Balancing mixed-model assembly lines for random sequences

Celso Gustavo Stall Sikora*

Moorweidenstr. 18, 20148, Hamburg, Institute for Operations Research, University of Hamburg

Abstract

Assembly lines are the standard choice for the manufacturing of large products and must combine efficiency with flexibility. Since multiple product variants are assembled on the same line, the variation in the processing time and the production sequence must be accounted for. This is especially hard in the planning phase of assembly lines when the demand is only forecast and the production sequence is not yet known. In this paper, an exact method and heuristic extensions are proposed to tackle the paced assembly-line balancing problem considering a random production sequence. Along with the assignment of tasks, the length of the stations is optimized in a Branch-and-Bound algorithm using Markov chains to evaluate the expected costs, which require the solution of a convex subproblem on each node. Product variants are modeled with a given probability, which can be either independent or correlated. Results of the model can be then further improved by optimizing the sequencing once the orders are known. The exact method can solve instances of small and medium-size containing up to 10^{12} product variants and the heuristic extensions can give good solutions with up to 10^{20} products. Results show that the optimization of station lengths is more important for the total cost than the difference between optimal and good task assignment solutions.

Keywords: flexible manufacturing systems, Mixed-model assembly-line balancing, Markov chains, Random sequence, Decomposition, mixed-model assembly line

This is the peer-reviewed, accepted version of the paper. European Journal of Operational Research, 2024, 314 (2), 597–611. DOI 10.1016/j.ejor.2023.10.008

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

1. Introduction

An assembly line is a production arrangement highly used for the manufacturing of large products, especially in the automotive and home appliances industries. Due to the large investment required to build such a system, its optimization is of great practical interest (Battaïa & Dolgui, 2013). Such large investments can be associated with either semi-automated systems or equipment to hold tools and/or workpieces in place for operations.

The efficiency of assembly lines is strongly related to how well the workload is divided (or balanced) between the multiple stations. The optimization problem derived from the task-station assignment is a classic in the field of operations research and is named assembly-line balancing problem (ALBP) (Boysen et al., 2007).

Although the assembly line (as well as the ALBP) was initially conceptualized for the production of a single product in the automotive industry, market pressure requires the offer of a large spectrum of sizes, capabilities, and customizations (Boysen et al., 2008). The assembly of multiple product variants in a single system must deal with the dichotomy of being efficient and flexible. Such requirements result in the so-called mass customization and there are reports from automotive manufacturers with combinations of up to 10^{32} possible variants (Boysen et al., 2008).

The production under a high number of varieties must be organized in a way such that not only average processing times stay within bounds, but also the dynamic effect of the production sequence is accounted for. Measures to compensate for deviations due to product varieties depend on the employed system. Paced lines are built with a conveyor belt moving at a slow but constant speed (Scholl & Klein, 1997). The workers perform their tasks while the product is accessible within their station. In such paced lines, reactions to processing-time variations are the stoppage of the line, employment of utility workers, or finishing overload at the end of the line (Sikora, 2022). Another assembly-line configuration is based on an unpaced transport of products. In such lines, the stations are fixed and the products only are moved between stations when the tasks are finished (Lopes et al., 2020b).

Accounting for the production sequencing explicitly can be computationally expensive since the resulting optimization decisions consist of the combined assembly line and production sequencing problem. In works such as Lopes et al. (2018), the number of products which can be sequenced is limited to at most a couple of dozens. Therefore, it is not uncommon that the effect of multiple models is tackled indirectly. Merengo et al. (1999), for instance, propose the use of methods named horizontal and vertical balancing. These methods aim at the minimization of processing-time differences within and between stations, respectively.

Besides the computational aspects of integrating assembly-line balancing and sequencing problems, practical considerations may hinder such an endeavour. While the assembly line must be planned and built in advance, the production sequence is strongly linked to the customers' orders. Hence, both decisions (balancing and sequencing) have different time frames (Boysen et al., 2008) and the exact demand is

usually not known in the planning phase (when the ALB is solved before the operational phase of the line). The demand or production sequence uncertainty is then another factor to make the balancing of real assembly lines difficult.

1.1. Contributions

The contribution of this paper is an exact method for the balancing of a mixed-model assembly line and the optimization of station lengths when the production sequence is not yet known in the planning stage. Instead of considering an average product, the sequence of products is modelled explicitly in the form of a random sequence of products. The approach aims for a solution that performs the best when no control over the production sequence is assumed. This is an extreme case which is mostly not applied in industry but serves as a proxy solution when the orders are yet unknown. When orders are later known before the operation phase, the optimization of the sequencing can further reduce the operative cost of the line.

The modelled system consists of a paced line operating with utility workers as a reactive measure. This decision allows the decomposition of the problem with respect to stations, allowing the computation of the random sequences locally using Markov chains. The products are conceptualized as a combination of options, that is, alternatives which can be ordered by the customer. With such an option-based approach within a decomposition procedure, instances with 10^{12} product variants are solved exactly and even lines with over 10^{20} variants can be computed heuristically with proposed heuristic variants of the solution method. It is to notice that the formulation using Markov chains require a specific variant for the next model does not depend on the previous variants from the sequence. This memoryless process is an assumption of the presented method and is inspired of a production based on random customers' orders that are independent from each other.

Another advantage of the proposed method is the modelling of the station length as a variable. This way, the space of the station can be used as a planned reaction measure along with the utility workers. In the results of the heuristic variants of the solution method, it is clear that selecting the optimal station length is even more important for a good solution than selecting the optimal balancing solution with a non-optimal station length. Furthermore, the method allows the treatment of correlated data. Effects such as a positive (or negative) correlation between pairs of options can be modelled without greatly affecting the solution time of the algorithm.

A final contribution is due to the framework using Markov chains to evaluate convex subproblems and the use of dynamic bounds on the enumeration.

This work is an extension of a chapter of a PhD thesis (Sikora, 2022). The further contributions not included in the thesis are the development of the heuristic methods, the adaptation for dealing with non-independent probabilities, and further tests considering methods of the literature are considered.

1.2. Structure of the paper

The remainder of the paper is organized as follows. In Section 2, the literature on the mixed-model assembly-line balancing problem under uncertainties is discussed. In particular, the contributions dealing with uncertainty in the demand or production sequence are detailed. The formulation and assumptions of the problem dealt with here are described in Section 3. The exact calculation of the operative costs caused by a random sequence is modelled by a Markov chain, which is described in Section 4. The formulation of product variant probabilities and the constrains on correlated probabilities are described in Section 5. The proposed exact Branch-and-Bound algorithm is then introduced in Section 6 along with heuristic adaptations of the solution procedure. The experiments to evaluate the algorithms are described in Section 7 and the paper is concluded in Section 8.

2. Literature

The intersection of the assembly-line balancing literature and uncertainty has existed since not much later than the deterministic problem is presented. Moodie & Young (1965) introduced stochastic processing times, which are still today the main inspiration for uncertainty modelling in assembly-line balancing (Sikora, 2022).

Although there are multiple sources of uncertainty in assembly lines, most surveys such as the one from Battaïa & Dolgui (2013) simply divide the literature between deterministic and stochastic models. Boysen et al. (2007), in their assembly-line balancing problem classification, mention two categories which are affected by uncertainty: stochastic processing times (with notation t^{sto}) and the observance of the cycle-time restriction with a given probability (*prob*, in their notation). The latter modelling approach includes the modelling of the cycle time restriction as a chance constraint with respect to stochastic processing times or when multiple products are assembled.

A further classification of uncertainties is not discussed even in the more recent systematic review of Eghtesadifard et al. (2020) nor in the survey by Boysen et al. (2022), although both papers point out the increasing importance of uncertainty in the literature. Even in the survey on uncertainties for the balancing of assembly and disassembly lines presented by Bentaha & Battaïa (2015), only uncertain processing times are discussed. Such uncertain times are divided between stochastic processing times (usually modelled with a normal distribution) (Urban & Chiang, 2006), fuzzy processing times (Zacharia & Nearchou, 2013), and interval processing times (Pereira & Álvarez-Miranda, 2018). These interval processing times $t^{int} = [a, b]$ refer to robust optimization approaches, in which solutions should remain feasible even if the processing time of a number Γ of tasks in a station assumes the upper bound b.

Although uncertain processing times are often described in the literature, there are arguments undermining their practical importance in the automotive industry. Sternatz (2014) states that the processing times are generally obtained by standardized MTM (Methods-Time Measurement) time blocks, which

are based on the performance of the working population. In the German industry, processing times are accorded with worker unions (Boysen et al., 2008), so they can often be regarded as deterministic.

Even though the processing times of the individual operations are often deterministic, there are still reasonable arguments for modelling the station loads as uncertain. Modern assembly lines are not used to produce a single product but a family of different product models, which can account for 10^{32} variants (Boysen et al., 2008). To simplify such mixed-model problems, some authors aggregate all their information into a single model problem, as introduced by Vrat & Virani (1976). Although the simplification allows the easy modelling of multiple product variants, their aggregation does not allow considering the production sequence, which can greatly influence the productivity of the assembly lines (Lopes et al., 2020b). Several approaches in the literature consider the combined problem of balancing the assembly line and sequencing the production order. The methods differ based on the implementation type of the line, which can be paced, unpaced synchronous (Karabati & Sayın, 2003), unpaced asynchronous (Öztürk et al., 2012, 2015; Lopes et al., 2019), or even hybrid (Lopes et al., 2018). The exact solution of the combination of problems is, however, very limited in terms of the size of the instances. All cited references consider the production of at most 7 different products.

In a recent literature review, Sikora (2022) surveyed 85 papers up to 2020 dealing with uncertainties in assembly-line balancing. Of the 85 references, 66 of them deal with a single product while only 19 references consider the more realistic mixed-model assembly-line problem. These last contributions, along with recent publications, are discussed here in more detail.

In Sikora (2022), the literature on mixed-model assembly-line balancing with uncertainties is divided into three categories: uncertainty in processing time, uncertainty of demand, and uncertainty in production sequence. From the first group, some authors reduce the problem by considering the average product (McMullen & Tarasewich, 2003) or by treating each product independently (Hop, 2006). Hop (2006), for instance, models the processing times as fuzzy numbers, and the cycle time is computed independently for each different product. To consider the interrelations between the different products, Özcan et al. (2011), Dong et al. (2014), and Tiacci & Mimmi (2018) consider and solve a sequencing problem along with the balancing. The three papers differ in their solution approach and assembly-line structure, but they all tackle the combined balancing and sequencing of assembly lines with stochastic processing times.

A second and very recent stream of research focuses on designing assembly lines for uncertain demand. This uncertainty is generally tackled by considering a set of possible scenarios. Some decisions must be made before the demand is realized, while reactions such as rebalancing (reassignment of tasks) or the production sequencing can be adjusted after the demand is known. In Li & Gao (2014), Chica et al. (2016), and Chica et al. (2019), the multiple products are reduced to an average model in each demand scenario. The tasks of this average model must then be assigned between the multiple workstations. For Li & Gao (2014), overwork can be used in individual scenarios to compensate for deviancy. Chica et al. (2016, 2019) use a robust approach with multiple objective functions which, between other objectives, minimizes the

required production time that surpasses the cycle time. Another trend in the literature is to consider the reassignment of tasks as a reaction to demand variation. Yang & Gao (2016) assign tasks to regions and not stations individually. In each region, workers are trained to perform all tasks, so that they can be easily reassigned based on the demand. Fisel et al. (2019) also consider the possibility of rebalancing in a multi-objective approach. One innovation is to consider line flexibility (ease of rebalancing) as an optimization objective. A third contribution is due to Lopes et al. (2021), who solve the single-product assembly-line balancing problem considering two or more target cycle times. The tasks are assigned to a position along the line while the stations' limits are defined after knowing the demand level. This way, the line is optimized for both scenarios without requiring the need of moving and reinstalling equipment. Liu et al. (2021) propose a fuzzy formulation for uncertain demand. The fuzzy demand is then tackled using a multi-objective formulation. Finally, Sikora (2021) uses the production sequencing as the reaction for uncertain demand in a combined two-stage stochastic assembly-line balancing and sequencing problem.

Another very recent contribution on ALB with uncertain production sequence is due to Tremblet et al. (2023), who consider a multi-product assembly line. In such lines, a single product is performed at each time, while a set-up between different products adapts the line for the next batch of orders. Tremblet et al. (2023) consider an unknown sequence of batches and minimizes the reassignment of tasks within station (rebalancing) that is used as set-up.

The final category of the classification given in Sikora (2022) considers the uncertainty of the production sequence. In the papers by Bukchin et al. (2002), Manavizadeh et al. (2012), Tiacci (2015a), Tiacci (2015b), and Lopes et al. (2020a), the objective is to balance an assembly line under a random sequence of products. While Bukchin et al. (2002) and Manavizadeh et al. (2012) rely on an approximation function for the line performance, Tiacci (2015a), Tiacci (2015b), and Lopes et al. (2020a) propose integrated optimization and simulation algorithms. These works all present heuristic approaches for unpaced assembly lines.

From the literature review, it can be observed that a large part of the literature focuses on uncertain processing times while only a few model other uncertainty factors. The production sequence of modern assembly lines is hardly known in the planning phase and may consider a huge number of products (Sikora, 2021). Therefore, this paper presents an exact method to solve the assembly-line balancing problem under random sequences. The novelty in comparison to the literature is threefold: first, no other method can solve the problem exactly; second, this is the first approach to optimize the balancing of paced lines under random sequences; third, the station lengths are explicitly accounted for and optimized.

3. Problem definition

The problem definition is based on distributing tasks $i \in I$ over the available stations $j \in J$ in order to minimize the operative costs. Each task i has a set of options O_i which can be selected by the customer. For instance, a car can be produced without a sunroof, with a normal sunroof, or a panoramic sunroof.

The assembly of the sunroof is then a task with three options (no roof, normal, or panoramic). A product variant is defined by selecting one option for each task. One example of a product variant based on three tasks is a vehicle containing a normal sunroof, 5 doors, and leather seats. The number of product variants which can be produced on an assembly line is given by the product of the number of options available. The nomenclature of the sets, variables, and parameters used in the problem definition are summarized in Table 1.

Table 1: Nomenclature of sets, data, and variables of the formulation.

Sets	**Meaning**
I	Set of tasks i
J	Set of stations j
O_i	Set of options o to be selected for task i
$Prec$	Set of precedence relations (i_1, i_2)
Data	**Meaning**
$D_{i,o}$	Duration time of option o of task i
$D^T_{i,o}$	Probability of option o of task i be ordered by a customer
D^{avg}_i	Average processing time of task i
CT	Cycle time
c_1	Weight of the first component of the objective function
c_2	Weight of the second component of the objective function
Variables	**Meaning**
x_{ij}	Balancing variables: 1 if task i is assigned to station j, 0 otherwise
Len_j	Length of the station j: continuous

In paced lines, the speed of the conveyor is the same for all stations, which defines the productivity or, in other words, the cycle time. The cycle time is defined as the time gap between two adjacent products in an assembly line and not necessary needs to be equal to the time a product remains in a station. If the station is planned longer than the cycle time, some complex models requiring more than the cycle time can still be assembled in the station. This excess of work must, however, be compensated by assembly less complex models. One illustration of the scheduling within a station is given in Fig. 1. The longer stations are, more scheduling flexibility is allowed in the station.

Let D_{io} denote the integer processing time of task i for option o. Each option o of task i is considered to be selected with a given probability P^T_{io}. Based on these probabilities, the average processing time of a task i is defined as D^{avg}_i.

The balancing problem consists of finding the best assignment for each task. These assignments must obey ordering restrictions in form of precedence relations contained in the set $(i_1, i_2) \in Prec$. The operative costs are considered as the line-length cost and the expected utility-work cost. The line-length cost is a linear cost in the sum of the stations' lengths and represents the operative costs and opportunity costs of the investment and space. The expected utility-work cost is a function of the expected utility work. The model for the optimization problem is given in (1) - (5). The model exhibits two sets of

variables, the assignment variables x_{ij}, representing a task i at the station j, and the length variable of each station j (Len_j).

$$\textbf{Minimize } c_1 \cdot \sum_{j \in J} Len_j + c_2 \cdot \sum_{j \in J} E\left(C(x_{1j}, ..., x_{|I|j}, Len_j)\right) \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall\, i \in\ I \tag{2}$$

$$\sum_{k \in J : k \leq j} x_{i_1 k} \geq \sum_{k \in J : k \leq j} x_{i_2 k} \qquad \forall\, (i_1, i_2) \in Prec, j \in J \tag{3}$$

$$\sum_{i \in I} D_i^{avg} \cdot x_{ij} \leq CT \qquad \forall\, j \in J \tag{4}$$

$$x_{ij} \in \{0, 1\}, Len_j \geq CT \qquad \forall\, i \in I, j \in J \tag{5}$$

The objective function is described in (1). The modelled assembly-line cost considers the line-length costs and the expected utility work of an assignment $E\left(C(x_{1j}, ..., x_{nj}, Len_j)\right)$ under a random sequence of products. Both costs are summed using appropriate cost factors c_1 and c_2. Here, the planner must decide on a statistic measure to evaluate the solution quality. In this paper, the expected cost is used as the objective function. Such expected cost formulation is justified by the repetitive use of the assembly line along months or years of operation, during which a possible new production sequence may be applied daily or even in each shift. A robust approach can be better for some problematic sequences of products but would not result in the most productive assembly line. Furthermore, utility workers assure every sequence is feasible. Eqs. (2) are the occurrence restrictions. The precedence constraints are modelled in Ineqs. (3). Ineqs. (4) are restrictions on the expected processing time in each station. The utility workers should compensate for a sequence of heavy loaded products and not to be part of the average production time. Therefore, the average processing time of the station is limited by the cycle time (CT) but stations can be planned to be as long as necessary. Note that the length of the stations is minimized in the objective function and there is a trade-off between station length and expected utility work. Finally, the assignment variables x_{ij} are binary, while the station length Len_j is not smaller than the space equivalent of the cycle time (5). That is, length measurements are expressed with time units representing the displacement of the conveyor belt within the time interval. Restrictions (2) to (5) are standard in ALBP literature. The novelty of the problem in question is the exact computing of the stochastic term in the objective function.

Figure 1 contains an example of the inner scheduling of a station in a paced line. In the diagram, the abscissa represents the station length and the duration of the processing of each product. In the illustration, the worker starts at the left boundary of the station (position 0) with the processing of product $P1$. In this station, $P1$ requires less time than the cycle time, so the worker must wait until the next product enters the station. When the next product $P2$ arrives at the station, the worker is already at the station's left boundary. If a product requires more processing time than the cycle time, the worker can continue the operations if the station is long enough. The processing of $P3$ starts immediately after

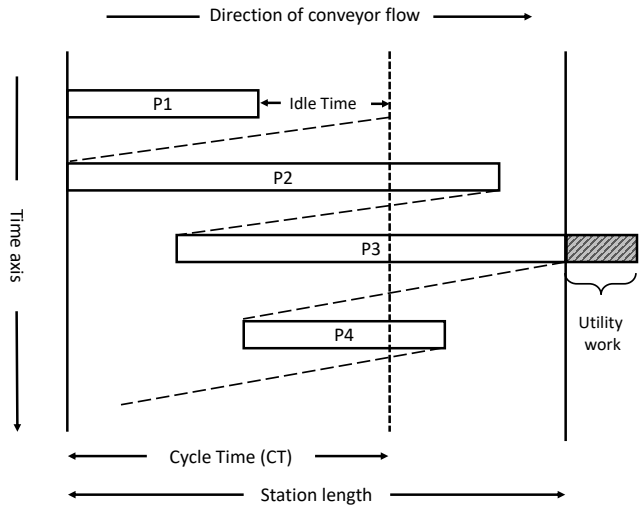


Figure 1: Realization of a production sequence in a paced assembly line. Utility work is used to assure processing within station bounds.

$P2$ is concluded, as the movement times are small and can often be disregarded (Boysen et al., 2009). The starting position of $P3$, however, is not the left boundary of the station, since the assembly of $P2$ finished after the cycle-time mark. If the worker is not able to finish an operation within the station borders, a utility worker must be called for aid. The utility worker can perform the remaining tasks and free the worker for the next product. The example ends with product $P4$.

The schedule displayed in Figure 1 is optimal for the given sequence in relation to the amount of utility work used. The optimal policy is proven by Yano & Rachamadugu (1991) and dictates that the worker should complete each product before moving to the next. Utility work should only be applied if the processing of a product were to exceed the station length.

The calculation of the expected amount of utility work for the presented system is modelled in Section 4. In order to model such a system using Markov chains, the following assumptions are necessary.

- The processing times of option o in task i D_{io} are integer.
- Utility work can always assure that every piece arrives ready at the next station.
- Processing times are independent of the sequence in which tasks are performed.

The assumptions are important in order to decompose the problem into independent stations and to yield a finite number of scenarios.

4. Evaluating the expected utility work

For each task assignment and station length, the expected utility work for a random sequence of products can be estimated using simulation or calculated with a Markov Chain. The combination of the

options of the assigned tasks results in the product variants processed in the station. For instance, an assignment of two tasks with two and three options, respectively, would result in six different product variants. The probability of a product variant $m \in M$ is derived by the probability of the combination of the options and is referenced as P^m. These probabilities can consider independent options or correlations between the selected options. The processing time of a variant m in a station (D_{mj}) is given by

$$D_{mj} = \sum_{i \in I} D_{io^m} \cdot x_{ij} \; \forall \; j \in J, \; m \in M,$$

which is the sum of the processing times of each option o^m in the variant m of the tasks assigned to the station j. In this section, all stations are modelled independently as Markov chains following Sikora (2022). Therefore, the station index j is dropped in the following. The idea of using Markov Chains to evaluate the solutions to subproblems in stochastic environments is inspired by Gwiggner (2020).

To model the position of the worker as a Markov process, his or her end position after producing each piece is selected as a reference point. Fig. 2 illustrates that the position of a piece is only dependent on its processing time and the end position of the previous piece. As the worker returns the equivalent of CT units of time to start working on the next piece, the resulting displacement is $D_m - CT$, which is labelled as Δ_m. The position of a worker (Pos) respects station boundaries: the end position of a piece is at least equal to the cycle time (otherwise idle time occurs, see Fig. 1) and limited by station length (Len), resulting in the set $\{CT; Len\}$. To simplify the notation, $\overline{Pos}$ (and $\overline{Len}$) is a translation of Pos (and Len) CT units to the left, so that $\overline{Pos} \in \{0, \overline{Len}\}$. The transition matrix of the Markov Process can then be expressed by

$$T = \begin{array}{c} \\ 0 \\ \vdots \\ i \\ \vdots \\ \overline{Len} \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & \cdots & j & \cdots & \overline{Len} \end{array} \\ \begin{pmatrix} P(\Delta \leq 0) & \cdots & P(\Delta = j) & \cdots & P(\Delta \geq \overline{Len}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P(\Delta \leq -i) & \cdots & P(\Delta = j - i) & \cdots & P(\Delta \geq \overline{Len} - i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P(\Delta \leq -\overline{Len}) & \cdots & P(\Delta = j - \overline{Len}) & \cdots & P(\Delta \geq 0) \end{pmatrix} \end{array}$$

where rows represent the end position after processing the previous piece and columns are associated with the worker position after processing the piece. The value of Δ represents the displacement and links both the initial and final state and $P(\cdot)$ stands for the probability that the products have a displacement of Δ within the specified conditions.

The transition matrix is defined for the positions in $\{0, \overline{Len}\}$. If a piece were to be finished before 0, idle time occurs and the worker has to wait for the next piece. A violation of the upper bound is corrected by using utility work. That is, an extra worker helps with the processing so that the piece is finished before the station boundary. The amount of necessary utility work is given by

$$U(\Delta_m, \overline{Pos}) = \text{Max}(0, \overline{Pos} + \Delta_m - \overline{Len}). \tag{6}$$

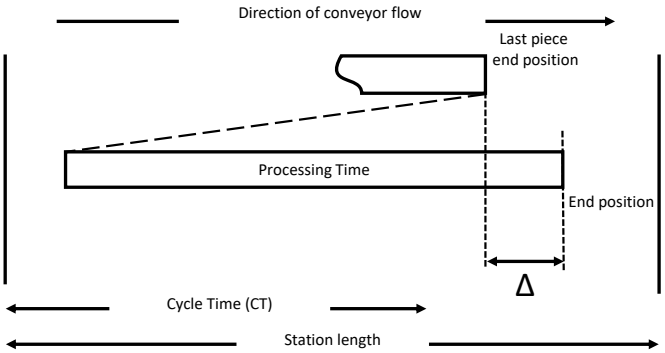


Figure 2: The final position of a piece is determined by the last position and a Δ.

The transition matrix T can be used to calculate the stationary probabilities w_p of a worker being in a given position $p \in \{0, \overline{Len}\}$ by solving the system of linear equations $w = T \cdot w$ and $\sum_{p=0}^{\overline{Len}} w_p = 1$ (Cechin & Corso, 2019). The probability P^m for the occurrence of each product model m and the stationary probabilities are used in

$$E\left(C(x_{1j}, ..., x_{|I|j}, Len_j)\right) = \sum_{m \in M} \sum_{p=0}^{\overline{Len_j}} P^m \cdot w_p \cdot U(\Delta_m, p) \tag{7}$$

to calculate the expected utility work for a given assignment and station length. Function (7) is a piece-wise linear and convex function in station length (Sikora, 2022) when all processing times are integer.

An example of the expected-utility-work function values is shown in Figure 3. In the plot, the expected utility work is shown for each station length between 15 and 35 LU along with an exponential interpolation of the expected utility work values. Each point of the plot represents the value obtained by solving the Markov chain's system for a specific station length. For the plot, independent probabilities are used.

5. Modeling product variant probabilities

In the problem definition, it is assumed that the share of each option of each task is given. However, there are multiple possible arrangements of products that fulfil this assumption. More specifically, the definition of the variant probabilities P^m of Eq. 7 is given in this section build based on the combinations of variants derived from two tasks. Let $P^T_{i,o}$ be the probability of option o of a task i being ordered (data from manufacturer), while $P^C_{o,p}$ represents the probability of a product containing options o for the first task and p for the second task.

Formally, the conditions for the model probabilities are given in Equations (8) to (11). Let i_1 and i_2 be two tasks assigned to a station with a set of options O^{i_1} and O^{i_2}. The conditions for the model probabilities are that they add up to 100% (eq. (8)) and the sum of the product variant probabilities containing a specific option o_k is equal to the probability of such option being chosen (eqs. (9) and (10)). Furthermore, the probabilities cannot be negative (ineq. (11)).

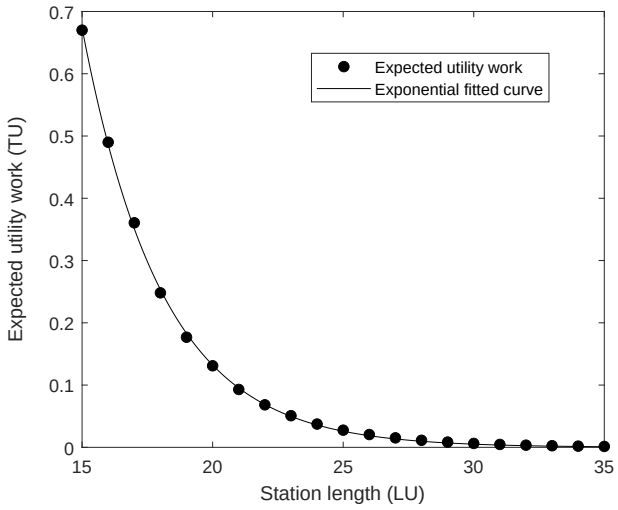


Figure 3: Expected utility-work cost with respect to station length for an example with $CT = 15$ and exponential interpolation of the exp. utility work values.

$$\sum_{k \in O^{i_1}} \sum_{l \in O^{i_2}} P^C_{k,l} = 1 \tag{8}$$

$$\sum_{k \in O^{i_1}} P^C_{k,l} = P^T_{i_2,l} \quad \forall\, l \in O_{i_2} \tag{9}$$

$$\sum_{l \in O^{i_2}} P^C_{k,l} = P^T_{i_1,k} \quad \forall\, k \in O_{i_1} \tag{10}$$

$$P^C_{k,l} \geq 0 \quad \forall\, k \in O_{i_1}, l \in O_{i_2} \tag{11}$$

The system of equations (8) to (10) plus the non-negativity constraint (11) define all feasible product variant probabilities with or without correlation. For stations with more than two tasks, the restrictions must be applied to each pair of tasks.

One solution of the system is given by independent probabilities. Given two correlated tasks i_1 and i_2 with n and l options, respectively, matrix

$$IP_{n \times l} = \begin{bmatrix} P^{i_1}_1 \cdot P^{i_2}_1 & P^{i_1}_1 \cdot P^{i_2}_2 & \cdots & P^{i_1}_1 \cdot P^{i_2}_l \\ P^{i_1}_2 \cdot P^{i_2}_1 & P^{i_1}_2 \cdot P^{i_2}_2 & \cdots & P^{i_1}_2 \cdot P^{i_2}_l \\ \vdots & \vdots & \ddots & \vdots \\ P^{i_1}_n \cdot P^{i_2}_1 & P^{i_1}_n \cdot P^{i_2}_2 & \cdots & P^{i_1}_n \cdot P^{i_2}_l \end{bmatrix}$$

is a valid solution.

Theorem 1. *Matrix IP represents a valid solution for the system (8) to (11).*

Proof: The values of each row and column of IP correspond to the probabilities of product variants containing the corresponding options. The sum of all elements in each row k is equal to $P^{i_1}_k \cdot \sum_{p=1}^{l} P^{i_2}_p =$

$P_k^{i_1}$. Therefore, equation (9) is fulfilled. The same reasoning can be applied analogously to columns. Furthermore, the probabilities are non-negative per definition. □

For independent probabilities, the probability of each combination used in (7) is given by a product of the individual probabilities. The formula

$$P_j^m = \prod_{i \in I | x_{ij}=1} \prod_{o \in O_i} P_{io}^T \; \forall \; j \in J, \; m \in M$$

uses the index j for the considered station to reflect the assignment variables x_{ij}. Here, P_j^m represents the probability of occurrence of a variant m in a station, while the superscript T in P_{io}^T refers to the probability that option o of task i is selected.

The correlated options used in this paper are based on a joint probability change matrix

$$Corr_{n \times l} = \begin{bmatrix} a_{11} & a_{12} & \cdots & -\sum_{j=1}^{l-1} a_{1j} \\ a_{21} & a_{22} & \cdots & -\sum_{j=1}^{l-1} a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ -\sum_{i=1}^{n-1} a_{i1} & -\sum_{i=1}^{n-1} a_{i2} & \cdots & \sum_{i=1}^{n-1} \sum_{j=1}^{l-1} a_{ij} \end{bmatrix}$$

and the matrix $CP = IP + \epsilon \cdot Corr$ containing correlated probabilities.

Theorem 2. *Matrix CP represents a valid solution for the system (8) to (11) for an adequate ϵ.*

Proof: As the sum of each row and column of matrix $Corr$ is equal to zero, the addition of matrix $\epsilon \cdot Corr$ does not change the sum of rows and columns. Therefore CP also fulfills (9) and (10) (see Theorem 1). Let $h = \min_{(k \in O^{i_1}, l \in O^{i_2} | Corr(k,l) < 0)} \left\{ P_k^{i_1} \cdot P_l^{i_2} / (-Corr(k,l)) \right\}$. Per construction, h is the maximal value of ϵ that can be added from $Corr$ until the first element of CP turns to zero. For any value of $0 \leq \epsilon \leq h$, all values in CP are non-negative. □

The proposed method does not consider the correlation factor between tasks but uses an additive term to control the product variant probabilities within the constraints. The method is intended to show the feasibility of using correlated data within the algorithm. The modelling of real data based on correlation factors is left as future work.

To illustrate the effect of correlation in the definition of customer orders, an example is described. Consider a station in which tasks i_1 and i_2 are assigned. Both tasks offer customers two options each. For the example, consider that Option 1 and Option 2 of the first task are ordered in 60% and 40% of the products, respectively ($P_{1,1}^T = 0.6$ and $P_{1,2}^T = 0.4$). For the second task, both options are demanded equally ($P_{2,1}^T = P_{2,2}^T = 0.5$). Based on these probabilities, if the customer preferences are considered independent, four variants are expected to be sold with probabilities $P_{1,1}^C = P_{1,2}^C = 0.6 \cdot 0.5 = 0.3$ and $P_{2,1}^C = P_{2,2}^C = 0.4 \cdot 0.5 = 0.2$. Such arrangement is displayed on the left side of Figure 4.

Suppose that the product variant with Option 2 for both tasks exhibits a long processing time and mounting this variant (2-2) may cause utility work. For this station, a correlation between both tasks

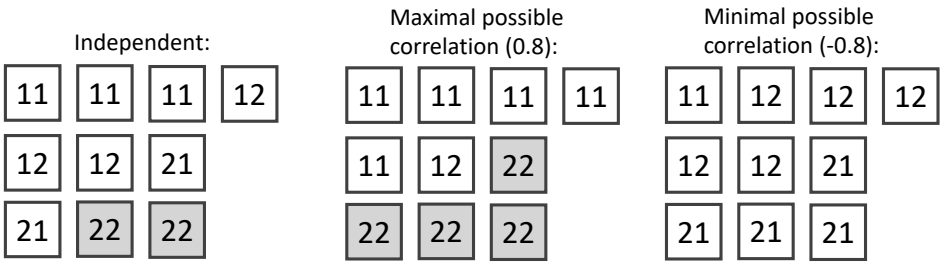


Figure 4: Example of possible product distributions considering two tasks with two options each with and without positive or negative correlation between tasks.

resulting in more 2-2 variants than expected in an independent distribution can be troublesome. Although the share of options for each task is given, the share of each of the variants (combinations 1-1, 1-2, 2-1, and 2-2) depends on how strongly the tasks are correlated. It is possible to generate feasible variant distributions with $P^C_{2,2} = 0$ to 0.4. The worst-case scenario in respect to utility work occurs when $P^C_{2,2} = 0.4$, depicted in the center of Figure 4. $P^C_{2,2} > 0.4$ is impossible since only 40% of the variants contain option 2 for task 1. The correlation between the tasks can also be negative. In this case, the combination of the second options appears less often than in the independent case. The limiting case for the decrease is shown in the right side of Figure 4, in which $P^C_{2,2} = 0$. For this example, the maximal correlation between the tasks is 0.8, while the minimal correlation is -0.8. Note that the correlation of 0.8 (or -0.8) are calculated for the values of Fig. 4 and are obtained by specific values for the a_{ij} coefficients.

6. Solution procedure

The proposed algorithm for the balancing of assembly lines for random sequences is based on a Branch-and-Bound algorithm presented in Sikora (2022). The structure of the procedure is illustrated in Fig. 5. Nodes consist of a set of tasks, whose enumeration is described in section 6.1. For each node, the station length is optimized using an exponential fitting algorithm described in section 6.2. For each combination of task assignment and station length, a Markov chain (section 4) is used to determine the expected utility work and consequently the assignment cost. The exact algorithm is described in section 6.3 and adapted into a heuristic in section 6.4.

6.1. Node enumeration scheme

A solution of the formulation of section 3 is considered feasible if all tasks are assigned, task assignments respect precedence relations, and stations' expected processing times stay within the cycle-time limit. As only the expected processing time is relevant for feasibility, the solution space of the stochastic problem is identical to a deterministic problem considering the expected processing times.

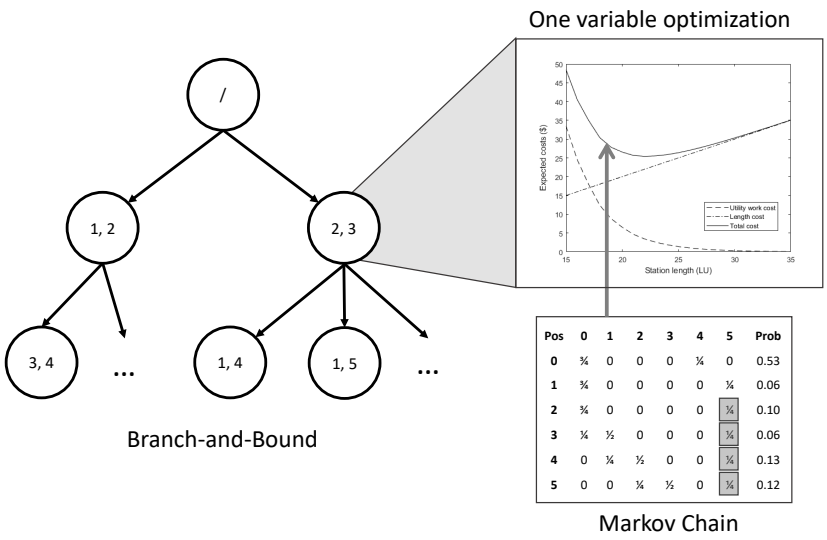


Figure 5: Illustration of the algorithm structure.

Among the branching schemes for the deterministic ALBP, the most efficient methods are based on the station-oriented enumeration schemes (Scholl & Klein, 1997; Sewell & Jacobson, 2012). In this scheme, each node of the tree consists of a set of tasks assigned to a station. Each level k of the enumeration tree consists of the nodes of possible assignments to station k. Modern solution procedures for SALBP and ALBP in general, such as the Branch-Bound and Remember (Sewell & Jacobson, 2012) heavily relies on good lower bounds and dominance rules for the algorithm efficiency. Although the SALBP rules cannot be applied directly to the ALBP with random sequence of products, adapted rules are introduced for the problem.

Algorithm 1 is used to enumerate the assignment alternatives of a station for a set of available tasks L. This set contains the tasks whose predecessors are already assigned so that their assignment respects the precedence constraints. Algorithm 1 performs a recursive depth-first search to enumerate all the possible assignments. The algorithm is an adaptation of the enumeration of Hoffmann (1992) and Fleszar & Hindi (2003).

During the node exploration, feasibility and dominance rules are checked. The precedence relations are obeyed in the construction of the set L. Whenever a task is added or removed to the set of assigned tasks A, set L is updated. That is, elements are added or removed from L so that all tasks in L can be directly assigned next. The second restriction is the expected processing time, which must be smaller than or equal to the cycle time. This restriction is implemented in the if-clause of line 3 in algorithm 1.

In the enumeration process, the idle time of the assignments is tracked. The expected idle time of a station is the difference between the cycle time and the expected processing time in the station. For the

whole line, the expected idle time is given by

$$CT \cdot |J| - \sum_{i \in I} D_i^{avg}$$

which is distributed among the stations. In the enumeration process, assignments that use more than the available idle time are infeasible, since the not yet assigned tasks cannot fit the remaining stations. This condition is checked in the if-clause of line 6 and is an extension of known lower bound techniques for SALBP-1 (Scholl & Klein, 1997).

For the deterministic SALBP, Jackson (1956) presented the maximum-load rule for station assignments. This dominance rule states that assignments with non-maximal station loads are dominated by maximum loads for the SALBP-1 variant. In the stochastic case, this rule does not apply, because the expected processing time may require much more utility work than assignments with positive expected idle time. Therefore, algorithm 1 uses a proposed weaker version of this dominance rule: the maximum upper-load rule. This rule considers the longest processing times of each task i (D_i^{max}) and does not consider assignments whose processing time in the worst-case scenario is small enough that another task could be added without causing utility work. This rule is implemented in the if-clause of line 7. This restriction needs to be waived for the last station of the enumeration process since set L is then empty.

Algorithm 1: OnePackingSearch($q, IdleTime, L, A$)

```
1  for r: q to |L| do
2      i = L_r
3      if D_A^avg + D_i^avg ≤ CT then
4          A = A ∪ {i}
5          Update L
6          if CT − D_A^avg ≤ IdleTime then
7              if D_A^max + Min_{k∈[i,|L|]}(D_{L_k}^max) > CT then
8                  RemainingIdleTime = IdleTime − (CT − D_A^avg)
9                  CreateNode(A, RemainingIdleTime, L \ A)
10             end
11             OnePackingSearch(r + 1, RemainingIdleTime, L, A)
12             Update L
13             A = A \ {i}
14         end
15     end
16 end
```

In Algorithm 1, function 'OnePackingSearch' is responsible for the enumeration of the assignments, stored in variable A. For the feasible station loads, function 'CreateNode' stores the given assignment, the remaining idle time for the next stations, and the list of tasks that may be assigned next ($L \setminus A$).

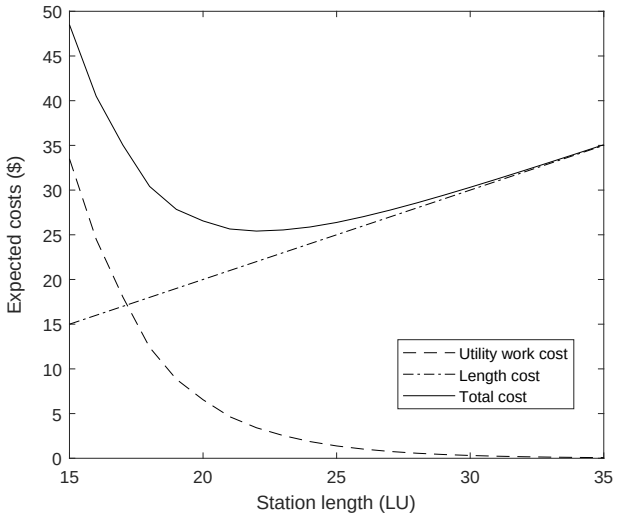


Figure 6: Expected utility-work cost, line cost, and expected total cost with respect to station length for an example with $CT = 15$.

These nodes are stored in a tree and indexing can eliminate the need of enumerating identical assignments repeatedly.

6.2. Length optimization

Each assignment or node of the enumeration tree has a cost compounded by the length cost and the expected utility-work cost of this station. The expected utility work is calculated by a Markov chain presented in Section 4 and is dependent on the station length. The expected utility-work cost is a convex and piece-wise linear, non-increasing function in station length (Sikora, 2022), while the line-length cost is a linear function. The total cost function, as in the objective function of the model (1), is the sum of both costs considering weights c_1 and c_2, respectively. An example of the total cost curve is given in Figure 6.

To obtain the minimal cost of a given assignment, the optimal length of the station must be determined. In this section, a one-variable optimization algorithm is presented. As the processing-time data is considered integer (and, as a consequence, Δ-values are integer), the optimal length is also integer (Sikora, 2022).

The structure of the one variable optimization is shown in Algorithm 2. As the total cost function is convex, but no information on the derivatives is available, the procedure performs the evaluation of a length value per iteration and uses information from the other three known length values to determine the new search interval. This structure is based on search procedures such as the Fibonacci Search or Golden Section Search (Kiefer, 1953). Differently from methods for arbitrary curves, the proposed algorithm uses information on the shape of the curve to select the newly evaluated point. As the expected utility work curve resembles an exponential shape, an exponential fitting is used to approximate the optimal solution

and accelerate the search procedure. Algorithm 2 also contains a correcting function for the cases in which the exponential fitting provides approximations outside the search interval. Finally, the search interval is updated based on the newly evaluated search length.

Algorithm 2: LengthOptIteration($A, LB, Pivot, UB, TC_{LB}, TC_{Pivot}, TC_{UB}$)

```
/* Approximate total cost curve and calculate next point.                    */
```
1 Sol = ExponentialFitting($LB, Pivot, UB, TC_{LB}, TC_{Pivot}, TC_{UB}$)

```
/* Round and assure Sol is within known bounds.                              */
```
2 CorrectWithinBounds($Sol, LB, Pivot, UB$)

```
/* Calculate the expected utility work.                                      */
```
3 $TC_{Sol} = c_1 \cdot Sol + c_2 \cdot$ MarkovChain(A, Sol)

```
/* Update bounds.                                                            */
```
4 UpdateBounds($Sol, LB, Pivot, UB, TC_{Sol}, TC_{LB}, TC_{Pivot}, TC_{UB}$)

In the 'ExponentialFitting' function, the known points ($Pivot$, LB, and UB) and their respective total costs (TC_{Pivot}, TC_{LB}, and TC_{UB}) are used to approximate the total-cost curve by a weighted sum of the exponential and linear curves in the form

$$\tilde{E}(Len) = c_1 \cdot Len + c_2 \cdot a \cdot e^{-b \cdot Len}.$$

Among many forms to fit the curve to the values, the version of the algorithm that performed the best in the empirical tests uses only two points (P_1 and P_2) for the fitting. One of the points is $Pivot$ and the second is either the lower bound (LB) or the upper bound (UB), whichever has the smallest distance to $Pivot$. The two-point fitting results in parameters

$$b = \ln\left(\frac{TC_{P_1} - c_1 \cdot P_1}{TC_{P_2} - c_1 \cdot P_2}\right) / (P_2 - P_1)$$

and

$$\ln a = \ln(TC_{P_1} - c_1 \cdot P_1) + b \cdot P_1.$$

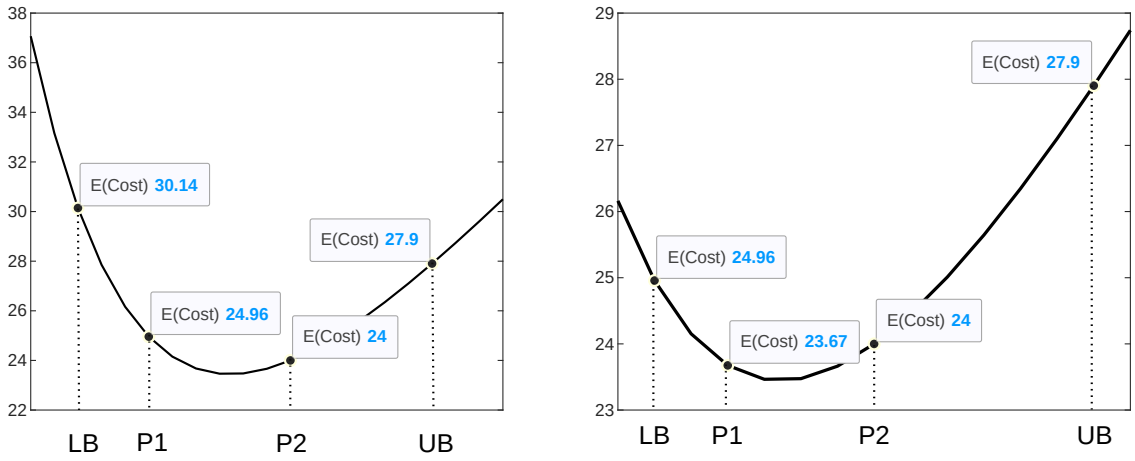
The optimal solution (Sikora, 2022) for

$$\frac{\mathrm{d}\tilde{E}(Len)}{\mathrm{d}Len} = 0$$

is given by

$$Sol = (\ln(c_2/c_1) + \ln a + \ln b)\,/b.$$

Although the fitting of the utility work with an exponential curve provides a good approximation, the algorithm does not always converge to the optimal solution. Therefore, the 'CorrectWithinBounds' function is used to make sure the newly tested point is within the search interval. This function rounds the value of Sol if its value lays in the interval $[LB, UB]$ and is different to $Pivot$. In the case of a bound violation (either $Sol > UB$ or $Sol < LB$), Sol is corrected to the midpoint of P_1 and P_2 rounded down. Finally, if the approximation gives $Sol = Pivot$, then Sol is set to the midpoint of the largest of



(a) Case 1: a lower bound update.

(b) Case 2: an upper bound update.

Figure 7: Illustrated cases for the bounds update function.

the intervals $[LB, Pivot]$ or $[Pivot, UB]$. This way, at the beginning of each iteration, Sol is an integer non-evaluated point within the search interval. The calculation of the total cost of Sol uses the procedure of section 4.

The remaining step in the procedure is given by the function 'UpdateBounds'. In this procedure, the search interval is updated based on the evaluation of the new point Sol. The procedure is illustrated in Fig. 7. From the four known points, three are stored for the next iteration. Either the lower or upper bound is updated with one of the internal points (Sol or $Pivot$). The case in which the left bound is updated is shown in Fig. 7a, while the upper bound update is depicted in Fig. 7b.

Algorithm 2 requires as initialization the value of three points (LB, UB, and $Pivot$). As a lower bound on the length, the cycle time (CT) is used, while the first $Pivot$-value is $CT + 1$. The utility work evaluation for $Len = CT$ requires only the weighted sum of the processing times longer than the cycle time since there is no length flexibility. For $Len = CT + 1$, the corresponding Markov chain has only two states (Position 0 and 1) and is also fast to be calculated. An initial upper bound for the length is given by

$$UB = \left\lfloor \frac{\min\left(TC_{LB}, TC_{Pivot}\right)}{c_1} \right\rfloor .$$

This bound is the equivalent in cost of the better of the LB or $Pivot$ solution if only length costs are considered. As utility work costs are not considered, UB must be an upper bound for the length of the station. To avoid calculating a large Markov Chain, TC_{UB} is initiated with an infinite value and UB is not used for the exponential fitting until an upper bound with the finite cost is calculated during the execution of the algorithm.

6.3. Iterative computation of nodes

With the node enumeration (Sec. 6.1) and node evaluation (Sec. 6.2) procedures defined, the last piece of the solution procedure is the search strategy.

Empirical tests show that the computational time required to evaluate a node is larger than the time required to compute the feasibility of child nodes in the enumeration procedure. This is rather unfortunate in the case of paths with feasible assignments for stations at the beginning of the line and infeasible nodes in deeper levels of the search tree. Therefore, the proposed search procedure generates the search tree nodes based on their feasibility only before their evaluation is computed.

Another drawback of the procedure is the lack of a known lower bound for a given assignment before the evaluation of the node. Since the utility work is the result of a Markov process and length is also optimized for each node, it is difficult to infer a proper lower bound for the total cost just based on the balancing variables. If, however, lower and upper bounds for the optimal length for a given assignment are known, a lower bound for the cost can be calculated by

$$TC_{Node} \geq c_1 \cdot LB + (TC_{UB} - c_1 \cdot UB).$$

That is, a valid lower bound is the length cost associated with the length cost of the LB and the utility-work cost of the UB (that is, total cost minus length cost). This lower bound is justified by the fact that length cost increases in the length, while utility-work cost decreases for increments of lengths.

Along with the node lower bound on the total cost, a lower bound for the line cost can be defined for each node. The line-cost lower bound of a node is its node-cost lower bound plus the minimal lower bound on costs of paths from the node to a leaf node and from the node to the root node. That is, a line-cost lower bound of a node is the lower bound of the cost of a path from the root node to a leaf node (a feasible assignment of the last station) passing through that node. A node lower bound (or upper bound) is shortened by 'Node LB (or UB)' and a line lower bound is denoted by 'Line LB'.

Based on the problem characteristics and the proposed lower bound on total cost, the proposed algorithm employs an iterative exploration of nodes. That is, the nodes are not solved until convergence one at a time. The algorithm loops between the feasible nodes and calculates one iteration of the 'Length-OptIteration' at a time for each node. The new interval is used to update lower bounds and potentially prune nodes in the process.

A flowchart of the algorithm is provided in the supplementary information.

6.4. Heuristic adaptations: Cost approximation and Beam search algorithm

The enumeration of the whole tree and the evaluation of nodes can require a long computational time. Therefore, heuristic versions of the algorithm are also proposed by simplifying some of the steps of the algorithm. Two components of the algorithm which can be relaxed for heuristic solutions are the enumeration of nodes and the evaluation of these nodes.

The heuristic evaluation of assignments is performed by leaving the length search before it converges. A very fast form of obtaining a cost approximation is calculating just the first iteration of Algorithm 2. The exponential fitting is calculated based on length CT (cycle time) and $CT + 1$, which require a Markov chain with only one and two states, respectively. The exponential fitting is then used to calculate the approximated optimal length and its cost. This approximated cost is used as the node cost in the heuristic search in Step 3 of Fig. A.8 to find a promising initial solution for the exact procedure. A heuristic version of the algorithm can be adapted by only relying on this approximated value.

The second search simplification can be performed in the enumeration of the nodes. Instead of the enumeration of all possible assignments, only a subset is considered. For this, all assignments are generated for a given length of the tree with Algorithm 1, but only a small portion of promising nodes is selected in a Beam Search Algorithm and the rest is discarded (Blum, 2008).

The selection of nodes plays an important role in the quality of the heuristic. For the proposed algorithm, some proxies for good solutions are identified and used to select the nodes. Candidate nodes are sorted with respect to a set of proxies and the best nodes of each proxy are selected. Selected proxies based on empirical tests are:

- Least deviation to the perfect division of idle time between all stations
- Most loaded assignments (for feasibility)
- Least deviation to the perfect division of processing time variance
- Weighted average of rules 1 and 3.

Based on the presented algorithm and the two simplifications, four possible algorithms are defined. The exact algorithm is denoted as 'Exact'. The version using the heuristic evaluation is named 'Approx.', the Beam Search algorithm is named 'Incomp. Enumeration' and the Beam Search using the heuristic evaluation is called 'Incomp. Approx.'.

7. Computational experiments

7.1. Dataset

In order to test the exact algorithm and the related heuristics, a dataset based on the 8 to 297-tasks SALBP instances from Scholl (1993) and proposed by Sikora (2022) is used. The selected instances contain at least 20 tasks, which are adapted to mixed-model problems. In the adaptation, each task can randomly present 1, 2, or 3 options with equal probabilities. The processing times of the second and third options are random uniform integer values between 50% and 150% of the original task time. Additionally, the second option has a probability of 20% of presenting no processing time. The probability of each option being selected in an order is calculated by a random draw of a uniform distribution. The individual values are divided by their sum and rounded to two decimal places accordingly to sum up to 1. The cost

factor c_1 is set to 1, while values $\{5, 10, 20, 50\}$ are used for c_2. All instances and results are included in the supplementary material and further details are given in Sikora (2022). A summary of the instances containing their name, number of tasks (I), number of stations (J), and the density of the precedence graph (order strength or OS) is given in table 2.

Note that considering up to three options per task already sums up to a huge amount of product variants. Since a product variant is defined as a selection of an option per task, the number of possible products is given by the product of the number of options. Instance Wee-Mag containing 75 tasks (each with 2 options) has, for instance, a number of around $2.84 \cdot 10^{19}$ product variants.

All methods are implemented in Visual Basic 15.0 on an Intel i7 8700K processor with 6 cores at 4.0 GHz and 32 GB of RAM.

Table 2: Summary of the instances used in the dataset order by the number of tasks.

Name	$\|I\|$	$\|J\|$	OS
Mitchell	21	3 - 8	0.709
Roszieg	25	4 - 10	0.717
Heskia	28	3 - 7	0.225
Buxey	29	7 - 12	0.507
Sawyer	30	7 - 11	0.448
Lutz1	32	8 - 10	0.835
Gunther	35	6 - 9	0.595
Kilbrid	45	3 - 10	0.446
Hahn	53	3 - 10	0.838
Warnecke	58	3 - 24	0.591
Tonge	70	3 - 20	0.594
Wee-Mag	75	11 - 30	0.227
Arc83	83	5 - 19	0.591
Lutz2	89	9 - 28	0.776
Lutz3	89	5 - 20	0.776

Instances based on the dataset from the more recent and systematic (Otto et al., 2013) are considered in 7.4.

7.2. Computational results in relation to the cost factor

The summary of the results of the instances solved exactly within a reasonable time is given in Table 3. Each row of this and the subsequent tables represents a set of instances based on an SALBP instance. The values presented in the table relate to the average results for a collection of station numbers. Row Buxey 7-12, for instance, refers to the average values of the Buxey instances containing 7 to 12 stations. In Table 3, the optimal length of the assembly line (Len), the expected amount of utility work measured in percentage of the cycle time (% UW), and the solution time (Time) are shown for the different cost factors (c_2).

From Table 3, the expected behaviour about the optimal length is observed. When the utility work is more costly, the planner should allow longer stations. Within a factor of 10 (from $c_2 = 5$ to $c_2 = 50$),

Table 3: Solution time, optimal length and percentage of utility work relative to the cycle time.

Instance	$\|J\|$	$c_2 = 5$			$c_2 = 10$			$c_2 = 20$			$c_2 = 50$		
		Len	% UW	Time	Len	% UW	Time	Len	% UW	Time	Len	% UW	Time
Mitchell	3 - 8	112.2	5.20	< 0.1	116.0	2.48	< 0.1	120.0	1.17	< 0.1	124.0	0.50	< 0.1
Roszieg	4 - 10	121.1	6.25	< 0.1	123.9	3.73	< 0.1	128.4	1.66	< 0.1	132.4	0.87	< 0.1
Heskia	3 - 8	1005.3	14.24	56.3	1081.7	7.84	102.9	1147.8	5.14	219.1	1281.0	2.69	563.6
Buxey	7 - 12	340.8	2.30	0.2	343.5	1.28	0.2	347.2	0.54	0.2	350.5	0.25	0.3
Sawyer	7 - 11	333.6	5.40	1.2	340.2	2.65	1.3	346.2	1.47	1.4	355.8	0.56	1.6
Lutz1	8 - 10	14404.7	7.43	< 0.1	14778.7	4.22	0.6	15222.7	2.29	5.9	15929.3	0.93	33.2
Gunther	6 - 9	469.8	15.90	1.6	504.5	7.98	2.0	538.3	4.12	2.8	583.0	1.84	3.6
Hahn	3 - 10	14814	5.54	239.1	15202.8	3.31	300.6	15722.4	1.73	653.4	16387.3	0.83	1758.4
Average		3886.0	7.37	50.3	3995.4	4.00	67.6	4131.6	2.16	146.2	4321.3	1.03	390.5

an average length increase of 11.2% is taken into account to reduce the expected utility work from an average of 7.37% of the cycle time to just 1.03%.

Interesting from a computational perspective are the differences in the required running time. The shift to longer stations due to higher utility-work costs greatly affects the size of the Markov chains. Since each length unit represents one possible worker position, larger linear systems must be solved for $c_2 = 50$. On average, the running times increase from 50.3 to 390.5 seconds per instance. For further comparisons, only the results for the largest cost factor ($c_2 = 50$) are displayed. The results for all instances and all c_2 values are included in the supplementary material.

7.3. Comparison of the proposed solution methods

In Table 4, the results of the proposed exact algorithm are compared with the heuristic versions of the solution procedure for $c_2 = 50$. The comparison with other methods for assembly line balancing, namely horizontal and vertical balancing (Merengo et al., 1999) are included in the supplementary material along with the results for the other cost factors. For the incomplete enumeration, the 20 best solutions for each selection criterion are used in each level of the tree search. The results of the complete enumeration are only considered for a subset of the instances due to solution time restrictions. The average values on the last two rows consider 'small' instances as the ones solved by all methods and 'all' instances as all of the listed instances in the table. Some incomplete enumeration methods could not find a feasible solution for instance Heskia 8, so this instance is omitted from the results report. The methods deliver a feasible solution for all other instances. The entries for larger instances, which are not solved by the exact algorithm, are displayed as $-$.

Table 4 shows the trade-off in solution quality and running time when the solution method is used as a heuristic procedure. When incomplete enumeration is allowed, all instances with up to 89 tasks (Lutz2 and Lutz3) are solved for any cost parameter. Larger instances with up to 148 tasks (about 10^{44} product variants) can also be solved within the memory and time limits, however not for every cost factor and every number of stations. Therefore, the instances which are not fully solved by 'Incomplete' and 'Incom. Approx.' are only listed in the supplementary material.

Table 4: Summary of the results for the exact algorithm and the heuristic procedures.

		Methods ($c_2 = 50$)							
		Exact		Approx.		Incomplete		Incom. Approx.	
Instance	$\|J\|$	Cost	Time	Cost	Time	Cost	Time	Cost	Time
Mitchell	3 - 8	129.3	< 0.1	144.3	< 0.1	129.5	0.3	146.2	0.2
Roszieg	4 - 10	139.3	< 0.1	148.1	< 0.1	142.3	0.3	149	0.2
Heskia	3 - 7	1474.7	675.5	1692.2	22.4	1587.4	1.0	1734.7	0.3
Buxey	7 - 12	354.9	0.3	394.5	0.2	377.3	0.3	411.8	0.2
Sawyer	7 - 11	366.1	1.6	463.6	0.6	410.2	0.2	457.1	0.2
Lutz1	8 - 10	16681.2	33.2	17820.3	9.0	16681.2	37.5	17820.3	9.5
Gunther	6 - 9	640.6	3.6	820.5	0.8	763.9	0.2	826.4	0.1
Kilbrid	3 - 10	-	-	-	-	720.5	4.0	861.9	3.0
Hahn	3 - 10	17360.5	1758.4	17952.6	82.1	17426.4	167.6	18387.7	40.1
Warnecke	3 - 24	-	-	-	-	2252.2	31.6	2505.1	7.0
Tonge	3 - 20	-	-	-	-	5031.8	880.5	5557.5	138
Wee-Mag	11 - 30	-	-	-	-	2179.7	173.4	2395.6	152.2
Arc83	5 - 19	-	-	-	-	109346	37287	120811.6	12610
Lutz2	9 - 28	-	-	-	-	672.4	0.8	781	0.6
Lutz3	5 - 20	-	-	-	-	2009.7	69.8	2252.5	22.8
Average (small)		4649.4	399.3	4895.7	18.3	4694.0	33.3	4982.3	8.1
Average (all)		-	-	-	-	12771.8	3570.2	14067.5	1199.9

By comparing the results of the different methods, the incomplete enumeration shows a good performance in both solution quality and time terms. On average for the small solutions, the cost increased by less than 1% by considering only a subset of the nodes. This difference, however, should increase with the problem size since the radius of 20 candidate solutions per performance criterion is constant for all tested instances. The relative amount of selected nodes then reduces for larger instances.

The incomplete enumeration produces better results for each instance constellation when compared to the 'Approx.' method. This result shows that optimizing the lengths of the stations plays a decisive role in the assembly-line costs since exploring all possible assignments with the approximated evaluation delivers worse results. Furthermore, the full enumeration of all task assignment possibilities accounts for a huge number of nodes. Even with the fast approximated evaluation, the size of the search tree is prohibitive for larger instances as implemented.

7.4. Instances based on Otto's dataset

In this section, the algorithm is evaluated based on the systematic dataset proposed by Otto et al. (2013). Otto's dataset is built with instances containing 20, 50, 100, and 1000 tasks which are named, respectively, small, medium, large, and very large. For the dataset, three properties of instances are explored: the form of the precedence relation graph, the order strength of the graph, and the processing time distribution of tasks. In respect to the form of the graph, Otto et al. (2013) consider bottleneck tasks and chains of tasks. Bottleneck tasks are those with several predecessors and successors. Such tasks work as anchors that partially divide the tasks precedence graph before and after them. Chains are sequences of tasks with exactly one predecessor and one successor. Both bottlenecks and chains are

observed in real instances (Otto et al., 2013) and are therefore used for molding the precedence graphs in the dataset. Otto's instances that contain bottlenecks are named 'BN', chains receive the acronym 'CH', and 'MIXED' stand for instances with both bottlenecks and chains. The Order Strength (OS) of a instance is a measure on how restricting the precedence relations are. A value of 0 represent an instance without any precedence relation, while a value of 1 results in an instance with a single feasible sequence for the tasks. The instances from Otto's are generated with OS of 0.2, 0.6, and 0.9. Finally, the distribution of the tasks' processing times are defined as peak at the bottom ('PB'), peak in the middle ('PM') or bimodal ('BI'). Peak at the bottom instances contain tasks with expected processing time of 10% of the cycle time. These instances present a large number of tasks per station since many tasks can be grouped within the cycle time. For the peak in the middle instances, the expected processing time is 50% of the cycle time. With relative large tasks, most of the stations present only one or two tasks, resulting in long lines. Finally, the bimodal distribution considers a mixture of 'PB' and 'PM' for the tasks' processing times. The instances from this dataset are adapted following the same procedure as described in Section 7.1.

In order to test the algorithm, one instance of each constellation of parameters is selected for the tests. The results are displayed in Table 5 for all small and medium instances. The first three columns show the instance properties for the precedence graph form (Prec.), the Order Strength (OS), and the Time distribution. For both the small and medium instances, the number of the instance from Otto's dataset, as well as the optimal cost and solution time are displayed. All instances assume the same cycle time of 1000 time units.

Note that all small instances are solved within few minutes. For the middle instances, the algorithm is not able to provide the optimal solution for some constellations of parameters. Mainly for the instances with low order strength, the enumeration tree is larger than the available 32 GB of memory and the solution procedure is aborted. For the medium instances, all instances with OS = 0.2 are not solved as well as instance 076 with OS = 0.6.

The most influential parameter on solution time is the order strength. Although the algorithm running time greatly varies from instance to instance, the OS is the only parameter that shows a clear trend of increasing difficulty (for low OS). The cost of the optimal solution is strongly linked to the time distribution used for the processing times. PB instances require many less stations since the tasks can be grouped together, while PM instances require long lines. This is reflected in the optimal solution, since most of the cost is based on the sum of the stations' length. Instances with bimodal distribution are in the middle of the spectrum.

Instances with even more tasks, such as the large and very large instances from the dataset, are not solvable with the proposed algorithm as presented. The enumeration of the tree requires more memory than the allocated and the solution procedure is aborted.

Table 5: Solution time and optimal solution for adapted instances from Otto's dataset. The instances correspond to the line cost $c_2 = 50$. Instances without result values have run out of memory and therefore no solution is reported.

Instance properties			Small instances			Medium instances		
Prec.	OS	Time dist.	Inst. No.	Cost	Time	Inst. No.	Cost	Time
BN	0.2	PB	001	3034.4	3.3	001		
BN	0.2	PM	016	12292.4	362.5	026		
BN	0.2	BI	041	6000.0	74.1	051		
BN	0.6	PB	066	3071.5	9.6	076		
BN	0.6	PM	091	11258.4	126.2	101	30664.8	298.6
BN	0.6	BI	116	5000.0	0.11	126	12000.0	3526
CH	0.2	PB	141	3029.3	97.4	151		
CH	0.2	PM	166	12122.6	60.0	176		
CH	0.2	BI	191	4000.0	44.7	201		
CH	0.6	PB	216	3010.5	1.33	226	7047.4	3975
CH	0.6	PM	241	13000.0	106.3	251	28322.2	1169
CH	0.6	BI	266	6159.3	4.0	276	15114.3	539.4
MIXED	0.2	PB	291	3000.0	161.5	301		
MIXED	0.2	PM	316	10048.1	187.2	326		
MIXED	0.2	BI	341	6000.0	347.2	351		
MIXED	0.6	PB	366	3757.7	1.6	376	7019.4	20155
MIXED	0.6	PM	391	11887.1	85.6	401	28606.2	1272
MIXED	0.6	BI	416	6000.0	3.8	426	11109.1	4723
MIXED	0.9	PB	441	3013.1	0.56	451	8091.2	1.03
MIXED	0.9	PM	466	13019.7	278.3	476	29205.4	482.8
MIXED	0.9	BI	491	6394.6	7.9	501	13200.8	1134

7.5. Comparison of the alternative solution methods

As there is no other solution method that exactly solves ALBPs considering random sequencing for the minimization of utility work, a direct comparison with other methods is not possible. There are, however, several optimization approaches for mixed-model assembly lines that indirectly deal with the production sequence. Merengo et al. (1999) describe horizontal and vertical balancing, two formulations for the ALBP aiming at the minimization of the differences between models either between or within stations. In the vertical balancing, the deviations of processing times between stations are focused. Considering L_j^{avg} the average processing time in station j and L_{max}^{avg} the processing time of the most loaded station, the vertical balancing objective function is given by

$$Obj_{Ver} : \sum_{j \in J} (L_{max}^{avg} - L_j^{avg}) \tag{12}$$

to be minimized. The horizontal balancing aims at the minimization of the model differences within each station. In Merengo et al. (1999), a root of the squared differences is used. Here, this objective function is simplified by

$$Obj_{Hor} : L_{max}, \tag{13}$$

that is, the maximal load of any product variant, in order to allow its solution in a commercial solver. Besides the objective function, the formulations consider cycle time and precedence constraints. The instances are solved using Gurobi 9.1 with a time limit of 3,600 seconds. The task assignments obtained by the horizontal and vertical balancing solution are then used in the proposed framework to optimize the station length. The summary of the results is displayed in Table 6.

Table 6: Assembly line cost obtained from different solution methods.

Instance	NS	$c_2 = 50$			
		Exact	Inc. Enum.	Horizontal	Vertical
Mitchell	3 - 8	129.3	129.5	137.8	132.3
Roszieg	4 - 10	139.3	142.3	149.7	145.1
Heskia	3 - 7	1509.1	1587.4	1639.3	1620.4
Buxey	7 - 12	354.9	377.3	386.8	379.2
Sawyer	7 - 11	366.1	410.2	411.1	404.8
Lutz1	8 - 10	16681.2	16681.2	17526.4	17268.7
Gunther	6 - 9	640.6	763.9	751.4	731.3
Kilbrid	3 - 10	-	720.5	670.6	630.5
Hahn	3 - 10	17360.5	17426.4	17820.9	18064.4
Warnecke	3 - 24	-	2252.2	2087.1	2006.9
Tonge	3 - 20	-	5031.8	4889.4	4533.6
Wee-Mag	11 - 30	-	2179.7	2019.9	1825.7
Lutz2	9 - 28	-	672.4	672.1	635.7
Lutz3	5 - 20	-	2009.7	2021.1	1916.6
Avgerage (small)		4649.4	4694.0	4831.8	4851.3
Avgerage (all)		-	2983.9	2959.9	2865.9

According to the results in Table 6, the incomplete enumeration can achieve better results for the small instances in comparison to the indirect formulation. For larger instances, however, the behaviour is inverted. These results point to two possible interpretations. The first is that the implementing settings for the incomplete enumeration are less efficient for larger instances. The width of the beam search (20 nodes for each of the four criteria) should be increased for larger instances if better solutions are necessary. The selection of the number of nodes treated in each level is flexible and presents a trade-off between solution time and quality. The second interpretation is that the indirect objective functions are adequate for the setting using random sequences and justify their role in the literature (Boysen et al., 2008).

8. Conclusion

In this paper, an exact method for solving the mixed-model assembly-line balancing problem for random sequences is presented. The products are modelled as a combination of multiple options for each component. Instances with up to 10^{12} possible products are solved exactly and the proposed heuristics extend such numbers to above 10^{20} possible product variants. This is possible due to the modelling of stations and worker positions using Markov chains and a set of assumptions that assures a finite number of states and the problem decomposability based on stations. Furthermore, the modelling approach allows the use of non-independent probabilities for the product variants.

The experiments show that the assembly-line balancing and the optimization of the station length are intrinsically related. The results of an incomplete enumeration of assignments with optimal station lengths are, for the presented dataset, better than considering all possible assignments without their respective best length. This comparison emphasizes the importance of treating together both aspects of a single problem.

For practitioners, the proposed model aids with the design of more resilient assembly lines when the production sequence is not yet known or planned. By considering a random sequence of products, the results can be considered a worst-case scenario for the application since optimizing the production sequence can only improve the system when considering the asymptotic behavior of an infinite sequence of products. The use of utility work can then be further reduced by optimizing the sequencing of the production. The stochastic solution, therefore, poses an important upper bound for the real production costs. Furthermore, the comparison of the algorithm results with indirect measures such as horizontal and vertical balancing (Merengo et al., 1999) shows that methods using the indirect measures result in similar solution quality.

The proposed algorithm poses a new framework for modelling assembly line problems and can be extended as further work. The presented method can be coupled to different classes of models including and extended to other assembly line structures. Furthermore, the results of the method are evaluated based on a random sequence only. It is left to further work whether such solutions also perform well when the planner can actively optimize the production sequence. Furthermore, evaluation of finite production sequences, such as the variants produced within a shift, can also be tackled in future works with other objective functions other than the expected costs. For finite sequences, computing the worst-case scenario can provide robust solutions for the balancing of mixed-model assembly lines.

The modelling framework can also be extended to other problem in which a Markov chain is used to evaluate the effect of stochastic factors. Furthermore, the use of dynamic lower bounds can be used in other contexts in which the evaluation of a node consists of solving a convex optimization problem.

The presented approach brings drawbacks based on the modelling approach. First, the efficiency of the algorithm strongly relies on the fact that the production sequence can be seen as a Markovian process. This requires that the random sequence is independent of the products selected before. Furthermore, all

tasks are considered independent to each other, otherwise the problem cannot be decomposed stationwise.

References

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, *142*, 259–277. doi:10.1016/j.ijpe.2012.10.020.

Bentaha, M. L., & Battaïa, O. (2015). A bibliographic review of production line design and balancing under uncertainty. *IFAC-PapersOnLine*, *48*, 70–75. doi:10.1016/J.IFACOL.2015.06.060.

Blum, C. (2008). Beam-ACO for simple assembly line balancing. *INFORMS Journal on Computing*, *20*, 618–627. doi:10.1287/ijoc.1080.0271.

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, *183*, 674–693. doi:10.1016/j.ejor.2006.10.010.

Boysen, N., Fliedner, M., & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics*, *111*, 509–528. doi:10.1016/j.ijpe.2007.02.026.

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

Boysen, N., Schulze, P., & Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, *301*, 797–814. doi:10.1016/j.ejor.2021.11.043.

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

Cechin, R. B., & Corso, L. L. (2019). High-order multivariate Markov chain applied in Dow Jones and IBOVESPA indexes. *Pesquisa Operacional*, *39*, 205–223. doi:10.1590/0101-7438.2019.039.01.0205.

Chica, M., Bautista, J., & de Armas, J. (2019). Benefits of robust multiobjective optimization for flexible automotive assembly line balancing. *Flexible Services and Manufacturing Journal*, *31*, 75–103. doi:10.1007/s10696-018-9309-y.

Chica, M., Bautista, J., Cordón, Ó., & Damas, S. (2016). A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand. *Omega*, *58*, 55–68. doi:10.1016/j.omega.2015.04.003.

Dong, J., Zhang, L., Xiao, T., & Mao, H. (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*, 7529–7548. doi:10.1080/00207543.2014.944280.

Eghtesadifard, M., Khalifeh, M., & Khorram, M. (2020). A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. *Computers and Industrial Engineering*, *139*, 106182. doi:https://doi.org/10.1016/j.cie.2019.106182.

Fisel, J., Exner, Y., Stricker, N., & Lanza, G. (2019). Changeability and flexibility of assembly line

balancing as a multi-objective optimization problem. *Journal of Manufacturing Systems*, *53*, 150–158. doi:https://doi.org/10.1016/j.jmsy.2019.09.012.

Fleszar, K., & Hindi, K. S. (2003). An enumerative heuristic and reduction methods for the assembly line balancing problem. *European Journal of Operational Research*, *145*, 606–620. doi:10.1016/S0377-2217(02)00204-7.

Gwiggner, C. (2020). Personal communication: A priori sequencing with uncertain release dates.

Hoffmann, T. R. (1992). Eureka: A hybrid system for assembly line balancing. *Management Science*, *38*, 39–47.

Hop, N. V. (2006). A heuristic solution for fuzzy mixed-model line balancing problem. *European Journal of Operational Research*, *168*, 798–810. doi:10.1016/j.ejor.2004.07.029.

Jackson, J. (1956). A computing procedure for a line balancing problem. *Management Science*, *2*, 261–271.

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

Kiefer, J. (1953). Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, *4*, 502–506.

Li, J., & Gao, J. (2014). Balancing manual mixed-model assembly lines using overtime work in a demand variation environment. *International Journal of Production Research*, *52*, 3552–3567. doi:10.1080/00207543.2013.874603.

Liu, X., Yang, X., & Lei, M. (2021). Optimisation of mixed-model assembly line balancing problem under uncertain demand. *Journal of Manufacturing Systems*, *59*, 214–227. doi:10.1016/j.jmsy.2021.02.019.

Lopes, T. C., Michels, A. S., Lüders, R., & Magatão, L. (2020a). A simheuristic approach for throughput maximization of asynchronous buffered stochastic mixed-model assembly lines. *Computers and Operations Research*, *115*, 104863. doi:10.1016/j.cor.2019.104863.

Lopes, T. C., Michels, A. S., Sikora, C. G. S., Brauner, N., & Magatão, L. (2021). Assembly line balancing for two cycle times: Anticipating demand fluctuations. *Computers and Industrial Engineering*, *162*. doi:10.1016/j.cie.2021.107685.

Lopes, T. C., Michels, A. S., Sikora, C. G. S., & Magatão, L. (2019). Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations. *Journal of Manufacturing Systems*, *50*, 193–200. doi:10.1016/j.jmsy.2019.01.001.

Lopes, T. C., Michels, A. S., Sikora, C. G. S., Molina, R. G., & Magatão, L. (2018). Balancing and cyclically sequencing synchronous, asynchronous, and hybrid unpaced assembly lines. *International Journal of Production Economics*, *203*, 216–224. doi:10.1016/j.ijpe.2018.06.012.

Lopes, T. C., Sikora, C. G. S., Michels, A. S., & Magatão, L. (2020b). Mixed-model assembly lines balancing with given buffers and product sequence: model, formulation comparisons, and case study.

Annals of Operations Research, *286*, 475–500. doi:10.1007/s10479-017-2711-0.

Manavizadeh, N., Rabbani, M., Moshtaghi, D., & Jolai, F. (2012). Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms. *Expert Systems with Applications*, *39*, 12026–12031. doi:10.1016/j.eswa.2012.03.044.

McMullen, P. R., & Tarasewich, P. (2003). Using Ant Techniques to Solve the Assembly Line Balancing Problem. *IIE Transactions*, *35*, 605–617. doi:10.1080/07408170304354.

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

Moodie, C. L., & Young, H. H. (1965). A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering*, *16*, 23–29.

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

Özcan, U., Kellegöz, T., & Toklu, B. (2011). A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. *International Journal of Production Research*, *49*, 1605–1626. doi:10.1080/00207541003690090.

Öztürk, C., Tunali, S., Hnich, B., & Örnek, A. (2012). Balancing and scheduling of flexible mixed model assembly lines with parallel stations. *International Journal of Advanced Manufacturing Technology*, *67*, 255–2591. doi:10.1007/s00170-012-4675-1.

Öztürk, C., Tunali, S., Hnich, B., & Örnek, A. (2015). Cyclic scheduling of flexible mixed model assembly lines with parallel stations. *Journal of Manufacturing Systems*, *36*, 147–158. doi:10.1007/s00170-012-4675-1.

Pereira, J., & Álvarez-Miranda, E. (2018). An exact approach for the robust assembly line balancing problem. *Omega*, *78*, 85–98. doi:https://doi.org/10.1016/j.omega.2017.08.020.

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

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

Sewell, E. C., & Jacobson, S. H. (2012). A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem. *INFORMS Journal on Computing*, *24*, 433–442. doi:10.1287/ijoc.1110.0462.

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

Sikora, C. G. S. (2022). Balancing Under No Sequencing Control. In *Assembly-Line Balancing under Demand Uncertainty* (pp. 97–131). Wiesbaden: Springer Gabler. doi:10.1007/978-3-658-36282-9_5.

Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. *European Journal of Operational Research*, *235*, 740–754.

Tiacci, L. (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. doi:10.1016/j.ijpe.2014.05.005.

Tiacci, L. (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. doi:10.1016/J.IJPE.2015.01.022.

Tiacci, L., & Mimmi, M. (2018). Integrating ergonomic risks evaluation through OCRA index and balancing/sequencing decisions for mixed model stochastic asynchronous assembly lines. *Omega*, *78*, 112–138. doi:10.1016/j.omega.2017.08.011.

Tremblet, D., Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N., & Dolgui, A. (2023). Optimizing task reassignments for reconfigurable multi-model assembly lines with unknown order of product arrival. *Journal of Manufacturing Systems*, *67*, 190–200. doi:https://doi.org/10.1016/j.jmsy.2023.02.001.

Urban, T. L., & Chiang, W.-C. (2006). An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. *European Journal of Operational Research*, *168*, 771–782. doi:10.1016/j.ejor.2004.07.027.

Vrat, P., & Virani, A. (1976). A cost model for optimal mix of balanced stochastic assembly line and the modular assembly system for a customer oriented production system. *International Journal of Production Research*, *14*, 445–463. doi:10.1080/00207547608956618.

Yang, C., & Gao, J. (2016). Balancing mixed-model assembly lines using adjacent cross-training in a demand variation environment. *Computers and Operations Research*, *65*, 139–148. doi:10.1016/j.cor.2015.07.007.

Yano, C. A., & Rachamadugu, R. (1991). Sequencing to Minimize Work Overload in Assembly Lines with Product Options. *Management Science*, *37*, 572–586. doi:10.1287/mnsc.37.5.572.

Zacharia, P., & Nearchou, A. C. (2013). A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. *Computers and Operations Research*, *40*, 3033–3044. doi:https://doi.org/10.1016/j.cor.2013.07.012.

Appendix A. Flowchart of the solution algorithm

The proposed Branch-and-Bound algorithm is outlined in Fig. A.8. The procedure can be described in 12 steps:

1. Load input data: cycle time, precedence relations, processing times of options, and probabilities of options.
2. Enumerate nodes: described in Subsection 6.1.

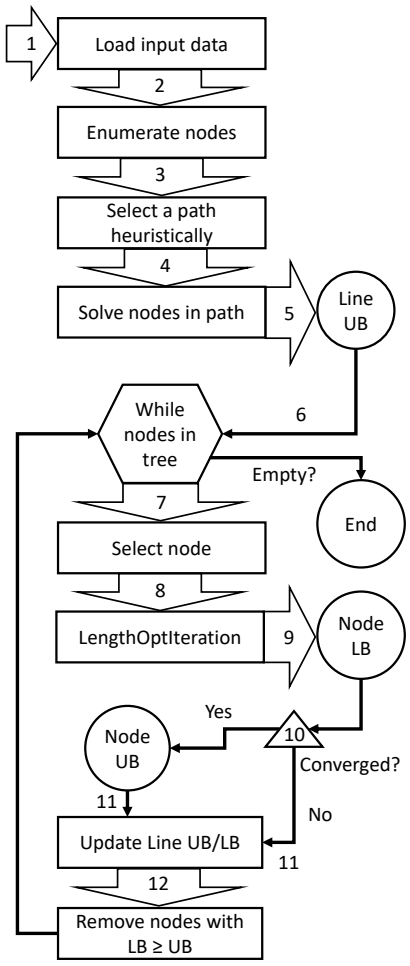


Figure A.8: Flowchart of the proposed algorithm.

3. Set a path based on the cost approximation: to provide an upper bound, one path is selected heuristically. The first iteration of 'LengthOptIteration' (Subsection 6.2) is calculated. The result of the first exponential fitting is used as an approximation for the total cost. The path with the lowest approximated cost is selected as the first solution.
4. Solve nodes in the path: the nodes on the initial path are solved to convergence.
5. Line UB: the sum of the cost of the solved nodes in the path provides an upper bound on the line cost.
6. While nodes in the tree: The main loop of the procedure. The iterative exploration of nodes is performed until all nodes are pruned.
7. Select node: a node is selected for exploration. In the implementation, the nodes with the lowest line lower bound (sum of the lower bound of the node, its parents, and children, from the root to leaf nodes) are selected. In computers with multiple cores, more than one node may be selected at the same time, since their computations are independent.

8. LengthOptIteration: an iteration of the one-variable optimization is performed for the selected node.
9. Node LB: update the cost lower bound of the node.
10. If clause: decides whether the node is solved. If the search interval $[LB, UB]$ has unexplored lengths, go directly to 11. If the one-variable search is completed, the node upper bound on cost is updated before starting step 11.
11. Update Line UB/LB: new bounds on the selected node are propagated to the cost bounds of parents and child nodes.
12. Remove nodes with $LB \geq UB$: if a newly actualized line cost lower bound is greater than or equal to the line-cost upper bound for any node, this node is pruned. After the pruning, go back to step 6.

Appendix B. Instance generation with correlated tasks

The dataset presented in Section 7.1 is extended to consider the correlations between task. The selected formulation to consider correlation consists of modelling the relationship between each pair of tasks and their options individually. For each pair of tasks, correlation factors are assigned with a 50% chance. The task pairs randomly selected to be correlated present a matrix of additive factors, whose size depends on their number of options.

As the proposed instances contain tasks of up to three options, three cases of correlation between two tasks must be accounted for: 2x2, 2x3 (or the transposed 3x2), and 3x3. The joint probability change matrices described in Section 5 are

$$Corr_{2\times 2} = \begin{bmatrix} a_1 & -a_1 \\ -a_1 & a_1 \end{bmatrix}$$

$$Corr_{2\times 3} = \begin{bmatrix} a_1 & a_2 & -a_1 - a_2 \\ -a_1 & -a_2 & a_1 + a_2 \end{bmatrix}$$

$$Corr_{3\times 3} = \begin{bmatrix} a_1 & a_2 & -a_1 - a_2 \\ a_3 & a_4 & -a_3 - a_4 \\ -a_1 - a_3 & -a_2 - a_4 & a_1 + a_2 + a_3 + a_4 \end{bmatrix}$$

for the corresponding number of options in each task. The values a_1, a_2, a_3, and a_4 represent the degrees of freedom in the system (8) to (11).

For the instances, random values between -1 and 1 are drawn from a uniform distribution. Whenever a value based on the sum of terms would be greater than one, the values of the corresponding row or column are divided by their sum. This way, all elements of the matrices are generated within the interval [-1, 1].

When more than a pair of correlated tasks are present in one station, the variant probabilities must account for all probability change factors. Assuming a station with tasks 1 to M, all correlated to each other, the probability of a given model containing options $o_1, o_2, \cdots, o_M$ is given by

$$P^C_{o_1,o_2,\cdots,o_M} = P^T_{1,o_1} \cdot P^T_{2,o_2} \cdot (\cdots) \cdot P^T_{M,o_M} + \epsilon \cdot \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Corr_{i,j}(o_i, o_j) \right),$$

in which $Corr_{i,j}$ is the probability change matrix derived for tasks i and j and $Corr_{i,j}(o_i, o_j)$ corresponds to the entry in matrix $Corr_{i,j}$ for two selected options o_i and o_j.

The value of ϵ must be selected in a way that no correlated probability is negative. The instances are generated using two settings for ϵ. One considers the maximal value ϵ that can be taken (named $\max \epsilon$). For the second setting, ϵ assumes half of its allowed maximal value (50% $\max \epsilon$).

Appendix C. Extended computational results with correlated products

In this section, further computational results are displayed containing correlation between tasks.

The summary of the results is displayed in Table C.7 for the instances with $c_2 = 50$. In the table, the average cost per group is shown along with the number of 'Wins' which counts for the configuration (independent, 50% $\max \epsilon$, or $\max \epsilon$) that achieves the smallest cost in each instance. The two points obtained by setting 50% $\max \epsilon$ are due to a tie when all methods achieve the same solution.

Table C.7: Comparison of the optimal assembly line cost for $c_2 = 50$ for uncorrelated and correlated options. 'Cost' represent the average line cost for the instances of the row, 'Win' shows the number of instances, for which the configuration achieves the smallest cost.

Instance	NS	Normal		50% $\max \epsilon$		$\max \epsilon$	
		Cost	Win	Cost	Win	Cost	Win
Mitchell	3 - 8	129.27	3	128.94	1	128.16	4
Roszieg	4 - 10	139.26	3	139.46	0	139.62	4
Heskia	3 - 8	1509.14	2	1510.54	0	1502.28	4
Buxey	7 - 12	354.94	2	354.71	0	353.40	4
Sawyer	7 - 11	366.05	1	365.01	0	358.63	4
Lutz1	8 - 10	16681.20	1	16663.54	0	16642.31	2
Gunther	6 - 9	640.64	1	641.85	0	637.49	3
Hahn	3 - 10	17360.52	4	17348.03	1	17333.95	5
Avg. / Total		4583.46	17	4580.19	2	4573.82	30

Counter-intuitively, the presence of correlation between options can **decrease** the line cost for a large portion of the instances. Inspection of the result data shows that the algorithm is able to exploit the correlation in the assignments. The instance with the largest difference between 'Normal' and '$\max \epsilon$' is used as an example to show such an effect. Instance Sawyer with 8 stations contains 30 tasks and a

cycle time of 41 time units. The data of the tasks assigned to one station in particular, their options' processing times and their probabilities are given on the left side of Table C.8. As the tasks contain one, two, and two options, respectively, four product variants can be specified. They are named '111', '112', '121', and '122' based on the options selected for each task and they are described on the right side of Table C.8. Note that model '112' requires a processing time of 48 time units, which is much larger than the cycle time (41 t.u.). In the random generation of instance Sawyer 8, a negative probability change factor between option 2 of task 20 and option 1 of task 21 is observed. By using the maximal value for ϵ, the probability of model '112' is zero for the correlated probabilities.

Table C.8: Description of the tasks and options of a station load of instance Sawyer 8 and the resulting product variants and their probabilities.

Task	No. Options	Task duration		Option prob.		Product	Processing time	Indep. Prob.	Corr. Prob.
		1	2	1	2				
10	1	4		1		111	41	0.24	0.48
20	2	16	0	0.48	0.52	112	48	0.24	0
21	2	21	28	0.5	0.5	121	25	0.26	0.02
						122	32	0.26	0.50

The assignment of tasks 10, 20, and 21 to one station is very costly when considering independent probabilities but this assignment is possible without the need for utility work if there is a negative correlation between the 'problematic' options. With these results and the example, it is shown that the correlation between options can be integrated into an exact algorithm. Furthermore, the correlations can be exploited to achieve even better solutions by joining tasks with beneficial correlated options. This improvement possibility may not be achievable if all options requiring long processing times are positively correlated to each other. Here, more research on the distributions of vehicle sales can be beneficial to the development of algorithms whose modelling is representative of the real correlated values.