Incorporating car-sequencing rules in the planning of mixed-model assembly lines

Celso Gustavo Stall Sikora[a] and Lorenzo Tiacci[b]

[a]University of Hamburg, Institute for Operations Research, Moorweidenstr. 18, 20148 Hamburg, Germany; [b]Università degli Studi di Perugia, Dipartimento di Ingegneria, Via Duranti 93, 06125 Perugia, Italy

ARTICLE HISTORY
Compiled July 3, 2026

ABSTRACT
Efficient configurations of assembly lines involve balancing the workload across stations and strongly depend on the sequence in which the products are processed. While these processes are interconnected, they are typically addressed separately due to the complexities and different time frames involved. Even if some literature methods combine balancing and sequencing problems, these are limited in scalability, often constrained to small-scale instances with a limited number of products. This restricts their applicability to real-world industrial scenarios, where the number of product configurations can be vast. Another branch of the literature assumes a random sequence, which is a pessimistic view of a real setting. To fill this gap, we propose a genetic algorithm for the integration of the balancing problem with a semi-random production sequence that respects selected car-sequencing rules. This paper addresses the combined balancing and car sequencing rules selection problem for large-scale assembly lines with uncertain production sequences. By incorporating rules related to car sequencing, we reduce the randomness of a simulated production sequence used to evaluate the efficiency of an assembly balancing solution without the burden of solving an optimization problem. The impact of incorporating rules is evaluated through a discrete-event simulator, demonstrating significant improvements in line performance.

KEYWORDS
Mixed-model assembly-line balancing; Genetic algorithm; Discrete event simulation; Unpaced lines; Car sequencing

This is the peer-reviewed, accepted version of the paper. International Journal of Production Research, 2025, 63 (6), 2114–2132. DOI 10.1080/00207543.2024.2396021

1. Introduction

In the manufacturing industry, assembly lines play a crucial role in efficiently producing a wide range of products (Boysen, Fliedner, and Scholl 2007). An efficient configuration of an assembly line involves two key components: balancing the workload across

CONTACT C. G. S. Sikora. Email: celso.sikora@uni-hamburg.de

stations (Boysen, Fliedner, and Scholl 2008) and determining the sequence in which the products are processed (Boysen, Fliedner, and Scholl 2009a). While these components are interconnected, they are typically addressed separately due to the inherent complexities and different time frames involved (Sikora 2021).

Most assembly lines are designed to handle multiple products or a diverse family of products (Boysen, Fliedner, and Scholl 2008), making the production sequence a vital factor in achieving optimal line efficiency (Lopes et al. 2020b). However, at the stage of assembly line balancing, the specific production sequence is usually unknown as it depends on dynamic factors such as demand fluctuations and unforeseen operational constraints (Sikora 2021). Planners often face the challenge of building assembly lines based on demand forecasts, with the realized production sequence varying on a regular basis, sometimes even weekly.

To address the combined balancing and sequencing problem, several approaches have been proposed in the literature. These methods offer tailored solutions to different assembly line configurations, considering factors such as station types (e.g., synchronous, asynchronous, or hybrid) and stochastic processing times (see the most recent surveys by Boysen, Schulze, and Scholl (2022) and Battaïa and Dolgui (2022)). However, a common limitation of these existing methods is their scalability. The majority of heuristic and exact approaches are constrained to small-scale instances with a limited number of products, which significantly restricts their applicability to real-world industrial scenarios where the number of product configurations can easily be vast, reaching into the millions, billions, or even more. Boysen, Fliedner, and Scholl (2009b) report that an automotive manufacturer allows theoretically over 10^{24} possible variations of a vehicle.

Another branch of the literature addresses the balancing problem by assuming a random product sequence. This approach allows for the optimization of assembly lines based on statistical characteristics rather than specific production sequences. However, it fails to capture the reality of assembly line operations, which lies between the extremes of complete randomness and total control over the production sequence.

Therefore, this paper aims to fill the gap in the literature by proposing an integrated approach for balancing and (indirectly) sequencing optimization in large-scale assembly lines with uncertain production sequences. Our objective is to develop a framework that effectively addresses the inherent complexities of simultaneously optimizing balancing and consider sequencing decisions, while accounting for the practical constraints and variability encountered in real-world assembly line operations. For that, production sequences enforcing car sequencing rules are simulated to evaluate the quality of the assembly line balancing solutions.

In the subsequent sections, we will review the existing literature on assembly line balancing (Subsection 2.1) and car sequencing (Subsection 2.2), discussing the limitations of current approaches and highlighting the need for a comprehensive solution for large-scale assembly lines with uncertain production sequences. We will then present our proposed framework, which combines assembly line balancing with car sequencing in Section 3 along with the solution method consisting of a genetic algorithm and a simulator in Section 4. The experimental design is described in Section 5 leading to our result reports described in Section 6. Finally, the conclusions and directions for further work are summarized in Section 7.

2. Literature review

2.1. *Mixed model assembly line balancing*

The literature on assembly-line balancing has witnessed significant developments in addressing different aspects of practical lines and the combinations and extensions of problems (Battaïa and Dolgui 2022). A significant part of these advances focus on the assembly of multiple products or a wide family of homogeneous products (Boysen, Fliedner, and Scholl 2008), since the investment and labor costs are then shared among the different products. In some applications, there can be a product type that is the bottleneck (Sikora et al. 2017) or a product representing a large proportion of the relative demand (Sikora, Lopes, and Magatão 2017) that justify designing the assembly line for a single product. More generally, however, the methods have to consider the multiple products specifically (Boysen, Fliedner, and Scholl 2007).

When products requiring different processing times are assembled together, the production sequence plays a crucial role in determining the line efficiency (Lopes et al. 2020b). Furthermore, the type of the line (or more specificity product transport system) greatly effect how well a line operates. In unpaced lines, the movement of pieces is discrete. The products are stopped in each station and are moved forward to the next station after the tasks are finished. If the next station is still occupied by another product, the station is said to be blocked and the product has to wait in the station. An empty station that is waiting the product of a previous station is said to be starved. One example of the scheduling of a line of three stations and four products (M1, M2, M3, and M4) is displayed in Fig. 1. Each row represents a station and the abscissa represents time. Each darker block stands for the processing time of a product. The blocks without any filling (m2, m3, and m4) represents the time in which a station is blocked. The empty space between M2 and M3 in the third row shows starvation of Station 3.

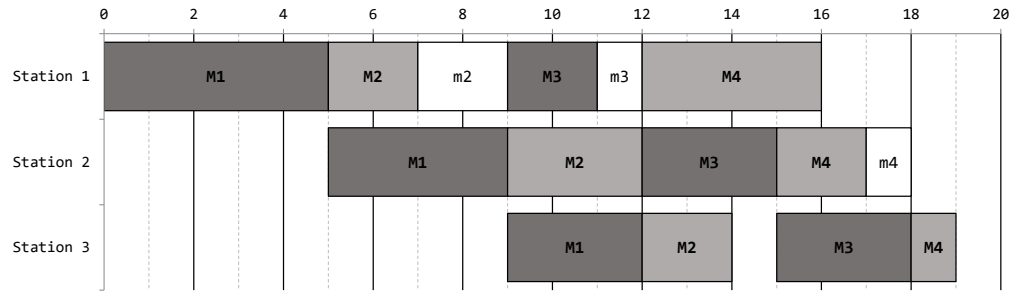

Figure 1. Example of the scheduling of an unpaced line with four products from Sikora (2022a). Every row represent a station while the x-axis represents time from left to right. Each block stands for a workpiece and its length is equal to the processing time in the corresponding station. While blocks show blockages. Alt text: Diagram with blocks representing the processing, blocking, and starvation times in three stations for four products. Every row represent a station while the x-axis represents time from left to right. Each block stands for a workpiece and its length is equal to the processing time in the corresponding station. While blocks show blockages.

Note that in the example of Fig. 1, products are moved whenever they are finished and the next station is free. Such transport system is said unpaced and asynchronous, since each station can expedite the product further independently. Unpaced synchronous lines also have a discrete movement, but the products of all stations are

moved at the same instant (Lopes et al. 2018a).

There are also paced assembly lines, for which the conveyor belt is set to a constant speed. Stations are defined as a length of the conveyor belt, in which the tasks are performed during the period the product is reachable. The worker moves along with the conveyor and returns to the start of the station after every cycle time. If the station is longer than the equivalent length of a cycle time (the length a product advances during a cycle time), the extra length serves as a buffer and allow longer processing times to be compensated with more simple products in the production sequence.

As the operation of assembly lines depends on the transport system used, optimization methods that integrate the balancing and the sequencing problems are usually tailored to these different assembly line configurations:

- Karabati and Sayın (2003) propose an MILP model and a heuristic for balancing unpaced synchronous assembly lines.
- Özcan, Kellegöz, and Toklu (2011) and Dong et al. (2014) propose an heuristic method for optimizing U-shaped lines considering stochastic processing times. Özcan, Kellegöz, and Toklu (2011) use an indirect objective function to minimize workload deviations while Dong et al. (2014) specifically considered paced assembly lines and minimized the expected utility work when stochastic processing times exceeded the cycle time.
- Öztürk et al. (2012) and Öztürk et al. (2015) developed a Constraint Programming and MILP formulations for unpaced synchronous lines with simple and parallel stations, respectively.
- Tiacci and Mimmi (2018) model unpaced asynchronous lines with parallel stations and stochastic processing times in a genetic algorithm.
- Lopes et al. (2018a) and Lopes et al. (2019) propose improved MILP formulations for unpaced lines. Lopes et al. (2018a) deal with both synchronous, asynchronous, or even hybrid lines, while Lopes et al. (2019) focus on asynchronous lines with parallel stations.
- Sikora (2021) optimizes a paced assembly line with uncertain demand. The problem is modeled as a two-stage stochastic problem. The balancing problem is solved in the first stage and the production sequencing problem is dealt with in the second stage, after the demand is known.

Even if the literature provides solution methods for the combined assembly line balancing and production sequence problem, there are strong limitations on their implementation in real systems. First, the combined problem is much harder to solve than considering the balancing problem independently. The instances solved by the contributions previously mentioned considered at most 7 product models (different types of products that can be assembled) in their instances. Nowadays, the combinations of options and add-ons that can be ordered result on millions, billions, or even more different products (Boysen, Fliedner, and Scholl 2008). This large number of different products are, however, not the main driver of complexity since in each station only a few different operations are observed. Nevertheless, modeling both balancing and sequencing problems together remains a challenge due to the immense combinatorial complexity.

A second limitation of the combined problem is that both problems do not have the same time frame. The design of an assembly line is a medium to long-term decision (Boysen, Fliedner, and Scholl 2008), while the production sequencing is a short-term operative problem (Boysen, Fliedner, and Scholl 2009a). In fact, the demand (and

product sequence) can differ daily or weekly depending on customers orders (Sikora 2021). Therefore, the lack of knowledge of the exact demand or the products to be sequenced when designing a line before its operation is a further challenge in a real application.

A way to mitigate the uncertainty on the future production sequence but still consider the scheduling effect of having multiple models is to consider a random sequence. Operating under such a sequence has been referred as Make-to-Order concept (Bukchin, Dar-El, and Rubinovitz 2002) and would be equivalent to producing whatever the customers purchase in the (random) sequence their products are ordered. The literature development in this stream consists of:

- Bukchin, Dar-El, and Rubinovitz (2002) and Manavizadeh et al. (2012) use a "bottleneck measure" as an approximation for the efficiency of a line. Bukchin, Dar-El, and Rubinovitz (2002) propose a heuristic method for finding good balancing solutions, which is improved in a metaheuristic framework by Manavizadeh et al. (2012).
- Tiacci (2015a) and Tiacci (2015b) couple a genetic algorithm with a simulator to evaluate balancing solutions in a simulated random sequence. Both contributions consider parallel stations and stochastic processing times. The allocation of buffers is also considered in Tiacci (2015b).
- Lopes et al. (2020a) also use a simulator to evaluate solutions as Tiacci (2015a). The assembly line problem is tackled with an iterative local search and a simpler simulator to quickly test the quality of solutions.
- Lopes et al. (2018b) propose a method based on Markov chains to calculate exactly the expected cycle time of unpaced lines instead of using simulation. The method is, however, very limited because the size of the Markov chains grows very fast relative to the number of stations.
- Sikora (2022b) adapted the Markov chains to precisely calculate the expected utility work for a random sequence, albeit limited to paced assembly lines. The method assumes that all products can be finished within each station, so that stations are treated independently. Instances with up to 53 tasks and over 10^{12} products are solved exactly.

Both these literature streams (combined balancing and sequencing problem / balancing problem for a random sequence) have their merits and their drawbacks. While the methods for the combined problem cannot solve large instances, designing the assembly line for a random sequence is over pessimistic. While this method offers computational convenience, it comes with significant limitations that detract from its practical value in real-world assembly line settings. Therefore, we propose an intermediary formulation that does not solve the sequencing problem but restricts the randomness of the production sequence. One example of partial sequencing is resequencing (Boysen, Fliedner, and Scholl 2009a), in which a disturbed (or random) sequence can be rearranged in a buffer (Lübben, Pries, and Sikora 2023). In our proposed method, we consider car sequencing rules, described in Subsection 2.2, to constraint the randomness of a simulated product sequence. Enforcing sequencing rules is a compromise about assuring a good production sequence without requiring the solution of a integrated sequencing problem.

2.2. *Car sequencing*

The optimization of the production sequence in mixed-model assembly lines is not an easy task. The production rate of large manufacturers can easily be bellow one vehicle per minute (Emde and Gendreau 2017) so the sequencing of the daily production alone already requires considering over 1000 products. The Car Sequencing (CS) Problem was introduced by Parrello, Kabat, and Wos (1986) and is an approach to find good production sequences using simple rules instead of focusing on an optimal solution. The illustrative example of this inaugural paper portray the mounting of the air conditioning. Suppose that historical data shows that fewer than 60% of the customers order a vehicle with air conditioning and the mounting of air conditioning requires a long processing time – five times the cycle time in the example of Parrello, Kabat, and Wos (1986). A possible solution for such case is to have three workers assigned to assure enough the installing capacity of 60%. If in a sequence of five products, up to three vehicles require air conditioning, a worker is assigned to each model. If a fourth vehicle with air conditioning is planned within five products, there will not be an idle worker for the installation. Therefore, a car sequencing rule 3:5 (at most three out of five) can be defined for the production is respect to the feature 'air conditioning'.

A feasible production sequence may have to obey a collection of car sequencing rules denoted by H:N (H out of N). For instance, vehicles may be ordered with or without sunroof, air conditioning, or cruise control. Each such a critical component (or option) requires a specific rule. Table 1 shows an example from Fliedner and Boysen (2008) and shows a feasible sequence of products respecting rules 1:2, 1:3, and 3:5. Each vehicle is represented with a column and while each row reflects a critical component. Cells with an X denote a vehicle with an option while $-$ stands for a product without such option. A sequence is feasible if for every partial sequence of N_j vehicles respects the rule for option j. If a ninth vehicle were to be added in the sequence, it must not contain any of the three options. Otherwise the window based on products 8-9, 7-9, and 5-9 would present 2, 2, and 4 vehicles with the three components, respectively.

Table 1. Example of a sequence respecting car-sequencing rules based on Example 4 of Fliedner and Boysen (2008).

	Product							
Rule	1	2	3	4	5	6	7	8
1:2	X	-	X	-	-	X	-	X
1:3	X	-	-	X	-	-	X	-
3:5	-	X	X	-	X	-	X	X

The CS problem is described as often used in the industry (Golle, Rothlauf, and Boysen 2014). In fact, from the scheduling literature on the automotive industry, most of the real instances are based on this problem (Gagné, Gravel, and Price 2006). CS instances from the *Renault Group* were even the focus of the ROADEF Challenge 2006 (Solnon et al. 2008). The ROADEF Challenge is organized by the French Society of Operations Research and foster a competition to solve real industrial optimization instances.

The literature on solution methods for CS is dominated by heuristic and metaheuritic methods (Gagné, Gravel, and Price 2006; Solnon et al. 2008; Golle, Rothlauf, and Boysen 2015; Sun and Fan 2018; Thiruvady et al. 2020) but also presents some exact methods such as branch-and-bound (Fliedner and Boysen 2008) and constraint programming (Dincbas, Simonis, and Van Hentenryck 1988). Recently, Sun et al. (2022) compared the performance of different exact and heuristic methods for

the available benchmarks and proposed new harder instances. The authors also propose a visualization method for the instances' characteristics and discussed about the drivers of instance difficulty.

Even though the literature on CS is fruitful, there are some inherent disadvantages and shortcomings of using this method to solve the sequencing of the automotive production. If no sequence rule is violated, then usually the production is feasible. However, the real cost (cycle time, amount of utility work, rework, etc.) required if a rule cannot be fulfilled depends on the objective function used for CS (Fliedner and Boysen 2008; Golle, Rothlauf, and Boysen 2014). In the literature, multiple forms of counting the number of violations are proposed (Fliedner and Boysen 2008) but none of them provides the real effect in the production (Golle, Rothlauf, and Boysen 2014). A comparison between CS and the Mixed-Model-Sequencing (MMS) Problem is discussed in Golle, Rothlauf, and Boysen (2014). The MMS explicitly schedules the position of the worker in each cycle time. The MMS requires more data than the simple H:N rules but computes directly the effect of the sequence. Even if the MMS deliver better solutions, Golle, Rothlauf, and Boysen (2014) discuss that the industry still rely more on CS due to its simplicity. The sequencing rules H:N are easy to understand and to communicate between the production and marketing departments.

Another shortcoming of the CS is the assumption that only two kinds of products are produced in a station (Golle, Boysen, and Rothlauf 2010). As the introductory example, the rules are defined based on only two options in a station. If three or more options are to be produced in a station, the rules H:N cannot be applied as described. Golle, Boysen, and Rothlauf (2010) explore this case and provide a method to generate sequencing rules for this case. The rules for multiple products, however, do not exactly translate the simple H:N rules and may require multiple rules.

Even with these disadvantages, we use CS rules to simulate pseudo random sequences close to sequences that would be planned in real applications.

3. Problem description

The proposed problem consists in finding a good assignment for a set of tasks $i \in T$ to P work centres of an unpaced line operating under a partially random sequence. The definition of the tasks, the production sequence, the notation, and the sequencing rules are given in the following sections.

3.1. *Tasks and product variants*

A set of tasks T (numbered with $i = 0, \ldots, |T| - 1$) has to be performed within the production line in order to assembly the required products. Tasks also can have options, which can be either be an optional component (such as sunroof) or two different variations of a component (such as a regular seat or a heated seat). For each task, there is at most two options: the one requiring the least amount of processing time (t_{iL}) is named L (Lowest Work Intensive option), while the most time consuming task (t_{iH}) is indexed with H (Highest Work intensive option). We assume that there are three type of tasks. The **normal tasks** are identical for all products and do not present a second option. For one of such task i, the relative demand (d) of option L is then 100% or $d_{iL} = 1$, while $d_{iH} = 0$. The second and third type of tasks present different options. The **variant task** allows two different components to be mounted in the product. The relative demand of products with the L component is $0 < d_{iL} < 1$,

while the H variant has the relative demand of $d_{iH} = 1 - d_{iL}$. One example of such task can be the mounting of a regular seat or a heated seat in a vehicle. Finally, the third type of tasks are named **optional tasks**. These tasks represent a component that is not required in each product and can be ordered as an add on, such as a sunroof. We denote the optional variant as the H variant, whose demand is $0 < d_{iH} < 1$, while the variant L has the processing time $t_{iL} = 0$.

A product (or product model) is described by the specifications of all components as ordered by the costumer. A model can be, for instance, a vehicle without sunroof, with air conditioning, and with cruise control. The number of models that can be produced is equal to 2^{T^*}, in which T^* represents the number of tasks with two options. In other words, a variant is defined by a sequence $(j_0, j_1, ..., j_{|T|-1})$ selected among the possible set of options for each task.

As at most two options are available for each task, we use the notation of Table 1 and use X for the most complex option and $-$ for the least demanding option in terms of processing time. Product 1 in Table 1 is therefore referenced as $(X, X, -)$.

Besides the processing times, the average sales proportion of each option is also available as given data. Let $d_{ij} \in [0, 1]$ be the relative demand of option j for the component mounted with task i. The value of d_{ij} can be seen as the probability that option j is purchased by the client. These probabilities are assumed to be valid in average while variations in the daily production are allowed. One sample instance can be found in Table 3 and a sample solution is given along with the algorithm description in Section 4.

3.2. *Production sequence*

As the time frame of the ALB and the sequencing problems are different, there is not an obvious way to model the production sequence for the medium to long term design of assembly lines. In the literature (section 2.1) there are papers that assume the production sequence to be random, while others assume total control of the sequencing in the planning phase by solving a combined ALB and sequencing problem. Both of these conditions are unrealistic since the planner can enforce influence on the production sequence but the solution of a real sequencing problem with potentially billions of product variants is not manageable in the planning phase. In this work, we propose a condition in between by using partial random sequences.

Suppose each task i presenting two options has a meaningful CS rule in the form $H_i : N_i$, that is, at most H_i of products containing the most complex option (in terms of processing time) of task i is allowed in a sequence of N_i products. Assume that R is a set containing some of the defined sequencing rules. A partial random sequence is defined as a sequence of products drawn randomly based on the probability distributions for each task while obeying a set of sequencing constraints R. In the process of generating sequences, if one sequencing rule were to be violated, the option is corrected to the most simple one. We use the sequence of Fig. 1 as example. The selection of the 8^{th} product $(X, -, X)$ would also be feasible for the products $(-, -, -)$, $(X, -, -)$, and $(-, -, X)$. For the 8^{th} product, the only prohibited assignment is to have a complex option in the second row due to the 1:3 rule. For the 9^{th} model, the only feasible product is $(-, -, -)$ as any X option would violate a CS rule.

A list of T^* CS rules (T^* is the number of tasks with two options) may result in a highly restricted CS problem to be solved. Therefore, we assume that only a subset of the CS rules should be considered in the planning of an assembly line. The production

sequence is then simulated by random draws while respecting the selected rules.

The combined problem consists of the assignment of tasks of the assembly line plus selecting the most critical components to define the CS rules.

As described in Section 2.2, one of the disadvantages of CS is that CS rules only work as intended originally if there is two options within a station. We relax this condition and allow multiple tasks to be assigned in a station. A drawback is that a feasible sequence without any CS violation can still be a bad production sequence. We show this in one example: If, for instance, the balancing solution assigns the tasks related to the air conditioning and the cruise control to the same station and only one the CS rule for air conditioning is selected, any sequence in respect to cruise control is allowed. That is, even if the products with air conditioning are sequenced well, a long sequence with products all with cruise control is still possible.

By solving the ALBP along with the selection of CS rules, it is expected that good solutions will identify the critical components (options that without control in the sequence lead to bad solutions), apply CS rules to them, and possibly assign the critical components to different stations. A solution to the problem not only provides the design of an assembly line but also points out what practitioners have to pay attention to in terms of sequencing.

3.3. *Notation and assumptions*

A summary of the notation introduced in the explanation of the tasks along with the variables that are yet to be introduced in the algorithm is shown in Table 2.

The following assumptions are required to described the problem defined further in Subsection 3.4.

- A set of $|T|$ tasks has to be performed in the line in order to complete each product.
- Tasks are either **normal** tasks (the same task for all products), **variant** tasks (presents two options with non-zero processing times), or **optional** tasks (the most basic product does not require this task / do not has a component, while the most complex product requires processing time greater than zero. The task is described as optional since not all products require them).
- The time needed to perform the option type j of task i is stochastic, normally distributed, with mean value t_{ij} and standard deviation $\sigma_{ij} = cv \cdot t_{ij}$, where cv is the coefficient of variation.
- The assembly line is composed by a series of P Work Centres (WCs). Each WC consists of either one Workstation (WS), for the case of non-paralleling, or multiple parallel WSs.
- In each Workstation there is a human operator that performs one or more tasks. An annual cost CW is associated to each operator.
- The way tasks are assigned to WCs is constrained by the precedence relations among tasks. Precedence constraints among tasks are defined by a precedence diagram (see Figure 2).
- A task can be performed only if all its predecessors have already been performed. Precedence constraints can be mathematically formulated as follows (adapted from Miltenburg and Wijngaard (1994)). Given a set of tasks $T = \{i | i = 0, 1, \ldots, |T| - 1\}$, a set of precedence constraints $C = \{(x, y) |$ task x must be completed before task $y\}$, find a collection of subsets of T, $(T_0, T_1, \ldots, T_{P-1})$ where $T_k = \{i |$ task i is done at work centre $k\}$, that satisfy the following con-

Table 2. Parameters and variables of the problem.

Sets	
T	the set of tasks with index i ($i = 0, \ldots, N-1$)
T^*	subset of the tasks containing two options
V	the set of variant tasks ($V \subseteq T$)
O	the set of optional tasks ($O \subseteq T$ and $V \cap O = \emptyset$)
Problem parameters	
L	index for the lowest work intensive option of a task
H	index for the highest work intensive option of a task
P	the total number of work centres in the line (index $k = 0, \ldots, P-1$)
ct	imposed cycle time
t_{ij}	the average time required to execute the task i on option j [time units]
σ_{ij}	the standard deviation of the time required to execute the task i on option j [time units]
cv	the coefficient of variation of t_{ij}
d_{ij}	the demand proportion for option j of task i
r_i	possible sequencing rule for task i
CW	annual cost of a worker [€/year]
CE	annual cost of a piece of equipment [€/year]
Decision makers' parameters	
ζ	the penalty factor related to the cycle time constraint
Variables	
T_k	the set of tasks assigned to work centre k
N_k	the number of tasks assigned to work centre k
W_k	the number of workers (or workstations) in work centre k
E_k	the number of pieces of equipment in work centre k ($E_k = W_k \cdot N_k$)
Performances measurements	
ct_{eff}	the realized average cycle time [time units]
DC	the design cost of the line
NDC	the normalized design cost

ditions:

$$\bigcup_{k=1}^{P-1} T_k = T \tag{1}$$

$$T_k \bigcap T_w = \emptyset \qquad \forall k \neq w \in \{0, 1, ..., P-1\} \tag{2}$$

$$\text{if } (x, y) \in C, x \in T_k, y \in T_w, \text{then } k \leq w, \forall x, y \in T \tag{3}$$

Equation (1) ensures that all tasks are assigned to a work centre. Conditions (2) ensure that each task is assigned only once. Conditions (3) ensure that the precedence constraints are not violated. In fact, if a task x must be completed before a task y, it must be assigned in the same WC where task y is done or in a preceding WC.

- In the case of paralleling, all of the WC's tasks are not split among the WSs, but each WS completes them all. Thus, the addition of one (or more) WS and the related operator who performs the same set of operations increases the WC's production capacity.
- Each task requires a different piece of equipment to be performed, to which a

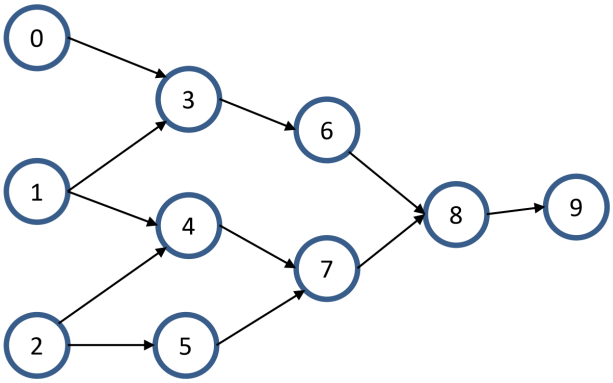

Figure 2. Exemple of a precedence diagram where nodes represent tasks numbered from 0 to 9 and arcs represent precedence relations between tasks. Alt text: A graph with nodes and arcs representing the precedence relations of an assembly line.

cost CE is associated. Thus, if one station is parallelized there is an extra cost due to equipment duplication.

- There are no buffers between WC.
- The line is asynchronous, that is blockage as well as starvation are possible.
- The first WC is never starved (there is always raw material for the first WC) and the last station is never blocked (there is always storage space for the finished product).
- One WC with multiple WSs is considered busy if every WS inside is busy. If a WS finishes its work on a workpiece while the subsequent WC is still busy the workpiece cannot move on, and remains in the current WS keeping it busy; the WS will be released (i.e., it will be able to process another workpiece) only when the workpiece leaves the WS.
- Products enter the line with a sequence able to comply with the demand proportion of each option of each task.
- While respecting the demand proportion of each option of each task, variant and optional tasks are characterized by a possible sequencing rule r_i: the rule $r_i = H_i : N_i$ means that the high work intensive option of task i can be performed at most H_i times every N_i products (see Subsection 3.2).

A sample of a problem instance is displayed in Table 3.

Table 3. Data of an example of an instance of the problem with 10 tasks, with imposed cycle time of 14 time units. The table contains average completion times, scheduling rules and demand proportions of tasks.

task#	type	t_{iL}	t_{iH}	r_i	d_{iL}	d_{iH}
0	Variant	7	9	4:6	0.334	0.666
1	Normal	3			1	0
2	Optional	0	7	3:5	0.500	0.500
3	Variant	9	14	1:4	0.867	0.133
4	Normal	10			1	0
5	Variant	6	9	1:3	0.777	0.223
6	Normal	5			1	0
7	Optional	0	10	2:5	0.687	0.313
8	Normal	4			1	0
9	Normal	3			1	0

3.4. *The problem: solving the MALBP using car sequencing rules*

The goal is to create a line configuration that minimizes a cost-oriented objective function while satisfying task precedence constraints. A line configuration is defined by the elements that reflect the problem's variables:

- the number of work centres in the line, P
- the number of workstations in each work centre, W_k
- the tasks assigned to each work centre
- which are the variant tasks in V and the optional tasks in O for which a sequencing rule is applied.

Task precedence relations reflect which tasks must be done before others and limit how tasks are assigned to WCs. The imposed cycle time (ct) is equal to the inverse target line throughput. Because the line is asynchronous, the imposed cycle time has no direct influence on the line's speed. On the contrary, the line configuration must be built in such a way that the average effective cycle time (ct_{eff}) achieved by the line is close to the imposed one (ct) while also having the lowest design cost possible. To achieve this goal, the compliance with the performance objective (ct close to ct_{eff}) is inserted in a cost-based objective function known as Normalized Design Cost (Tiacci, Saetta, and Martini 2006).

The Design Cost, i.e. the overall annual cost for workers and equipment of a line layout, is used to evaluate costs:

$$DC = \sum_{k=0}^{P-1} (CE \cdot E_k + CW \cdot W_k) \tag{4}$$

The performance objective is met by imposing a penalty on line configurations whose effective average cycle time exceeds the prescribed one. This is accomplished by computing the Normalized Design Cost (NDC). If $ct_{eff} \leq ct$, the NDC matches the Design Cost; otherwise, it is increased:

$$NDC = \begin{cases} DC & \text{if } ct_{eff} \leq ct \\ DC \left[1 + \zeta \left(\frac{ct_{eff} - ct}{ct}\right)^2\right] & \text{if } ct_{eff} > ct \end{cases} \tag{5}$$

where ζ is a penalty factor that can be assigned based on the needed compliance with ct. The cycle time constraint grows more stringent as assumes higher values. The NDC reflects the fact that in real-world situations, a small deviation in average cycle time from the theoretical one is permitted, especially if this results in consistent savings in terms of line configuration.

The cost-oriented mindset that many managers and practitioners adopt in businesses serves as the driving force behind the decision to use a singular objective function for a multi-objective problem rather than Pareto frontiers or other approaches. Managers can directly relate the failure to meet performance target values to costs by using the penalty coefficient, whose value can be opportunely set by decision-makers.

4. The genetic algorithm approach

This section explains how the given problem has been solved using a genetic algorithm combined with an event/object oriented simulator to determine an individual's fitness.

The necessity to overcome the limitation of utilizing conventional metrics of the performance of the line that are not simulation-based and are only weakly associated to its true value spurred the notion to couple a balancing algorithm with a simulator. Battaïa and Dolgui (2022) noted in a recent review on new trends and formulations about line balancing that simulation has been used much more frequently in recent years. They also noted that this is consistent with the development of digital twins and the desire to have a digital model that is identical to the physical one in order to aid in decision-making. The decision to use a genetic algorithm coupled with a simulator is based on the ability of GAs to have a strong exploratory component and to be easily integrated with a simulator for fitness assessment.

A genetic algorithm frequently converges on a good solution by using natural selection and survival of the fittest to increase the quality of the initial population of individuals, each individual representing a feasible solution of the problem. The method starts with a population of random solutions and then uses genetic operators, such as crossover and mutation, to generate new ones. Each individual of the new generation is evaluated through a fitness function, which measures how well the solution solve the problem. The fittest individuals are chosen to reproduce and generate new children who inherit the characteristics of their parents. This process continues for a number of iterations, each iteration corresponding to a new generation of individuals. At the end of the iterations, the individual in the final population with the highest fitness is considered the final solution.

The proposed genetic algorithm is an extension of the algorithms described in Tiacci (2015a, 2022), and utilizes three chromosomes to represent and individual, which corresponds to a feasible solution of the problem. Although the algorithm has similarities with already published methods, its description is fully given to provide a self-contained explanation of the method. The novel developments are related to the modeling and treating of the sequence rules within the genetic algorithm and the simulation model, which has required to change the form of and the meaning of the third chromosome, as described in the next subsection.

4.1. *Representation and decoding*

Figure 3 shows the chromosomal representation of an individual which corresponds to a feasible solution of the sample instance of Table 3. An individual is represented through three chromosomes. The first chromosome CH1 is formed by a number of genes equal to the number of tasks ($|T|$) and can be seen as a vector representing an ordered sequence of all tasks. Because their order corresponds to the order in which they are assigned to WCs, the sequence must always adhere to the precedence criteria.

The second chromosome CH2 is also a vector with $|T|$ elements that determines which WC gets assigned the duty. The WCs on this chromosome are numbered in ascending order, beginning with the first in the line and ending with the last. As a result, when a task is allocated to a different WS than the previous task, the matching value in the second chromosome is increased by one. Using Figure 3 as an example, tasks 1 and 0 will be assigned to WC#0, tasks 3, 2 and 5 to WC#1, tasks 4 and 7 to WC#2 and tasks 6, 8, and 9 to WC#3. It is necessary to determine the number

of workers (i.e. WSs) in each WC after all tasks have been delegated to WCs. This is accomplished by requiring each WC to complete all jobs with a specific probability, *pc*. To do this, the so-called "average model" is taken into account. The time required to complete each job i is first computed using the weighted average completion time $\hat{t}_i$ and the weighted standard deviation $\hat{\sigma}_i$:

$$\hat{t}_i = \sum_j d_{ij} t_{ij}, \qquad \hat{\sigma}_i = \sqrt{\sum_j d_{ij} \sigma_{ij}^2} \tag{6}$$

The total amount of time required for a WC to complete all of the tasks allocated to it will thus follow a normal distribution $\mathcal{N}\left(\sum_i \hat{t}_i, \sqrt{\sum_i \hat{\sigma}_i^2}\right)$, with a mean value equal to the total of the average task times, and a standard deviation equal to $\sqrt{\sum_i \hat{\sigma}_i^2}$. The predicted time T_{limit} required to finish all the tasks given to a WC with a probability pc can be calculated using the inverse normal distribution function. The number of employees W_k assigned to the WC will equal the minimal integer value causing $ct \cdot W_k$ to exceed T_{limit}. Table 4 demonstrates how the number of workers W_k for the individual depicted in the top of Fig. 3 is determined, considering an imposed cycle time $ct = 14$, and $pc = 80\%$. It is noteworthy how in the example shown in Table 4, the various tasks have a different coefficient of variation *cv*. This shows that the algorithm can work even without all tasks necessarily having the same *cv*. In generating the instances with which we then tested the algorithm, we assumed for simplicity's sake the same *cv* for all tasks, but this does not preclude the algorithm from being applied in the event that the experimental data allow us to determine different standard deviations for the various tasks.

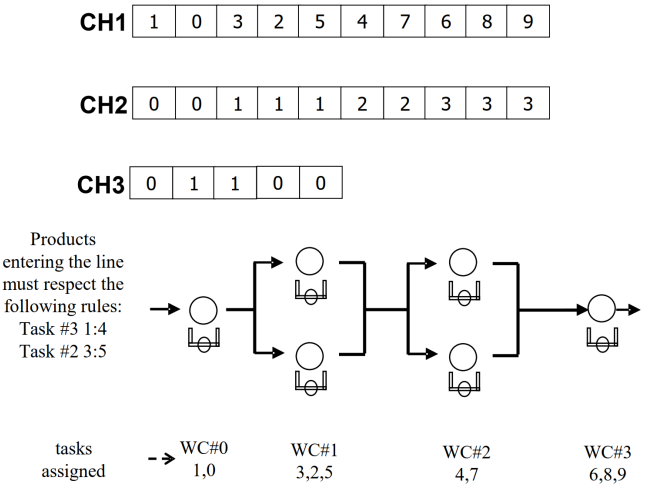

Figure 3. The decoding procedure and the line configuration corresponding to the chromosomal representation. The three rows of integer values are displayed symbolizing the three chromosomes of the algorithm. The line structure coded in the top part of the figure is shown in the bottom of the picture. Four stages of the line are manned with either single or parallel stations. The first and fourth station has a single worker, the middle stations have two workers. Alt text: Three rows of integer values are displayed symbolizing the three chromosomes of the algorithm. The line structure coded in the top part of the figure is shown in the bottom of the picture. Four stages of the line are manned with either single or parallel stations. The first and fourth station has a single worker, the middle stations have two workers.

Table 4. Calculating the number of workers W_k in each WC ($ct = 14, pc = 80\%$).

WC_k	task	$\hat{t}_i$	$\hat{\sigma}_i$	$\sum_i \hat{t}_i$	$\sqrt{\sum_i \hat{\sigma}_i^2}$	T_{limit}	W_k
0	1	3.00	0.90	11.33	2.67	13.58	1
	0	8.33	2.52				
1	3	9.67	2.94	19.83	3.97	23.10	2
	2	3.50	1.48				
	5	6.67	2.04				
2	4	10.00	3.00	13.13	3.44	16.02	2
	7	3.13	1.68				
3	6	5.00	1.50	12.00	2.12	13.79	1
	8	4.00	1.20				
	9	3.00	0.90				

The third chromosome CH3 provides information on which of the variant and optional tasks should adopt the corresponding sequencing rule. In Tiacci (2015a), the third chromosome represented the fixed sequence of models entering the line expressed in terms of the Minimum Part Set, i.e. the shortest sequence of models able to satisfy the proportion of demand for each model. By repeating this sequence during the planning horizon, the total demand for each model is satisfied. In the present case, however, we do not have to decide on an exact sequence of models entering the line, but rather determine whether or not to adopt a scheduling rule for each variant and optional task.

To this aim, the third chromosome has a number of genes equal to the sum of variant and optional tasks. Each gene of the chromosome contains a binary value, 1 if the sequencing rule is selected, 0 if the sequencing rule is not selected. Each gene is associated to a corresponding variant/optional task following the order in which they appear in CH1. In the sample instance, there are 3 variant tasks and 2 optional tasks, so that the length of the chromosome equals 5. In the example, the variants/optional tasks appear in CH1 in the following order: 0, 3, 2, 5, 7. This means that, being CH3 = {0, 1, 1, 0, 0}, for the variant task #0, the variant task #5 and the optional task #7 the corresponding sequencing rules are not selected, while for the variant task #3 and the optional task #2 the corresponding sequencing rules ($1:4$ and $3:5$) are applied.

4.2. *Initial population*

The initial population, composed by N_P individuals, is created randomly. For each individual, the first chromosome is generated, respecting the precedence constraints, following the procedure adopted by Kim Y. K., Kim, and Kim (1996):

(1) Begin by forming an initial set of tasks that have no predecessors and create an empty string, which will serve as an empty chromosome.
(2) If the available set is empty, terminate the process. Otherwise, proceed to Step 3.
(3) Randomly select a task from the available set and add it to the string.
(4) Update the available set by removing the selected task. Additionally, include every immediate successor of the task in the available set if all of its immediate predecessors are already present in the string. Return to Step 2.

Note that in Step 4, the available set is updated in accordance with precedence constraints to ensure the generation of a feasible sequence at all times.

To construct the second chromosome, the first position of the vector is originally set to 0. The following place can then have the same value as the previous one, or a value increased by 1. This choice is picked at random, with a probability of 50%.

The generation of the third chromosome is performed by generating a number of genes equal to the sum of the variant and optional tasks. Each gene can assume a value of 1 or 0 with equal probability, which represents, as described in the preceding section, the selection or non-selection of the rule for the corresponding variant/optional task in $CH1$.

4.3. *Crossover operator*

The Crossover operator is used during the reproduction phase. This involves two individuals, identified as parent1 (p1) and parent2 (p2), who generate two children, s1 and s2. All three chromosomes are subjected to the Crossover operator.

A single point crossover operator is utilized for the first chromosome (CH1) (top of Fig. 4). In parent1, a random cut point c is chosen (p1). Following the cut point, the elements are transferred into the matching location in offspring s1. These elements are then removed from parent2 (p2), and the remaining elements of p2 are copied in the same order as they appear in p2 into the beginning places of s1. This procedure ensures the development of a feasible sequence that complies with precedence constraints.

A different technique is used for the second chromosome (CH2) (middle of Fig. 4). The same cut point c is used as in the first chromosome. The values in the genes preceding the cut point are copied from p1 to s1 in the same order. Let d indicate the difference between the last copied gene from p1 and the corresponding gene from p2. The values in p2 to the right of the cut point are then transferred, but with an additional increment of d, into the identical locations in s1. Thus, the first value in s1 after the cut point can either match the value of the gene (position) before it or, at most, increase by 1.

The crossover applied to the third chromosome CH3 (bottom of Fig. 4) simply consists in concatenating a part of the chromosome from each parent. A crossover point c' is randomly selected along the chromosomes of the parent individuals. The chromosomes of the parents are divided at the crossover point. The genetic material before the crossover point in p1 is combined with the genetic material after the crossover point in parent 2 to form the chromosome of the first offspring. The second offspring s2 is generated inverting p1 and p2 roles.

4.4. *Mutation operator*

The operator is applied to each of the individual's three chromosomes when a mutation takes place. For the first chromosome a task is randomly selected (for instance, task 5 in top of Fig. 5), and its location within the chromosome is swapped with another task. To maintain a feasible sequence of tasks, in the new position the selected task must come before all of its immediate predecessors and come after all of its immediate successors. The feasible insertion locations form a substring within the parent chromosome and are sequential. Random selection is used to choose one of the possible insertion places.

The mutation takes place as follows when it comes to the second chromosome (middle of Fig. 5): a random gene within the chromosome is picked. Let p stand for the

value in this gene and l for the value in the gene before it. Obviously, $l \leq p$. All values from p forward are increased by one if $l = p$. This signifies the division of a single Work Centre (WC) into two. On the contrary, all values from p forward are decreased by one if $l < p$. This denotes the merger of two WCs into one.

The third chromosome is mutated by switching two randomly selected genes (bottom of Fig. 5).

4.5. *Fitness evaluation: the Assembly Line Simulator (ALS)*

The fitness function, which assesses the quality of an individual, is known as the Normalize Design Cost (eq. (5)). The evaluation of fitness is conducted in two phases.

The first phase involves calculating the design cost of the assembly line represented by an individual. This is accomplished by applying the decoding procedure outlined in Subsection 4.1. This procedure gathers information regarding the number of working centers (WCs) in the assembly line, the number of workers in each WC, the tasks assigned to each WC (including the corresponding equipment), and the sequencing rules for option/variant tasks. We will refer to this collection of information as the "line configuration". The design cost (DC) of the line configuration associated with the individual can be straightforwardly calculated using eq. (4).

The second phase involves calculating the Normalized Design Cost, which penalizes solutions that have an effective cycle time exceeding the target cycle time. To determine the effective cycle time of the line configuration represented by the individual, a simulation run must be performed. For this purpose, the Assembly Line Simulator (ALS) (Tiacci 2012) has been integrated into the fitness evaluation phase of the genetic algorithm. ALS is a Java-based discrete event simulator for assembly lines, developed under the object-oriented simulation paradigm. Object-oriented simulation has gained importance in the context of the Industry 4.0 paradigm, where optimization through simulation, real-time simulation, automated decision systems based on simulation, and online scenario analysis play significant roles (Tiacci 2020). The object-oriented architecture of ALS enables it to calculate the average cycle time of a complex assembly line directly. It does so by receiving input arrays that represent task durations, the line configuration, and the sequence of products entering the line, eliminating the need for manual reconstruction of the simulation model as the line configuration changes.

A specific version of ALS has been developed in order to consider the different data structure needed to represent the normal and the high work intensive options that characterize optional/variant tasks, with respect to the classic concept of 'model'. This required changing the source code of ALS, consisting in introducing specific attributes in the Load Class that allow determining, when creating a load, which of the two options, normal or high work intensive, must performed for each task. Furthermore, a different method for generating random sequences of loads entering the line has been implemented. In fact, the simulator cannot simple generate random sequence based on the options proportions d_{iL} and d_{iH} and expect the sequencing rules to be obeyed, because sequences of loads with more than H out of N optional or variant tasks are prohibited if the corresponding rule is selected. Therefore, the method to generate random sequences of loads has been adjusted in order to maintain the given options proportion and at the same time to respect the rules selected for each task. In order to do so, it is necessary to correct the probabilities of d_{iL} and d_{iH} with which the options N and H respectively are generated, in order to consider precisely that when a sequence does not comply with the selected rule, it must be discarded. Subsection

4.6 illustrates how the corrected probabilities were calculated.

4.6. *Correction of the model probabilities*

When simulating a random sequence under car-sequencing rules of a product with an option j with a relative demand of d_{ij}, one cannot simply draw the option with the same probability d_{ij}. As a random draw also contains sequences that are forbidden, the resulting simulation will result in less products with the option as required. In this section, we propose a corrected probability θ for the random draw of an option H (represented with X), whose resulting relative demand is equal to the required d_{iH}.

The calculation of the corrected probabilities θ can be performed by using Markov chains to model all possible subsequences. We illustrate the procedure for the rules $1:3$ and $2:3$ and show the resulting formulas for all rules up until 6 products in Table 5.

For a rule of the type $H:N$, consider all possible sequence of products with length $N-1$. Depending on the amount of products with a given option, the next product can be drawn with an option with a probability θ or a product without the option must be enforced.

For the case of rules $1:3$ and $2:3$, four possible subsequences with length 2 (3-1) are: $--$, $-X$, $X-$, and XX. Here, $-$ stands for a product with the simplest option j and X for a product with the most complex option. A sequence $-X$ stands for a first product without the option and a second product with the option. Suppose a station has observed the subsequence $-X$.Without any restrictions, the next product can have any option j, resulting in the sequences $-XX$ or $-X-$. Considering only the last two products, the subsequences are expressed as XX and $X-$. Depending on the sequencing rules, some of these transitions can be prohibited. For instance, if the rule $1:3$ must be considered, after the subsequence $-X$, the next model cannot contain the option and the next subsequence must be $X-$.

The possible transitions can be transcribed in a matrix along with the probability θ of a model containing option j. For rule $1:3$, matrix

$$\mathbf{T_{1:3}} = \begin{array}{c} \\ -- \\ -X \\ X- \\ XX \end{array} \begin{array}{c} \begin{array}{cccc} -- & -X & X- & XX \end{array} \\ \begin{pmatrix} 1-\theta & 0 & 1 & 0 \\ \theta & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

represent the transition probability, while matrix

$$\mathbf{T_{2:3}} = \begin{array}{c} \\ -- \\ -X \\ X- \\ XX \end{array} \begin{array}{c} \begin{array}{cccc} -- & -X & X- & XX \end{array} \\ \begin{pmatrix} 1-\theta & 0 & 1-\theta & 0 \\ \theta & 0 & \theta & 0 \\ 0 & 1-\theta & 0 & 1 \\ 0 & \theta & 0 & 0 \end{pmatrix} \end{array}$$

can be used to describe a $2:3$ rule. These matrices are transition matrices from Markov processes and each column describes the transition for the current state to the next state depicted in the rows. The first column of $\mathbf{T_{1:3}}$, for instance, shows that

after a subsequence $--$, there is a probability of θ to transition to a state $-X$, that is, a product with option H entered, while the system can remain in state $--$ (the next product does not have the option H) with probability $1-\theta$. For all other states, the transition is deterministic, since only one product can contain the option in each sequence of 3 products. For the rule $2:3$, more states are possible. The state XX, however, requires that the next product does not contain the option, as depicted in the last column of $\mathbf{T_{2:3}}$.

Transition matrices from Markov processes can be used to calculate the steady-state probabilities of the states in which the system occupies. By solving the system $xT = x$, the vector x represents how often each subsequence occurs. If these values are multiplied with the amount of products with an option in each subsequence, we obtain the expected amount of products with the given option. For a rule of the type $X:3$ (X is 1 or 2), the number of products with an option is given by

$$\mathbf{Prod_{X:3}} = \begin{matrix} -- \\ -X \\ X- \\ XX \end{matrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

Vector $\mathbf{Prod_{X:3}}$ represent the expected number of products with an option X in every subsequence of length 2 and is ordered in the same way as matrices$\mathbf{T_{1:3}}$ and $\mathbf{T_{2:3}}$.

Finally, as subsequences of length $N-1$ are considered, we can write

$$x^T Prod_{H:N} = (N-1)d_{iH}. \tag{7}$$

x_T is a function of θ, while $Prod_{H:N}$ and d_{iH} are numeric vectors and (N-1) a scalar. Equation (7) can be solved in respect to θ, the corrected probability of a given option. In this paper, we provide the resulting expression of solving the Markov Chain linear system (finding x) and substituting in eq. (7) for every rule up until $N = 6$ in Table 5. The equations were solved using Wolfram Mathematica. For further sequences, the same steps (solve linear equations of the Markov Chain in function of θ and using Eq. (7) should be used).

5. Experiment design

In order to test the effectiveness of incorporating car sequencing rules in the planning of mixed-model assembly lines, we compare the proposed approach, which provides the possibility to use specific rules for variant and optional tasks, with the random approach. To this aim, two versions of the algorithm has been tested on 45 benchmark instances from the literature (see the next section for details):

- GA-Rules. This is the genetic algorithm approach presented in this paper, able to balance the line and to specify which rules among variant and optional tasks must be adopted in order to minimize the Normalized Design Cost.
- GA-Random. This algorithm, presented in Tiacci (2015a), adopts the same approach of GA-rules, but does not provide the possibility to select the rules for optional and variant tasks. In this way, the sequence of models entering the line is random, drawn based on a uniform distribution for the given demand proportions.

Rule	Equation
1:2	$\frac{\theta}{1+\theta} = d_{iH}$
1:3	$\frac{2\theta}{1+2\theta} = 2d_{iH}$
2:3	$\frac{2\theta(1+\theta)}{1+\theta+\theta^2} = 2d_{iH}$
1:4	$\frac{3\theta}{1+3\theta} = 3d_{iH}$
2:4	$\frac{3\theta(1+2\theta)}{1+2\theta+3\theta^2} = 3d_{iH}$
3:4	$\frac{3\theta(1+\theta+\theta^2)}{1+\theta+\theta^2+\theta^3} = 3d_{iH}$
1:5	$\frac{4\theta}{1+4\theta} = 4d_{iH}$
2:5	$\frac{4\theta(1+3\theta)}{1+3\theta+6\theta^2} = 4d_{iH}$
3:5	$\frac{4\theta(1+2\theta+3\theta^2)}{1+2\theta+3\theta^2+4\theta^3} = 4d_{iH}$
4:5	$\frac{4\theta(1+\theta+\theta^2+\theta^3)}{1+\theta+\theta^2+\theta^3+\theta^4} = 4d_{iH}$
1:6	$\frac{5\theta}{1+5\theta} = 5d_{iH}$
2:6	$\frac{5\theta(1+4\theta)}{1+4\theta+10\theta^2} = 5d_{iH}$
3:6	$\frac{5\theta(1+3\theta+6\theta^2)}{1+3\theta+6\theta^2+10\theta^3} = 5d_{iH}$
4:6	$\frac{5\theta(1+2\theta+3\theta^2+4\theta^3)}{1+2\theta+3\theta^2+4\theta^3+5\theta^4} = 5d_{iH}$
5:6	$\frac{5\theta(1+\theta+\theta^2+\theta^3+\theta^4)}{1+\theta+\theta^2+\theta^3+\theta^4+\theta^5} = 5d_{iH}$

Table 5. Formulas for the corrected model probability θ for rules with at most 6 products for an expected relative demand of d_{iH}.

Because of the stochastic nature of the algorithm and the simulator ALS, the solution found for each instance changes if the procedure is repeated using a different random seed. For this reason, each instance has been solved 20 times by each of the two versions of the algorithm.

The genetic algorithm has been implemented in Java, and the experiments have been performed on a PC with 8GB RAM and an Intel Core i5-2400 Cup at 3.1 GHz.

5.1. *Benchmark instances*

The two versions of the algorithm, GA-rules and GA-random, have been tested on a series of benchmark instances proposed in the literature. We chose 5 instances (Barthold, Gunther, Roszieg, Scholl, and WeeMag - (Bartholdi 1993; Gunther 1983; Rosenberg and Ziegler 1992; Scholl 1993; Wee and Magazine 1981)) with a number of tasks varying from 25 to 297, whose precedence diagrams, imposed cycle times, and task times t_i for a single model can be downloaded from the data set of Scholl (1993).

We generate the normal, variant, and optional tasks by taking into account two degrees of variability, that we called respectively $v1$ and $v2$.

The first degree of variability $v1$ refers to the proportion of normal, variant, and optional tasks in each instance. A low $v1$ variability indicates that this proportion equals 70% for normal, 15% for variant and 15% for optional tasks in an instance; while for medium $v1$ variability the proportion changes to 50%, 25% and 25% respectively and for high $v1$ variability to 40%, 30% and 30% (see Table 6).

The second degree of variability $v2$ refers to how time consuming is the most work-intensive option of variant and optional tasks. For variant tasks, we took as reference the time required by the less work-intensive option of each task, which is taken equal to the original task time of the instances ($t_{iL} = t_i$). Then, the corresponding average task time for the most work intensive option has been generated through the formula

Table 6. Values of parameter $v1$ for the generation of the benchmark instances.

	% of normal tasks	% of variant tasks	% of optional tasks
$v1-$low	70%	15%	15%
$v1-$medium	50%	25%	25%
$v1-$high	40%	30%	30%

$t_{iH} = t_{iL}(1+f)$, where f is randomly generated in a range that depends on the degree of variability: $f \in [0.1, 0.3]$ for low $v2$, $f \in [0.3, 0.7]$ for medium $v2$ and $f \in [0.7, 1.0]$ for high $v2$. For optional tasks, we took as reference the imposed cycle time of the original instance, ct. Then the task time for each optional task has been generated as $t_{iH} = ct \cdot f$, where f is randomly generated in the same ranges than the preceding case (see Table 7).

Table 7. Values of parameter $v2$ for the generation of the benchmark instances.

	Demand proportion	Task time for variant tasks	Task time for optional tasks	f
$v2-$low				$f \in [0.1, 0.3]$
$v2-$medium	$d_{ij} \in \left[\frac{H-1}{N}, \frac{H}{N}\right]$	$t_{iH} = t_{iL}(1+f)$	$t_{iH} = ct \cdot f$	$f \in [0.3, 0.7]$
$v2-$high				$f \in [0.7, 1.0]$

For the definition of H:N sequencing rules, random values for H and N are drawn for the tasks with variant or optional tasks. N is uniformly randomly selected between 2 and 6, while H is uniformly selected from the set $\{1, ..., N-1\}$. After defining the values for H and N, the relative demand of each option is drawn from the interval $\left[\frac{H-1}{N}, \frac{H}{N}\right]$. Finally, the corrected probabilities when a car sequencing rule is selected are calculated based on the formulas of Table 5. Both the probabilities and corrected probabilities are rounded to three decimal values.

The task times for normal tasks have been taken equal to the original task times of each instance. By combining 5 different instances with three degree of variability $v1$ and three degree of variability $v2$, we generated a total number of $5 \cdot 3 \cdot 3 = 45$ instances.

5.2. *Tuning*

The genetic algorithm procedure provides a number of characteristic parameters (mutation rate (mr), crossover rate (cr), population size (N_P) and number of iterations (G)) that influence both the time required to find the solution, and the quality of the solution. The values for these parameters have been decided through a tuning procedure on the 5 benchmark instances with $v1 = v2 =$ medium.

The time required to find a solution largely depends on the number of decoding procedures D_p performed along all the iterations. This is particularly true in our case where each time an individual is decoded a simulation run must be performed. The expected number of decoding procedures performed by the presented algorithm is equal to:

$$D_p = N_P \left[cr + (1 + cr)mr\right] G \tag{8}$$

(8) is obtained considering that at the beginning of each one of the G iterations, the algorithm starts with a population of N_p decoded individuals. During the reproduction

phase, the number of new individuals generated by the crossover operator that have to be decodified equals $(N_p \cdot cr)$. These individuals are added to the population, whose size becomes $N_p \cdot (1+cr)$. Then each individual of the current population has a probability mr to undergo a mutation, so that the expected number of mutated individuals that have to be decodified equals $N_p \cdot (1+cr) \cdot mr$.

The tuning procedure consisted in finding the best combination of the parameters that brings to a total number of decoding procedures $D_p \approx 10000$, value that guarantees an acceptable solution time (between about 40 and 500 seconds, depending on the instance) using a PC with 32G RAM and an Intel Core i9-12900 Cpu at 2.4 GHz. The possible combinations have been determined in the following way: taking a fixed number of iterations $G = 150$, we explored the combinations of cr and mr in the range [0.1, 0.9] step 0.1. For each combination of cr and mr we calculated through eq. (8) the population size N_P that brings to the desired number of D_p. We found that the combination of the parameters bringing to solutions with the lowest fitness was, for all the 5 instances: $N_P = 40$, $cr = 0.5$, $mr = 0.9$, $G = 150$.

Another parameter, utilized in the decoding phase (see Subsection 4.1), is the probability of completion pc. The tuning procedure showed that best results are achieved assigning a value $pc = 0.5$.

Finally, the penalty coefficient ζ, which is a decision-makers' parameter introduced in Subsection 3.4, is not subject to the tuning procedure. In fact, there is not an 'optimal' value for this parameter, but rather a value satisfying the management in terms of desired deviation from the target cycle time. In our experiment we choose $\zeta = 20$, a value that heavily penalizes solutions with $ct_{eff} > ct$, thus forcing the algorithm to find solutions that substantially achieve the target cycle time.

6. Results

Table 8 displays the outcomes of the numerical experiment described in the previous section. The median, mean, and best value in 20 replications are presented for each of the 45 instances examined for both GA-Random and GA-Rules. The Mann-Whitney test was used to compare the statistics between the two versions. The null hypothesis of the test is that neither of the two versions outperforms the other (i.e. achieves a lower Normalize Design Cost). A p-value of less than or equal to 0.001 indicates that the null hypothesis must be rejected with a probability of 99.9%, and the alternative hypothesis, that one of the two algorithms outperforms the other, must be accepted. In this situation, the algorithm with the lower median value of NDC outperforms the others. The outcome of this comparison is reported in the "winner" column, showing which version of the algorithm is significantly superior. When the p value is greater than 0.001, we consider that the test is not significant (ns), which means that there is no significant difference between the algorithms' performances. We also indicated in the "%Best" column the NDC percentage decrease achieved by the algorithm that performs better than the other, measured on the best solution found by the two algorithms.

Results show that GA-Rules significantly outperforms GA-Random in 40 out of 45 instances, while in 5 instances the difference is not significant. However, even in this case, GA-Rules found a best solution with lower NDC than GA-Random in 4 out of 5 instances.

The higher savings can be observed, as expected, both when the proportion of variant and optional tasks in an instance increases ($v1$ increasing) and when the time

required by the most work-intensive option of variant and optional tasks increases ($v2$ increasing), reaching a maximum saving of about 20% in the instance by Bartholdi with $v1$ = Medium and v2 = High.

GA-Rules seems to performs particularly better for instances with a high number of tasks, achieving savings often higher than 10% in the instances by Bartholdi (148 tasks) and Scholl (297 tasks). However, even in the smaller instance by Roszieg (25 tasks) significant reductions of Normalized Design Cost can be achieved. For example, Table 9 shows the input data related to the instance Roszieg with $v1$ = medium and $v2$ = medium, with 4 variant tasks and 6 optional tasks. Table 10 shows the best solution found by GA-Rules and GA-Random. Although the numbers of workers in the line is the same (7 workers), the solution found by GA-Random provides the duplication of the last WC, so that the total Design Cost is higher due to the equipment duplication cost. On the contrary, the solution found by GA-Rules achieves an effective cycle time that respects the imposed one not by exploiting parallelism, but by selecting the opportune rules for some of the variant and optional tasks. This allows to decrease of more than 7% the Normalized Design Cost.

7. Conclusion

In this paper a novel approach for solving the Mixed model Line Balancing and Sequencing Problem in asynchronous assembly lines was presented. The approach utilizes rules that are practically employed in the car sequencing problem, specifically rules that relate to the frequency at which certain optional can be installed in each product. The advantage of incorporating these rules lies in the reduction of the number of variables of the problem. Instead of searching for a specific sequence among all the different models generated by the combination of various optional, the decision variable is simplified to the adoption or non-adoption of a rule.

The impact of incorporating these rules on the line's performance was evaluated through a parametric discrete-event simulator. The simulator is called by a Genetic Algorithm, that explores the solution space of the balancing and rule selection problems, each time the fitness of an individual has to be evaluated. The results of the proposed numerical experiment demonstrate that the adoption of these rules enables the design of cost-effective and highly efficient assembly lines.

Nevertheless the assumptions necessary for the modeling of the problem bring simplifications that are unrealistic in a real world scenario. In this modeling approach, the processing times of all tasks are considered to be independent to each other. Furthermore, the selection of the options are considered without any dependency on other tasks. This assumption is not realistic since high-end products are very likely to have options purchased together. The computational efficiency of the method, however, relies on such assumptions and extending them require further research. One example of a study considering correlation between options is shown in Sikora (2024).

Possible extensions to this work could include:

- Considering a maximum number of rules to be adopted, such as allowing only one rule per station. This would add an additional constraint to the problem and explore the impact of limited rule adoption on the overall line performance.
- Allowing the choice of multiple rules for each task. Currently, the approach assumes the adoption or non-adoption of a single rule for each task. By expanding the decision space to include multiple rule options, it would be possible to inves-

tigate the trade-offs and performance implications of different rule combinations.
- Relaxing assumptions that task processing times and selection of options' probabilities are independent.

By exploring these extensions, researchers can further enhance the understanding of the mixed model line balancing and sequencing problem, and potentially discover additional strategies to optimize assembly line efficiency and cost-effectiveness.

Data availability statement

Data that allow replicating the experiments and the findings of this study are openly available in ResearchGate at: Instances and results: 10.13140/RG.2.2.11702.88649

Disclosure statement

The author confirms that there are no relevant financial or non-financial competing interests to report.

References

Bartholdi, J J. 1993. "Balancing two-sided assembly lines: a case study." *International Journal of Production Research* 31 (10): 2447–2461. https://doi.org/10.1080/00207549308956868.

Battaïa, Olga, and Alexandre Dolgui. 2022. "Hybridizations in line balancing problems: A comprehensive review on new trends and formulations." *International Journal of Production Economics* 250: 108673. https://doi.org/https://doi.org/10.1016/j.ijpe.2022.108673.

Boysen, N, M Fliedner, and A Scholl. 2009a. "Sequencing mixed-model assembly lines: Survey, classification and model critique." *European Journal of Operational Research* 192 (2): 349–373. https://doi.org/10.1016/j.ejor.2007.09.013.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2007. "A classification of assembly line balancing problems." *European Journal of Operational Research* 183 (2): 674–693. https://doi.org/10.1016/j.ejor.2006.10.010.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2008. "Assembly line balancing: Which model to use when?" *International Journal of Production Economics* 111 (2): 509–528. https://doi.org/10.1016/j.ijpe.2007.02.026.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2009b. "Assembly line balancing: Joint precedence graphs under high product variety." *IIE Transactions* 41 (3): 183–193. https://doi.org/10.1080/07408170801965082.

Boysen, Nils, Philipp Schulze, and Armin Scholl. 2022. "Assembly line balancing: What happened in the last fifteen years?" *European Journal of Operational Research* 301 (3): 797–814. https://doi.org/10.1016/j.ejor.2021.11.043.

Bukchin, Joseph, Ezey M Dar-El, and Jacob Rubinovitz. 2002. "Mixed model assembly line design in a make-to-order environment." *Computers and Industrial Engineering* 41 (4): 405–421. https://doi.org/10.1016/S0360-8352(01)00065-1.

Dincbas, Mehmet, Helmut Simonis, and Pascal Van Hentenryck. 1988. "Solving the car-sequencing problem in constraint logic programming." In *Proceedings of the 8th European Conference on Artificial Intelligence*, ECAI'88, USA, 290–295. Pitman Publishing, Inc.

Dong, Jietao, Linxuan Zhang, Tianyuan Xiao, and Huachao Mao. 2014. "Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time." *International Journal of Production Research* 52 (24): 7529–7548. https://doi.org/10.1080/00207543.2014.944280.

Emde, Simon, and Michel Gendreau. 2017. "Scheduling in-house transport vehicles to feed parts to automotive assembly lines." *European Journal of Operational Research* 260 (1): 255–267. https://doi.org/10.1016/j.ejor.2016.12.012.

Fliedner, Malte, and Nils Boysen. 2008. "Solving the car sequencing problem via Branch & Bound." *European Journal of Operational Research* 191 (3): 1023–1042. https://doi.org/https://doi.org/10.1016/j.ejor.2007.04.045.

Gagné, Caroline, Marc Gravel, and Wilson L. Price. 2006. "Solving real car sequencing problems with ant colony optimization." *European Journal of Operational Research* 174 (3): 1427–1448. https://doi.org/10.1016/J.EJOR.2005.02.063.

Golle, Uli, Nils Boysen, and Franz Rothlauf. 2010. "Analysis and design of sequencing rules for car sequencing." *European Journal of Operational Research* 206 (3): 579–585. https://doi.org/10.1016/j.ejor.2010.03.019.

Golle, Uli, Franz Rothlauf, and Nils Boysen. 2014. "Car sequencing versus mixed-model sequencing: A computational study." *European Journal of Operational Research* 237 (1): 50–61. https://doi.org/https://doi.org/10.1016/j.ejor.2014.01.012.

Golle, Uli, Franz Rothlauf, and Nils Boysen. 2015. "Iterative beam search for car sequencing." *Annals of Operations Research* 226 (1): 239–254. https://doi.org/10.1007/s10479-014-1733-0.

Gunther, R. 1983. "Currently practiced formulations for the assembly line balance problem." *Journal of Operations Management* 3 (4): 209–221. https://doi.org/10.1016/0272-6963(83)90005-0.

Karabati, Selçuk, and Serpil Sayın. 2003. "Assembly line balancing in a mixed-model sequencing environment with synchronous transfers." *European Journal of Operational Research* 149 (2): 417–429. https://doi.org/10.1016/S0377-2217(02)00764-6.

Kim Y. K., Y J Kim, and Y Kim. 1996. "Genetic algorithms for assembly line balancing with various objectives." *Computers and Industrial Engineering* 30: 397–409.

Lopes, Thiago Cantos, Adalberto Sato Michels, Ricardo Lüders, and Leandro Magatão. 2020a. "A simheuristic approach for throughput maximization of asynchronous buffered stochastic mixed-model assembly lines." *Computers and Operations Research* 115: 104863. https://doi.org/10.1016/j.cor.2019.104863.

Lopes, Thiago Cantos, Adalberto Sato Michels, Celso Gustavo Stall Sikora, and Leandro Magatão. 2019. "Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations." *Journal of Manufacturing Systems* 50: 193–200. https://doi.org/10.1016/j.jmsy.2019.01.001.

Lopes, Thiago Cantos, Adalberto Sato Michels, Celso Gustavo Stall Sikora, Rafael Gobbi Molina, and Leandro Magatão. 2018a. "Balancing and cyclically sequencing synchronous, asynchronous, and hybrid unpaced assembly lines." *International Journal of Production Economics* 203: 216–224. https://doi.org/10.1016/j.ijpe.2018.06.012.

Lopes, Thiago Cantos, Celso Gustavo Stall Sikora, Adalberto Sato Michels, Arinei Carlos Lindbeck da Silva, Leandro Magatão, Arinei Carlos Lindbeck da Silva, and Leandro Magatão. 2018b. "Modeling the Stochastic Steady-State of Mixed-Model Asynchronous Assembly Lines with Markov Chains." In *Proceedings of the XIX Latin-Iberoamerican Conference on Operations Research (CLAIO)*, Lima, 183–190.

Lopes, Thiago Cantos, Celso Gustavo Stall Sikora, Adalberto Sato Michels, and Leandro Magatão. 2020b. "Mixed-model assembly lines balancing with given buffers and product sequence: model, formulation comparisons, and case study." *Annals of Operations Research* 286 (1-2): 475–500. https://doi.org/10.1007/s10479-017-2711-0.

Lübben, Malte, Sven Pries, and Celso Gustavo Stall Sikora. 2023. "Online resequencing of buffers for automotive assembly lines." *Computers and Industrial Engineering* 175: 108857. https://doi.org/10.1016/j.cie.2022.108857.

Manavizadeh, Neda, Masoud Rabbani, Davoud Moshtaghi, and Fariborz Jolai. 2012. "Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms." *Expert Systems with Applications* 39 (15): 12026–12031. https://doi.org/10.1016/j.eswa.2012.03.044.

Miltenburg, John, and J. Wijngaard. 1994. "U-line line balancing problem." *Management Science* 40 (10): 1378–1388.

Özcan, Uğur, Talip Kellegöz, and Bilal Toklu. 2011. "A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem." *International Journal of Production Research* 49 (6): 1605–1626. https://doi.org/10.1080/00207541003690090.

Öztürk, Cemalettin, Semra Tunali, Brahim Hnich, and Arslan Örnek. 2012. "Balancing and scheduling of flexible mixed model assembly lines with parallel stations." *International Journal of Advanced Manufacturing Technology* 67 (9-12): 255–2591. https://doi.org/10.1007/s00170-012-4675-1.

Öztürk, Cemalettin, Semra Tunali, Brahim Hnich, and Arslan Örnek. 2015. "Cyclic scheduling of flexible mixed model assembly lines with parallel stations." *Journal of Manufacturing Systems* 36 (3): 147–158. https://doi.org/10.1007/s00170-012-4675-1.

Parrello, Bruce D., Waldo C. Kabat, and L. Wos. 1986. "Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem." *Journal of Automated Reasoning* 2 (1): 1–42. https://doi.org/10.1007/BF00246021.

Rosenberg, O., and H. Ziegler. 1992. "A comparison of heuristic algorithms for cost-oriented assembly line balancing." *ZOR Zeitschrift für Operations Research Methods and Models of Operations Research* 36 (6): 477–495. https://doi.org/10.1007/BF01416240.

Scholl, Armin. 1993. *Data of Assembly Line Balancing Problems*. Technical Report. Darmstadt: Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).

Sikora, Celso Gustavo Stall. 2021. "Benders' decomposition for the balancing of assembly lines with stochastic demand." *European Journal of Operational Research* 292 (1): 108–124. https://doi.org/10.1016/j.ejor.2020.10.019.

Sikora, Celso Gustavo Stall. 2022a. *Assembly-Line Balancing under Demand Uncertainty*. Gabler Theses. Wiesbaden: Springer Fachmedien Wiesbaden.

Sikora, Celso Gustavo Stall. 2022b. "Balancing Under No Sequencing Control." In *Assembly-Line Balancing under Demand Uncertainty*, Gabler the ed., 97–131. Wiesbaden: Springer Gabler.

Sikora, Celso Gustavo Stall. 2024. "Balancing mixed-model assembly lines for random sequences." *European Journal of Operational Research* 314 (2): 597–611. https://doi.org/10.1016/j.ejor.2023.10.008.

Sikora, Celso Gustavo Stall, Thiago Cantos Lopes, and Leandro Magatão. 2017. "Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies." *European Journal of Operational Research* 259 (3): 949–971. https://doi.org/10.1016/j.ejor.2016.11.027.

Sikora, Celso Gustavo Stall, Thiago Cantos Lopes, Daniel Schibelbain, and Leandro Magatão. 2017. "Integer based formulation for the simple assembly line balancing problem with multiple identical tasks." *Computers and Industrial Engineering* 104: 134–144. https://doi.org/10.1016/j.cie.2016.12.026.

Solnon, Christine, Van Dat Cung, Alain Nguyen, and Christian Artigues. 2008. "The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem." *European Journal of Operational Research* 191 (3): 912–927. https://doi.org/10.1016/j.ejor.2007.04.033.

Sun, Hui, and Shujin Fan. 2018. "Car sequencing for mixed-model assembly lines with consideration of changeover complexity." *Journal of Manufacturing Systems* 46: 93–102. https://doi.org/https://doi.org/10.1016/j.jmsy.2017.11.009.

Sun, Yuan, Samuel Esler, Dhananjay Thiruvady, Andreas T. Ernst, Xiaodong Li, and Kerri Morgan. 2022. "Instance space analysis for the car sequencing problem." *Annals of Operations Research* https://doi.org/10.1007/s10479-022-04860-8.

Thiruvady, Dhananjay, Kerri Morgan, Amiza Amir, and Andreas T Ernst. 2020. "Large neighbourhood search based on mixed integer programming and ant colony optimisation for car sequencing." *International Journal of Production Research* 58 (9): 2696–2711. https://doi.org/10.1080/00207543.2019.1630765.

Tiacci, Lorenzo. 2012. "Event and object oriented simulation to fast evaluate operational objectives of mixed model assembly lines problems." *Simulation Modelling Practice and Theory* 24: 35–48. https://doi.org/10.1016/j.simpat.2012.01.004.

Tiacci, Lorenzo. 2015a. "Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times." *International Journal of Production Economics* 159: 319–333. https://doi.org/10.1016/j.ijpe.2014.05.005.

Tiacci, Lorenzo. 2015b. "Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times." *International Journal of Production Economics* 162: 201–215. https://doi.org/10.1016/J.IJPE.2015.01.022.

Tiacci, Lorenzo. 2020. "Object-oriented event-graph modeling formalism to simulate manufacturing systems in the Industry 4.0 era." *Simulation Modelling Practice and Theory* 99: 102027. https://doi.org/https://doi.org/10.1016/j.simpat.2019.102027.

Tiacci, Lorenzo. 2022. "Buffer allocation vs. sequencing optimization: which of the two is most effective to improve the efficiency of assembly lines?" *IFAC-PapersOnLine* 55 (10): 452–457. https://doi.org/https://doi.org/10.1016/j.ifacol.2022.09.435.

Tiacci, Lorenzo, and Mario Mimmi. 2018. "Integrating ergonomic risks evaluation through OCRA index and balancing/sequencing decisions for mixed model stochastic asynchronous assembly lines." *Omega* 78: 112–138. https://doi.org/10.1016/j.omega.2017.08.011.

Tiacci, Lorenzo, Stefano Saetta, and Andrea Martini. 2006. "Balancing mixed-model assembly lines with parallel workstations through a genetic algorithm approach." *International Journal of Industrial Engineering: Theory Applications and Practice* 13 (4): 402–411.

Wee, T.S., and A. Magazine. 1981. "An efficient branch and bound algorithm for an assembly line balancing problem - part II: maximize the production rate." .

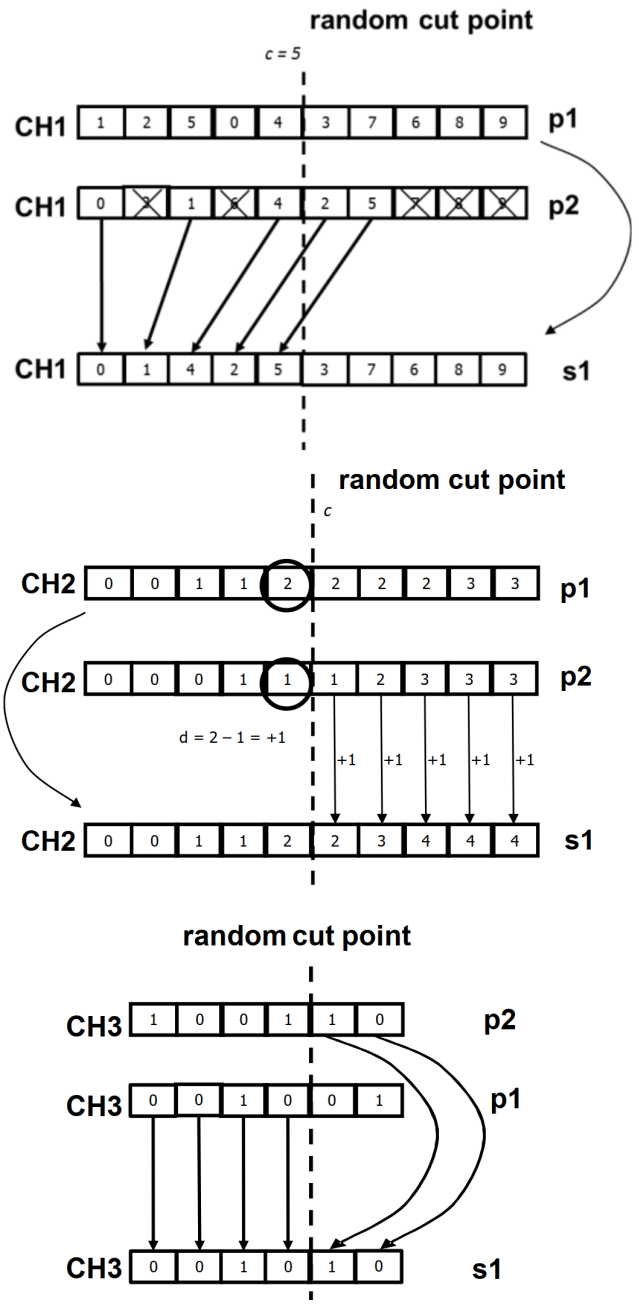

Figure 4. Description of the crossover operator for the three chromosomes. Alt text: Diagram exemplifying the crossover operator for the the three gene. For the the three genes, two chromosomes from parent 1 and 2 are shown at the top. An arbitrary cut point is selected. The resulting child chromosome is shown at the bottom for each gene.

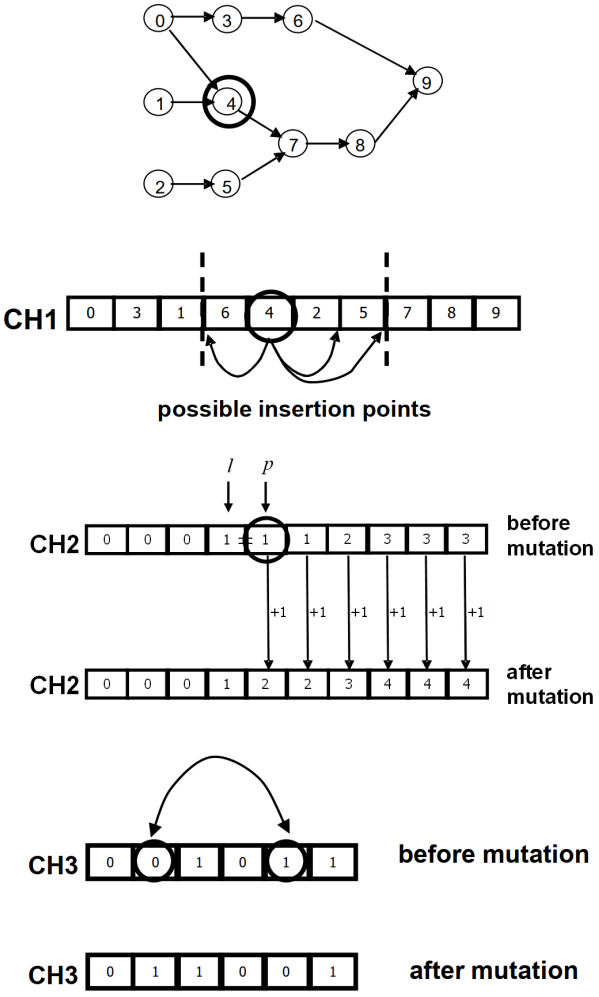

Figure 5. Mutation operator for the three chromosomes. Alt text: For the first chromosome, The figure show the possible options for a mutation in task 4 in a chromosome using arrows. The top of the figure also contains the precedence relations to justify the possible insertion points. The figure also contains the second and third chromosome before and after the mutation.

Table 8. GA-Rule vs GA-Random: results of the numerical experiments.

			GA-Random			GA-Rules					
Problem	v1	v2	Median	Mean	Best	Median	Mean	Best	p	%Best	Winner
		Low	315,001	317,857	315,000	315,000	315,037	315,000	0.015	0.00%	ns
	Low	Medium	331,256	331,687	319,297	317,807	321,436	315,000	0.000	-1.35%	GA-Rules
		High	362,943	363,411	354,550	360,004	359,991	345,800	0.003	-2.47%	ns
Roszieg		Low	315,175	316,555	315,000	315,000	313,119	294,955	0.000	-6.36%	GA-Rules
$\|T\| = 25$	Medium	Medium	345,000	344,934	340,265	315,000	315,289	315,000	0.000	-7.43%	GA-Rules
$ct = 18$		High	444,871	445,322	438,173	421,267	421,172	413,941	0.000	-5.53%	GA-Rules
		Low	315,000	315,290	315,000	285,522	286,290	285,000	0.000	-9.52%	GA-Rules
	High	Medium	360,000	360,253	357,000	315,275	316,623	315,000	0.000	-11.76%	GA-Rules
		High	452,273	452,958	442,884	417,081	417,015	406,054	0.000	-8.32%	GA-Rules
		Low	435,000	431,052	415,323	418,879	422,478	407,779	0.007	-1.82%	ns
	Low	Medium	503,494	507,026	497,086	496,567	500,010	493,587	0.001	-0.70%	ns
		High	506,778	507,516	495,141	485,151	487,283	479,069	0.000	-3.25%	GA-Rules
Gunther		Low	408,130	410,988	405,497	397,253	392,359	375,256	0.000	-7.46%	GA-Rules
$\|T\| = 35$	Medium	Medium	513,499	513,678	498,922	496,095	496,746	477,483	0.000	-4.30%	GA-Rules
$ct = 54$		High	541,809	541,975	532,215	493,500	494,844	482,211	0.000	-9.40%	GA-Rules
		Low	420,537	421,668	408,240	405,000	401,255	382,265	0.000	-6.36%	GA-Rules
	High	Medium	564,127	566,301	555,054	525,040	523,882	507,143	0.000	-8.63%	GA-Rules
		High	750,473	751,928	738,461	695,648	696,373	681,374	0.000	-7.73%	GA-Rules
		Low	1,898,728	1,901,654	1,868,590	1,873,069	1,882,020	1,853,908	0.004	-0.79%	ns
	Low	Medium	1,909,610	1,911,200	1,887,868	1,852,426	1,857,606	1,825,071	0.000	-3.33%	GA-Rules
		High	2,007,791	2,010,335	1,988,809	1,958,851	1,962,528	1,939,135	0.000	-2.50%	GA-Rules
WeeMag		Low	1,868,901	1,868,175	1,837,705	1,804,040	1,802,240	1,772,606	0.000	-3.54%	GA-Rules
$\|T\| = 75$	Medium	Medium	1,939,139	1,937,014	1,902,553	1,852,853	1,853,690	1,826,173	0.000	-4.01%	GA-Rules
$ct = 34$		High	2,102,922	2,106,435	2,084,221	2,013,652	2,017,588	1,999,381	0.000	-4.07%	GA-Rules
		Low	1,860,853	1,860,526	1,826,608	1,744,875	1,745,265	1,709,777	0.000	-6.40%	GA-Rules
	High	Medium	1,959,900	1,961,931	1,943,503	1,806,496	1,806,072	1,783,728	0.000	-8.22%	GA-Rules
		High	2,143,851	2,148,934	2,115,051	1,962,203	1,958,853	1,920,341	0.000	-9.21%	GA-Rules
		Low	954,324	953,391	924,372	954,000	949,647	897,125	0.135	-2.95%	ns
	Low	Medium	1,217,848	1,218,191	1,184,509	1,115,941	1,116,895	1,082,724	0.000	-8.59%	GA-Rules
		High	1,441,225	1,445,570	1,411,006	1,315,827	1,317,915	1,284,000	0.000	-9.00%	GA-Rules
Barthold		Low	1,044,347	1,048,048	1,024,798	1,002,656	997,585	954,001	0.000	-6.91%	GA-Rules
$\|T\| = 148$	Medium	Medium	1,379,059	1,377,254	1,348,072	1,221,809	1,223,213	1,194,000	0.000	-11.43%	GA-Rules
$ct = 470$		High	1,926,119	1,938,174	1,894,360	1,582,940	1,583,822	1,503,629	0.000	-20.63%	GA-Rules
		Low	1,109,997	1,108,588	1,080,990	985,755	989,862	954,000	0.000	-11.75%	GA-Rules
	High	Medium	1,726,835	1,727,263	1,689,819	1,483,293	1,476,133	1,435,616	0.000	-15.04%	GA-Rules
		High	2,373,922	2,377,306	2,325,248	1,959,452	1,959,274	1,872,354	0.000	-19.48%	GA-Rules
		Low	2,773,135	2,772,251	2,707,820	2,713,355	2,721,114	2,663,327	0.000	-1.64%	GA-Rules
	Low	Medium	3,311,501	3,315,614	3,231,285	3,144,066	3,141,175	3,027,219	0.000	-6.32%	GA-Rules
		High	3,928,392	3,931,985	3,842,414	3,631,212	3,617,062	3,547,149	0.000	-7.68%	GA-Rules
Scholl		Low	2,868,667	2,867,414	2,771,199	2,619,824	2,612,028	2,545,302	0.000	-8.15%	GA-Rules
$\|T\| = 297$	Medium	Medium	3,812,731	3,814,095	3,736,941	3,435,107	3,429,871	3,354,096	0.000	-10.24%	GA-Rules
$ct = 1515$		High	5,044,764	5,044,715	4,944,492	4,492,812	4,491,356	4,361,265	0.000	-11.80%	GA-Rules
		Low	2,988,299	2,973,994	2,873,710	2,710,117	2,703,008	2,630,258	0.000	-8.47%	GA-Rules
	High	Medium	4,268,655	4,281,380	4,196,104	3,687,305	3,697,723	3,543,944	0.000	-15.54%	GA-Rules
		High	5,794,569	5,797,360	5,662,646	4,894,202	4,909,700	4,675,356	0.000	-17.44%	GA-Rules

Table 9. Details of the instance Roszieg 25, $v1$ = medium, $v2$ = medium, ct = 18.

task#	type	t_{iL}	t_{iH}	r_i	d_{iL}	d_{iH}
0	Normal	4			1	0
1	Normal	3			1	0
2	Variant	9	15	1:3	0.985	0.015
3	Variant	5	8	1:2	0.679	0.321
4	Normal	9			1	0
5	Optional	0	10	3:5	0.554	0.446
6	Normal	8			1	0
7	Normal	7			1	0
8	Optional	0	11	1:6	0.953	0.047
9	Normal	1			1	0
10	Optional	0	11	1:2	0.966	0.034
11	Normal	1			1	0
12	Normal	5			1	0
13	Normal	3			1	0
14	Normal	5	8	1:2	0.591	0.409
15	Normal	3			1	0
16	Normal	13			1	0
17	Normal	5			1	0
18	Optional	0	12	1:6	0.863	0.137
19	Variant	3	4	3:5	0.555	0.445
20	Normal	7			1	0
21	Normal	5			1	0
22	Variant	3	4	3:5	0.538	0.462
23	Optional	0	8	1:3	0.808	0.192
24	Optional	0	6	2:6	0.792	0.208

Table 10. Best solutions from GA-Rules and GA-Random for Roszieg 25, $v1$ = medium, $v2$ = medium, ct = 18.

GA-Rules

WC#	WS assigned (W_i)	tasks assigned
0	1	0, 1, 2
1	1	3, 4
2	1	5, 7
3	1	8, 6, 9, 10, 12
4	1	11, 14, 13, 18
5	1	16
6	1	15, 22, 17, 19
7	1	24, 20, 23, 21
ct_{eff} = 17.99; DC = 315000; NDC = 315000		
Rules:	Task8 NO	Task19 NO
Task2 1:3	Task10 1:2	Task22 3:5
Task3 1:2	Task14 1:2	Task23 NO
Task5 NO	Task18 1:6	Task24 2:6

GA-Random

WC#	WS assigned (W_i)	tasks assigned
0	1	0, 1, 2
1	1	3, 4
2	1	5, 7, 8
3	1	6, 11
4	1	10, 12, 13, 9, 19, 15
5	1	14, 20
6	2	18, 16, 21, 17, 22, 24, 23
ct_{eff} = 18.45; DC = 336000; NDC = 340265		