# A Matheuristic for Makespan Minimization of Asynchronous Stochastic Mixed-Model Assembly Lines

**Thiago Cantos Lopes, Celso Gustavo Stall Sikora,**
**Adalberto Sato Michels, Leandro Magatão**
Federal University of Technology - Paraná (UTFPR)
Graduate Program in Electrical and Computer Engineering (CPGEI)
Av. Sete de Setembro, 3165, Curitiba-PR, Brasil, 80230-901
thiagolopes@alunos.utfpr.edu.br, sikora@alunos.utfpr.edu.br,
amichels@alunos.utfpr.edu.br, magatao@utfpr.edu.br

## RESUMO

Linhas de montagem de modelo misto são *layout* de produção orientados ao produto e compartilhadas entre um conjunto de modelos semelhantes de produtos. Linhas assíncronas permitem que peças se movam discreta e independentemente, quando o processamento na estação atual estiver concluído e a estação seguinte estiver livre. Se as sequências de produtos for desconhecida e além do controle dos gestores, a otimização da linha deve considerar fatores estocásticos. Diversos trabalhos propuseram funções objetivo para otimizar tais linhas de produção, mas poucos consideram incertezas associadas a sequências estocásticas. Este artigo apresenta uma *matheurística* desenvolvida para gerar soluções de balanceamento que otimizem a linha sob tais incertezas. A *matheurística* é composta por: um modelo em programação linear inteira mista, uma busca local e cortes de otimalidade redutores de campo de busca. O regime permanente estocástico é estimado pela média ponderada de tempos de ciclo estáveis de todas as sequências cíclicas de um tamanho específico. Testes em um conjunto de dados da literatura mostram que a *matheurística* proposta supera formulações anteriormente descritas por 2% em média, gerando melhores respostas em 323 de 350 instâncias. Um estudo de caso prático confirma as vantagens do método proposto e mostra que considerar sequências mais longas produz melhores resultados. Além disso, a *matheurística* demonstrou sinergia entre seus elementos, produzindo respostas melhores que a soma de suas partes em 61 de 350 instâncias.

**PALAVRAS CHAVE. Balanceamento de Linhas de Produção de Modelo Misto, Sequência de Produtos Estocástica, *Matheurística***
**Tópicos: AD&GP - PO na Administração e Gestão da Produção, IND - PO na Indústria**

## ABSTRACT

Mixed-model assembly lines are product-oriented layouts shared between a set of similar product models. Asynchronous lines allow pieces to move downstream discretely and independently, provided processing is completed at current station and the following station is empty. When product sequences are unknown and beyond the line managers control, line optimization must take stochastic factors into account. Numerous works have proposed goal functions to optimize mixed-model assembly lines, however, few consider uncertainties tied to stochastic product sequences. This paper presents a matheuristic designed to provide balancing solutions that optimize the production line under such uncertainties. The matheuristic is composed of: a mixed-integer linear programming model, a local search, and search-space reduction optimality cuts. The stochastic steady-state is estimated by averaging the stable cycle time of all possible cyclic product sequences of a certain size weighted by their probabilities. Tests on a literature dataset showed the proposed matheuristic outperforms previous literature formulations by 2% in average, generating better solutions on 323 out of 350 instances. A practical case study confirmed the advantages of the proposed formulation and showed that considering longer product sequences produces better results. Furthermore, the matheuristic has been shown to generate synergies between its elements, producing better answers than both of its parts in 61 out of 350 instances.

**KEYWORDS. Mixed-Model Assembly Line Balancing, Stochastic Product Sequence, Matheuristic**
**Paper Topics: AD&GP - OR in Administration & Production Management, IND - OR in Industry**

https://proceedings.science/p/85554?lang=pt-br

## 1. Introduction

Assembly lines are product-oriented layouts commonly used in industrial environments for mass-production of standardized products or large-scale production of similar products [Scholl, 1999]. Multiple authors have studied different aspects of assembly line balancing problems, ranging from lower-bounds and optimization techniques [Scholl and Becker, 2006] to strengthened mixed-integer linear programming models [Ritt and Costa, 2015].

These lines can be classified according to numerous features such as product diversity [Battaïa and Dolgui, 2013]: single model lines are dedicated to one product type exclusively, while multi-model and mixed-model lines are shared between a set of them. Multi-model lines have relevant set-up times when product models change, while mixed-model lines do not, allowing more flexibility. Multiple authors have analyzed and presented algorithms, heuristics, and mathematical models for the mixed-model variants [Decker, 1993; Merengo et al., 1999; Karabati and Sayin, 2003; Sawik, 2012].

Mixed-model assembly lines can be further classified in terms of line pace [Boysen et al., 2008]: in paced (continuous) lines, a conveyor moves the product pieces constantly downstream, while in unpaced lines product movement is discrete and can occur either synchronously (all pieces move forward together) or asynchronously (product pieces move independently).

The medium/long term balancing problem is often followed by the short-term sequencing optimization problem [Boysen et al., 2009]. If demand is relatively stable, it is also possible to simultaneously balance and cyclically sequence an assembly line [Öztürk et al., 2015]. However, in make-to-order environments [Bukchin et al., 2002], product sequences are not controllable by the manager: market fluctuations and other practical considerations might impose a specific product sequence. For example: in the automobile industry, it is common for the paint-shop to practically determine the factories' product sequence. This occurs because set-up times are much more relevant in that part of the factory and changing product sequences between factory sections is often considered too costly or problem-prone. Therefore, managers of the other parts of the factory may not have a say in product sequences. Furthermore, in the paint-shop, product color is the key factor for sequencing, and the vehicle's physical model is independent of its color. This means that for other parts of the factory, the model of the next product to enter the line can be seen as a random variable [Bukchin et al., 2002; Tiacci, 2015].

If managers know the product sequences in advance, it is possible to optimally balance the line using a sequence aware formulation [Lopes et al., 2018]. However, it might be too costly to re-balance the line on a weekly basis, and product sequences are rarely known in large advances. Only one balancing solution can be implemented for a considerable period and there is usually no solution that can be simultaneously best for all possible different product sequences [Birge and Louveaux, 2011]. Hence, a balancing solution that optimizes the *expected* stochastic makespan is desired. To find such a solution is the focus of the present paper.

This paper is structured as follows: Section 2 defines the studied problem. Section 3 presents a new Matheuristic based on a Mixed-Integer Linear programming (MILP) model (Section 3.1), search-space reduction constraints (Section 3.2), and a local search procedure (Section 3.3). Section 4 presents previous literature Benchmark formulations. Section 5 presents two case studies that compare the proposed matheuristic to previous formulations: a combinatorial dataset case study (Section 5.1), and a practical one (Section 5.2). Lastly, Section 6 summarizes the paper's major contributions and conclusions.

## 2. Problem Statement

A mixed-model assembly line balancing problem consists primarily on assigning a set of tasks $T$ to a set of sequential stations $S$. Technological constraints impose direct precedence relations between tasks. These relations are given by a binary matrix $R_{[t_1,t_2]}$, whose values are set to one when task $t_1$ must precede task $t_2$. A feasible task-station assignment must respect all such relations. A set of different product models $M$ is produced in the line. Task assignments to stations are the same for all product models in order to benefit from worker specialization, and

precedence relations are also the same for all product models. Therefore, any solution that respects precedence relations is feasible for any product sequence. In that sense, the problem is closer to stochastic programming [Birge and Louveaux, 2011] than to robust optimization [Ben-Tal et al., 2009].

The duration (measured in time units) of each task $t$ is different for each product model $m$ produced in the line and given by the parameter matrix $D_{[t,m]}$. For all product models, the processing time at each station is given by the sum of duration of tasks assigned to it. Each product piece that is produced in the line moves asynchronously and sequentially between stations, starting at the first one and moving to the next station when two conditions are verified: processing at current station is complete, and the following station is empty. This means that blockages and starvations may occur: a piece is blocked when it has concluded processing, but the following station is still occupied by another piece, and a station is starved when a piece departs from it and the next piece is not yet ready to depart from the previous station. The tasks' processing times are considered deterministic parameters, and the stochastic nature of the line refers to the product sequence.

Each product model $m$ has a demand rate parameter $O_m$ such that the sum of demand rates equals one. The product sequence is considered a memoryless stochastic process: each new piece that enters the line has the probability $O_m$ of being product model $m$ [Bukchin et al., 2002]. Lets consider the example where the factory is set to produce 50000 products, and the goal is to minimize the expected makespan. A scenario is defined as a model sequence of 50000 products. Each scenario has easily computable makespan and probability values. If all scenarios could be explicitly computed, it would be possible to have an exact measure of the expected makespan [Birge and Louveaux, 2011]. However, there is an exponentially large number of scenarios ($|M|^{50000}$) and, therefore, it is impractical to do so via standard stochastic programming. Given a balancing solution, the expected makespan can be approximated by multiple independent simulations.

The problem's goal is to produce a balancing solution that minimizes the line's expected makespan. Due to the large number of pieces produced, steady-state optimization is key to produce low makespan solutions. However, the random product sequences difficult the determination of such stochastic steady-state. The following section presents a matheuristic devised around approximations for it.

## 3. The Proposed Matheuristic Procedure

The proposed procedure consists of a dynamic combination of three elements: a mathematical model (Mixed-Integer Linear Programming) described in Section 3.1; optimality cuts described in Section 3.2; and a local search procedure presented in Section 3.3. The way these elements interact with one another is presented by Figure 1.

The Proposed Model (PM) approximates the stochastic steady-state by explicitly considering all possible product model sequences up to $N$ pieces ($|M|^N$ sequences). The stochastic steady-state cycle time is approximated by the weighted sum of the stable steady-state cycle time of each product sequence and used as an indirect performance. The weights in this sum are defined based on the occupation rates of the sequence's product models. Section 3.1 describes the proposed model in more detail.

The Optimality Cuts (OC) consist of search-space reductions designed to accelerate the Proposed Model and the Local Search. They are dynamically strengthened whenever a new incumbent solution is found. These constraints are defined based on an incumbent solution with known expected makespan $Mkpn_{inc}$. A correspondent incumbent realized cycle time value $C^R_{inc}$ is defined by dividing the makespan by the number of produced pieces. This value is used to demonstrate that some task-station assignments will necessarily lead to worse solutions than the current incumbent. This information is then codified in constraints whose violations imply in sub-optimality in regard to the incumbent solution. Section 3.2 describes the developed optimality cuts in more detail.

https://proceedings.science/p/85554?lang=pt-br

The Local Search (LS) procedure performs a neighborhood search on solutions, either randomly generated by a priority-based rule or provided by the mathematical model. Its core idea consists on re-assigning all tasks allocated to a pair of adjacent stations: all possible re-assignments (of these tasks to the station pair) are tested and the best re-assignment is chosen as the next trial solution. In order to evaluate each re-assignment, a set of quick deterministic simulations is performed for each sequence considered in the MILP model and the observed cycle times are weighted by their probabilities and summed. Local *minima* are added to a Tabu list to avoid repeating trial solutions. Section 3.3 describes LS in more detail.

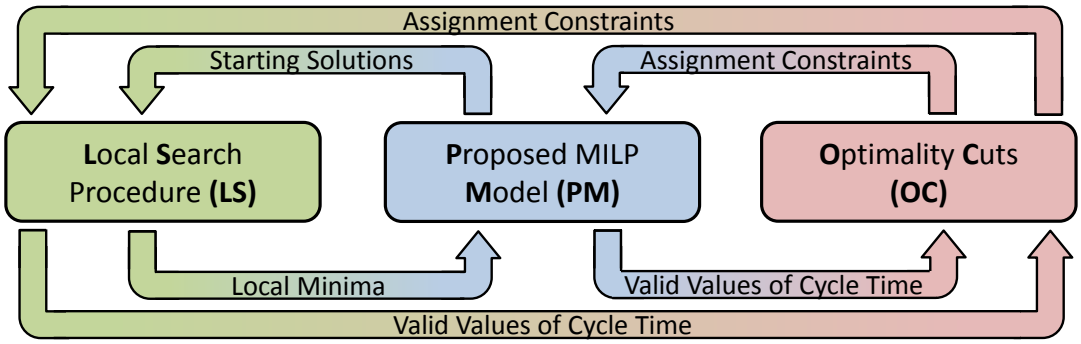



**Figure 1:** Proposed matheuristic flowchart

The interactions between PM, LS and OC (indicated by the labeled arrows in Figure 1) are the following: Whenever PM finds a new incumbent, that solution is set as the next starting solution for LS. Whenever LS finds a new best solution it can also inform PM. Each local *minima* found by LS or new solution found by PM can be tested by longer more exhaustive simulations to verify its expected makespan, $Mkpn_{inc}$. Whenever a new incumbent value of $Mkpn_{inc}$ is obtained, OC can add tighter optimality cuts during execution to PM and can be used to prune neighborhoods of LS. Gurobi, a state of the art solver, is used to solve the mathematical model. The solver offers callbacks, which allow the user to access the incumbent solution whenever one is found and add constraints during execution as "Lazy Constraints". Cplex, another state of the art MILP solver, also enables equivalent procedures.

### 3.1. Mathematical Model

The employed model is based on Lopes et al. [2018]'s formulation for the Sequence-Aware Mixed-Model Assembly Line Balancing Problem: the model considering a set $K$ of possible product sequences $k$, each having a probability parameter $Prob_k$ defined by the product of the demand rates of the $N$ models present in the sequence. Let the parameter set $M_{[p,k]}$ state the model of the $p^{\text{th}}$ piece of sequence $k$. The central binary decision variable $x_{[t,s]}$ is set to one when task $t$ is assigned to station $s$. This binary variable set controls the processing time of each product model $Tx_{[m,s]}$. For each product sequence $k$, the entry and departure times of each piece $p$ (from set $P = \{1, \ldots, N\}$) at/from each station $s$ is controlled by the variables $Tin_{[p,s,k]}$ and $Tout_{[p,s,k]}$, and the cycle time of each sequence $k$ is measured by the variable $C_k$. The proposed model is presented by Expressions 1 to 8.

$$\text{Minimize} \quad \sum_{k \in K} Prob_k \cdot C_k \tag{1}$$

$$\sum_{s \in S} x_{[t,s]} = 1 \qquad \forall\, t \in T \tag{2}$$

$$\sum_{s_1=1}^{s} x_{[t_1,s_1]} \geq \sum_{s_2=1}^{s} x_{[t_2,s_2]} \qquad \forall\, s \in S,\ t_1, t_2 \in T :\ R_{[t_1,t_2]} = 1 \tag{3}$$

https://proceedings.science/p/85554?lang=pt-br

$$\sum_{t \in T} D_{[t,m]} \cdot x_{[t,s]} = Tx_{[m,s]} \qquad \forall\, m \in M,\ s \in S \tag{4}$$

$$Tout_{[p,s,k]} \geq Tin_{[p,s,k]} + Tx_{[M_{[p,k]},s]} \qquad \forall\, p \in P,\ s \in S, k \in K \tag{5}$$

$$Tout_{[p,s-1,k]} = Tin_{[p,s,k]} \qquad \forall\, p \in P,\ s \in S : s > 1, k \in K \tag{6}$$

$$Tout_{[p-1,s,k]} \leq Tin_{[p,s,k]} \qquad \forall\, p \in P : p > 1,\ s \in S, k \in K \tag{7}$$

$$Tout_{[N,s,k]} \leq Tin_{[1,s,k]} + C_k \qquad \forall\, s \in S, k \in K \tag{8}$$

The goal function is defined by Expression 1. The balancing constraints are stated by Equation 2 (all tasks must be performed) and Inequality 3 (strengthened precedence relations by Ritt and Costa [2015]). Equation 4 ties model-station processing times to the balancing variables. Inequalities 5 and 7, and Equation 6 define the discrete asynchronous flow of pieces, and Inequality 8 controls the cycle time of each considered product sequence $k$. The proposed model's rationale is minimizing the expected average stochastic steady-state cycle time by approximating it to a weighted sum of deterministic cyclical sequences cycle times $C_k$. The weights in this sum are the probabilities of occurrence of each sequence $k$, as given by Equation 9 in terms of the occupation rates $O_m$ of each model $m$ in the sequence.

$$Prob_k = \prod_{p \in P} O_{[M_{[p,k]}]} \qquad \forall\, k \in K \tag{9}$$

The proposed formulation is an indirect performance measure for the makespan. Its core rationale is that, when a large number of pieces is produced, the makespan will be dictated by the expected cycle time. This means that $\sum_{k \in K} Prob_k \cdot C_k$ should approximate the realized cycle time ($C^R$). Notice that it is possible to extend this formulation to all possible sequences for the target number of pieces. By minimizing the expected departure time of the last piece at the last station ($Tout_{[N,|S|]}$) weighted by the probabilities of each sequence, one could explicitly minimize the expected makespan exactly [Birge and Louveaux, 2011]. However, the number of sequences grows exponentially with the sequence size ($|K| = |M|^N$) and, consequently, the number of variables and constraints, meaning that directly and explicitly optimizing the makespan of thousands of pieces would be computationally intractable with the proposed model.

### 3.2. Optimality Cuts

The developed optimality cuts are based on a lower bound analysis for the expected cycle time of a solution ($C^R_{sol}$), which equals the expected makespan ($Mkpn_{sol}$) divided by the number of produced pieces. For each station, the expected cycle time can be written as the sum of expected processing times and expected idle times (blockages and starvations). Idle times are always non-negative, and hence, by ignoring them, the lower bound for $C^R_{sol}$ stated by Inequality 10 is defined.

$$\text{LB}\left[C^R_{sol}\right] = \max_{s \in S} \left( \sum_{m \in M} O_m \cdot Tx_{[m,s]} \right) \tag{10}$$

If a trial solution's LB $\left[C^R_{sol}\right]$ is greater than the incumbent $C^R_{inc}$, then the trial solution is necessarily dominated by the incumbent. Any task-station assignment that leads to a higher lower bound for LB $\left[C^R_{sol}\right]$ than $C^R_{inc}$ can, therefore, be discarded. In order to make this evaluation, the mixed-model problem is translated to a virtual single model problem: the processing times $D_t$ of each task $t$ are defined as the weighted (by demand ratios) average of the processing times $D_{[t,m]}$ across models. For each new incumbent values of cycle time realized for the mixed-model ($C^R_{inc}$), tighter values of earliest ($E_{S_t}$) and latest ($L_{S_t}$) stations [Scholl and Becker, 2006] can be defined for each task $t$. This allows the definition of the first optimality cut, stated by Equation 11.

$$x_{[t,s]} = 0 \qquad \forall\, t \in T,\ s \in S : s < E_{S_t}(C^R_{inc}) \cup s > L_{S_t}(C^R_{inc}) \tag{11}$$

https://proceedings.science/p/85554?lang=pt-br

Similarly, by analyzing assignments of pair of tasks to the same station, a second type of optimality cut can be defined. Let $R^*_{[t_1,t_2]}$ represent the generalized precedence matrix obtained by the transitive closure [Gersting, 2006] of the precedence relations in the given precedence matrix $R_{[t_1,t_2]}$. Equations 12, 13, and 14 define the total processing times before ($\Delta B$), between ($\Delta I$), and after ($\Delta F$) each task pair ($t_1$, $t_2$). Inequality 15 states the second optimality cut, applicable when assigning two tasks to the same station $s$ would necessarily imply on LB $\left[C^R_{sol}\right]$ higher than $C^R_{inc}$ before, at, or after station $s$ (Inequalities 16, 17, and 18 respectively).

$$\Delta B_{[t_1,t_2]} = \sum_{t_3 \in T/\{t_1,t_2\}} (R^*_{[t_3,t_1]} \vee R^*_{[t_3,t_2]}) \cdot D_{t_3} \tag{12}$$

$$\Delta I_{[t_1,t_2]} = \sum_{t_3 \in T/\{t_1,t_2\}} (R^*_{[t_1,t_3]} \wedge R^*_{[t_3,t_2]}) \cdot D_{t_3} \tag{13}$$

$$\Delta F_{[t_1,t_2]} = \sum_{t_3 \in T/\{t_1,t_2\}} (R^*_{[t_1,t_3]} \vee R^*_{[t_2,t_3]}) \cdot D_{t_3} \tag{14}$$

$$x_{t_1,s} + x_{t_2,s} \leq 1 \;\; \forall\, s \in S,\; t_1, t_2 \in T \;:\; (16) \vee (17) \vee (18) \tag{15}$$

$$s < \frac{D_{t_1} + D_{t_2} + \Delta B_{[t_1,t_2]}}{C^R_{inc}} \tag{16}$$

$$D_{t_1} + D_{t_2} + \Delta I_{[t_1,t_2]} > C^R_{inc} \tag{17}$$

$$|S| + 1 - s < \frac{D_{t_1} + D_{t_2} + \Delta F_{[t_1,t_2]}}{C^R_{inc}} \tag{18}$$

Violation of constraints (Equation 11 and Inequality 15) imply that $C^R_{sol}$ is higher than $C^R_{inc}$ for the virtual single model problem. Given that such problem does not take into consideration blockages and starvations associated to the original mixed-model problem, its cycle time (and makespan) is a lower bound on the realized cycle time (and expected makespan) of the original problem. Hence, violating these constraints leads to sub-optimality in regard to the current incumbent solution.

### 3.3. Local Search Procedure

The proposed local search procedure combines elements from GRASP and Tabu Search. Initial trial solutions are randomly generated according to a priority rule or provided to the method by the MILP model, when it finds an incumbent. For each pair of adjacent stations $(s, s+1) \in S$, the neighborhood is defined by verifying all the possible ways to re-assign the subset of tasks assigned on the current trial solution to said station pair. The best neighbor to the current trial solution becomes the new trial solution. Each local *minima* is then added to a permanent Tabu list that prevents the repetition of final trial solutions. The local search high-level pseudo-code is presented by Algorithm 1.

The initial solution can either be provided by the mathematical model or be generated by a priority based rule: Assign randomized priorities to tasks; one by one, assign the $i^{th}$ task with highest priority (whose predecessors were already assigned) to the $\lfloor 1 + (i-1) \cdot (|S|-1)/|T| \rfloor^{th}$ station. Unassigning tasks from a station pair consists of removing all tasks assigned to it and defining a sub-problem: how to best assign that subset of tasks to the station pair. A valid re-assignment consists of a task-station allocation for each unassigned task to one station of the pair, in a way that precedence relations and optimality cuts are respected. If OC are used, it is important to only activate them in the local search after the current solution "*sol*" respects the OC. Otherwise, it is possible that starting solutions have no direct neighbors that respect them and, therefore, the algorithm would not perform the neighborhood search.

Lastly, the performance evaluation consists of a weighted sum of results obtained from quick deterministic simulations. Based on the task-station assignments, each solution defines the

https://proceedings.science/p/85554?lang=pt-br

**Algorithm 1** Local Search

**while** Time Limit is not Reached **do**
  $sol \leftarrow$ Get Initial Solution
  $ini \leftarrow null$
  **while** $ini \neq sol$ **do**
    $ini \leftarrow sol$
    $best \leftarrow sol$
    **for all** Adjacent station pair $(s, s+1) \in S$ **do**
      $sol \leftarrow ini$
      Unassign all tasks from $s$ and $s+1$ in $sol$
      **for all** Valid Reassignment $trial_{sol} \notin$ Local *Minima* **do**
        **if** Performance($trial_{sol}$)≥Performance($best$) **then** $best \leftarrow trial_{sol}$
        **end if**
      **end for**
    **end for**
    $sol \leftarrow best$
  **end while**
  Local *Minima*.Add($sol$)
  $C^R_{(inc)} \leftarrow \min\left(C^R_{(inc)}, C^R_{(sol)}\right)$
**end while**

model-station processing times $Tx_{[m,s]}$. For each sequence $k \in K$, an asynchronous line can be simulated by computing departure times from each station by each product piece. The departure times of each piece depend on their processing times and on the times that the previous piece left each station (potential blockages). The time each piece departs from each station $Tout_{[p,s]}$ can be computed as $\max(\max(Tout_{[p,s-1]}, Tout_{[p-1,s]}) + Tx_{[m_p,s]},\ Tout_{[p-1,s+1]})$. By simulating two replications of each sequence, the values of $C_k$ can be estimated by the difference between departures at the last station [Öztürk et al., 2015] of the end of each replication. By weighting these values by $Prob_k$ for each sequence, their sum defines a deterministic approximation for Expression 1.

## 4. Literature Benchmarks

Various goal functions and formulations have been presented to optimize the cycle time of mixed-model assembly lines [Becker and Scholl, 2006; Boysen et al., 2008; Battaïa and Dolgui, 2013]. In this section, four of them will be discussed: horizontal balancing, vertical balancing, maximum sub-cycle time minimization, and truncated makespan minimization.

A common approach to mixed-model balancing consists on smoothing model-wise processing times across stations. This is commonly referred to as Horizontal Balancing, hereafter abbreviated as "HBal". Decker [1993] proposes the use of the maximum $Tx_{[m,s]}$ deviation from the model-wise mean as a measure of horizontal balancing, as expressed by Expression 19. Multiple authors have considered different measures (often non-linear ones) for horizontal balancing [Becker and Scholl, 2006], but they share a similar rationale: homogenize processing times of models across stations.

$$\text{Minimize} \quad \max_{m \in M}\left(\max_{s \in S}\left(Tx_{[m,s]} - \sum_{t \in T}\frac{D_{[t,m]}}{|S|}\right)\right) \tag{19}$$

Another common optimization approach to mixed-model line balancing is based on the station-wise weighted average processing times. Each station's (weighted) average can be seen as a potential bottleneck. By homogenizing said averages, Expression 20 (proposed by Merengo et al. [1999]) seeks to increase line productivity. This rationale is commonly referred to as Vertical Balancing and is hereafter abbreviated as "VBal".

$$\text{Minimize} \quad \sum_{s_1 \in S}\left(\max_{s \in S}\left(\sum_{m \in M} O_m \cdot Tx_{[m,s]}\right) - \sum_{m \in M} O_m \cdot Tx_{[m,s_1]}\right) \tag{20}$$

https://proceedings.science/p/85554?lang=pt-br

Another approach, proposed by Karabati and Sayin [2003] consists of minimizing the Maximum Sub-cycle Time (hereafter abbreviated as "MST") measured by the highest model-station processing time. This performance measure is presented by Expression 21.

$$\text{Minimize} \quad \max_{m \in M} \left( \max_{s \in S} Tx_{[m,s]} \right) \tag{21}$$

Some authors [Sawik, 2012; Öztürk et al., 2015] present Makespan minimization formulations for simultaneous balancing-sequencing variants of this problem. As discussed in Section 3.1, it is impractical to explicitly minimize the expected makespan for a large production target. However, by limiting the analysis to a smaller number of products, it is possible to minimize the truncated expected makespan as stated by Expression 22. Notice that this formulation (hereafter abbreviated as "Mkpn") is similar in structure to the goal function of the proposed MILP model (Expression 1).

$$\text{Minimize} \quad \sum_{k \in K} Prob_k \cdot Tout_{[N,|S|,k]} \tag{22}$$

The proposed performance measure (Secton 3.1) and all herein-above measures are indirect performance measures. As Tiacci [2015] discusses, it is difficult to directly measure throughput of mixed-model assembly lines. The true problem goal is to maximize the line's expected efficiency. The proposed formulation seeks to optimize the expected steady-state. Horizontal balancing (HBal), Vertical balancing (VBal), and Maximum Sub-cycle Time (MST) formulations are also (supposed) means to achieve high and stable throughput. The makespan minimization (Mkpn) formulation would be a direct performance measure if it were possible to explicitly enumerate all possible product sequences. Given that it is impractical to do so, tests are required to verify which formulation produces the best results for the considered problem variant.

## 5. Experiments and Results

Two case studies were conducted to test and compare the proposed formulation and matheuristic to benchmark literature alternatives: Section 5.1 presents a combinatorial case study, which consists on applying these techniques to a dataset proposed by Lopes et al. [2018] which is based on data provided by Otto et al. [2013]. Section 5.2 presents a practical case study based on data from a car-seat assembly line.

### 5.1. Combinatorial Dataset

All the model alternatives described in Section 4 were implemented in the same solver (Gurobi 7.5) and tests were run on the same hardware (an Intel Core i7-3770 CPU 3.40GHz (8 CPUs) 16 GB RAM). The time limit was set to 900 seconds per instance/method. The tested dataset is composed of 350 instances (175 small and 175 large ones). In order to verify the relative importance of each part of the proposed matheuristic procedure (Full), tests were performed using only the proposed mathematical model (PM) and the proposed local search (LS). In total, 2450 instance-method executions were solved, requiring over 230 hours of computational effort. For each instance, the best answer found (by all methods) was set as a reference and deviations were calculated for each method. The number of best solutions found was also counted for each method. If only one method found a best solution, this is counted as an exclusive best solution for that method. For these tests, the parameter $N$ was set as 3. Table 1 summarizes the results of the tests.

These results show that the proposed matheuristic procedure (Full) outperformed the all tested literature alternatives in all considered measures: Number of best solutions found (both total and exclusive), average deviation from best solution, and maximum deviation from best solution. In fact, even when the instance-wise best out of the four benchmarks is considered (Min4), the proposed matheuristic outperforms it by 1.3% in average. Furthermore, by verifying the performance of PM and LS, it is possible to infer that these methods generate synergies when combined: the results of the Full matheuristic evidence that it is better than the sum of its parts (PM+LS considers the best answer out of PM or LS for each instance). For example, the total number of best

https://proceedings.science/p/85554?lang=pt-br

**Table 1:** Results summary for the combinatorial dataset case study with 2450 executions

Average and Maximum Deviations from the Best Solution

| Dataset | Method | Proposed Matheuristic | | | | Previous Benchmark alternatives | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Deviation | Full | PM | LS | PM+LS | MST | Vbal | Hbal | Mkpn | Min4 |
| Small | Max. | 0.1% | 2.8% | 0.3% | 0.3% | 8.6% | 23.1% | 12.3% | 8.8% | 4.9% |
| | Avg. | 0.001% | 0.4% | 0.01% | 0.01% | 3.0% | 10.9% | 4.6% | 3.2% | 1.8% |
| Large | Max. | 0.5% | 2.6% | 0.8% | 0.7% | 5.2% | 20.8% | 11.1% | 6.1% | 3.5% |
| | Avg. | 0.01% | 0.4% | 0.08% | 0.1% | 1.0% | 10.8% | 2.3% | 2.5% | 0.8% |
| Both | Avg. | 0.01% | 0.4% | 0.04% | 0.03% | 2.0% | 10.8% | 3.4% | 2.8% | 1.3% |

Number of Best Solutions Found

| Dataset | Method | Proposed Matheuristic | | | | Previous Benchmark alternatives | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Number | Full | PM | LS | PM+LS | MST | Vbal | Hbal | Mkpn | Min4 |
| Small | Exclusive | 8 | 0 | 5 | - | 0 | 0 | 0 | 0 | - |
| | Total | 169 | 52 | 165 | 167 | 2 | 0 | 1 | 1 | 4 |
| Large | Exclusive | 53 | 4 | 4 | - | 12 | 0 | 1 | 0 | - |
| | Total | 154 | 37 | 99 | 109 | 12 | 0 | 1 | 1 | 14 |
| Both | Exclusive | 61 | 4 | 9 | - | 12 | 0 | 1 | 0 | - |
| | Total | 323 | 89 | 264 | 276 | 14 | 0 | 2 | 2 | 18 |

answers found by the Full matheuristic (323) is higher than the sum of those found by PM and LS (276). Furthermore, by separating the results between small and large instances it is possible to verify that this synergy was stronger for larger instances. A Wilcoxon test [Wilcoxon, 1945] confirms that these result differences are not symmetrically distributed around zero, with a z-score of 4.15 (p-value $< 0.01\%$). Hence, this test corroborates the synergy claims.

This superior result quality reflects the success of the proposed formulation to represent stochastic steady-state: By adding scheduling variables, the proposed formulation is capable of approximating stochastic steady-state better than alternative rationales: low maximum processing times (MST), station-wise weighted averages (Vbal) and model-wise differences (Hbal) are all desirable features, but not goals in themselves. These are only supposed means to achieve high and stable throughput. Furthermore, the matheuristic outperformed the truncated Mkpn by 2.8% in average, demonstrating that balancing a line to minimize the expected stochastic makespan of few pieces is not the same as optimizing the stochastic makespan of thousands of pieces. In the later case, the (difficult to estimate) stochastic steady-state is much more relevant.

Further tests were performed in order to verify the influence of the optimality cuts (OC): both PM and LS were executed again on one in every five instances, both with and without OC. The average differences in results is reported by Table 2. While OC did not lead to significantly better answers (less than 0.1% difference in average for both PM and LS) it did lead to significant speed-ups: The OC accelerated PM's time to optimality by 9.66%, and reduced its final integer gap by 4.2%. Notice that optimality and gap refer here to the mathematical model described in Section 3.1, which is an approximation of the stochastic problem. OC also accelerated LS: the number of local *minima* found was increased by 94.1%. Furthermore, with OC, the time for LS to reach the same answer was reduced by 33.6%. These tests demonstrated synergies between OC and both optimization techniques: the mathematical model (PM) and local search procedure (LS).

**Table 2:** Average Improvements due to Optimality Cuts

| Mathematical Model - PM | |
|---|---|
| Solution Quality ($CT^R_{inc}$) | 0.08% |
| Time to Optimality | 9.66% |
| Integer Gap Reduction | 4.20% |

| Local Search Procedure - LS | |
|---|---|
| Solution Quality ($CT^R_{inc}$) | 0.02% |
| Time to Same Solution | 33.6% |
| Found Local *Minima* | 94.1% |

## 5.2. Practical Case Study

The proposed model (presented in Section 3.1) and the literature alternatives (described in Section 4) were also applied to data from a real mixed-model car-seat assembly line located on

https://proceedings.science/p/85554?lang=pt-br

the outskirts of Curitiba-PR (Brazil). Two product models are produced in that assembly line: a more common and simpler one (with demand rate of 84.4%) and a less common more complex one (with demand rate of 15.6%). The problem's main practical constraints are: some tasks are fixed to specific stations due to heavy machinery that cannot be moved, and other tasks are restricted in assignments to a subset of stations that have the necessary equipment to perform them. These two constraint types, when combined to the precedence relations, lead to significant search-space reductions. This line also had been previously optimized by the cyclical sequence aware formulation described by Lopes et al. [2016]. That work used the most likely cyclical product sequence (5-1), which might be a reasonable approximation, but does not necessarily reflect the demand rates: 84.4% and 15.6% reflect a demand rate of approximately 5.41 to 1. In order to enhance the case study, Lopes et al. [2016]'s model was applied to both the (5-1) and (6-1) most likely cyclical sequences.

Each formulation was applied to the line and the balancing solution for each method was tested via simulation. A total of 5000 strictly different random seeds were used to define product sequences (scenarios with 50000 products each), and each method's solution was simulated for all scenarios. This approximated the expected makespan, and, therefore, the realized average cycle times (makespan divided by the number of produced pieces) for stochastic product sequences. The average, minimum, maximum, standard deviation result values, and CPU time required are summarized in Table 3. Naturally, both for PM and Mkpn when higher values of $N$ are used, higher CPU time requirements are expected.

**Table 3:** Realized Cycle Time values with 5000 different scenarios - Practical Case Study

| Type of Formulation / Method | | Mean | Std. Dev. | Min. | Max. | Time (s) |
|---|---|---|---|---|---|---|
| Scheduling Unaware Formulations | MST | 2038.3 | 0.16 | 2037.3 | 2038.8 | 0.1 |
| | Hbal | 1829.1 | 1.10 | 1824.6 | 1833.1 | 0.2 |
| | Vbal | 1669.5 | 3.01 | 1657.6 | 1680.0 | 4.0 |
| Truncated Makespan Minimization with product sequences of size $N$ | $N$=2 | 1624.3 | 2.24 | 1615.1 | 1632.8 | 1.5 |
| | $N$=3 | 1647.0 | 1.96 | 1639.1 | 1654.5 | 1.1 |
| | $N$=4 | 1642.5 | 1.90 | 1634.9 | 1649.9 | 2.8 |
| | $N$=5 | 1635.4 | 1.82 | 1628.0 | 1642.6 | 7.9 |
| | $N$=6 | 1647.0 | 1.39 | 1641.3 | 1652.8 | 10.5 |
| | $N$=7 | 1625.6 | 1.60 | 1619.0 | 1632.6 | 28.5 |
| | $N$=8 | 1625.6 | 1.60 | 1619.0 | 1632.6 | 112.9 |
| | $N$=9 | 1625.6 | 1.60 | 1619.0 | 1632.6 | 559.1 |
| Single Most Likely Cyclical Sequences | (5-1) | 1632.7 | 2.10 | 1624.0 | 1641.6 | 8.0 |
| | (6-1) | 1678.0 | 2.33 | 1668.6 | 1686.9 | 3.2 |
| Proposed Stochastic Steady-State Model with product sequences of size $N$ | $N$=2 | 1619.0 | 2.89 | 1607.1 | 1629.7 | 7.4 |
| | $N$=3 | 1606.9 | 2.46 | 1596.9 | 1616.6 | 6.0 |
| | $N$=4 | 1598.0 | 2.30 | 1588.3 | 1607.2 | 4.2 |
| | $N$=5 | 1594.6 | 2.26 | 1585.0 | 1603.9 | 11.5 |
| | $N$=6 | 1593.8 | 2.25 | 1584.3 | 1603.2 | 8.9 |
| | $N$=7 | 1593.8 | 2.25 | 1584.3 | 1603.2 | 28.3 |
| | $N$=8 | 1593.8 | 2.25 | 1584.3 | 1603.2 | 41.6 |
| | $N$=9 | 1593.8 | 2.25 | 1584.3 | 1603.2 | 157.2 |

The obtained results reinforce the advantages of the proposed formulation: In average, minimum and maximum cases, the proposed model obtained better solutions than all alternatives. By explicitly considering scheduling possibilities and weighting their performance by the probabilities of each scenario, the model performs better than the literature alternatives and better than simply considering any of the most likely cyclical sequences.

Furthermore, as the number of replications increases, the proposed representation becomes more accurate, leading to better results: Notice that the maximum value obtained by the proposed formulation with $N \geq 6$ was better than those obtained with smaller values of $N$. Notice that this consistency was not observed by the truncated Mkpn formulation (For which the best answer was obtained with $N = 2$), reinforcing an advantage of the proposed formulation: If more

https://proceedings.science/p/85554?lang=pt-br

computing power is added so that larger values of $N$ are tractable, then the proposed formulation can offer a more accurate representation of the stochastic steady-state. This also led to a more "robust" solution: notice that PM's ($N \geq 6$) worse performance (1603.2) is still better than the best performance of any previous benchmark (1615.1 by Mkpn with $N = 2$). Furthermore, the difference (11.9) is over five times higher than the standard deviation (2.25) of PM's solution, hence, it is statistically significant.

## 6. Conclusions

This paper presents a matheuristic for makespan optimization of stochastic asynchronous mixed-model assembly lines. These lines are product-oriented production layouts that are often subject to product sequence uncertainties (make-to-order environments): the model of each product that enters the line is a memoryless random variable, whose probabilities are given by market demand ratios. The goal is to minimize the makespan of a large number of products (e.g. 50000), which is heavily tied to the stochastic steady-state behavior. The matheuristic combines a mixed-integer programming model, a local search procedure, and search-space reduction optimality cuts to generate solutions for the problem. The proposed goal functions for the model and local search are approximations to the stochastic steady-state that consider all possible product sequences up to a certain size, and weight their performance by the probability of each sequence. The search-space reduction optimality cuts are valid in the sense that solutions that violate them are necessarily dominated by the current incumbent.

Tests performed on a 350-instance dataset showed that solutions found by the proposed matheuristic outperformed four literature benchmark models using the same computing time limit, hardware and software conditions. The matheuristic outperformed each of the benchmarks by at least 2% in solution quality. If, for each instance, the matheuristic is compared to the best out of the four benchmarks, it still outperforms them by 1.3% in average. This highlights the matheuristic success of both representing stochastic steady-state and minimizing the expected makespan.

The parts of the matheuristic (mathematical model, and local search procedure) were also executed separately. This allowed to verify that the proposed matheuristic was better than the sum of its parts: solutions reached by the full proposed matheuristic were better than the best solution found by its MILP model and local search separately in 61/350 instances with up to 0.7% improvements. Tests verified that the proposed optimality cuts contributed to both the mathematical model and the local search procedure: the model's average integer gap was 4.2% smaller, and time to optimality was 9.7% smaller in average; the local search time to reach the same solution was reduced by 33.6% and the number of local *minima* found was increased by 94.1%. Hence, a triple inter-synergy was demonstrated to operate in the matheuristic: Each of the employed techniques combines their strengths and compensate the weaknesses of one another.

A practical case study was conducted, verifying anew the better performance of the proposed formulation. This case study shows that considering larger product sequences improves the quality of the stochastic steady-state representation obtained by the proposed formulation. Furthermore, multiple (5000) independent simulations verified the performance superiority of the solution found by the proposed formulation: The worst performance of the best solution found by the matheuristic's model was statistically significantly better than the best performance of all solutions found by previous benchmarks in the practical case study.

Further works should seek to explore potential formulation symmetries that allow speed-ups of the proposed mathematical model and the local search procedure. Further computational tests should also verify the impact of buffers on how to best distribute tasks on stochastic asynchronous mixed-model assembly lines.

## Acknowledgments

The authors thank the financial support from Fundação Araucária (Agreement 041/2017 FA – UTFPR – RENAULT), and CNPq (Grants 406507/2016-3 and 307211/2017-7).

https://proceedings.science/p/85554?lang=pt-br

## References


Battaïa, O. and Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2):259–277. ISSN 09255273.

Becker, C. and Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694–715. ISSN 03772217.

Ben-Tal, A., Ghaoui, L. E., and Nemirovski, A. (2009). *Robust Optimization*. Princeton University Press, Princeton, 1st edition. ISBN 9780691143682.

Birge, J. R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer, New York, 2nd edition. ISBN 9781461402367.

Boysen, N., Fliedner, M., and Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 111(2):509–528. ISSN 09255273.

Boysen, N., Fliedner, M., and Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349–373. ISSN 03772217.

Bukchin, J., Dar-el, E. M., and Rubinovitz, J. (2002). Mixed model assembly line design in a make-to-order environment. *Computers and Industrial Engineering*, 41:405–421.

Decker, M. (1993). Capacity smoothing and sequencing for mixed-model lines. *International Journal of Production Economics*, 30-31:31–42. ISSN 09255273.

Gersting, J. L. (2006). *Mathematical Structures for Computer Science*. W. H. Freeman, 7th edition. ISBN 9780716768647.

Karabati, S. and Sayin, S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, 149(2):417–429. ISSN 03772217.

Lopes, T. C., Sikora, C. G. S., and Magatão, L. (2016). Buffer and Cyclical Product Sequence Aware Assembly Line Balancing Problem: Model and Steady-State Balancing Case Study. In *Annals of the XLVIII SBPO*, p. 3458–3469, Vitória-ES, Brazil.

Lopes, T. C., Sikora, C. G. S., Michels, A. S., and Magatão, L. (2018). Mixed-Model Assembly Line Balancing with Given Buffers and Product Sequence: Model, Formulation Comparisons and Case Study. *Annals of Operations Research*, 1(1):1–26.

Merengo, C., Nava, F., and Pozzetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835–2860. ISSN 1366588X.

Otto, A., Otto, C., and 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(1):33–45. ISSN 0377-2217.

Öztürk, C., Tunali, S., Hnich, B., and Örnek, A. (2015). Cyclic scheduling of flexible mixed model assembly lines with parallel stations. *Journal of Manufacturing Systems*, 36(3):147–158. ISSN 02786125.

Ritt, M. and Costa, A. M. (2015). Improved integer programming models for simple assembly line balancing and related problems. *International Transactions in Operational Research*, 1:1–15. ISSN 14753995.

Sawik, T. (2012). Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. *International Journal of Production Research*, 50(18):5017–5034. ISSN 00207543.

Scholl, A. (1999). *Balancing and sequencing assembly lines*. Physica, Heidelberg, 2nd edition.

Scholl, A. and Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666–693. ISSN 03772217.

Tiacci, L. (2015). 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. ISSN 09255273.

Wilcoxon, F. (1945). Individual comparisons by ranking. *Biometrics Bulletin*, 1(6):80–83.


https://proceedings.science/p/85554?lang=pt-br