Annals of Operations Research accepted manuscript.
DOI: 10.1007/s10479-017-2711-0

Mixed-Model Assembly Lines Balancing with Given Buffers and Product Sequence

Model, Formulation Comparisons, and Case Study

Thiago Cantos Lopes · Celso Gustavo Stall Sikora · Adalberto Sato Michels · Leandro Magatão

Abstract Asynchronous assembly lines are productive layouts in which products move sequentially between stations when processing at current station is complete, and the following station is empty. When these conditions are not verified, downstream starvations and upstream blockages can occur. Buffers are often employed to minimize these problems, which are particularly relevant when the line is shared between a set of different products models (mixed-model lines). If the sequence of such models is cyclical, a steady-state production rate is eventually reached. However, determining (and, therefore, optimizing) such steady-state is challenging. This led to the development of indirect performance measures for mixed-model lines by many authors. In this paper, a direct performance measure is presented with a mixed-integer linear programming (MILP) model and compared to previous formulations. The model is also applied to a practical case study and to a new dataset (with 1050 instances), allowing general assertions on the problem. All instances are solved with a universal solver and solutions are validated with a simulation software. Tests on the dataset instances confirmed the observations made on the case study: the proposed formulation produced solutions with higher production rate in 82% of the instances and tied the remaining ones, not being outperformed a single time. A triple interdependency of task balancing, product sequencing, and buffer allocation is demonstrated. Cyclical schedules show how buffers are able to compensate differences between models across stations and lead to the conclusion that the propagation of differences of models between stations can generate scheduling bottlenecks (blockages and starvation).

The authors would like to thank the financial support from Fundação Araucária - Agreements 141/2015, 06/2016 and 041/2017 FA-UTFPR-RENAULT, and CNPq (grant 406507/2016-3)

L. Magatão
Federal University of Technology - Paraná (UTFPR)
Graduate Program in Electrical and Computer Engineering (CPGEI)
Av. Sete de Setembro, 3165, Curitiba-PR, Brazil, 80230-901
Tel.: +55-41-3310-4701
E-mail: magatao@utfpr.edu.br

Keywords Mixed-Model Assembly Line Balancing · Cyclical Steady-State Optimization · Buffers · Asynchronous Lines · Performance Measures

1 Introduction

Assembly lines are product oriented layouts, in which products (or pieces) move through the workstations on the line in a sequential manner. In each workstation, a set of tasks is performed before the next workstation's tasks are allowed to begin. An important decision problem related to managing such lines is to balance the distribution of tasks amongst stations (Scholl and Becker, 2006). Common goals for this problem include minimizing the number of workstations required for a given production rate (Type-1) and maximizing the production rate for a given number of workstations (Type-2). This paper will focus on a Type-2 variant of such problems in which a cyclical product sequence and a set of finite buffers are given as parameters. This paper is an extended version of the work presented by Lopes et al (2016) to which the authors have added comparisons to further literature formulations and performance measures for mixed-model assembly lines, as well as a more general study on a new dataset.

Assembly lines are often not dedicated exclusively to a single product, but rather shared by a set of products (Scholl, 1999). A common example is found in assembly lines of automotive factories with different vehicle models. Lines with more than one product model can be further classified as in two different types (Boysen et al, 2008): if set-up times between models are significant and must be taken into account, significant size batches of each product go through the line in an alternated fashion, and the problem is called a multi-model assembly line; If those set-up times are small and negligible, models flow through the line in a more mixed fashion, such line is called a mixed-model assembly line. This work will focus on the later type.

Authors approach mixed-model balancing differently (Becker and Scholl, 2006): some (Thomopoulos, 1970; Merengo et al, 1999) state that it is important to minimize differences between model processing times across stations (station smoothing, horizontal balancing), others (Merengo et al, 1999; Matanachai and Yano, 2001; Pastor et al, 2002) state that it is best to equalize average workloads between stations allowing more differences between models (vertical balancing). Bukchin (1998) presents comparisons between several performance measures for mixed-model balancing. These formulations are indirect goals, and were developed because measuring the line throughput directly is a challenge.

Assembly lines can be further classified in terms of the line flow control (Boysen et al, 2008): paced lines (continuous) have a conveyor belt that moves all pieces constantly and together; In unpaced synchronous lines, all products move together, but discretely one station forward when all pieces have completed processing at their current stations; In unpaced asynchronous lines, pieces move discretely and almost independently: each piece can move when processing is completed at its current station and the next station is available. This work will focus on the later type of line control, which offers interesting possibilities and challenges linked to the unpaced flow of pieces in and out of stations and buffers: On the one hand, products might be blocked (when they are done processing, but the next station is occupied) and stations starved (when a piece leaves the station and the previous

station has not completed processing). On the other hand, the independent flow of pieces can help compensate the differences between models processing times.

Blockages and starvations produce disturbances on the assembly line that must be taken into account. If the product sequence is cyclical, the system will converge towards steady-state. However, such state is not trivially determinable due to the aforementioned disturbances. Furthermore, if the line starts empty, transient-stage disturbances further complicate such determination. Figures 1 and 2 compare the steady-state cyclical behavior to the transient behavior on a two-model (flowing through the line on a 5-1 pattern) assembly line with seven stations and single unitary buffers between them[1]. Notice that the replications of the piece set behave (six pieces, five of model 1 and one of model 2) identically in Figure 1, and differently in Figure 2: The behavior is stable on the cyclical schedule and unstable on the transient one, in particular on the first station and buffer: idle times occur at the first station due to downstream blockages, which are mainly originated at the second and third workstations.

The optimization of a cyclical mixed-model flow-shop consists in combining traditional simpler problems, namely line balancing (first studied by Salveson (1955)), model sequencing (first modeled by Bard et al (1992)), and buffers allocation (first studied by Koenigsberg (1959)). It has been repeatedly stated that, if possible, it is best to attempt to deal with these aspects simultaneously (Sawik, 2000; Boysen et al, 2008, 2009b) in order to achieve better final results. Nevertheless, these problems are often dealt with independently as some reviews for mainly balancing (Battaïa and Dolgui, 2013), sequencing (Boysen et al, 2009b), buffer allocation (Demir et al, 2014), and cyclical scheduling (Levner et al, 2010) suggest. Most authors focus separately on variations of each of this problem's aspects. Some examples: Scholl et al (2009); Gurevsky et al (2012); Kellegöz (2016) present variations of balancing problems, Leu et al (1997); Heath et al (2013); Golle et al (2015) present variations of sequencing problems, and Karabati and Kouvelis (1994); Spinellis and Papadopoulos (2000); Gurgur (2013) present variations of buffer allocation problems. Karabati and Kouvelis (1994), in particular, reinforced the previously mentioned interconnection between these problems by showing that buffer allocation based exclusively on sequence-independent information often leads to suboptimal results.

Nonetheless, some authors combine these problems, mostly two at a time, either using decomposition strategies or (meta)heuristic procedures (Battini et al, 2009; Özcan et al, 2010; Tiacci, 2015). In general, however, most balancing models leave buffers and product sequences out of the picture, whereas in most sequencing models as well as in most cyclical flow-shop models, processing times are considered parameters. There are, however, some noteworthy exceptions: Sawik (2004) presents a formulation for balancing and scheduling pieces in an assembly system in which stations are not necessarily serial. Öztürk et al (2013) presented a model for mixed-model assembly lines that linked balancing and sequencing, on (possibly buffered) asynchronous lines with a makespan minimization goal, but did not take the cyclical aspect into account. Sawik (2012) presents formulations that compare makespan results of different scheduling strategies. Öztürk et al (2015) seek to optimize steady-state results, but also use a makespan minimiza-

[1] The schedules illustrated by Figures 1 and 2 are generated by the optimization procedures discussed at Section 3.2.2.

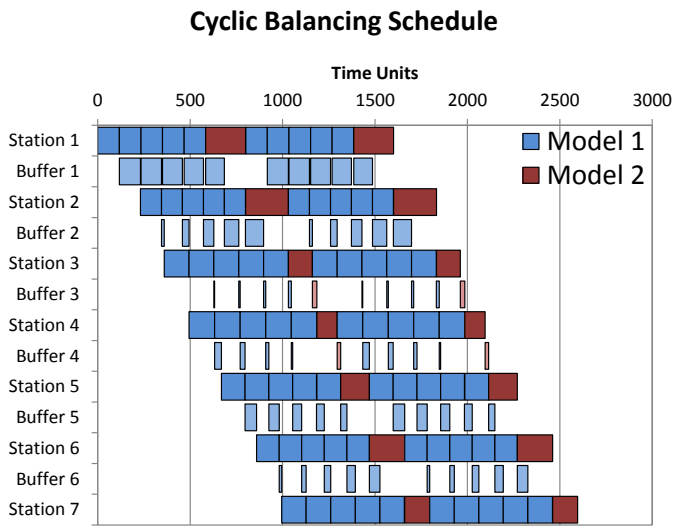

Fig. 1: Cyclical schedule representing steady-state: Darker colors represent processing times and lighter colors represent waiting times. Buffer and station behavior are identical in both replications. Source: Lopes et al (2016).

tion goal for a series of replications of the *minimal part set* (*MPS*). The minimal part set is defined as the smallest set of pieces that can be repeated indefinitely to reach a production target: if one needs to produce 5000 units of product A and 1000 units of product B, the *MPS* is five units of product A and one unit of product B. These makespan minimization approaches (Sawik, 2012; Öztürk et al, 2015) only represent the transient-state for the first n cycles and are not guaranteed to properly predict the steady-state behavior due to the aforementioned disturbances (blockages, starvations, and empty line start). The other previously mentioned approaches (station smoothing, horizontal balancing, and vertical balancing) are indirect performance measures, and are not guaranteed to optimize cyclical steady-state either.

Thus far, mixed-model balancing models still fail to properly represent steady-state. There are many different approaches which argue their specific (and different) goal functions will also lead to efficiency maximization. This might not always be the case in a flow shop cyclical schedule, either due to aforementioned transient effects (as the line starting "empty"), or due to inherent disturbances associated with asynchronous lines (blockages and starvations). Moreover, a deeper investigation of single unitary buffer might offer interesting insights on steady-state behavior and convergence, as suggested by Figure 2.

In this paper, goal function formulations applied on mixed-model assembly line models and previously described in the literature are discussed and compared to the authors' proposed cyclical steady-state formulation (Lopes et al, 2016). Buffer allocation and a cyclical product sequence are given as parameters, and the goal is to optimize the cyclical steady-state taking such aspects into account. Tests reported herein provide evidences that these features should be considered

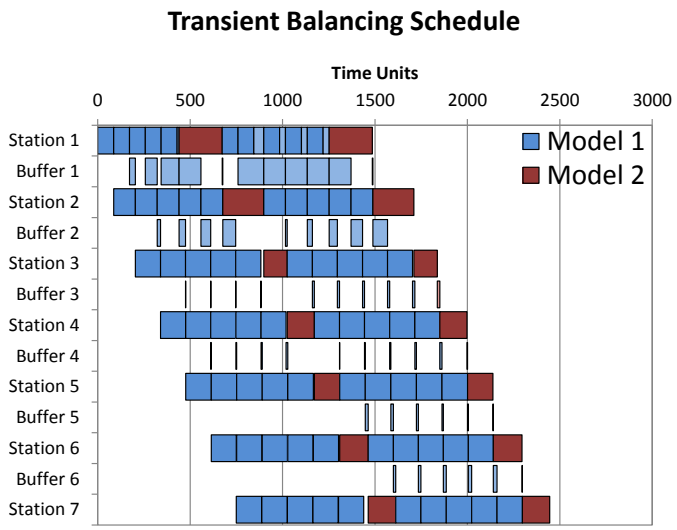

Fig. 2: Transient schedule representing initial state: Darker colors represent processing times and lighter colors represent waiting times. Buffer and station behavior are different in each replication. Source: Lopes et al (2016).

simultaneously. The problem at hand is a *buffer-and-cyclical-product-sequence-aware* mixed-model assembly line balancing problem. The presented model can be iteratively used to compare different options of layouts and cyclical sequences (some variations are tested in the case study). The model developed by the authors is presented and a case study tests such model with data from a real-world assembly line located on the outskirts of Curitiba-PR (Brazil). Section 2 presents the base formulation from Lopes et al (2016), along with the extensions required to compare it to previous performance measures. Section 3 presents the results of applying both the proposed formulation and different goal functions to the case study's data, insights drawn from such tests are discussed. Furthermore, a new dataset is presented and tested to verify the generality of such insights. Section 4 summarizes the main conclusions drawn from this paper, and provides directions for further works.

2 Mathematical Model

In order to describe the *buffer-and-cyclical-product-sequence-aware* problem, a mathematical model developed with mixed-integer programming is employed. The proposed model is based on the following assumptions:

1. Tasks are indivisible and performed on the same stations for all product models.
2. There are precedence relations between some tasks, reflecting technological constraints.
3. Some of them might be restricted to a subset of stations, reflecting practical constraints (i.e. the problem is assignment bounded).

4. The product sequence and buffer layout are given, and the product sequence cycles indefinitely.
5. Transportation times between stations are small and can be ignored.
6. Products flow asynchronously and sequentially in the line between stations.
7. The goal is to balance the line in a way that maximizes its efficiency, minimizing the average *steady-state* cycle time.

The following sections describe the parameters, variables and constraints of the model developed based on these assumptions.

2.1 Parameters, Sets and Variables

The developed mathematical model is a variation of mixed-model assembly line balancing models (Scholl, 1999). Therefore, it contains a set of basic elements, such as tasks, precedence relations, stations, and models (product's type). The mathematical model uses as parameter the product sequence of the $|P|$ pieces in the global set (batch). In other words, there is a fixed sequence of products that repeats cyclically.

Table 1: Nomenclature: Parameters, sets and variables. The domains D on which each set is defined are also presented.

Parameter Sets$_{[D]}$	**Element**	**Meaning**
S	s	Station s, ranging from 1 to $\|S\|$ (number of stations).
S_b	s_b	Buffer set, $S \supset S_b$, lists all buffers s_b
P	p	Piece p, ranging from 1 no $\|P\|$ (number of pieces).
T	t	Task t, ranging from 1 to $\|T\|$ (number of tasks).
M	m	Model m, ranging from 1 to $\|M\|$ (number of models).
$n_{[M]}$	n_m	Number of pieces of each model, $\sum_m n_m = \|P\|$.
$m_{[P]}$	$m_{[p]}$	Model m of the p-th piece in the sequence.
$d_{[T,M]}$	$d_{[t,\ m]}$	Duration (time units, T.U.) of each task t for each model m.
TT	$(t_1,\ t_2)$	Set for precedence relations between tasks, each pair $(t_1,\ t_2)$ defines a precedence relation.
F	$(t,\ s)$	Indicates that the task t can be assigned to the station s, if s is a buffer then no task can be assigned to it.
Variable Sets$_{[D]}$	**Type**	**Meaning**
$TS_{[F]}$	Binary	1 if task t is assigned to station s, 0 otherwise
$Tin_{[P,S]}$	Real+	Time (T.U.) at which the p-th piece enters the station s
$Tx_{[P,S]}$	Real+	Processing time (T.U.) for the p-th piece in the station s
$Tout_{[P,S]}$	Real+	Time (T.U.) at which the p-th piece leaves the station s
CT_{PX}	Real+	Steady-state cycle time, according to the proposed formulation.
$px_{[M,S]}$	Real+	Processing time of model m at the station s
$Px_{[S]}$	Real+	Weighted average processing time at the station s
$TAx_{[M]}$	Real+	Theoretical average processing time of the model m

Some practical constraints mean that not every task t can be assigned at every station s. In order to describe these constraints, the set F is employed: F lists

all tuples (t, s) that represent feasible task-station allocations. Each such tuple indicates that the task t can be assigned to the station s. The set F serves two purposes: First, it allows buffers to be represented as stations to which no task can be assigned to. Second, it allows practical features to be incorporated into the model (e.g., some tasks were fixed to stations due to heavy equipment that could not be moved). Table 1 lists all the names and interpretations of parameters, variables, and sets employed in the model.

2.2 Constraints

The developed mathematical model uses a balancing core to control processing times for pieces in stations. The balancing core of the model follows a standard multi-model set up with assignment bounded occurrence and precedence constraints defined for the TS variable set, the feasible $(t,\ s)$ tuples on F, and the precedence $(t_1,\ t_2)$ tuples on TT. Equation 1 establishes the assignment bounded occurrence constraint, while Inequality 2 models the precedence relations. Notice that F must be defined in a way that if a particular station s_b is a buffer, then no task can be assigned to it. In other words, the set F contains no tuple $(t,\ s_b)$ with buffer stations $s_b \in S_b$.

$$\sum_{(t,\ s)\ \in\ F} TS_{[t,\ s]} \ =\ 1 \qquad \forall\, t\ \in\ T \tag{1}$$

$$\sum_{(t_1,\ s)\ \in\ F} s \cdot TS_{[t_1,\ s]} \ \leq \sum_{(t_2,\ s)\ \in\ F} s \cdot TS_{[t_2,\ s]} \qquad \forall\, (t_1,\ t_2)\ \in\ TT \tag{2}$$

The balancing core variables are also employed to determine the processing time $Tx_{[p,\ s]}$ of each piece p in each station s. This is done by combining the information of the piece p's model m_p with that of the duration parameters $d_{[t,\ m]}$ in time units of each task t for each model m. Equation 3 ties the piece-station processing times with the model's balancing core.

$$Tx_{[p,\ s]} \ =\ \sum_{(t,\ s)\ \in\ F} d_{[t,\ m_{[p]}]} \cdot TS_{[t,\ s]} \qquad \forall\, p\ \in\ P,\ s\ \in\ S \tag{3}$$

The processing time is then employed to bind together the model's last two variable sets, namely $Tin_{[p,\ s]}$ and $Tout_{[p,\ s]}$. These variables carry the time at which each piece p enters (Tin) and leaves ($Tout$) the station s. There are three logical constraints required to properly tie these variable sets. First, a piece can only leave a station after it's been processed (stated by Inequality 4).

$$Tout_{[p,\ s]} \ \geq\ Tin_{[p,\ s]} + Tx_{[p,\ s]} \qquad \forall\, p\ \in\ P,\ s\ \in\ S \tag{4}$$

Second, when a piece leaves one station it enters the next one, as stated by Inequality 5. Notice that this constraint can be easily adapted to include movement times between stations, if such times are deemed relevant.

$$Tin_{[p,\ s]} \ =\ Tout_{[p,\ s-1]} \qquad \forall\, p\ \in\ P,\ s\ \in\ S,\ s > 1 \tag{5}$$

Third, a piece may only enter one station after the previous piece has left said station as stated by Inequality 6. This constraint prevents two pieces from simultaneously occupying the same station.

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

Notice that, while the first and second time constraints tie each p^{th} piece's variables to its other variables, the third one tie each p^{th} piece's variables to its predecessor, the $(p-1)^{th}$ piece. In order to make a model that represents a steady-state operation, it is necessary to also tie the last piece ($|P|$) to the first one (1).

The proposed measure of steady-state mean cycle time CT_{PX} ties the first and last piece together as the gap between them is linked to the flow shop's average production rate. This cycle time is bounded by throughput time for each station s, as stated by Inequality 7. This bound is the adapted version of Inequality 6: it ties the end of one set of products to the beginning of the following set: just as the piece p is tied to the previous piece $p-1$, the piece 1 ties to the last piece (of a "previous" set) $|P|$, taking into account the shop's average steady-state cycle time: CT_{PX}.

$$|P| \cdot CT_{PX} \ + \ Tin_{[1,\ s]} \ \geq \ Tout_{[|P|,\ s]} \qquad \forall\ s \ \in \ S \tag{7}$$

This station-wise bound for steady-state cycle time is scheduling aware, because it takes into account scheduling variables. Ignoring such variables, it is possible to imagine stations operating at a theoretical maximum efficiency (never waiting for a piece to enter or leave it) as a bound to steady-state behavior. Under this consideration, each station can be seen as a potential bottleneck that bounds the maximum average steady-state cycle time. Another way of saying it is: Even though the average processing time does not define the steady-state cycle time, it does inferiorly bound it (in a scheduling unaware manner). This bound is stated by Inequality 8.

$$|P| \cdot CT_{PX} \ \geq \sum_{(t,\ s)\ \in\ F,\ p\ \in\ P} d_{[t,\ m_p]} \cdot TS_{[t,\ s]} \qquad \forall\ s \ \in \ S \tag{8}$$

As this bound for cycle time does not take scheduling variables into account, it can be referred to as "LB-CT". Being unaware of product sequences and buffers, this lower bound merely defines an inferior limit to steady-state, rather than being the key to steady-state behavior.

Lastly, an initialization constraint can be added for the first piece at the first station: the entry time of the first piece in the first station is set to zero, as stated by Equation 9.

$$Tin_{[1,\ 1]} \ = \ 0 \tag{9}$$

2.3 Objective Function

As stated by Tiacci (2015), realistic line features (such as mixed-models) make throughput difficult to estimate directly. This leads to multiple authors employing an indirect performance measure (Thomopoulos, 1970; Bukchin, 1998). However, the problem's true goal is to maximize productivity (throughput), or equivalently, to minimize the average steady-state cycle time. In this paper, a measure of such

cycle time is proposed (CT_{PX}), and the herein-above formulation seeks to minimize it, as stated by Expression 10.

$$\text{Minimize} \quad CT_{PX} \tag{10}$$

The assumption here is that the number of pieces to be processed is large and the only relevant goal is to optimize the steady-state behavior. The proposed measure for that behavior is given by Inequality 7. Previously described indirect performance measures proposed by other authors are presented in Section 2.4, their results are discussed in Section 3.2. A more detailed comparison with the makespan minimization goal (Sawik, 2012; Öztürk et al, 2015) is discussed on Section 3.2.2.

In these following sections, the proposed formulation, described by Expressions 1 to 10, is referred to as PX, the realized value of steady state cycle time as CT, and the proposed measure CT_{PX}. These notation differences are needed, as an actual comparison between CT and CT_{PX} is only discussed in Section 3.1.2.

2.4 Alternative Goal Functions and Performance Measures

As discussed in Section 1, different authors have employed different goal functions when approaching mixed-model balancing. Two variables are defined in terms of the balancing decision variable: The processing time of each model m at each station s can be determined by adding the task times of the tasks performed on it, as stated by Equation 11.

$$px_{[m,s]} = \sum_{(t,\ s)\ \in\ F} d_{[t,\ m]} \cdot TS_{[t,\ s]} \qquad \forall\ m\ \in\ M,\ s\ \in\ S \tag{11}$$

Second, the weighted average processing time can be determined by taking into account the demand ratio of each model, as stated by Equation 12.

$$Px_s = \sum_{m\ \in\ M} \frac{n_m}{|P|} \cdot px_{[m,s]} \qquad \forall\ s\ \in\ S \tag{12}$$

Last, a parameter TAx_m is defined for each model m. It states the theoretical minimum average processing of each model. Such value is calculated in accordance to Equation 13.

$$TAx_m = \sum_{t\ \in\ T} \frac{d_{[t,m]}}{|S|} \qquad \forall\ m\ \in\ M \tag{13}$$

The first objective to be compared to the proposed formulation is the "Station Smoothing" measure. This objective was proposed by Thomopoulos (1970) and stated by Equation 14. The goal is to minimize the fluctuation of processing times at stations. This goal function will be referred to as SX.

$$\text{Minimize } SX = \sum_{m\ \in\ M} n_m \cdot \sum_{s\ \in\ S \setminus S_b} |TAx_m - px_{m,s}| \tag{14}$$

The second objective is often called "Vertical Balancing" (hereafter referred to as VX) as proposed by Merengo et al (1999) and stated by Equation 15. The

idea behind this goal is to establish more balanced total workloads across stations, allowing more differences between models. This goal function minimizes the differences between the weighted average processing times.

$$\text{Minimize } VX \;=\; \sum_{s \,\in\, S \setminus S_b} \left(\max_{k \,\in\, S \setminus S_b} Px_k - Px_s \right) \tag{15}$$

Vertical Balancing (Equation 15) argues that the main relevant aspect of mixed-model balancing is the weighted average processing times at stations. Its rationale is: "If the average processing times are similar across stations, no workstation will behave as an important bottleneck". Vertical balancing does not take the differences between models within a station directly into account.

The third objective is commonly called "Horizontal Balancing" (hereafter referred to as HX). The idea behind this goal is to establish more balanced workloads across models, allowing more differences between stations. This goal function minimizes the differences between processing times of each model and the largest processing time in each station. The formulation presented by Equation 16 was proposed by Merengo et al (1999).

$$\text{Minimize } HX \;=\; \sum_{s \,\in\, S \setminus S_b} \frac{\sum_{m \in M} (\max_{k \in M} px_{[k,s]} - px_{[m,s]}) \cdot n_m}{|P| \cdot \max_{m \in M} px_{[m,s]}} \tag{16}$$

Horizontal Balancing (Equation 16) argues that the main relevant aspect is how model's processing times differ in each workstation. Its rationale is: "If models all have similar processing times at each station, product sequence matters less and productivity is less dependent on it". Horizontal balancing does not take weighted average processing times directly into account.

One should note that there are other many variants of horizontal balancing and vertical balancing proposed by other authors (Matanachai and Yano, 2001; Pastor et al, 2002). They are, however, similar in nature to the presented ones.

The fourth alternative goal function considered is the makespan minimization variant discussed in Section 3.2.2, proposed by Sawik (2012); Öztürk et al (2015). This approach consists on taking two replications of the *MPS* into account and minimizing the completion of the last piece, as stated by Equation 17. To the best of the authors' knowledge, prior to this work, this was the most sophisticated balancing goal that uses scheduling information.

$$\text{Minimize } MX \;=\; Tout_{[|P|,|S|]} \tag{17}$$

A last performance measure is hereafter indicated, a probabilistic estimate first presented by Dar-El et al (1999). It consists on computing the probability of each model being the bottleneck at each station and multiplying this probability by the processing time of said model at said station. First, the variable β compares each model at each station, as stated by Equation 18.

$$\beta_{[m_1,s_1,m_2,s_2]} \;=\; \begin{cases} 1 & \text{if } px_{[m_1,s_1]} > px_{[m_2,s_2]} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Next, the probability of each model m_1 at station s_1 having a higher load than station s_2 is computed. Let that probability be stated by $prob1_{m_1,s_1,s_2}$, as presented by Equation 19.

$$prob1_{m_1,s_1,s_2} = \sum_{m_2 \in M} \frac{n_{m_2} \cdot \beta[m_1, s_1, m_2, s_2]}{|P|} \tag{19}$$

The product of these probabilities in every station represents the probability of model m being the line's bottleneck at station s. That probability ($prob2_{[m,s]}$) is stated by Equation 20.

$$prob2_{[m,s]} = \prod_{s_k \in S, s_k \neq s} prob1_{[m,s,s_k]} \tag{20}$$

At any given time, the probability of a model being in a station can be estimated by its occupation rate. This allows one to finally define the probabilistically expected cycle time ($P.E.$) as stated by Equation 21. Notice that this formulation assumes that processing times at stations are the root source of bottlenecks. This is further discussed in Section 3.

$$P.E. = \sum_{s \in S} \sum_{m \in M} \frac{n_m}{|P|} \cdot prob2_{[m,s]} \cdot px_{[m,s]} \tag{21}$$

This probabilistic estimate assumes a random sequence of products (Dar-El et al, 1999), in the problem at hand, the product sequence is known a priori. Furthermore, the non-linearity present at Equation 20 makes it difficult to incorporate this formulation to the main MILP model presented in Section 2.2. Therefore, the $P.E.$ measure will only be compared to the experienced steady-state results of each formulation.

3 Results and Discussions

The proposed model is applied to a data from an automobile seat assembly line on the outskirts of Curitiba-PR (Brazil), extending the case studies presented by Lopes et al (2016). The problem has *seven workstations* and *two product models* with relative demands of approximately *five to one*. The most common model is simpler and its tasks required, in average, less time. Some tasks are fixed, reducing the problem's degrees of freedom (such fixed tasks are controlled by the F set). Problem data (task processing times, precedence relations, and task-assignment possibilities), along with solution tuples generated by the proposed formulation, is available at the *Supporting Information* for reproducibility purposes.

3.1 Case Study Description

The case study is centered on a car seat assembly line that produced two product models: a simpler one and a more complex one with higher processing times in average. Demand rates are stable in a rate of 5 units of the simpler product model to 1 unit of the more complex one. Due to internal reasons of the factory, such as a relatively constant internal demand for both products, the cyclical product

sequences could be set in one of two configurations: Either the more complex product type is produced every six pieces (on a 5-1 pattern), or a batch of five units is produced every thirty pieces (on a 25-5 pattern). Intermediary configurations are regarded as confusing due to the loss of regularity and larger batches are deemed unpractical due to internal demands of the factory. The company also wanted to evaluate the impact of buffers: initially, the line did not have any internal storage. Company specialists suspected that a buffer between the second and third workstations would be very useful, but wanted to compare that it to installing buffers between every two stations.

The case study aimed at **establishing the best steady-state results as for given product sequences and buffer layouts**. By comparing multiple scenarios, one can verify the influence of sequencing and buffer allocation on solution quality. Two different product sequences with the same demand rate are considered: The first one (S1) is a "Mixed-Model" sequence, in that products alternate more frequently and, therefore, product interfaces are more relevant. The second one (S2) is a "Multi-Model" sequence, in that products alternate less often and interfaces are less relevant. Notice that despite the label "Multi-Model", set-up times between models are assumed to be negligible in both cases. Figure 3 illustrates the two cyclical sequences that are tested.

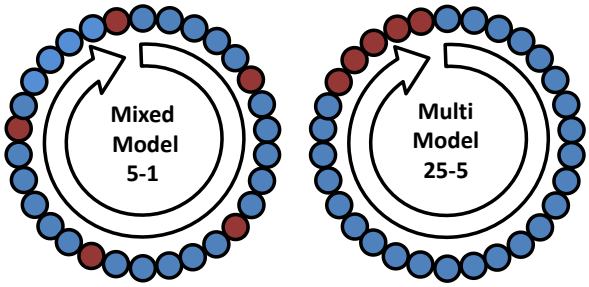

Fig. 3: "Mixed-Model" (S1) and "Multi-Model" (S2) sequences.

Three different buffer layouts are considered to test the influence of internal storage: The first one (L1) is a layout with no buffers between stations; The second one (L2), an intermediary layout with only one single buffer between the second and third stations; The last one (L3), a fully buffered line with one buffer between each station pair. The buffer layouts are illustrated by Figure 4. Notice that all buffers are single unitary buffers: each can only hold one piece at a time.

The combination of the two sequences (S1, S2) with the three layouts (L1, L2, L3) generate six scenarios, one for each layout under each product sequence (e.g. S1L1, S2L1, S1L2, ...).

3.1.1 Case Study Results

For each of the generated scenarios, the MILP model described in Section 2 is employed to generate a balancing solution that optimizes the steady-state. The optimal balancing solution generated for each scenario is applied to all other scenarios

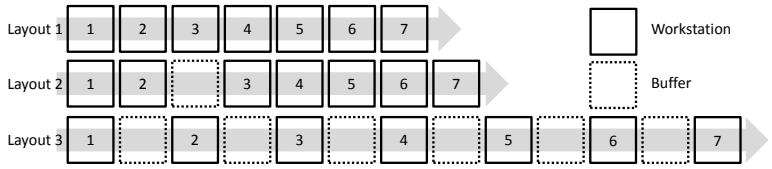

Fig. 4: Buffer layouts: L1, L2, and L3.

in order to verify how well solutions perform in configurations different than those they are designed to optimize. This gave rise to for a total of thirty-six cases. These are converted to model instances and solved to optimality using a universal solver (IBM ILOG CPLEX, available at *www.ibm.com*). Table 2 presents the realized values of steady-state cycle time for each one of the 36 combinations. Boldfaced values refer to steady-state behavior of each scenario with the optimal balancing obtained for that scenario. Numbers discussed in the text are highlighted in *italic* fonts.

Notice that for every scenario (line i) in Table 2, the best answer is the one obtained by the balancing designed to optimize it (column i). However, balancing solutions (column j) can behave both better and worse in scenarios other than the ones they are designed to optimize (line j): For instance, when the S1-L2 solution is applied to S1-L3, the realized cycle time (142.68) is better than the one obtained by applying it to S1-L2 (143.87), but not as good as the one designed to S1-L3 applied to S1-L3 (133.48). Also, while the solutions with lowest steady-state cycle time also have low values of LB-CT (Inequality 8), some cases with low LB-CT have high steady-state cycle times (for instance: S2-L3, with LB-CT of 135.48, applied to S1-L1 realized a stable CT of 168.45). This confirms an expected behavior: low values of LB-CT are **required** for low values of steady-state cycle time, but they are **not sufficient**.

Table 2: Values of CT_{PX} for every scenario (rows) when the optimal balancing solutions from every other scenario (columns) is applied to them.

			Balancing from Sequence and Layout					
			Mixed-Model Seq (5-1): S1			Multi-Model Seq (25-5): S2		
			L1	L2	L3	L1	L2	L3
Applied to	S1	L1	**156.15**	166.33	*172.20*	*165.20*	*163.55*	*168.45*
		L2	155.28	***143.87***	152.52	155.78	155.78	152.35
		L3	153.20	*142.68*	***133.48***	140.53	140.53	135.48
	S2	L1	158.65	159.85	157.48	***149.02***	149.02	154.62
		L2	155.36	155.28	152.87	144.75	**144.75**	150.09
		L3	153.20	151.96	*146.14*	140.53	140.53	**135.48**
LB-CT			153.20	142.68	133.48	140.53	140.53	*135.48*

Solutions designed for buffered layouts tend to behave poorly when applied to unbuffered layouts: S1-L3 has the best steady-state behavior (133.48), however, when its balancing is applied to S1-L1, it displays the worst steady-state behavior

(172.20). Furthermore, Table 2 shows that buffer layouts can influence the optimal sequencing decisions: For L1, the sequence that offers the best steady-state behavior is S2 (149.02), but for L2 and L3, the sequence S1 outperforms S2 (143.87 and 133.48).

Cases S2-L1 and S2-L2, in particular, generated very similar answers: When applied to all scenarios other than S1-L1 (in which S2-L2 (163.55) outperformed S2-L1 (165.2)), both generated equal answers. This demonstrates, that for some scenarios, there might exist **multiple optimal** balancing answers: two different balancing solutions provided the same steady-state cycle time for five out of six scenarios.

Table 3 presents the (model-station) processing times of all six solutions. Notice that the best solutions for the fully buffered cases (S1-L3 and S2-L3) display steady-state cycle times lower than the highest station-wise processing times of individual models. These cases also have very similar cycle times despite the different cyclical sequences: it may also be unintuitive how five pieces of model M2 (with much higher processing times in some stations than M1) can go through the line (Sequence S2) with only a minor impact on steady-state cycle time: How can single buffers compensate those differences? In order to further clarify how this occurs, steady-state schedules are presented for thirty pieces of both the S1-L3 case (Figure 5) and the S2-L3 case (Figure 6).

Table 3: Processing times for each model, at each station, for each layout and each cyclical sequence.

	Mixed-Model Sequence - S1								
	Layout 1			Layout 2			Layout 3		
Station	M_1	M_2	LB-CT	M_1	M_2	LB-CT	M_1	M_2	LB-CT
1	121.0	231.7	139.45	105.1	187.3	118.80	116.3	218.1	133.27
2	110.5	216.0	128.08	104.6	**255.5**	129.75	113.6	**231.8**	133.30
3	121.9	108.0	119.58	136.6	139.5	137.08	134.7	127.0	133.42
4	126.7	120.9	125.73	130.9	141.1	132.60	**138.2**	109.6	133.43
5	129.2	127.7	128.95	136.9	137.7	137.03	128.6	155.8	133.13
6	**139.8**	130.6	138.27	135.0	136.9	135.32	122.0	190.9	**133.48**
7	137.1	**233.7**	**153.20**	**137.1**	170.6	**142.68**	132.8	135.4	133.23

	Multi-Model Sequence - S2								
	Layout 1			Layout 2			Layout 3		
Station	M_1	M_2	LB-CT	M_1	M_2	LB-CT	M_1	M_2	LB-CT
1	123.5	**217.5**	139.17	123.5	**217.5**	139.17	117.7	**222.9**	135.23
2	125.3	216.0	140.42	125.3	216.0	140.42	118.8	216.0	135.00
3	126.3	175.7	134.53	126.3	165.8	132.88	123.7	127.6	124.35
4	127.7	77.3	119.30	127.7	87.2	120.95	126.4	169.6	133.60
5	**128.0**	144.0	130.67	**128.0**	144.0	130.67	132.9	141.7	134.37
6	127.8	132.9	128.65	127.8	132.9	128.65	**135.7**	132.9	135.23
7	127.6	205.2	**140.53**	127.6	205.2	**140.53**	131.0	157.9	**135.48**

Notice that in Figure 5 the buffer utilization changes as the minimal part set goes through the line. Buffer 1 fills after the first piece is processed at Station 1

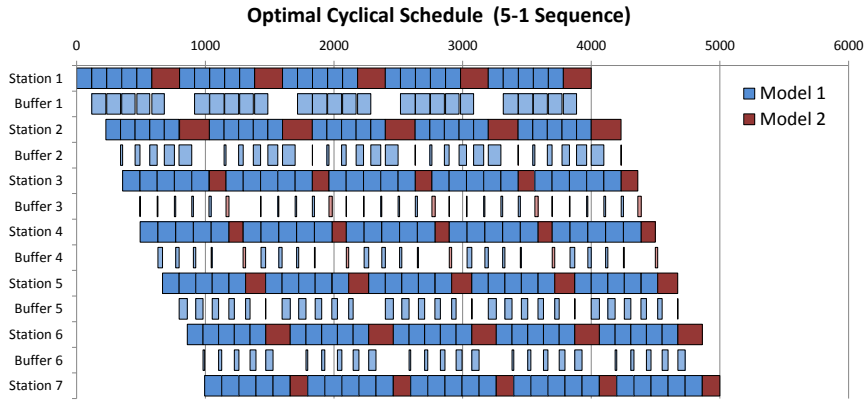

Fig. 5: Steady-state schedule for 30 pieces under the "Mixed-Model" cyclical sequence. Darker colors represent processing times and lighter colors represent waiting times.

and stays full until the model two piece arrives. The second buffer is used initially for a small time, but this duration increases as the *MPS* passes through the line. Naturally, every time a full *MPS* passes through the system, the occupation returns to the initial configuration. This is required for the schedule to be representative of the steady-state behavior.

Figure 6 further clarifies how buffers reduce blockages and starvations caused by model differences (as shown by Table 3): buffers also display an **evolving pattern** of relative utilization. The main difference to the S1-L3 scenario is how incremental these changes are: the first five buffers in the S2-L3 case slowly build up the relative utilization as M1 pieces pass through the line. Buffer 6, however, displays the opposite behavior: the buffer decreases relative utilization as M1 pieces are produced and increases it as M2 pieces are produced.

Notice that the third station on the S2-L3 case has significantly longer idle times than any station in the S1-L3 case. This is expected as its average processing time (124.35, indicated in Table 3) is much lower than the cycle time (135.48, indicated in Table 2), one could try to see this as result of bad balancing. However, the fact that this is an optimal steady-state schedule for this scenario means that no other balancing can produce less unproductive times. For example, applying S1-L3 (that has a LB-CT of 133.48, indicated in Table 2) to S2-L3 leads to a steady-state cycle time of 146.14.

Lastly, by running the presented model, with Inequality 8, but without Inequality 7, a global value of LB-CT is achieved. This bound is 133.48 time units, which matched the value of the S1-L3 (Table 3) case. This means that a **finite number of buffers** allowed the system to reach the **maximum theoretical productivity**.

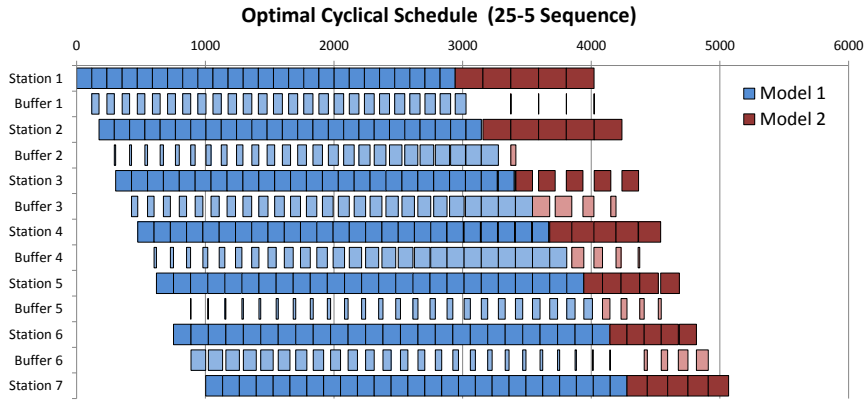

Fig. 6: Steady-state schedule for 30 pieces under the "Multi-Model" cyclical sequence. Darker colors represent processing times and lighter colors represent waiting times.

3.1.2 Result Validation via Simulation

The optimal steady-state cycle times predicted by the proposed model are validated using a simulation software (Simio, available at *www.simio.com*). The product sequence and processing times are deterministic and, therefore, the results are deterministic. As a result, a detailed statistical analysis of the data is not required, contrary to stochastic cases (Alexander et al, 2010). The predicted values for average steady-state cycle time matched perfectly with the steady-state conditions observed in the simulation (i.e. $CT = CT_{PX}$). Figure 7 shows the convergence behavior of the simulation for the S1-L3 case, reproducible with the input data shown in Table 3. Notice that model 2 takes three cycles to achieve steady-state, while model 1 takes only two. Model 1's initial cycle (138.2) time is not only higher than its final one (132.8) but also slightly higher than model 2's final one (136.9). Figure 7 demonstrates that the time between pieces at the last station (and, therefore, the cycle time) **gradually converges** towards steady-state (and, therefore, towards the steady-state cycle time). The simulation allows the verification of an interesting behavior: the initial cycle times of both models (138.2, and 311.1) are higher than the one verified at the steady-state for both models (132.8, and 136.9).

These simulations confirmed that the difference between the realized cycle time and the value of LB-CT can be explained by the blockages and starvation disturbances, which happen cyclically due to the cyclical product sequence. Naturally, the simulation model required a warm-up period as the initial emptiness of the line meant that the first pieces took longer to go through the system, as shown by Figure 7. Furthermore, as the first pieces passed, buffers are not yet able to compensate differences between models as well as they do in steady-state. This meant that the first couple of replications of the *MPS* are not necessarily representative of the steady-state, and that one cannot always measure model-wise steady-state cycle time based on the behavior of such early replications. Steady-state behav-

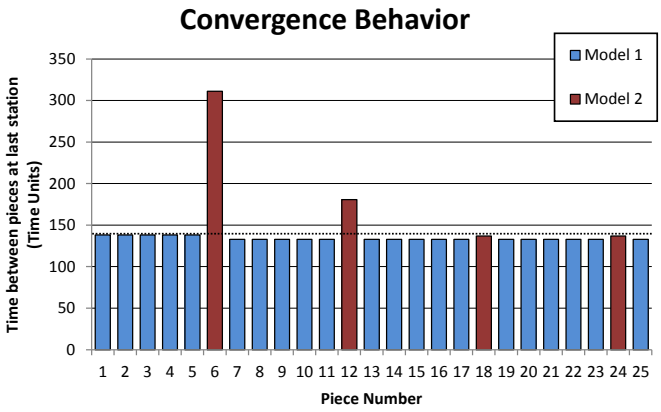

Fig. 7: Convergence behavior (scenario S1-L3) verified via simulation. After the third replication, steady-state is achieved. Source: Lopes et al (2016).

ior (in this case time between pieces output) can be "better" than transient-state behavior, despite the naturally expected (and verified) higher *time in system* of later pieces. This "**empty line effect**" is a disturbance associated with the initial transient-state and is the key to explain the convergence behavior.

3.2 Comparisons to Other Formulations

In order to verify how well the different formulations behave, each goal function is implemented and their answers tested for steady-state behavior. Tests are performed for each buffer layout and each cyclical product sequence presented in the main case study described in Section 3.1. This section is subdivided in three parts: First, Section 3.2.1 will present and discuss the results obtained by the **scheduling unaware** literature formulations. Second, Section 3.2.2 will compare the proposed formulation to the makespan minimization (**scheduling aware**) formulation described in the literature. Last, Section 3.2.3 will evaluate how well the literature described **probabilistic estimate** of cycle time behaves in regard to the **experienced cycle time** of the tested cases.

3.2.1 Scheduling Unaware Formulations

Out of the alternative goal functions presented in Section 2.4, three do not employ scheduling variables (e.g., $Tin_{[p,\ s]}$ or $Tout_{[p,\ s]}$), namely Station Smoothing (SX), Horizontal Balancing (HX) and Vertical Balancing (VX). The MX goal function is the subject of Section 3.2.2.

Three composed variants are also tested: First, $VX + SX$ is the goal function that originates from adding VX to SX. Second, $HX + CE$ employs the same goal function as HX and the additional constraint stated by Inequality 22, binding the

processing time value of each model at each station to an upper bound value[2]. Last, $HX + CA$ is similar to $HX + CE$ but employs the additional constraint stated by Inequality 23, binding only the weighted average processing time by an upper bound value[3].

$$px_{[m,s]} \leq MaxProcTime \qquad \forall\, s \in S,\ m \in M \tag{22}$$

$$Px_s \leq MaxWProcTime \qquad \forall\, s \in S \tag{23}$$

The main difficulty presented by Equation 16 is the non-linearity associated with the division by the variable $px_{[m,s]}$. In the case studies, an iterative process is employed: A trial solution is used, and the values of $\max_{m \in M} px_{[m,s]}$ are set as parameters for the next execution. Each variable $px_{[m,s]}$ is then limited by the value of $(\max_{m \in M} px_{[m,s]})$ multiplied by $(1 + k)$. Where k is a parameter whose starting value is 0.2 and decreased (by 0.01) over the iterations. This led to solutions with progressively lower values of HX. Once $k = 0$ is reached, HX reached a local minimum value. This procedure is not guaranteed to generate optimal solutions in regard to HX, but did generate better results (lower values of HX) than the initial trial solution, justifying its use.

Table 4: Realized cycle times obtained for each goal function, including the proposed formulation (PX).

Seq.	Layout	SX	VX	HX	SX+VX	HX+CE	HX+CA	MX	PX
S1	L1	167.28	172.20	364.47	170.23	171.68	177.62	156.73	156.15
	L2	154.70	154.55	364.47	154.70	171.68	158.37	143.87	143.87
	L3	145.00	134.57	364.47	141.33	170.27	158.37	138.93	133.48
S2	L1	154.35	159.55	364.47	151.42	179.83	187.48	149.02	149.02
	L2	149.95	154.94	364.47	147.01	174.15	182.38	144.75	144.75
	L3	146.31	148.41	364.47	142.64	170.27	170.79	136.38	135.48

Table 4 compares the observed steady-state cycle time of each formulation to those obtained by the proposed formulation and those obtained by the makespan minimization formulation, for a total of 48 cases. By comparing the base formulations (SX, VX and HX) it is rather clear that horizontal balancing is outperformed by both the station smoothing and the vertical balancing approaches. In the buffered cases (Layout 3 in particular), VX reached solutions comparable to the much more sophisticated scheduling aware formulations (MX and PX): VX applied to S1-L3 obtained a result (134.57) between PX's one (133.48) and MX's one (138.93).

Table 5 presents the processing times obtained on optimal solutions (or local minimum in HX's case). Notice how HX presents an interesting problem: Some stations have average processing times substantially higher (over five times) than

[2] The value 250 time units is employed based on the optimal processing time from answers obtained by other formulations

[3] The value 160 time units is employed based on observed cycle time values of other formulations

Table 5: Processing times of solutions obtained according to each formulation, for each model at each station.

	Station Smoothing - SX			Vertical Balancing - VX			Horizontal Balancing - HX		
Station	M_1	M_2	LB-CT	M_1	M_2	LB-CT	M_1	M_2	LB-CT
1	126.0	184.6	135.77	119.9	200.4	133.32	95.2	95.2	95.2
2	126.4	**238.0**	**145.00**	110.5	**247.3**	133.3	**317.4**	**599.8**	**364.47**
3	**132.1**	160.9	136.90	134.2	129.2	133.37	41.9	42.1	41.93
4	123.7	129.2	124.62	**138.2**	109.6	133.43	29.4	29.3	29.38
5	126.6	146.3	129.88	128.6	155.8	133.13	94.9	94.7	94.87
6	123.8	144.7	127.28	122	190.9	**133.48**	150.1	149.6	150.02
7	127.6	164.9	133.82	132.8	135.4	133.23	157.3	157.9	157.4

	Composed - SX+VX			Composed - HX+CE			Composed - HX+CA		
Station	M_1	M_2	LB-CT	M_1	M_2	LB-CT	M_1	M_2	LB-CT
1	126.0	206.6	139.43	122.1	229	139.92	105.7	102.9	105.23
2	126.4	**216.0**	**141.33**	98.8	**248.1**	123.68	115.2	**374.2**	158.37
3	**132.1**	160.9	136.90	83.5	83.6	83.52	83.5	83.6	83.52
4	123.5	128.6	124.35	89.3	89.3	89.3	134.3	136	134.58
5	126.2	153.9	130.82	169.2	169.3	169.22	**151.6**	152.6	151.77
6	124.4	155.4	129.57	153.1	178.7	157.37	142.8	142.7	142.78
7	127.6	147.2	130.87	**170.2**	170.6	**170.27**	153.1	176.6	**157.02**

others. This is due to the fact that the goal function only measures the difference between models. HX **concentrates** processing times **differences** in a single station, which is the most loaded one. This is not a coincidence: Analyzing Equation 16 one can notice that differences between processing times count less if they occur in loaded stations. This means HX tends to intentionally load a station with very high processing time in one model and concentrate the differences between models as much as possible in there: Indeed, aside from that station, processing times are very similar across models. Vertical balancing (VX) presents solutions with better potential as the **average processing times** are **lower**, but the realization of said potential depends heavily on scheduling, product sequence, and buffer layout. In scenarios without buffers, VX produced steady-state behaviors very distant from optimal behavior. Station smoothing (SX) presents model-wise processing times that are roughly **stable across stations**, but worse values of LB-CT than VX. The SX solution realizes better values of cycle times than VX in four out of six cases. However, in the S1-L3 case, the better processing time distribution of VX outperforms SX substantially (134.57 versus 145).

In most cases the presented composed variants ($SX+VX$, $HX+CE$ and $HX+CA$) did outperform the base variants. However, the reason for the improvement in each case ought to be further discussed. The main problem with HX is its tendency to accumulate high processing time in one station. This means that HX is less capable of accumulating processing time in one station when combined to either CE or CA. The $VX+SX$ combination, on the other hand, only outperforms both SX and VX in the S2 cases. In four cases $VX + SX$ outperforms SX and in other four it outperforms VX. In that sense, $VX + SX$ seems to combine

the formulations strengths. One should note, however, that this is not enough to optimize steady-state as both MX and PX produce better steady-state results (than $VX + SX$) in all scenarios: this indicates that, in some cases **neither low average** processing times **nor smoothed processing times** across stations **are guaranteed to lead**, on their own, **to steady-state efficiency**.

In all scenarios, the **scheduling unaware** formulations are **dominated by *PX***, indicating a weakness of these formulations: Not being able to explicitly take into account the product sequence and buffers.

3.2.2 Goal Function Comparison and MPS Replications

As discussed in Sections 1 and 2, makespan minimization (Sawik, 2012; Öztürk et al, 2015) for some *MPSs* is not necessarily the same as steady-state optimization. This is confirmed by Table 4, which shows that for three cases (S1-L1, S1-L3, and S2-L3) the realized cyclical behavior is worse when applying the makespan minimization goal (for two *MPS* replications) than when applying the proposed formulation. It is expected, however, that as more replications are taken into account these two goals are likely to **converge** towards the same results. In order to test that hypothesis, the scenario with most significant differences between the results of each formulation (S1-L3) is tested for up to forty *MPS* replications. The results are summarized by Table 6.

Table 6 shows the realized values of both steady-state cycle time (CT) and makespan (MX), when minimizing both PX and MX. For one replication, 1611.7 is the minimal makespan. PX obtains 1752.6, meaning the initial behavior is better when minimizing MX. However, the realized values of CT when minimizing makespan are higher than those obtained by the proposed formulation (144.28 versus 133.48), meaning the stable behavior is worse when minimizing MX.

These tests show that the proposed formulation produces better steady-state results, but longer makespan for the n first replications. However, as the number n of considered replications is increased, the difference in both makespan and steady-state behavior between formulations decrease. Notice, however, that **the proposed formulation achieves the best steady-state behavior without the need to consider multiple replications** of the piece set. This confirms the previously mentioned hypothesis, but does not explain the results on its own: Why do makespan minimization formulations produce different results for different numbers of replications?

Figure 1 and Figure 2 in Section 1 offer further insights as they present the cyclical schedules for optimal answers of the proposed formulation and the makespan minimization formulation, respectively. As expected, the makespan is smaller on Figure 2, but the steady-state result is worse (about 3% higher average cycle time, as presented by Table 6). Notice how the replications of the *MPS* do not display the same behavior if the makespan minimization formulation is employed.

The probable explanation for this behavior difference lies in the transient-state disturbances, or "empty line effect" (also verified in the simulation model as the key to the convergence behavior): just as in simulation models, the line is started "empty", and this can be exploited by the makespan minimization goal. This gain might, however, come at a cost for the steady-state behavior: makespan minimization generates a dispute between exploiting the "empty line effect" and optimizing

Table 6: Comparison between makespan and realized steady-state cycle time (CT) reached for the S1-L3 scenario by the proposed formulation (PX) and makespan minimization formulation (MX).

Goal:	Makespan Minimization (MX)			Proposed Formulation (PX)		
Values:	MX	CT	CPU Time	MX	CT	CPU Time
No. Rep.						
1	1611.7	144.28	5”	1752.6	133.48	6”
2	2445.3	138.93	6”	2596.8	133.48	8”
3	3277.7	137.37	7”	3397.7	133.48	9”
4	4101.1	137.07	9”	4198.6	133.48	9”
5	4923.0	136.62	12”	4999.5	133.48	11”
10	8954.1	133.62	30”	9002.0	133.48	18”
20	16971	133.62	2’54”	17011	133.48	48”
40	33005	133.62	7’23”	33029	133.48	1’20”

how well the pieces flow through the line when such effect does not exist. This is also concluded by Öztürk et al (2015), who discussed the advantages of considering multiple product replications. The proposed formulation (PX), however, is **insensitive** to the empty line effect, and achieves steady-state optimization without having to subject itself to multiple replications of the piece set.

In industrial practical cases, the number of pieces to be produced might be on the order of thousands (5000 pieces of model 1, 1000 pieces of model 2). If that is the case, not only the behavior of the initial pieces is negligible, but it might also be unpractical to optimize makespan for so many pieces simply due the large size of the problem, measured in number of variables and constraints: As the number of replications increased, so did the required CPU time to solve both models (see Table 6). In that case the best goal is clearly to optimize cyclical behavior, which can be performed by the proposed model with one MPS replication, regardless of the total number of pieces to be produced. Notice that, as shown by Table 6, even with forty MPS replications the makespan minimization formulation does not reach the optimal steady-state behavior. This reinforces the contribution of the proposed formulation.

If the number of produced pieces is smaller, one could argue that it’s interesting to take advantage over the “empty line effect” to reduce the makespan. However, this effect can often be unreliable: Factories usually continue in one day the work with pieces from the previous day and, therefore, pieces that are already in the line make the “empty line effect” (observed in both the simulation model and in the makespan minimization model) inexistent as the previous day’s pieces tend to slow down the first replications of the current workpieces. If that is the case, even for smaller total number of produced pieces it might be best to optimize steady-state behavior instead of the makespan, even if one could take the total number of replications into account.

3.2.3 Probabilistic Estimate

This section discusses the probabilistic estimate of cycle time $P.E.$ introduced by Dar-El et al (1999). The values of $P.E.$ are calculated for each solution of the

proposed method and also of those obtained by the scheduling unaware formulations. Then, P.E. values are compared to the realized cycle time those solutions offered. Table 2 and Table 4 present the values of six steady-state results for each of the 12 solutions. Such minimum and maximum values of realized cycle time are compared to the probabilistic estimate of that each solution in Table 7, which summarizes results from 72 cases.

Table 7: Best and worst steady-state cycle times for each solution, compared to the corresponding probabilistic estimate. Boldfaced values are higher than the estimate.

Solution	Min	Max	P.E.	Solution (PX)	Min	Max	P.E.
SX	145.00	**167.28**	165.60	S1-L1	153.20	158.65	177.44
VX	134.57	172.20	172.50	S1-L2	142.68	166.33	170.20
HX	364.47	364.47	364.47	S1-L3	133.48	**172.20**	172.07
SX+VX	141.33	**170.23**	164.77	S2-L1	140.53	165.20	167.00
HX+CE	170.27	179.83	192.37	S2-L2	140.53	163.55	168.67
HX+CA	158.37	187.48	193.21	S2-L3	135.48	**168.45**	158.20

Notice how, for almost every case, the estimate is conservative: higher than the realized cycle time. This could lead one to suppose that it can be used as an upper bound for steady-state solution quality. However, there are problems with that line argument: First, this estimate is based on random product sequence; Second, the measure is insensitive to buffers capacity of assisting scheduling and, therefore, the correspondence between realized cycle time and the probabilistic estimate is not high. Lastly, in some cases ($SX + VX$, S1-L1, for instance) the estimate is better than the realized result. All these "pathological results" occurred for the same scenario: S1-L1. In order to clarify why this happened, one of these "pathological" cases ($SX + VX$) is compared to a "healthy" one (S1-L1) in, respectively, Figure 8 and Figure 9. Some insights on the reasons why this occurred: In Figure 8, the bottleneck station is the sixth one, even though the processing times on both models are not large. The station is a bottleneck due to starvations that are **propagated** and generate idle times between pieces, this can be seen as a scheduling bottleneck: Almost ironically, the sixth station blocks the previous stations because it is starved by them. In Figure 9, the differences between processing times absorb the blockages and prevent them from propagating idle times. Notice that the replications of the piece set are closer to each other in Figure 9 than in Figure 8: Blockages propagate through stations in Figure 8 and generate worse steady-state behavior.

This implies on a rather unintuitive conclusion: In some specific cases, differences between models is a **desirable feature**. Naturally, the benefit is not the difference itself, but rather what it allows when scheduling comes into play. This is naturally easier when buffers are added. Furthermore, Figure 8 offers a clear example on how and why a station without high processing times can be the system's bottleneck.

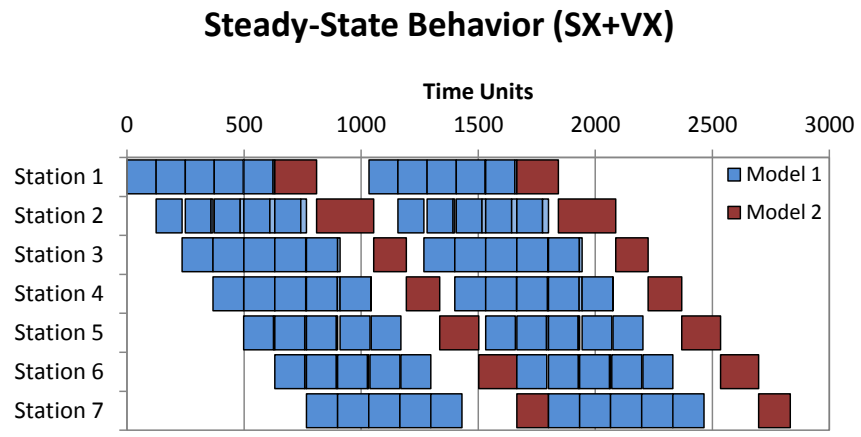

Fig. 8: Steady-State schedule for the $VX + SX$ formulation: full schedule repeats every 1021.4 time units. Darker colors represent processing times and lighter colors represent waiting times.

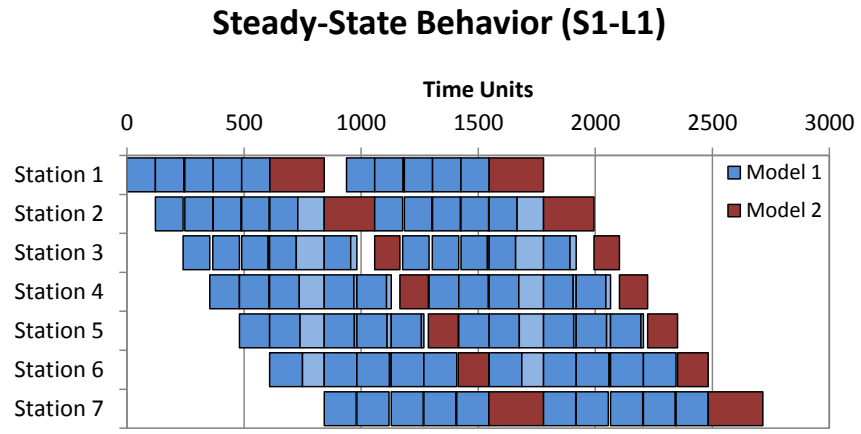

Fig. 9: Steady-State schedule for the proposed formulation: full schedule repeats every 940.2 time units. Darker colors represent processing times and lighter colors represent waiting times.

3.3 Generality Case Study

The results described in Section 3.1.1 and Section 3.2 refer to variations of a single instance: One set of data vectors describing tasks, models, and demand rates is tested with two different product sequences and three buffer layouts. It is necessary to verify whether or not these results hold in general. Furthermore, task-balancing often occurs before model sequencing (Boysen et al, 2009a) contrary to the case study, in which the product sequence is known as a parameter.

Therefore, if the procedure described by Boysen et al (2009a) is adopted, product sequences will be optimized for a certain balancing solution. This poses the question: Can the proposed formulation lead to production rate improvements for a product sequence that is itself optimized for a previous balancing solution? Given the relative proximity between the results obtained by the proposed formulation (PX) and the makespan minimization ones (MX), one can also ask: How do these sequence-aware formulations behave in other instances?

3.3.1 Data Generation

In order to answer the questions herein above mentioned and to verify the generality of the results presented in the previous sections, new tests are performed on a larger dataset. The SALBP instances from the dataset presented by Otto et al (2013) are used to generate mixed-model ones, as described hereafter: The SALBP instances are designed for the type-1 variation of the problem, with a cycle time of 1000 time units. The instances vary in size, task time distributions, precedence graph structure, and ordering strength. Otto et al (2013) present three types of task time distributions: peak in the bottom, peak in the middle, and bimodal. The first two generate instances with a normal distribution of task times centered on either a low or medium value (relative to 1000 time units). The later generates task times that are most likely to be small, but have a small probability of being much larger. This type of task distribution is chosen to generate mixed-model instances due to the following reason: If the task times of multiple SALBP instances are converted into task times of different product models, most tasks will have similar small processing times, and the tasks that have higher processing times will differ between models. This emulates a rather interesting mixed-model characteristic.

In order to restrict the comparisons to optimal solutions of each formulation, only the small (all of which had 20 tasks) cases from Otto et al (2013) are used. No filter is applied to the structure of the precedence diagrams or to its ordering strength. A total 175 bimodal and small SALBP instances are used. These are grouped five-by-five sequentially, generating 35 groups of task processing times. For each group, five task-properties data vectors are generated: one for the precedence diagram of each of the SALBP instances. For all instances, the demand rates are presumed to be equal, 20% for each product, with an MPS of $(1, 1, 1, 1, 1)$. This leads to a total of 175 mixed-model data vectors.

In order to optimize the production rate, the number of stations must be given. A specific value is chosen (seven workstations), based on the main case study. Three buffer layouts are considered: *Empty*, with no buffers; *Half*, with single unitary buffers before every even station (3 buffers in total); and *Full*, with single unitary buffers between every station (six buffers in total). This leads to a total of 525 individual *buffered mixed-model instances*.

The cyclical product sequence could be arbitrarily chosen. But one goal of this dataset is to verify if the proposed formulation can lead to production rate improvements for a product sequence that is itself optimized for a previous balancing solution. Therefore, for each *buffered mixed-model instance* two specific cyclical sequences are produced: the ones that lead to the best production rates for the SX and VX formulations. This generates a total of 1050 *sequence-aware instances* for the formulations MX and PX. This paper's Supporting Information provides additional details about the data generation.

3.3.2 Generality Study Results

Each of the 525 *buffered mixed-model instances* is solved for two sequence unaware goal functions (SX, VX)[4], generating an optimal balancing solution according to said goal. The best possible cyclical sequence is then defined for the balancing solution, this can be done rather quickly by comparing the 24 possible alternatives[5]. This sequence solution (the best possible one for either SX or VX) is then used to generate a sequence-aware instance that is solved by both proposed model (PX) and the makespan minimization model (MX). The cycle time obtained by each formulation (PX, MX, SX, VX) can then be compared to every other formulation. These results are summarized by Table 8.

Table 8: Ratios between realized values of CT, average obtained values of CT, and number of cases in which it reached the theoretical bound LB-CT, for each formulation and buffer layout.

	Buffers	SX	VX	MX	PX
Avg. Ratio to PX	Empty	1.047	1.095	1.032	1
	Half	1.06	1.069	1.048	1
	Full	1.098	1.002	1.066	1
Max. Ratio to PX	All Layouts	1.22	1.25	1.18	1
Min. Ratio to PX	All Layouts	1	1	1	1
Average CT	Empty	830.16	866.13	817.15	791.97
	Half	779.98	785.49	770.85	735.55
	Full	759.8	691.86	736.85	691.08
No. CT=LB-CT	All Layouts	1	151	11	263

Table 8 presents the average values of cycle time for each formulation (SX, VX, MX, PX) and the (minimum, average, and maximum) ratios between those values and the CT value achieved by the proposed formulation (PX). Notice that the global minimum value for all cases is 1, meaning that *the proposed formulation is not outperformed in any instance*. These results showed that the other methods behave differently depending on the buffer layouts: the SX goal leads to better results than the VX goal in the *Empty* and *Half* buffer layouts, but is outperformed by VX in the *Full* Buffer layout. In this layout, VX is very close to PX, with a 0.2% average difference in realized cycle times, meaning that this is a good alternative when assembly lines have many buffers (e.g. in the Electronic Industry). However, it generated in average the worst results for the other layouts. The makespan minimization alternative (MX) generated good results (second only to PX) for the buffer layouts *Empty* and *Half*, but showed a large difference to the proposed formulation in the *Full* buffered scenario. This mimics the results obtained on Section 3.2, suggesting that when **more buffers** are

[4] Horizontal balancing HX is not tested due to its poor performance on the practical case study and to its non-linearities that prevent optimal solutions from being found.

[5] In an *MPS* of five units (1,1,1,1,1), by fixating the first model in the first position, one can generate 4! combinations of cyclical sequences. Each sequence requires a quick simulation of a few replications of the *MPS* going through the line. Alternatively, linear relaxations of the proposed model can be used.

present there is a **stronger dispute between** the optimization of **steady-state** behavior and **transient-state** behavior. These results also corroborate the observed interconnections between balancing and buffer allocation. Furthermore, a comparison between the realized values of cycle time and the minimal theoretical one (LB-CT) shows that, in 263 cases, a finite number of buffers allows the system to reach the maximum theoretical productivity.

The Probabilistic Estimate (P.E.) discussed in Section 3.2.3 is also compared to the realized cycle time of each formulation (SX, VX, MX, PX). The minimum, average and maximum ratios are presented by Table 9. Notice that the estimate repeats the conservative overall behavior that had been previously verified: In average for all methods the estimate is higher than the realized cycle time, and remains insensitive to buffers capacity of assisting scheduling. For some instances, however, the P.E. is up to 6% optimistic, repeating the inconsistencies that had been previously observed in Section 3.2.

Table 9: Minimum, Average and Maximum P.E. Ratios to Realized CT of each Method and each Buffer Layout

Buffer	Empty			Half			Full			All Layouts		
Method	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.
SX	1.16	1.07	**0.94**	1.23	1.14	1.03	1.38	1.17	1.03	1.38	1.12	**0.94**
VX	1.24	1.14	**0.98**	1.39	1.25	1.14	1.63	1.42	1.24	1.63	1.27	**0.98**
MX	1.31	1.17	1.06	1.42	1.24	1.1	1.52	1.31	1.13	1.52	1.24	1.06
PX	1.4	1.2	1.07	1.48	1.3	1.14	1.65	1.41	1.26	1.65	1.3	1.07

A detailed table with all output information of each of the 1050 individual executions (525 for SX, 525 for VX, each followed by MX and PX) is available at the *Supporting Information*, along with the data vectors. The results reported above summarize that information, answering the aforementioned generality questions: What is **observed** in the practical case study is **verified** anew in the combinatorial instances. In the majority of cases (82%), **it is possible to improve** the productivity of the line by using a sequence aware formulation (by in average 5%), **even when that sequence is optimal for a previous balancing solution**. This reinforces the main contribution of this paper: although sequencing is usually performed after balancing, when a cyclical sequence is known, the presented balancing formulation (PX) can offer better steady-state production rates.

4 Conclusions

This paper extends a previous work by the authors (Lopes et al, 2016) in which a model for balancing an asynchronous assembly line with given buffers and cyclical product sequence is presented. The model is applied to data from a car seat assembly line on the outskirts of Curitiba-PR (Brazil) in a practical case study. The proposed formulation accurately predicts steady-state configurations with only one replication of the product set, and is validated by a simulation model. The optimized result is aware of the buffer layout and product sequence (problem's scenario). Answers obtained with each tested scenario are applied to

all other scenarios, allowing one to state that a triple interdependency of optimal solutions exists connecting balancing, sequencing and buffer allocation (Table 2). Furthermore, comparisons between scenarios (Figure 5 and Figure 6) allowed to verify that buffers can aid scheduling mixed-model assembly lines whether product models change often or not: The relative utilization of buffers can slowly increase or decrease as rather large minimal part sets (up to 30 pieces tested) go through the line.

The steady-state results achieved by the proposed formulation are compared to those reached by previous literature described goal functions: station smoothing, vertical balancing, horizontal balancing, and makespan minimization. Variants of previous formulations are also tested. The case study has shown that optimizing a flow-shop with makespan minimization goal for a finite number of replications is not equivalent to optimizing for steady-state efficiency (Table 6 and Figure 2). Moreover, the makespan minimization formulation's answer converges towards the proposed formulation's answer as the number of considered replications increases.

A probabilistic estimate of cycle time (Dar-El et al, 1999) is tested, showing conservative values for most scenarios. In some scenarios, however, the estimate is optimistic. A closer investigation on the solution provided in such scenarios suggested (counter-intuitively) that differences between models can be a desirable feature, by helping to compensate via scheduling other differences between models on other stations. This case showed that the probabilistic estimate is unable to take into account scheduling-generated bottlenecks: Stations can propagate the differences between models and lead to steady-state bottlenecks in stations without high processing times on the individual models (Figure 8).

A combinatorial study with 1050 individual instances is conducted. The optimal answers of each instance in accordance to each formulation is analyzed, and compared to the probabilistic estimate. With these additional tests, the results verified on the practical case study are reinforced: First, balancing, sequencing and buffer allocation problems are interconnected. Second, makespan minimization of a finite number of replications does not necessarily optimize steady-state. Third, a finite number of buffers might allow an assembly line to reach maximum theoretical efficiency. Last, the proposed formulation generated solutions with steady-state cycle times in average 5% better than alternative formulations (station smoothing, vertical balancing and makespan minimization), and it is not outperformed in a single any instance.

Further works should seek to combine the degrees of freedom in balancing, sequencing and possibly buffer allocation to this cyclical form, and eventually take into account transportation times between stations, amongst other often unavoidable unproductive times currently left out of the model for simplicity's sake. Other problem variations such as U-shaped assembly lines, parallelism, continuous or synchronous pace might also have modeling opportunities that can be further explored by combining such degrees of freedom.

References

Alexander DR, Premachandra IM, Kimura T (2010) Transient and asymptotic behavior of synchronization processes in assembly-like queues. Annals of Operations Research 181:641–659, DOI 10.1007/s10479-010-0796-9

Bard JF, Dar-elj E, Shtub A (1992) An analytic framework for sequencing mixed model assembly lines. International Journal of Production Research 30(1):35–48, DOI 10.1080/00207549208942876

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

Battini D, Faccio M, Persona A, Sgarbossa F (2009) Balancing - sequencing procedure for a mixed model assembly system in case of finite buffer capacity. International Journal of Advanced Manufacturing Technology 44:345–359, DOI 10.1007/s00170-008-1823-8

Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168:694–715, DOI 10.1016/j.ejor.2004.07.023

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

Boysen N, Fliedner M, Scholl A (2009a) Production planning of mixed-model assembly lines: overview and extensions. Production Planning & Control: The Management of Operations 20(5):455–471, DOI 10.1080/09537280903011626

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

Bukchin J (1998) A comparative study of performance measures for throughput of a mixed model assembly line in a JIT environment. International Journal of Production Research 36(10):2669–2685

Dar-El EM, Herer YT, Masin M (1999) CONWIP-based production lines with multiple bottlenecks: performance and design implications. IIE Transactions 31:99–111

Demir L, Tunali S, Eliiyi DT (2014) The state of the art on buffer allocation problem: a comprehensive survey. Journal of Intelligent Manufacturing 25(3):371–392, DOI 10.1007/s10845-012-0687-9

Golle U, Rothlauf F, Boysen N (2015) Iterative beam search for car sequencing. Annals of Operations Research 226:239–254, DOI 10.1007/s10479-014-1733-0

Gurevsky E, Battaïa O, Dolgui A (2012) Balancing of simple assembly lines under variations of task processing times. Annals of Operations Research 201:265–286, DOI 10.1007/s10479-012-1203-5

Gurgur CZ (2013) Optimal configuration of a decentralized, market-driven production/inventory system. Annals of Operations Research 209:139–157, DOI 10.1007/s10479-011-0977-1

Heath SK, Bard JF, Morrice DJ (2013) A GRASP for simultaneously assigning and sequencing product families on flexible assembly lines. Annals of Operations Research 203:295–323, DOI 10.1007/s10479-012-1167-5

Karabati S, Kouvelis P (1994) The interface of buffer design and cyclic scheduling decisions in deterministic flow lines. Annals of Operations Research 50:295–317

Kellegöz T (2016) Assembly line balancing problems with multi-manned stations : a new mathematical formulation and Gantt based heuristic method. Annals of Operations Research 253(1):377–404, DOI 10.1007/s10479-016-2156-x

Koenigsberg E (1959) Production Lines and Internal Storage – A Review. Management Science 5(4):410–433, DOI 10.1287/mnsc.5.4.410

Leu Yy, Huang PY, Russell RS (1997) Using beam search techniques for sequencing mixed-model assembly lines. Annals of Operations Research 70:379–397, DOI 10.1023/A:10189386

Levner E, Kats V, Alcaide D, Pablo LD, Cheng TCE (2010) Complexity of cyclic scheduling problems: A state-of-the-art survey. Computers & Industrial Engineering 59(2):352–361, DOI 10.1016/j.cie.2010.03.013

Lopes TC, Sikora CGS, 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, Vitória-ES, Brazil, pp 3458–3469

Matanachai S, Yano CA (2001) Balancing mixed-model assembly lines to reduce work overload. IIE Transactions 33:29–42, DOI 10.1080/07408170108936804

Merengo C, Nava F, Pozzetti A (1999) Balancing and sequencing manual mixed-model assembly lines. International Journal of Production Research 37(12):2835–2860, DOI 10.1080/002075499190545

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(1):33–45, DOI 10.1016/j.ejor.2012.12.029

Özcan U, Çerçioglu H, Gökçen H, Toklu B (2010) Balancing and sequencing of parallel mixed-model assembly lines. International Journal of Production Research 48(17):5089–5113, DOI 10.1080/00207540903055735

Öztürk C, Tunali S, Hnich B, Örnek MA (2013) Cyclic Scheduling of Flexible Mixed Model Assembly Lines, vol 18. IFAC, DOI 10.3182/20130619-3-RU-3018.00413

Öztürk C, Tunalı 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 10.1016/j.jmsy.2015.05.004

Pastor R, Andrés C, Duran A, Péres M (2002) Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. The Journal of Operational Research Society 53(12):1317–1323, DOI 10.1057/palgrave.jors.2601457

Salveson ME (1955) The assembly line balancing problem. The Journal of Industrial Engineering 6:18–25

Sawik T (2000) Simultaneous versus sequential loading and scheduling of flexible assembly systems. International Journal of Production Research 34(14):3267–3282, DOI 10.1080/002075400418252

Sawik T (2004) Loading and scheduling of a flexible assembly system by mixed integer programming. European Journal of Operational Research 154:1–19, DOI 10.1016/S0377-2217(02)00795-6

Sawik T (2012) Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. International Journal of Production Research 50(18):5017–5034, DOI 10.1080/00207543.2011.627388

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

Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research 168:666–693, DOI 10.1016/j.ejor.2004.07.022

Scholl A, Boysen N, Fliedner M (2009) Optimally solving the alternative subgraphs assembly line balancing problem. Annals of Operations Research 172:243–258,

DOI 10.1007/s10479-009-0578-4
Spinellis DD, Papadopoulos CT (2000) A simulated annealing approach for buffer allocation in reliable production lines. Annals of Operations Research 93:373–384
Thomopoulos NT (1970) Mixed Model Line Balancing with Smoothed Station Assignments. Management Science 16(9):593–603
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, DOI 10.1016/j.ijpe.2015.01.022