0.00%	99.69	1.31%	90.84	0.79%	90.26	0.30%	75.90	0.55%	66.95	0.39%	59.93	0.02%
MX	0.2	PB	21.01	0.00%	19.98	1.31%	16.87	0.92%	15.78	0.53%	10.04	0.20%	7.32	0.08%	6.21	0.00%
MX	0.2	PM	312.80	0.00%	308.85	1.38%	299.24	0.76%	292.83	0.14%	263.45	0.97%	247.70	0.92%	231.27	0.05%
MX	0.2	BM	105.74	0.00%	103.26	1.32%	94.45	0.82%	93.27	0.36%	78.53	0.62%	70.26	0.44%	63.47	0.02%
MX	0.6	PB	16.54	0.00%	15.70	1.32%	12.94	0.94%	12.48	0.54%	6.70	0.11%	4.76	0.04%	4.01	0.00%
MX	0.6	PM	234.03	0.00%	230.88	1.41%	222.64	0.77%	218.32	0.09%	193.88	0.82%	180.39	0.74%	166.97	0.04%
MX	0.6	BM	104.36	0.00%	101.86	1.31%	93.05	0.80%	91.59	0.34%	73.21	0.63%	65.84	0.46%	58.26	0.02%
MX	0.9	PB	11.89	0.00%	11.29	1.36%	9.33	0.93%	8.61	0.60%	4.85	0.08%	3.62	0.03%	3.16	0.00%
MX	0.9	PM	184.94	0.00%	182.40	1.36%	175.53	0.74%	170.46	0.12%	151.67	0.71%	138.89	0.58%	128.55	0.03%
MX	0.9	BM	68.68	0.00%	66.86	1.33%	60.35	0.84%	57.84	0.38%	46.04	0.50%	39.45	0.27%	35.06	0.01%
Mean value			**121.86**	**0.00%**	**119.64**	**1.35%**	**113.04**	**0.83%**	**110.73**	**0.33%**	**94.10**	**0.51%**	**86.34**	**0.41%**	**79.42**	**0.02%**

Table 6: Results for horizontal instances.

GS: Graph structure, OS: Order strength, TD: Time distribution, BN: Bottleneck, CH: Chain, MX: Mixed, PB: Peak at the bottom, PM: Peak in the middle, BM: Bimodal.

Parameters			FiFo		Min PT		Alternating		Specific PT		Min UW		Lookahead Depth2		Lookahead Depth3	
GS	OS	TD	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value	UW	Error Value
BN	0.2	PB	26.26	0.00%	25.11	1.34%	22.93	0.93%	21.26	0.55%	7.12	0.07%	6.41	0.08%	5.91	0.01%
BN	0.2	PM	221.47	0.00%	217.84	1.45%	212.24	0.79%	210.69	0.08%	154.50	0.52%	154.04	0.79%	145.13	0.06%
BN	0.2	BM	87.68	0.00%	85.48	1.37%	81.17	0.82%	81.72	0.45%	51.99	0.30%	51.30	0.41%	48.11	0.03%
BN	0.6	PB	27.25	0.00%	26.12	1.36%	24.18	0.95%	23.02	0.62%	9.55	0.14%	8.29	0.12%	7.73	0.01%
BN	0.6	PM	181.84	0.00%	178.91	1.42%	174.72	0.77%	170.61	0.09%	125.88	0.40%	125.35	0.60%	118.45	0.04%
BN	0.6	BM	98.27	0.00%	95.75	1.30%	90.25	0.81%	90.28	0.33%	56.78	0.37%	56.33	0.53%	51.96	0.04%
CH	0.2	PB	23.31	0.00%	22.22	1.36%	20.45	0.94%	18.34	0.48%	5.75	0.06%	4.95	0.06%	4.54	0.01%
CH	0.2	PM	272.46	0.00%	267.94	1.38%	260.69	0.75%	261.36	0.09%	197.53	0.57%	197.36	0.84%	184.77	0.05%
CH	0.2	BM	91.72	0.00%	89.08	1.27%	84.49	0.80%	84.35	0.46%	49.71	0.29%	50.01	0.42%	46.43	0.04%
CH	0.6	PB	24.91	0.00%	23.91	1.32%	22.03	0.90%	20.77	0.52%	7.56	0.10%	6.58	0.10%	6.09	0.01%
CH	0.6	PM	164.98	0.00%	161.90	1.37%	157.65	0.74%	152.84	0.12%	108.54	0.42%	108.21	0.64%	99.87	0.04%
CH	0.6	BM	88.50	0.00%	86.02	1.31%	81.44	0.79%	79.48	0.31%	50.33	0.27%	49.11	0.39%	44.92	0.03%
MX	0.2	PB	22.98	0.00%	21.87	1.30%	20.11	0.92%	18.74	0.48%	5.83	0.06%	4.90	0.06%	4.50	0.01%
MX	0.2	PM	252.58	0.00%	248.27	1.39%	241.68	0.76%	239.12	0.12%	180.66	0.56%	179.50	0.80%	167.63	0.05%
MX	0.2	BM	101.87	0.00%	99.16	1.33%	94.03	0.82%	93.91	0.41%	59.42	0.33%	59.59	0.48%	55.06	0.04%
MX	0.6	PB	19.49	0.00%	18.71	1.32%	17.22	0.94%	16.05	0.52%	6.12	0.07%	5.42	0.07%	5.16	0.01%
MX	0.6	PM	175.52	0.00%	172.60	1.41%	168.15	0.77%	164.01	0.10%	118.68	0.39%	117.59	0.58%	109.51	0.04%
MX	0.6	BM	99.20	0.00%	96.41	1.31%	90.98	0.81%	90.96	0.36%	53.96	0.32%	53.29	0.47%	48.96	0.04%
MX	0.9	PB	14.84	0.00%	14.22	1.36%	13.17	0.93%	11.97	0.31%	5.12	0.06%	4.57	0.07%	4.16	0.01%
MX	0.9	PM	104.47	0.00%	102.50	1.36%	99.18	0.74%	94.73	0.11%	61.47	0.23%	60.59	0.33%	55.44	0.02%
MX	0.9	BM	59.65	0.00%	58.03	1.33%	54.54	0.84%	52.32	0.32%	32.41	0.22%	30.77	0.26%	27.95	0.02%
Mean value			**102.82**	**0.00%**	**100.57**	**1.35%**	**96.73**	**0.83%**	**95.07**	**0.33%**	**64.23**	**0.27%**	**63.53**	**0.39%**	**59.16**	**0.03%**

It can be noted that the results without sequencing (**FiFo**) exhibit the most utility work per order across all instances. According to this rule, none of the orders remains in the buffer for long, so that the options are all installed within their due dates and have an error value of 0%. In terms of utility, the **shortest PT** rule is slightly better for both types of instances (horizontal and vertical), but violates more often the due date of the options. This could be because orders with high processing times remain in the buffer for a long time and thus special combinations of options cannot be installed in time. This is confirmed by the mean error value of the **alternating** rule, as this selects the job with the most processing time in every second cycle to prevent a job from staying in the buffer too long. It is interesting to see that the utility work also decreases as this apparently results in a better distribution of processing times over time.

The other specification of the normal PT rule, the **shortest specific PT** rule, gives slightly better results. This can be attributed to the fact that, in addition to the original rule, this rule also processes the information on where the workers are located at the stations in the respective cycle.

This perspective on what information is used within the priority rule can also be applied to the **Min UW** rule, where again somewhat more information (lengths of the stations) is used for decision making. In comparison to the FiFo rule, the result for Min UW is better: 22.78% (vertical) and 37.53% (horizontal). The Min UW rule can be considered to be identical to a lookahead rule with the depth of 1. With this, the improvement for an additional search depth can be reflected. The improvement of the vertical instances in table 5 for the utility work and the error value is continuous when gaining more depth. But it must be noted that the results in table 6 differ from this estimation, as the mean error value (failure to meet option due dates) of the Min UW rule (equal to depth 1) is better than the mean error value of the lookahead rule with depth 2 (0.27% $<$ 0.39%). This can be explained by the number of ties that occur.

To gather information about the occurrences of ties, a subset of instances is used. Instead of 25 instances for every combination of parameters (GS, OS, and TD), just one instance out of 25 is used and only the horizontal instances are considered. With the Min UW rule, a tie occurs on average in 51% of the cycles because at least two products have the same score value. In 24% of the cycles, the product that has been in the buffer the longest is involved in the tie. In these cases, the FiFo rule is used as a tie-breaker. On the other hand, with the lookahead rule with depth 2, there are more tie situations (59%). However, the number of possible combinations in each cycle is also nine times higher ($B \cdot (B-1)$) for a buffer with 10 positions. The number of situations in which the FiFo rule is used as a tie breaker is the case in 22% of the cycles. This is a small but significant difference. As a consequence, the Min UW rule indirectly contributes more to keeping the due dates than the lookahead rule with depth 2.

The results of the studies can be used to evaluate the quality of the balancing solutions used in the instance generation. For the way the instances are built, the horizontal balancing is superior in terms of required utility work: Considering random sequences (FiFo), the horizontal balancing approach requires less UW in 303 of 525 instances. When comparing the results for the lookahead

algorithm with a depth of 3, the horizontal balancing wins 323 of 525 instances. Therefore, it can be concluded, that horizontal balancing provides a better assembly line implementation in terms of utility work.

It can also be seen in tables 5 and 6, that for those instances with short processing times (peak at the bottom), the vertical balancing leads to better results. On the other hand, for the instances with longer processing times (peak in the middle) or a bi-modal time distribution, the horizontal balancing is superior.

6.2. Evaluation of the different lookahead variants

For the evaluation of the different lookahead variants, a smaller set of instances with just one instance for every combination of parameters (GS, OS, and TD) is used. Furthermore, only the horizontal instances are considered. The results can be seen in table 7.

Table 7: Instance by instance result comparison for the horizontal instances for different variants of the lookahead rule. Only selected vertical instances are considered, one from each category (see parameters of table 6).

Instance	Lookahead Depth 3 regular			Lookahead Depth 3 with MP			Lookahead Depth 3 w/o virtual		
Name	UW	Error value	Runtime	UW	Error value	Runtime	UW	Error value	Runtime
n=50_1	10.29	0.01%	00:09:38	10.28	0.00%	01:07:06	11.79	0.00%	00:29:44
n=50_26	145.14	0.10%	00:02:29	145.25	0.00%	00:21:52	147.78	0.00%	00:09:16
n=50_51	198.81	0.09%	00:02:04	197.80	0.00%	00:21:10	200.30	0.00%	00:09:43
n=50_76	0.24	0.00%	00:03:29	0.25	0.00%	00:16:28	0.48	0.00%	00:05:23
n=50_101	33.77	0.01%	00:06:39	33.67	0.00%	00:54:56	34.61	0.00%	00:12:49
n=50_126	58.26	0.08%	00:03:38	58.27	0.00%	00:31:33	60.02	0.00%	00:11:00
n=50_151	3.95	0.00%	00:06:31	3.98	0.00%	00:37:05	4.14	0.00%	00:12:07
n=50_176	153.41	0.05%	00:02:03	153.01	0.00%	00:15:49	161.07	0.00%	00:07:27
n=50_201	40.87	0.03%	00:04:45	40.95	0.00%	00:45:31	43.70	0.00%	00:12:50
n=50_226	8.19	0.01%	00:10:24	8.18	0.00%	01:11:24	9.93	0.00%	00:29:24
n=50_251	117.54	0.06%	00:02:11	116.96	0.00%	00:17:46	120.89	0.00%	00:08:23
n=50_276	111.61	0.04%	00:01:51	111.69	0.00%	00:16:49	115.24	0.00%	00:07:55
n=50_301	1.80	0.00%	00:06:04	1.77	0.00%	00:36:46	2.21	0.00%	00:14:04
n=50_326	140.46	0.05%	00:02:08	141.28	0.00%	00:18:03	147.44	0.00%	00:08:21
n=50_351	23.35	0.00%	00:06:40	23.27	0.00%	00:48:34	24.01	0.00%	00:17:29
n=50_376	0.07	0.00%	00:01:35	0.06	0.00%	00:06:11	0.07	0.00%	00:01:31
n=50_401	49.37	0.05%	00:03:16	49.46	0.00%	00:32:08	51.87	0.00%	00:13:55
n=50_426	14.99	0.00%	00:11:57	14.99	0.00%	01:21:58	15.75	0.00%	00:22:47
n=50_451	0.68	0.00%	00:04:23	0.66	0.00%	00:23:04	1.01	0.00%	00:08:21
n=50_476	59.44	0.02%	00:03:08	59.30	0.00%	00:32:02	63.45	0.00%	00:12:42
n=50_501	1.56	0.00%	00:06:10	1.56	0.00%	00:33:41	2.36	0.00%	00:12:20
Mean value	**55.90**	**0.03%**	**00:04:49**	**55.84**	**0.00%**	**00:34:45**	**58.01**	**0.00%**	**00:12:44**

The mean value for the utility work for the normal lookahead rule (variant A) is 55.90 time units (TU) per cycle. In comparison, the result for the lookahead rule with the matching problem (variant B) is 55.84 TU per cycle. When the results for the instances are compared in a t-test on a 5% significance level, the p-value is 0.45395. It can therefore be concluded that the difference in terms of the utility work is not significant at this level. When looking at the error value for variant A, the number of delayed options is 0.03% per cycle. The number of delayed options in variant B is 0 per definition due to the matching problem. Here, a trade-off has to be made between a 86%

improvement in computing time and the acceptance of a small but existing error of 0.03% in the option due dates.

On the other hand, when the lookahead rule without virtual resequencing (variant C) is compared to the rule from variant A, the mean values for the utility work differ: Variant A requires 55.90 TU per cycle while variant B requires 58.01 TU per cycle. When comparing these two results in a t-test, the difference is significant on a 5% significance level with a p-value of 0.0001785. It can therefore be concluded that the sequencing can significantly be improved on average by 3.6% if virtual resequencing is considered. As a consequence, it can be helpful to decouple the options from the orders, but this should always be balanced against the effort that may be involved in virtual resequencing. For this computational study, the effort in terms of computing time was higher when virtual sequencing was omitted.

To show that these methods can be applied to an online problem, the computing times per cycle are shown in Table 8 as an example for one instance. It can be seen that the calculations for each decision, even for the lookahead approaches with a depth of 3 take much less than a second on average.

Table 8: Comparison of computational time

Instance n=50_1

Method	**No. of simulated cycles**	**time [hh:mm:ss]**	**Avg time per decision [s/cycle]**
FiFo	$66{\cdot}10^4$	00:00:04	$0.74{\cdot}10^{-10}$
Min PT	$58{\cdot}10^4$	00:00:04	$0.88{\cdot}10^{-10}$
Alternating	$54{\cdot}10^4$	00:00:04	$0.87{\cdot}10^{-10}$
Specific PT	$63{\cdot}10^4$	00:00:17	$3.17{\cdot}10^{-10}$
Min UW	$500{\cdot}10^4$	00:01:57	$2.71{\cdot}10^{-10}$
Lookahead Depth2	$500{\cdot}10^4$	00:05:15	$7.29{\cdot}10^{-10}$
Lookahead Depth3 (Var A)	$500{\cdot}10^4$	00:09:38	$13.37{\cdot}10^{-10}$
Lookahead Depth3 (Var B)	$500{\cdot}10^4$	01:07:06	$93.18{\cdot}10^{-10}$
Lookahead Depth3 (Var C)	$500{\cdot}10^4$	00:29:44	$41.30{\cdot}10^{-10}$

6.3. Comparison of different buffer sizes

Furthermore, the effects of the different buffer sizes are examined. For this purpose, the same dataset as in 6.2 is used and the different buffer sizes 2, 4, 6, 8, 10, 15 and 20 are investigated. The lookahead rule (variant A) with a depth of 3 is used as the selection rule. The results are displayed in figure 6.

It can be seen that as the buffer size increases, the selection rule succeeds in further reducing the UW. However, this effect decreases and the UW can only be reduced slightly between buffer sizes 15 and 20. On the other hand, the computational time increases significantly with increasing buffer size. In a trade-off between UW and computing time, a buffer size of 10 was chosen for the solution of the larger dataset.

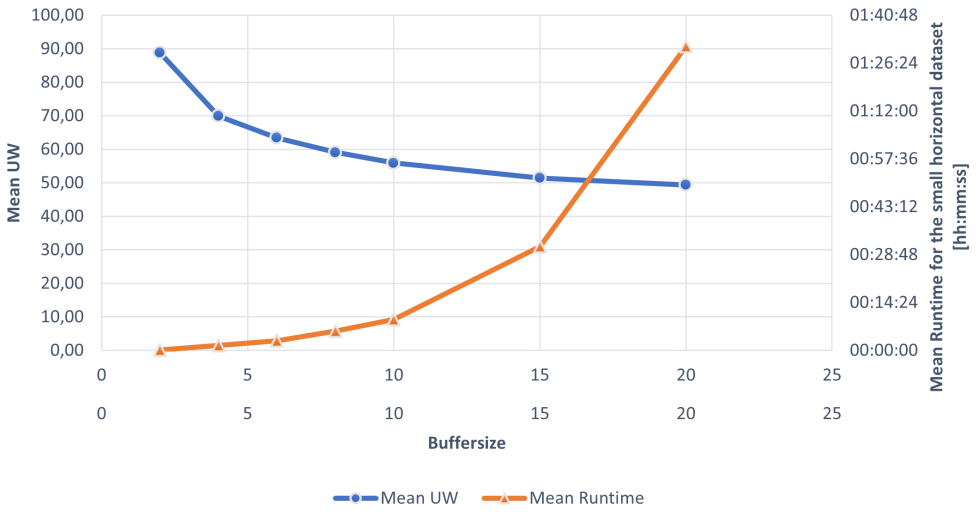

Figure 6: Plot of the UW and computational time for different buffer sizes for lookahead variant A with a depth of 3.

7. Conclusion

In this paper, the online resequencing of buffers with an uncertain input sequence is investigated. A lookahead procedure is proposed and compared to simple heuristics. Furthermore, we examine the differences between physical and a combination of physical and virtual resequencing and explain the trade-off between the best possible adherence to completion dates and computing time. Moreover, the calculations for each decision take much less than a second, which means that these approaches can be used online without any restriction.

It is shown that different balancing objectives, vertical and horizontal, which are used to create the instances for the simulation, influence the sequencing results. In our setting, the horizontal balancing approach requires less utility work and wins 323 of 525 instances. An exception to this conclusion is when the processing times are smaller in relation to the cycle time (PB). In this case, vertical balancing has a lower cost.

Between results of the priority rules, it is found that most rules slightly improve the baseline (FiFo). The more information processed in the respective rule, the greater is the improvement. Of the priority rules tested, the Min UW rule produces the best results, as the utility work is hereby directly considered. This rule can be improved with depth search. For a buffer of 10 positions, the results improved on average 59.8% (comparison of FiFo and lookahead with depth 3).

The comparison of the different lookahead variants shows that the acceptance of a small error value (0.03%) on average can decrease the computing time by 86% (variant A vs. variant B). Another finding is that the sequencing can significantly be improved (3.6%) when the options are decoupled from the orders and virtual sequencing is considered (variant A vs. variant C).

In conclusion, the use of a buffer before the final assembly can be used not only to restore but to improve the final assembly sequence. This improvement, however, can only be achieved if the manufacturer is able to adapt the part delivery systems for the production. For further work,

it would be interesting to evaluate this problem for lookahead rules with deeper search depths. Also, it could be worthwhile to investigate other approaches that process the information about the assembly line and the buffer content not via a priority rule, but in a neural network. One could train these with deep reinforcement methods (e.g. Deep Q-Learning). An interesting approach in this field is suggested by Stauder & Kühl (2021), who focus on making classification predictions about sequence scrambling effects with Artificial Intelligence. Hence, another worthwhile attempt would be to use such predictions to solve the problem shown here.

References

Bock, S., & Boysen, N. (2021). Integrated real-time control of mixed-model assembly lines and their part feeding processes. *Computers and Operations Research*, *132*. doi:10.1016/j.cor.2021.105344.

Boysen, N., Fliedner, M., & Scholl, A. (2009a). Production planning of mixed-model assembly lines: Overview and extensions. *Production Planning and Control*, *20*, 455–471. doi:10.1080/09537280903011626.

Boysen, N., Fliedner, M., & Scholl, A. (2009b). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, *192*, 349–373. doi:10.1016/j.ejor.2007.09.013.

Boysen, N., Kiel, M., & Scholl, A. (2011). Sequencing mixed-model assembly lines to minimise the number of work overload situations. *International Journal of Production Research*, *49*, 4735–4760. doi:10.1080/00207543.2010.507607.

Boysen, N., Scholl, A., & Wopperer, N. (2012). Resequencing of mixed-model assembly lines: Survey and research agenda. *European Journal of Operational Research*, *216*, 594–604. doi:10.1016/j.ejor.2011.08.009.

Cao, C., & Sun, H. (2019). Virtual Level Rescheduling for Automotive Mixed-model Assembly Lines with Beam Search Algorithms. *2019 IEEE 6th International Conference on Industrial Engineering and Applications, ICIEA 2019*, (pp. 225–229). doi:10.1109/IEA.2019.8714985.

Choi, W., & Shin, H. (1997). A real-time sequence control system for the level production of the automobile assembly line. *Computers and Industrial Engineering*, *33*, 769–772. doi:10.1016/s0360-8352(97)00249-0.

Corsten, H. (2016). *Produktionswirtschaft : Einführung in das industrielle Produktionsmanagement*. (14th ed.). Berlin: De Gruyter Oldenbourg.

Emde, S., & Gendreau, M. (2017). Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *European Journal of Operational Research*, *260*, 255–267. doi:10.1016/j.ejor.2016.12.012.

Ford, H. (2013). *The expanded and annotated My life and work : Henry Ford's universal code for world-class success*. Boca Raton, Fla. u.a.: CRC Press.

Franz, C., Hällgren, E. C., & Koberstein, A. (2014). Resequencing orders on mixed-model assembly lines: Heuristic approaches to minimise the number of overload situations. *International Journal of Production Research*, *52*, 5823–5840. doi:10.1080/00207543.2014.918293.

Franz, C., Koberstein, A., & Suhl, L. (2015). Dynamic resequencing at mixed-model assembly lines. *International Journal of Production Research*, *53*, 3433–3447. doi:10.1080/00207543.2014.993046.

Gujjula, R., & Günther, H. O. (2009). Resequencing mixed-model assembly lines under just-in-sequence constraints. *2009 International Conference on Computers and Industrial Engineering, CIE 2009*, (pp. 668–673). doi:10.1109/iccie.2009.5223797.

Gujjula, R., & Günther, H. O. (2010). An efficient heuristic to sequence mixed-model assembly lines. *IEEM2010 - IEEE International Conference on Industrial Engineering and Engineering Management*, (pp. 1209–1213). doi:10.1109/IEEM.2010.5674353.

Gujjula, R., Werk, S., & Günther, H. O. (2011). A heuristic based on Vogel's approximation method for sequencing mixed-model assembly lines. *International Journal of Production Research*, *49*, 6451–6468. doi:10.1080/00207543.2010.527384.

Günay, E. E., & Kula, U. (2020). A hybrid model for mix-bank buffer content determination in automobile industry. *European Journal of Industrial Engineering*, *14*, 544–572. doi:10.1504/EJIE.2020.108599.

Inman, R. R., & Schmeling, D. M. (2003). Algorithm for agile assembling-to-order in the automotive industry. *International Journal of Production Research*, *41*, 3831–3848. doi:10.1080/00207540310001595792.

Kampker, A., Kreiskother, K., & Schumacher, M. (2019). Mathematical model for proactive resequencing of mixed model assembly lines. *Procedia Manufacturing*, *33*, 438–445. doi:10.1016/j.promfg.2019.04.054.

Merengo, C., Nava, F., & Pozzetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, *37*, 2835–2860. doi:10.1080/002075499190545.

Meyr, H. (2004). Supply chain planning in the German automotive industry. *OR Spectrum*, *26*, 447–470. doi:10.1007/s00291-004-0168-4.

Mosadegh, H., Fatemi Ghomi, S. M., & Süer, G. A. (2017). Heuristic approaches for mixed-model sequencing problem with stochastic processing times. *International Journal of Production Research*, *55*, 2857–2880. doi:10.1080/00207543.2016.1223897.

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research*, *228*, 33–45. doi:10.1016/j.ejor.2012.12.029.

Scholl, A., & Klein, R. (1997). SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing. *INFORMS Journal on Computing*, *9*, 319–334. doi:10.1287/ijoc.9.4.319.

Schumacher, M., Kreiskoether, K. D., & Kampker, A. (2018). Proactive Resequencing of the Vehicle Order in Automotive Final Assembly to Minimize Utility Work. *Journal of Industrial and Intelligent Information*, *6*, 1–5. doi:10.18178/jiii.6.1.1-5.

Sikora, C. G. S. (2021). Benders' decomposition for the balancing of assembly lines with stochastic demand. *European Journal of Operational Research*, *292*, 108–124. doi:10.1016/j.ejor.2020.10.019.

Sikora, C. G. S. (2022). *Assembly-Line Balancing under Demand Uncertainty*. Wiesbaden: Springer Gabler. doi:10.1007/978-3-658-36282-9.

Stauder, M., & Kühl, N. (2021). Ai for in-line vehicle sequence controlling: development and evaluation of an adaptive machine learning artifact to predict sequence deviations in a mixed-model production line. *Flexible Services and Manufacturing Journal*, (pp. 1–39). doi:10.1007/s10696-021-09430-x.

Taube, F., & Minner, S. (2018). Resequencing mixed-model assembly lines with restoration to customer orders. *Omega (United Kingdom)*, *78*, 99–111. doi:10.1016/j.omega.2017.11.006.

Thomopoulos, N. T. (1967). Line Balancing-Sequencing for Mixed-Model Assembly. *Management Science*, *14*, B–59–B–75. doi:10.1287/mnsc.14.2.B59.

Tsai, L.-h. (1995). Mixed-Model Sequencing to Minimize Utility Work and the Risk of Conveyor Stoppage. *Management Science*, *41*, 485–495. doi:10.1287/mnsc.41.3.485.

Xu, Y., & Zhou, J.-g. (2016). A Virtual Resequencing Problem in Automobile Paint Shops. *Proceedings of the 22nd International Conference on Industrial Engineering and Engineering Management 2015*, (pp. 71–80). doi:10.2991/978-94-6239-180-2.

Yano, C. A., & Rachamadugu, R. (1991). Sequencing to minimize work overload in assembly lines with product options. *Management Science*, *37*, 572–586.