ORIGINAL ARTICLE - ACCEPTED MANUSCRIPT

# An Iterative Decomposition for Asynchronous Mixed-Model Assembly Lines: Combining Balancing, Sequencing, and Buffer Allocation

Thiago Cantos Lopes[a], Celso Gustavo Stall Sikora[b], Adalberto Sato Michels[a], Leandro Magatão[a]*

[a]: Graduate Program in Electrical and Computer Engineering (CPGEI), Federal University of Technology - Paraná (UTFPR), Curitiba, Brazil, 80230-901
[b]: Institute for Operations Research, Hamburg Business School, University of Hamburg, Germany

**ARTICLE HISTORY**
Compiled July 24, 2019

**ABSTRACT**
Asynchronous Mixed-Model Assembly lines are common production layouts dedicated to large-scale manufacturing of similar products. Cyclically scheduling such products is an interesting strategy to obtain high and stable throughput. In order to best optimize these lines, it is necessary to combine line balancing, model sequencing, and buffer allocation. However, few works integrate these three degrees of freedom, and evaluating steady-state performance as a consequence of these decisions is challenging. This paper presents a mathematical model that allows an exact steady-state performance evaluation of these lines, and hence their optimization. While the combination of degrees of freedom is advantageous, it is also computational costly. An iterative decomposition procedure based on alternation between two mathematical models and on optimality cuts is also presented. The decomposition is tested against the proposed mathematical model in a 700-instance dataset. The developed methods obtained 142 optimal answers. Results show that the decomposition outperforms the monolithic mathematical model, in particular for larger and harder instances in terms of solution quality. The optimality cuts are also shown to help the decomposition steps in terms of solution quality and time. Comparisons to a sequential procedure further demonstrate the importance of simultaneously optimizing the three degrees of freedom, as both the proposed model and the decomposition outperformed such procedure.

**KEYWORDS**
Assembly Line Balancing, Mixed-Model Sequencing, Buffer Allocation, Cyclical Scheduling, Steady-State Optimization, Decomposition

*Corresponding author. Email: magatao@utfpr.edu.br

## 1. Introduction

Assembly lines are product oriented production layouts commonly employed for large-scale manufacturing of similar products (Scholl 1999). While early versions of such lines were dedicated to a single product model, the increase in product diversity led to mixed-model assembly lines, which are shared between a set of similar products (Boysen, Fliedner, and Scholl 2009b). Substantial research has been conducted on the optimization of assembly lines, ranging from the simpler more theoretical cases (Scholl and Becker 2006) to more general ones (Becker and Scholl 2006) that remove some typical simplifying hypothesis. The focus of this paper is asynchronous mixed-model assembly lines, an unpaced variant in which each product moves to the next station when two conditions are met: processing at the current station is complete, and the next station is available (Boysen, Fliedner, and Scholl 2007). In these lines, three degrees of freedom are relevant: balancing, sequencing, and buffer allocation (Boysen, Fliedner, and Scholl 2008).

The first problem, assembly line balancing, consists of distributing a set of tasks amongst stations in a way that maximizes the lines' efficiency. Stations are sequential in nature and, usually, equally manned. Tasks have specific durations and precedence relations, meaning that some must be performed before others. Thus, precedence relations restrict task assignment. The sum of task processing times in each station is tied to the line's production rate. If only one product model is produced, the highest station-wise sum dictates the line's cycle time. When more than one product is produced in the same line, other factors such as line pace and product sequence become important. The line balancing problem was first introduced by Salveson (1955) and was recently object of a review by Battaïa and Dolgui (2013). The most common goals are the minimization of the number of stations given a minimal production rate or the minimization of the cycle time given the number of stations. In the particular case of asynchronous mixed-model assembly lines, the production rate may be difficult to determine (Tiacci 2015b), and may be a function of line balancing, product sequence, and buffer allocation.

The second problem, mixed-model sequencing, consists of defining the order in which product models are to be produced. Product models usually have known processing times in each station and given demand rates. While virtually any product sequence is technically feasible (Boysen, Fliedner, and Scholl 2009c), the order in which products are produced affects performance measures with relevant economic impacts. Cycle time is usually considered a parameter for sequencing problems. However, in throughput maximization variants, mixed-model sequencing can be an important optimization aspect, especially when combined to line balancing. Thomopoulos (1967) was one of the earliest authors to study mixed-model sequencing problem, and many variants have been defined since then (Boysen, Fliedner, and Scholl 2009c). In the particular context of relatively stable demands, cyclical scheduling is often employed to obtain a high and stable throughput (Karabati and Sayin 2003; Sawik 2012). Cyclical scheduling is often based on the Minimal Part Set (MPS) concept (Levner et al. 2010), which is defined as the smallest set of products that can be repeated indefinitely in order to meet the production target. For example, if the production target is 5000 units of product model 1 and 3000 units of product model 2, then the MPS is 5 units of product model 1 and 3 units of product model 2, which can be represented by the vector (5, 3).

The third problem, buffer allocation, consists of assigning internal storage spaces in order to increase the system's productivity. Buffers are units with the capacity of storing incomplete products between stages in a production system. Their presence can help prevent starvations and blockages, therefore, increasing throughput. Naturally, they imply costs. Buffer allocation problems usually assume that factors such as system design and workload distributions have already been solved (Demir, Tunali, and Eliiyi 2014). However, in the context of throughput maximization of mixed-model assembly lines, previous works have shown that buffers can affect the optimal workload distribution (Lopes et al. 2018b). The buffer allocation problem was first formalized by Koenigsberg (1959) and was recently reviewed by Demir, Tunali, and Eliiyi (2014). Common optimization goals are the minimization of the number of buffers required to meet a certain production rate, and the maximization of the production rate given the number of buffers.

Combining these degrees of freedom should lead to better results than optimizing them independently or sequentially (Boysen, Fliedner, and Scholl 2007, 2008). However, most works focus separately on each of these degrees of freedom (Boysen, Fliedner, and Scholl 2009c; Battaïa and Dolgui 2013; Demir, Tunali, and Eliiyi 2014). To the best of the authors' knowledge, no work has yet combined the three degrees of freedom to obtain an optimized solution as the approach herein presented. However, some papers combined two of these degrees, in particular balancing and sequencing. The strategies adopted by most authors when combining these degrees of freedom and optimizing mixed-model assembly lines are presented in Section 2.

In this paper, the number of buffers and workstations are considered as problem parameters, i.e. finite available resources. The goal is to determine a line balancing, a cyclical product sequence, and a buffer allocation that maximizes the line's throughput. The formulation presented by Lopes et al. (2018b) is extended to incorporate sequencing and buffer allocation decisions: a **new MILP model** is developed for mixed-model assembly lines, in which one is required to combine the three degrees of freedom to optimize the steady-state throughput. Therefore, the problem at hand is a mixed-model assembly line **B**alancing, **C**yclical **S**equencing and **B**uffer Allocation **P**roblem (hereafter abbreviated as **BCSB-P**). The three sub-problems are shown to be interconnected, and the presented MILP model presents a mathematical description of such connections. Due to the computational difficulties that arise from the combination of these degrees of liberty, an **Iterative Decomposition** procedure is presented. The decomposition is tested against the proposed monolithic model, proving to be very competitive. Both the decomposition and the model are also compared to a **sequential procedure** based on literature procedures for similar problems (Sawik 2004; Battini et al. 2009).

The paper is organized as follows. Section 2 provides a brief overview of recent related works. In Section 3 a context is yielded and the definition of a BCSB-P is explained. Section 4 presents the model that describes a cyclical steady-state optimization problem. Section 5 presents the proposed iterative decomposition. Section 6 presents the results of the comparisons between the monolithic model and the proposed decomposition. Lastly, the main conclusions drawn from this paper are summarized and future research directions are provided in Section 7.

## 2. Related Works

Works on optimization of assembly line balancing, sequencing, related and combined problems employ a large range of methods (mathematical models, decompositions, metaheuristics, etc.) to solve multiple variations of problems. The main focus of this section is not on the solution method applied, but rather on the problem's decision variables (balancing, sequencing, buffer allocation), and on the throughput performance measure used, i.e. the role cycle time plays in the problem's definition.

Most works that combine some of the three aforementioned degrees of freedom (balancing, sequencing, and buffer allocation) focus on variants of the balancing-sequencing combination. However, the role of cycle time as a performance definition and goal function varies substantially: Sawik (2002); Alghazi and Kurz (2018) employ the weighted average (or sum) of model-wise processing times in each station as a performance requirement.

Multiple authors consider as goal functions a measure of processing time deviations from an ideal average value: Kim, Kim, and Kim (2000a, 2006) apply such deviations as goal functions of a combined balancing-sequencing problem and Battini et al. (2009) use them as a goal function in a decomposition step in a problem that seeks to maximize production rates and minimize buffer requirements.

Workload smoothing is a performance measure that is also often employed along with other goal functions such as minimizing the number of workstations: Özcan et al. (2010) apply it for a problem definition with parallel assembly lines, and Hamzadayi and Yildiz (2012, 2013) use it for U-lines with and without parallel workstations.

Some balancing-sequencing works on continuous lines with utility work minimization goal also consider cycle time to be a parameter that affects the goal function: Kim, Kim, and Kim (2000b) present a base formulation for straight mixed-model lines, Mosadegh, Zandieh, and Fatemi Ghomi (2012) present a problem variant with station-dependent assembly times and Nilakantan et al. (2017) present a formulation for two-sided lines. Defersha and Mohebalizadehgashti (2018) present a balancing-sequencing formulation for mixed-model paced lines with the combined goal of minimizing line length, task duplications, and number of stations. In their formulation, time between products (cycle time) is also considered a parameter. Another common goal function for balancing-sequencing problems is makespan minimization for unpaced asynchronous lines (Öztürk et al. 2013, 2015). Biele and Mönch (2018) presented a cost-oriented balancing problem for when (non-cyclical) product sequence is given.

Many authors also consider cycle time as a parameter that must be respected by all product models in the line, regardless of product sequence: Zhong (2017) applies this definition for hull assembly line balancing (shipbuilding); Akpinar and Baykasoglu (2014); Akpinar, Elmi, and Bektas (2017) employ this definition for mixed-model problems with set-ups; Delice et al. (2017) uses it for two-sided assembly lines; Roshani and Nezami (2017) apply it for multi-manned lines; Kucukkoc and Zhang (2016) uses it for complex parallel lines variants; Dong, Zhang, and Xiao (2018) employs a chance-constraint version of this consideration for stochastic lines. The conversion of this parameter into a problem variable and its minimization is also a studied strategy (Karabati and Sayin 2003).

Tiacci (2015a,b, 2017) has presented a simulation-based performance evaluation to have a direct performance measure for balancing and buffer allocation on asynchronous stochastic assembly lines. Tiacci (2015a) argues that the above mentioned indirect performance measures (weighted average processing times, workload smoothing, workload deviations minimization, etc.) are not goals in themselves, but rather supposed means to achieve a high and stable throughput. This has been more recently corroborated by Lopes et al. (2018b), which presented a formulation for direct steady-state performance measure in deterministic asynchronous lines with given cyclical product sequence and given buffer allocation. Lopes et al. (2018b) also compared their formulation to indirect performance measures and confirmed that they do not necessarily optimize steady-state. Recent extensions of said formulation (Lopes et al. 2018a, 2019) incorporates sequencing decisions and parallel workstations, but do not take buffer allocation into account.

While Tiacci (2015b) presented a balancing and buffer allocation optimization procedure, its simulation-based procedure considers a random product sequence and does not include a mathematical model with decision variables and constraints. Lopes et al. (2018a)'s model, on the other hand, does not incorporate buffer allocation decisions. To the best of the authors' knowledge, no work has addressed the combined BCSB-P: Both the direct performance measure and the combination of balancing, sequencing, and buffer allocation degrees of freedom remain a challenge and a gap in the literature.

## 3. Problem Statement

The optimization of a cyclical asynchronous assembly line requires the combination of three degrees of freedom: balancing the assignment of tasks to stations; assigning some buffers to the line; and cyclical sequencing and scheduling of product models in the Minimal Part Set (MPS). Figure 1 conceptually depicts the studied BCSB-P. In this figure, the models M1, M2, and M3 (represented by circles, squares and triangles) are cyclically sequenced to enter to and depart from the line in a repeated pattern. Workstations (S1-S4) perform the required tasks, and available buffers (B1 and B2) are to be placed in two out of the three candidate spots. In this paper, the number of workstations and buffers is considered a parameter, and the goal is to maximize the line's productivity, i.e. a type-2 problem (Scholl 1999). A related problem can be defined as the minimization of the number of workstations and buffers given a maximum steady-state cycle time.

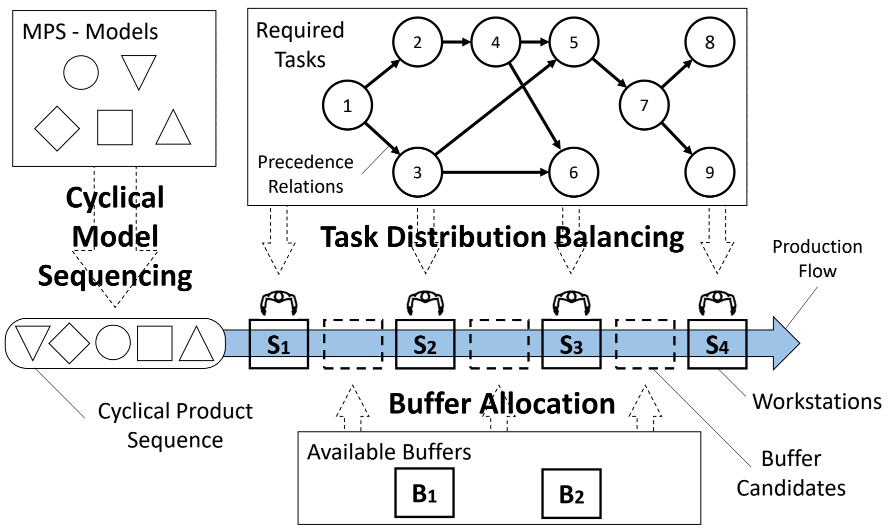


Figure 1.: Overview of the optimization problem

Line balancing must respect a set of precedence relations between tasks. If precedence relations are different across models, they can be combined into a joint precedence graph during pre-processing (Boysen, Fliedner, and Scholl 2009a). Furthermore, without loss of generality, tasks are assigned to the same station for all models: If a task can be performed at different stations for different product models, it can be replaced by a set of tasks with zero processing times for all but one model each (Boysen, Fliedner, and Scholl 2009a). For instance, in a line with 2 product models, consider a task $t$ with durations of 7 and 9 time units for models 1 and 2, respectively. If $t$ can be performed at different stations for each model, then it can be replaced by two tasks: $t_a$ with duration of 7 and 0 time units, and $t_b$ with duration of 0 and 9 time units. These tasks would have the same precedence relations as $t$.

A finite number of buffers must be assigned between stations. They help compensate model-wise deviations of processing times (Boysen, Fliedner, and Scholl 2008) by temporarily storing pieces between stations. In this paper, dummy stations called buffer candidates are added between stations to represent potential buffer allocation positions: a buffer candidate is only able to store product pieces when a buffer is assigned to that position.

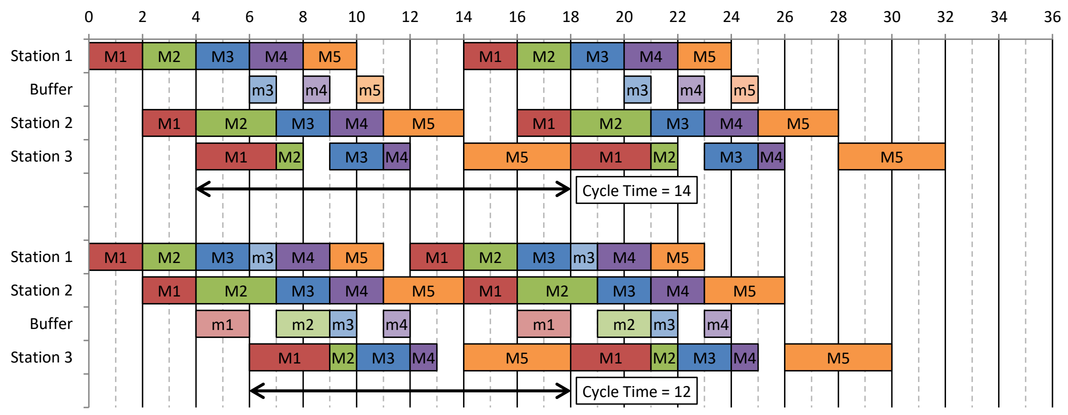
The product models in the minimal part set must be sequenced in a cyclical manner. Furthermore, the discrete entry and departure times of each product piece at each workstation must be cyclically scheduled. This scheduling ties balancing, buffer allocation, and sequencing as it must reflect the possibilities and restrictions of each of the problem's degrees of freedom. Furthermore, the scheduling variables should measure the line's steady-state performance. A recent work has shown how to measure such performance when product sequences and buffer layout are parameters (Lopes et al. 2018b). Such formulation must be extended to allow these aspects to become decision variables. Cyclical scheduling is NP-Hard (McCormick and Rao 1994) and hence, by assumption, the size of the minimal part set is assumed to be small enough so that cyclical scheduling given a balancing solution is quickly solved. This means that the core combinatorial difficulty of the problem lies on its balancing component, but that the evaluation of each balancing solution still requires solving a sequencing and buffer allocation subproblem.

In order to illustrate the importance of each degree of freedom (balancing, sequencing, and buffer allocation) in asynchronous cyclical schedules, Figure 2 is presented. Figures 2a and 2b, consider the same balancing solution, therefore, the processing times for each model at each station are constant parameters. In each of these Figures, two examples of cyclical schedules are presented with different buffer layout (Figure 2a) and product sequence (Figure 2b). Uppercase labels indicate processing times, while lowercase ones represent waiting (blocked) times. Two MPS replications are portrayed for an easier understanding of the cyclical nature of the schedules. In both Figures, arrows indicate the cycle time: the interval between entries of the first piece of each MPS at the latest station. It is clear from Figures 2a and 2b that buffer allocations and sequencing decisions can have a decisive influence on the steady-state CT value, even with a fixed set of balancing decisions. Consequently, the combination of these three degrees of freedom can provide even better solutions. Lastly, Figure 2c illustrates the influence of line balancing in solution quality. In it, two cyclical schedules are presented with the same sequencing and buffer allocation, but different line balancing solutions. This means different total processing time durations in each station and, therefore, different cycle time values.

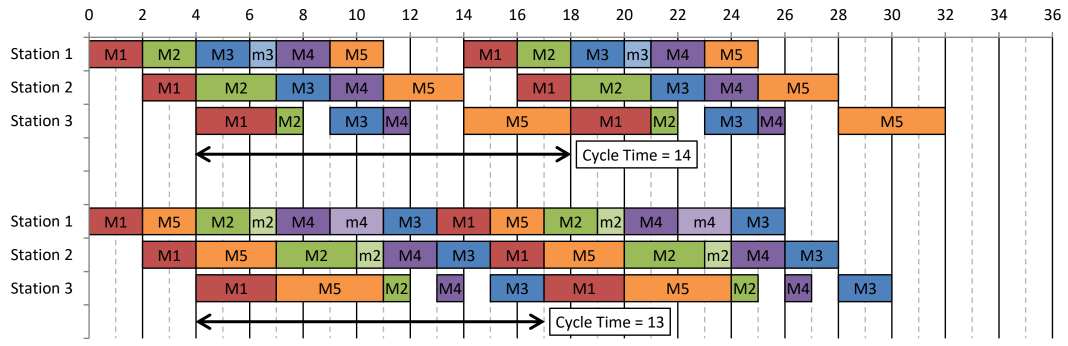
## 4. Monolithic Model

This Section presents the monolithic MILP model that combines balancing, sequencing, and buffer allocation. Table 1 presents the employed notation. The problem's goal is throughput maximization, hereby considered as cycle time minimization. This is possible as the throughput is the inverse of the average steady-state cycle time (Scholl 1999).

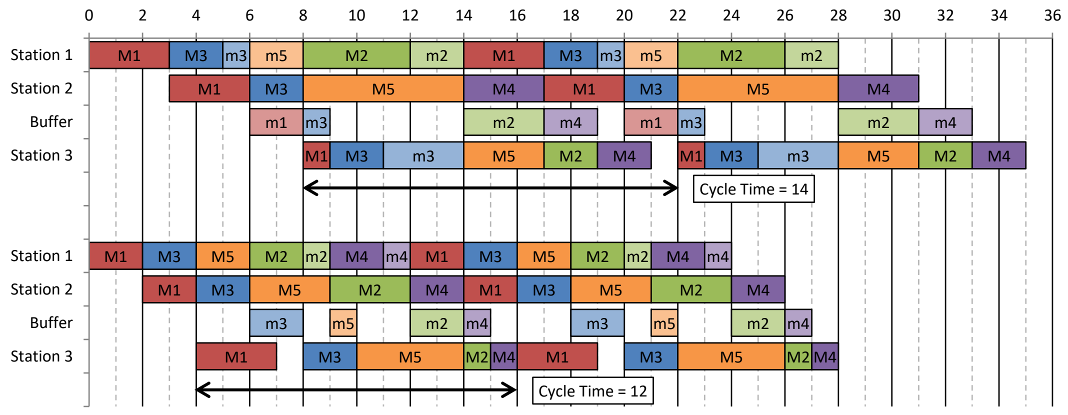
The proposed formulation's basic concept is to produce a cyclical schedule of $|P|$ product pieces such that: processing times are determined by balancing and sequencing decision variables, and scheduling constraints are affected by buffer allocation ones. Let $P$ be a set of positions in a product sequence that repeats indefinitely. Each (work)piece



(a) Examples of Cyclical Schedules with different buffer allocations.



(b) Examples of Cyclical Schedules with different product sequences.



(c) Examples of Cyclical Schedules with different task balancing.

Figure 2.: Examples of influences of each degree of freedom

| Parameter | Description |
|---|---|
| $T$ | Set of tasks $t$, $\|T\|$ states the number of tasks |
| $S$ | Set of stations $s$, $\|S/S_b\|$ states the number of workstations |
| $S_b$ | Set of buffer candidates $s_b$, $S_b \subset S$ |
| $\overleftarrow{s}$ | Immediate predecessor of station $s$ |
| $B$ | Number of available buffers, i.e. how many candidates can be buffers |
| $M$ | Set of product models $m$ |
| $N_m$ | Demand rate for model $m \in M$: integer number of model $m$ products in the MPS |
| $P$ | Set of product pieces $p$, $\|P\| = \sum_{m\in M} N_m$ states the number of pieces |
| $R$ | Set of precedence relations $(t_1, t_2)$, $t_1$ precedes $t_2$ |
| $D_{t,m}$ | Duration of task $t$ for model $m$, measured in Time Units [T.U.] |
| $\overline{CT}$ | Cycle time value for the incumbent solution |
| $BigM$ | A large number, a valid value of $\overline{CT}$ is high enough |
| | |
| Variable | Description |
| $x_{t,s}$ | Balancing binary variable, set to 1 if task $t$ is assigned to station $s$ |
| $y_{p,m}$ | Sequencing binary variable, set to 1 if model $m$ is the $p^{th}$ piece |
| $z_{s_b}$ | Buffer allocation binary variable, set to 1 if candidate $s_b$ is a buffer |
| $Tin_{p,s}$ | Continuous variable: Entry time of piece $p$ at station $s$ |
| $Tx_{p,s}$ | Continuous variable: Processing time of piece $p$ at station $s$ |
| $Tout_{p,s}$ | Continuous variable: Departure time of piece $p$ at station $s$ |
| $CT$ | Continuous variable: Average steady-state cycle time |
| $LBCT$ | Continuous variable: Lower bound on steady-state cycle time |

Table 1.: Employed nomenclature

$p$ represents a position in the cyclical product sequence. The monolithic model is defined by Expressions 1 to 13:

$$\text{Minimize} \quad CT \tag{1}$$

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

$$\sum_{s=1}^{s_k} x_{t_1,s} \geq \sum_{s=1}^{s_k} x_{t_2,s} \qquad \forall\, (t_1,t_2) \in R,\ s_k \in S \setminus S_b \tag{3}$$

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

$$\sum_{p\in P} y_{p,m} = N_m \qquad \forall\, m \in M \tag{5}$$

$$Tx_{p,s} \ \geq \ \sum_{t\in T} D_{t,m} \cdot x_{t,s} - BigM \cdot (1 - y_{p,m}) \qquad \forall\, m \in M,\ p \in P,\ s \in S \tag{6}$$

$$\sum_{p \in P} Tx_{p,s} \;=\; \sum_{t \in T, m \in M} N_m \cdot D_{t,m} \cdot x_{t,s} \qquad \forall\, s \,\in\, S \tag{7}$$

$$Tout_{p,s} \;\geq\; Tin_{p,s} + Tx_{p,s} \qquad \forall\, p \,\in\, P,\;\; s \,\in\, S \tag{8}$$

$$Tout_{p,\overleftarrow{s}} \;=\; Tin_{p,s} \qquad \forall\, p \,\in\, P,\; s \,\in\, S:\; s > 1 \tag{9}$$

$$Tin_{p,s} \;\geq\; Tout_{p-1,s} \qquad \forall\, s \,\in\, S,\; p \,\in\, P:\; p \,>\, 1 \tag{10}$$

$$Tout_{p,s} \;\leq\; Tin_{p,s} + BigM \cdot z_s \qquad \forall\, p \,\in\, P,\; s \,\in\, S_b \tag{11}$$

$$\sum_{s \in S_b} z_s \;\leq\; B \tag{12}$$

$$CT + Tin_{1,s} \;\geq\; Tout_{|P|,s} \qquad \forall\, s \,\in\, S \tag{13}$$

The goal function, i.e. cycle time minimization, is stated by the Expression 1. Equation 2 states task-assignment constraint, requiring every task to be performed at one non-buffer station ($S \setminus S_b$). Inequality 3 states the precedence relations, requiring that tasks be performed before their successors. Equation 4 states that a model is assigned to every position in the product sequence, i.e. to every piece. Equation 5 states the demand constraint: every product model $m$ will be present in $N_m$ positions in the sequence. Inequality 6 binds processing times of each piece $p$ at each station $s$ to balancing and sequencing decision variables, tying them to the sum of durations of tasks $t$ assigned to station $s$ for the product model $m$ associated to piece $p$. Equation 7 defines a logical cut for processing times within each station: The sum of processing times for all pieces equals the sum of processing times for all models, weighted by their demands. Inequality 8 states that pieces can only depart a station after entering it and being processes. Equation 9 states that a piece enters station $s$, when it departs from its immediate predecessor $\overleftarrow{s}$. Inequality 10 prevents two pieces from occupying the same station at the same time. Inequality 11 defines buffer allocation: If a candidate is chosen as a buffer, then pieces can stay on it after they enter it, otherwise they must depart immediately. Inequality 12 defines the buffer allocation limit. Lastly, Inequality 13 binds cycle time in each station to the elapsed time from the entry of the first piece to the departure of the last one. This proposed mathematical model, defined by Expressions 1 to 13, combines the balancing ($x_{t,s}$), sequencing ($y_{p,m}$), and buffer allocation ($z_s$) degrees of freedom. Therefore, it is referred to as monolithic model.

## 5. Iterative Decomposition Procedure

The monolithic model presented in the previous section tends to be difficult to solve, especially due to the constraints that employ the *BigM* relaxation method. When separated, each degree of freedom should be faster to solve, however, to measure performance algorithmically is challenging: it is difficult to determine cycle time value of the incumbent solution ($\overline{CT}$) directly from the binary decision variables $x_{t,s}$, $y_{p,m}$, and $z_{s_b}$ without solving a linear relaxation. Hence, in order to quickly obtain good solutions, an iterative decomposition procedure was developed. This is justified by the exact performance measurement allowed by the MILP formulation, and the expected faster exploration of search fields when part of the decision variables is fixed. Multiple authors present decompositions that first solve the (long-term) balancing problem then the associated (short-term) scheduling one (Sawik 2002; Battini et al. 2009). This rationale is combined to that of the model presented by Lopes et al. (2018b) in which the reverse occurs: balancing is optimized given sequencing and buffer parameters. This allows an iteration between two models, which is similar to the Fix-and-Optimize approach proposed by Helber and Sahling (2010): first the balancing variables are optimized by fixing sequencing and buffer allocation ones, then the reverse. Additionally, optimality cuts (hereafter presented) are added dynamically. These cuts are partly based on well-established graph analysis of precedence diagrams (Scholl and Becker 2006), and they function as (indirect) Combinatorial Benders' Cuts (Codato and Fischetti 2006). This type of cut consists of inequalities tied to subsystems of the linear problem and translate either optimality or feasibility requirements.

In this Section, the iterative decomposition and its notation are presented. The decomposition is based on the monolithic model (Section 4): in each step of the decomposition, some variables become parameters. Figure 3 presents a high-level overview of the iterative decomposition.

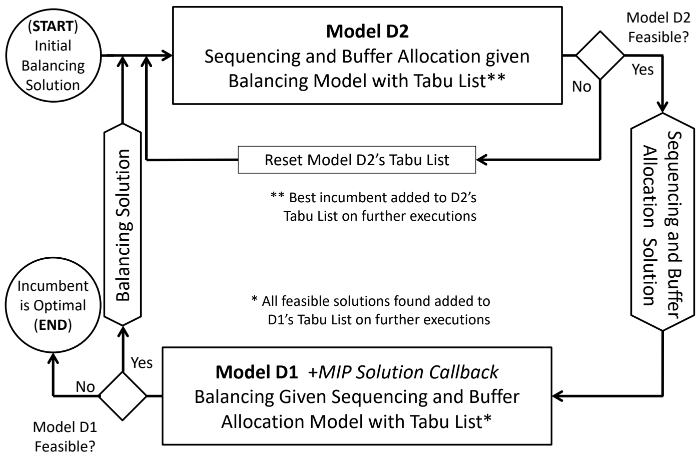

(START) Initial Balancing Solution
Model D2
Sequencing and Buffer Allocation given Balancing Model with Tabu List**
Model D2 Feasible?
Yes
No
Reset Model D2's Tabu List
** Best incumbent added to D2's Tabu List on further executions
Sequencing and Buffer Allocation Solution
Balancing Solution
Incumbent is Optimal (END)
* All feasible solutions found added to D1's Tabu List on further executions
Yes
No
Model D1 Feasible?
Model D1 +MIP Solution Callback
Balancing Given Sequencing and Buffer Allocation Model with Tabu List*

Figure 3.: Iterative Decomposition Overview

There are two main mathematical models in the decomposition. In Model D1, the sequencing and buffer allocation variables ($y_{p,m}$ and $z_s$) become binary parameters. In Model D2, the balancing variables ($x_{t,s}$) are set to a fixed configuration. Therefore, Model D1 does not require Constraints 4, 5, and 12, while Model D2 does not require Constraints 2, and 3. Nonetheless, the binary parameter values given to both D1 and D2 should be feasible in regard to these constraints. The proposed method ensures feasibility by having its parameters provided by incumbent answers of MILP models that contain these constraints.

The procedure starts by defining an initial balancing solution: By assumption, the set of tasks $T$ is topologically ordered according to the precedence relations. For instance, consider the diagram presented by Figure 1: each precedence constraints ties task pairs

$(t_a, t_b)$ such that $t_a < t_b$. This allows the initial balancing solution to be defined as follows: assign each task $t$ to station $\lfloor 1 + (t-1) \cdot |S/S_b|/|T| \rfloor$, for example, in an instance with 9 tasks and 3 workstations, tasks 1-3 will be assigned to station 1, tasks 4-6 will be assigned to station 2, and tasks 7-9 will be assigned to station 3. Because tasks are topologically ordered in regard to precedence relations, this initial balancing solution will automatically respect the precedence relations. Balancing solutions are used by Model D2 to generate a sequencing and buffer allocation solution. Model D2's solution is used by Model D1 to generate new balancing solutions. The process iterates as depicted in Figure 3.

In each iteration, the best incumbent solution provided by Model D2 is added to a Sequencing and Buffer Allocation Tabu list ($\text{TabuList}_{\text{D2}}$). Let $(\text{Tabu}_{\text{Seq}}, \text{Tabu}_{\text{Buff}}) \in \text{TabuList}_{\text{D2}}$ be a Tabu vector that contains the set of non-zero sequencing and buffer allocation variables, respectively. After the first time the Model D2 is run, further executions will include Constraint 14. This restriction demands at least one difference from each Tabu vector in the Tabu list, i.e., from each previously explored solution.

$$\sum_{(p,m)\,\in\,\text{Tabu}_{\text{Seq}}} y_{p,m} + \sum_{s\,\in\,\text{Tabu}_{\text{Buff}}} z_s \;\leq\; |P| + B - 1 \quad \forall\, (\text{Tabu}_{\text{Seq}}, \text{Tabu}_{\text{Buff}}) \;\in\; \text{TabuList}_{\text{D2}} \tag{14}$$

Similarly, every solution found by Model D1 is added to a Balancing Tabu List ($\text{TabuList}_{\text{D1}}$). However, Model D2 only adds optimal solutions to its Tabu List, and every feasible solution found by Model D1 is added to $\text{TabuList}_{\text{D1}}$. This is done by using a callback routine every time Model D1 reaches a new integer solution. The callback procedure is illustrated by Figure 4.

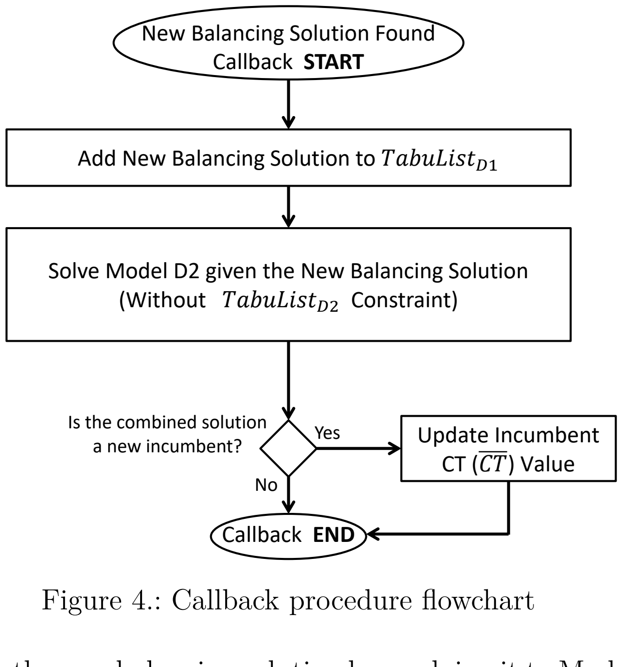

New Balancing Solution Found
Callback **START**
Add New Balancing Solution to $TabuList_{D1}$
Solve Model D2 given the New Balancing Solution
(Without $TabuList_{D2}$ Constraint)
Is the combined solution a new incumbent?
Yes
No
Update Incumbent CT ($\overline{CT}$) Value
Callback **END**

Figure 4.: Callback procedure flowchart

This callback tests the new balancing solution by applying it to Model D1 without the Tabu constraints (Inequality 14). This allows optimal sequencing and buffer allocation solutions to be obtained for that specific balancing solution. The value of cycle time is compared to the incumbent solution and replaces the previous value if the newly found solution is better than the incumbent.

After the first time the Model D1 is executed, Constraints 15, 16, and 17 are added to Model D1. The first demands at least one difference from each previously explored Tabu balancing solution $\text{Tabu}_{\text{Bal}}$ listed on Model D1's Tabu list $\text{TabuList}_{\text{D1}}$. The second demands the sum of processing times in each station to be smaller than the incumbent cycle time. The third one is based on the earliest and latest stations concept (Scholl and Becker 2006), defined for each task for the incumbent cycle time: $ES_{\left(t,\overline{CT}-1\right)}$ and $LS_{\left(t,\overline{CT}-1\right)}$. Such earliest and latest stations are computed by reducing the mixed-model problem into a single model one (Becker and Scholl 2006): First, the total processing times $\overline{D}_t$ for each task $t$ are computed as the sum of processing times across models weighted by their demands $N_m$. Then, the total processing times before and after each task are calculated in order to define the minimum number of stations before and after each task (graph analysis based on precedence relations) given a value of cycle time (Scholl and Becker 2006). This allows some binary variables to be set to zero, as stated by Equation 17, because they necessarily lead to violations of Inequality 16, otherwise.

$$\sum_{(t,s)\ \in\ \text{Tabu}_{\text{Bal}}} x_{t,s} \ \leq\ |T| - 1 \qquad \forall\ \text{Tabu}_{\text{Bal}}\ \in\ \text{TabuList}_{\text{D1}} \tag{15}$$

$$\sum_{p\ \in\ P} Tx_{p,s} \leq \overline{CT} - 1 \qquad \forall\ s\ \in\ S \tag{16}$$

$$x_{t,s} = 0 \qquad \forall\ t\ \in\ T,\ s\ \in\ S:\ s\ \notin \left[ES_{\left(t,\overline{CT}-1\right)}, LS_{\left(t,\overline{CT}-1\right)}\right] \tag{17}$$

By ignoring inefficiencies associated to blockages and starvations, the sum of processing times (Inequality 16) in each station is a lower bound on the cycle time: if that station operates at 100% efficiency, the MPS will take at least that much to be processed on it. This means that Inequality 16 and Equation 17 (whose violations imply in violations of Inequality 16) function as optimality cuts that act indirectly as Combinatorial Benders' Cuts (Codato and Fischetti 2006).

By demanding new solutions to respect these lower bound constraints, a large necessarily sub-optimal part of the search field is being discarded. Furthermore, if it is impossible for D1 to satisfy both the Tabu constraints and these performance constraints, then the incumbent solution must be optimal: each balancing Tabu solution was tested for optimal sequencing and buffer allocations, therefore, violating a Tabu constraint means testing a previously fully tested balancing solution; violating the lower bound constraint means the trial solution will have a cycle time value greater or equal to $\overline{CT}$, meaning the solution is either dominated or equivalent to the incumbent.

This means that the proposed iterative decomposition can prove the optimality of its answer, as long as enough computational time is offered. It is likely that such optimality proof would take longer than the monolithic model's one. However, the fact that the decomposition can prove a solution's optimality is relevant as it means the full search field is guaranteed to be eventually tested. That said, the focus of the iterative decomposition is finding good solutions within the time limit, not proving optimality of the best solution found.

If Model D2 is infeasible for a given $\text{TabuList}_{\text{D2}}$, then all sequences and buffer allocation possibilities have been tested. If in all iterations Model D1 was solved to optimality, then the incumbent is also optimal. However, Model D1 can require substantial time to be solved. Hence, during tests, Model D1's time limit was set to a relatively short time (60 seconds, as indicated in Section 6) in order to allow more iterations, and therefore, more product sequences to be considered. With the tested global time limit, in not a single instance Model D2 was infeasible. If that occurs for larger global time limits, then $\text{TabuList}_{\text{D2}}$ can be simply reset.

## 5.1. *Illustrative Example*

The proposed method is hereby illustrated with a small example. Table 2 presents both the instance data and the proposed decomposition iterations: The problem contains nine tasks ($|T| = 9$), five product models ($|M| = 5$), three workstations ($S = \{1,\ 1.5,\ 2,\ 2.5,\ 3\}$, $S_b = \{1.5,\ 2.5\}$, $|S \backslash S_b| = 3$) and one available buffer ($B = 1$). Demand rates are considered equal for all models, leading to an MPS of with one product piece of each. Task durations for each model ($D_{t,m}$) are presented by Table 2, as well as precedence relations ('Pre.' column). The example's precedence diagram is presented in Figure 1.

| Instance Data | | | | | | |
|---|---|---|---|---|---|---|
| | Models | | | | | |
| | 1 | 2 | 3 | 4 | 5 | |
| MPS | 1 | 1 | 1 | 1 | 1 | |
| Task | Durations | | | | | Pre. |
| 1 | 1 | 0 | 0 | 0 | 0 | - |
| 2 | 1 | 2 | 1 | 0 | 0 | 1 |
| 3 | 1 | 2 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 2 | 2 | 2 |
| 5 | 1 | 0 | 1 | 1 | 1 | 3,4 |
| 6 | 2 | 0 | 0 | 0 | 3 | 3,4 |
| 7 | 0 | 1 | 0 | 1 | 2 | 5 |
| 8 | 1 | 1 | 0 | 0 | 0 | 7 |
| 9 | 0 | 0 | 2 | 1 | 1 | 8 |
| Number of workstations: 3 | | | | | | |
| Available buffers: 1 | | | | | | |

| Method Iterations | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Assignments/Values | | | | | | | | |
| Initial Bal. | | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| Model D2 Iter. 1 | Bal. | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| | Seq. | 1 | 3 | 5 | 2 | 4 | | | | |
| | BA. | 2.5 | | | | | | | | |
| | $\overline{CT}$ | 14 | (First Incumbent) | | | | | | | |
| Model D1 Iter. 1 | Bal. | 1 | 1 | 2 | 1 | 2 | 3 | 2 | 3 | 3 |
| | Seq. | 1 | 3 | 5 | 2 | 4 | | | | |
| | BA. | 2.5 | | | | | | | | |
| | $\overline{CT}$ | 12 | (Second Incumbent) | | | | | | | |
| Model D2 Iter. - 2 does not improve solution | | | | | | | | | | |
| Model D1 Iter. 2 - infeasible | | | | | | | | | | |
| Therefore: current incumbent is optimal | | | | | | | | | | |

Table 2.: Illustrative Example data and Iterations

The initial balancing solution, obtained by assigning each task $t$ to station $\lfloor 1 + (t - 1) \cdot |S/S_b|/|T| \rfloor$, is presented as 'Initial Bal.'. In Table 2, the values presented for each incumbent are the workstations to which each task is assigned (balancing, or 'Bal.' line), the product model at each position of the sequence (sequencing, or 'Seq.' line), and the position at which the buffer is assigned (buffer allocation, or 'BA.' line). The cyclical schedules of each of these solutions are presented by Figure 2c. Thus, tasks 1-3 are initially assigned to station 1, tasks 4-6 to station 2, and tasks 7-9 to station 3. The first model D2 iteration uses the initial line balancing, leading to an incumbent with $\overline{CT} = 14$. The first model D1 iteration uses the incumbents' sequencing and buffer allocation, leading to a new incumbent with $\overline{CT} = 12$.

After the second incumbent is found, a second Model D2 iteration fails to improve the solution. The subsequent Model D1 iteration states that the model is infeasible, proving the optimality of the second incumbent. Table 3 helps to understand why the second Model D1 iteration was infeasible. It states the total task durations for each task ($\overline{D}_t = \sum_{m \in M} N_m \cdot D_{t,m}$ ) as well as the values of earliest and latest stations (Scholl and Becker 2006) for each task after each incumbent is found ($\text{ES}_{t,c}$, $\text{LS}_{t,c}$). These values are computed using the cycle time value $c$ equal to that of each incumbent minus one ($\overline{CT} - 1$).

| | Earliest and Latest Station for Each Task | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Task: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $c = \overline{CT} - 1$ | $\overline{D}_t$ | 1 | 4 | 4 | 5 | 4 | 5 | 4 | 2 | 4 |
| 13 | $\mathrm{ES}_{t,c}$ | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| | $\mathrm{LS}_{t,c}$ | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| 11 | $\mathrm{ES}_{t,c}$ | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 |
| | $\mathrm{LS}_{t,c}$ | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |

Table 3.: Illustrative example of earliest and latest stations after each incumbent

Table 3 reports that after the second incumbent is found, the first four tasks must be assigned to the first station ($\mathrm{ES}_{t,c}$=$\mathrm{LS}_{t,c}$=1). However, the sum of their processing times (Table 2) is 14, which is higher than the current incumbent cycle time ($\overline{CT} = 12$). Hence, Constraint 16 will be necessarily violated, leading to the aforementioned infeasible status for Model D1's second iteration.

## 6. Results

In order to verify how the proposed iterative decomposition compares to the proposed monolithic model, tests were performed on a 700-instance dataset. The dataset is based on Otto, Otto, and Scholl (2013)'s SALBP instances. Lopes et al. (2018b) combined single model instances of the same size to generate a set of mixed-model instances. This data set was extended by incorporating larger instances from Otto, Otto, and Scholl (2013). In total, there are 350 task properties data vectors (processing times, and precedence relations) half of which have 20 tasks and the other half 50 tasks. These data vectors can be further classified in regard to Order Strength (OS), a number between 0 and 1 that reflects how restricted task assignment is by the precedence relations (Scholl 1999). Each instance was tested with a fixed number of stations (10 for smaller cases, and 15 for the larger ones) and two numbers of buffers to be allocated (2 and 4 for the smaller cases, and 4 and 6 for the larger ones). All instances have five product models and the minimal part set was considered as one unit of each product model. Instance data is made available by the paper's supporting information.

Each of the 700 instances was solved with both the monolithic model (Mono) and the proposed iterative decomposition (Dec) with a time limit of 1800 seconds. All executions employed the same hardware and software conditions: Gurobi 7.5 was used to solve the models using a Core i7-3770 CPU (3.4GHz) and 16 GB RAM. The best solutions found by both methods are made available by the paper's supporting information. Table 4 summarizes the upper bound results of the executions by comparing answers of Mono and Dec. In Table 4, $N$ indicates the number of instances with the respective parameters (size, number of buffers and OS). Section 6.1 presents information on lower bounds and integrality gaps. The solutions found by Dec and Mono were compared in regard to the goal function (cycle time) for each instance: When only one of the methods found a best solution for a specific instance, that solution counts as an exclusive best solution. The column *Nbest* reports the number of best solutions found by each method, and the number of exclusive best ones is reported in parenthesis. Similarly, the number of optimal answers is reported under the *Nopt* column. Dec often found the optimal answer (same cycle time as the optimal one by Mono), but could not prove its optimality. The number of instances in which each method was able to prove on its own the optimality of the answer it found is reported in parenthesis under the *Nopt* column.

| Dataset | | | | Nbest(exclusive) | | Nopt(proven) | | $CT_{Mono}$ - $CT_{Dec}$ | (T.U.) | |
|---|---|---|---|---|---|---|---|---|---|---|
| Size | Buffers | OS | $N$ | Dec | Mono | Dec | Mono | Min. | Avg. | Max. |
| Small \|T\|=20 \|S\|=10 | 2 | 0.2 | 75 | 68(55) | 20(7) | 10(5) | 13(13) | -55 | 26.8 | 97 |
| | | 0.6 | 75 | 65(34) | 41(10) | 20(0) | 23(23) | -27 | 15.2 | 92 |
| | | 0.9 | 25 | 25(0) | 25(0) | 25(2) | 25(25) | 0 | 0 | 0 |
| | 4 | 0.2 | 75 | 62(43) | 32(13) | 13(4) | 15(15) | -42 | 14.0 | 67 |
| | | 0.6 | 75 | 69(28) | 47(6) | 39(16) | 41(41) | -11 | 13.9 | 93 |
| | | 0.9 | 25 | 25(0) | 25(0) | 25(17) | 25(25) | 0 | 0 | 0 |
| Large \|T\|=50 \|S\|=15 | 4 | 0.2 | 75 | 75(75) | 0(0) | 0(0) | 0(0) | 31 | 116.0 | 263 |
| | | 0.6 | 75 | 75(75) | 0(0) | 0(0) | 0(0) | 9 | 106.4 | 256 |
| | | 0.9 | 25 | 24(23) | 2(1) | 0(0) | 0(0) | -35 | 55.0 | 150 |
| | 6 | 0.2 | 75 | 74(74) | 1(1) | 0(0) | 0(0) | -1 | 109.8 | 233 |
| | | 0.6 | 75 | 71(70) | 5(4) | 0(0) | 0(0) | -36 | 91.2 | 214 |
| | | 0.9 | 25 | 22(19) | 6(3) | 0(0) | 0(0) | -29 | 23.3 | 128 |

Table 4.: Results Summary - comparisons between **Dec**omposition and **Mono**lithical model

The combined balancing, cyclical scheduling, and buffer allocation problem is complex and hard to define lower bounds for. Hence, it is expected that only small very restrictive instances (high ordering strength) will be solved to optimality. Indeed, all small-size instances with high OS (0.9) were solved to optimality, but only a minority of instances with low OS (0.2), and none of the large instances. By comparing cycle time upper bounds, one can verify that the decomposition produced better answers in more instances than the monolithic model: In all problem sets, the decomposition found more or at least as many best answers as the monolithic model. In fact, the only sets in which they tied were the small ones with high OS. In all other sets (low and medium OS), the decomposition produced more best answers, both total and exclusive ones, than the monolithic model. In 70.9% (496/700) of instances, the decomposition outperformed the monolithic model, and the reverse only happened in 6.5% (45/700) of instances.

Furthermore, Table 4 presents the differences in cycle time of the best solutions obtained by the decomposition and the model: maximum, average, and minimum. Positive differences indicate that the decomposition obtained a better answer. It is clear that in larger and less restrictive instances, the decomposition produced more substantial differences. Notice that in some cases the monolithic model did outperform the decomposition by a significant margin, but those were the exception: both the average differences and the number of best solutions found indicate the superiority of the decomposition.

A drawback of the decomposition is the systematic lack of lower bounds and the consequent difficulty in proving optimality of its solutions. For instance, in the smaller dataset, while the number of optimal solutions found was similar, only in a minority of cases did the decomposition proved on its own the optimality of its solutions and most of these cases had high OS. Section 6.1 further explores these lower bound questions.

Table 4 also presents an interesting behavior regarding buffers: in the small dataset, more instances with a higher number of buffers were solved to optimality by both the monolithic model and the decomposition. This is likely tied to the fact that more buffers enable lower cycle times, due to the decrease of blockages and starvations they allow. With a lower incumbent cycle time, performance constraints (Inequalities 7 and 16) can more easily discard parts of the search space. This suggests that with larger numbers of buffers, the highest sum of processing times becomes a better approximation of the realized cycle time. In other words: the naive lower bound on cycle time approaches the optimal value of cycle time when more buffers are added.

It has been repeatedly stated that combining the degrees of liberty provides better answers (Boysen, Fliedner, and Scholl 2008; Lopes et al. 2018b). However, doing so explicitly and in a single mathematical model might lead to computational difficulties and intractability, in particular for larger instances. In that regard, the proposed iterative decomposition combines these degrees of freedom by iteratively transforming part of the decision variables in parameters. This results in more easily tractable mathematical models, which have led to better performance: when instances are large and have a less restricted search-space, the decomposition generated better answers than the monolithic model. For small and very restricted instances, the decomposition still found many optimal solutions, even though it had significant difficulties to prove their optimality.

## 6.1. *Sequential Procedure, Lower Bounds, and Integrality Gaps*

In order to evaluate how well the proposed decomposition and monolithic model combine these degrees of liberty, a sequential method was also implemented and executed with the same hardware and software conditions. This method (hereafter abbreviated as 'Seq') consisted of a two-stage process: First, a virtual single model assembly line balancing problem is solved. This model, seeks to minimize the $LBCT$ variable, bounded by the Inequality 18. The only other constraints are the occurrence (Inequality 2) and precedence relations (Inequality 3). The best answer obtained within the time limit (900 seconds) is then solved by the Model D2 to provide the optimal sequencing and buffer allocation solutions. This sequential approach mimics the procedure described by Sawik (2004) and employs the same virtual model definition used by Battini et al. (2009).

$$LBCT \geq \sum_{t \in T} \sum_{m \in M} N_m \cdot D_{t,m} \cdot x_{t,s} \qquad \forall\, s \in S \tag{18}$$

It is easy to show that a lower bound to $LBCT$ is a lower bound to the cycle time ($CT$) of the original problem: if each station operates at 100% efficiency, each MPS will take at least the total processing time assigned to the most loaded station (Inequality 18) to flow through the line in the steady-state. Hence the MPS-wise cycle time is higher or equal to $LBCT$. Thus, if $LBCT$ is higher than the lower bound obtained by the monolithic model, its value can be used as a lower bound of the original problem. If the optimal $LBCT$ value is unknown, then the best bound obtained by the virtual single model is employed for the comparisons. This allows integer gaps to be calculated for the **Seq**uential, the **Dec**omposition and the **Mono**lithic best answers. The average integer gaps are presented by Table 5. Notice that the average Dec gap (5.3%) is 19% smaller than the average Mono gap (6.6%) and 68% smaller than the average Seq gap (16.9%). These gaps tend to be smaller for instances with more buffers: For instance, the average gap for large instances with six buffers is lower than that of the same instances with four for all values of ordering strength. This might be explained by the fact that more buffers allow better cyclical schedules and the assumption that lower bounds might not change much, since they are very difficult to compute even without taking the buffer information into account. Furthermore, instances with higher ordering strength tend to have smaller gaps. This is explained by the smaller search field, which makes finding better solutions and bounds easier.

Table 5 also presents lower bound comparisons between the monolithic model and

| Dataset | | | | Integrality Gaps | | | LB comparisons | | Gap Best $-$ *LBCT* | |
|---|---|---|---|---|---|---|---|---|---|---|
| Size | Buffers | OS | N | Seq | Dec | Mono | Rate | $\text{N}_{\text{Imp.}}$ | All | Optimal |
| Small $\lvert T\rvert=20$ $\lvert S\rvert=10$ | 2 | 0.2 | 75 | 20.1% | 5.7% | 6.6% | 99.2% | 30 | 7.6% | 7.8% |
| | | 0.6 | 75 | 18.4% | 3.9% | 4.4% | 97.0% | 18 | 7.6% | 6.8% |
| | | 0.9 | 25 | 15.0% | 0.0% | 0.0% | 92.3% | 0 | 7.7% | 7.7% |
| | 4 | 0.2 | 75 | 13.2% | 3.0% | 3.5% | 99.4% | 24 | 4.1% | 2.4% |
| | | 0.6 | 75 | 11.7% | 1.5% | 2.0% | 98.4% | 23 | 3.8% | 3.7% |
| | | 0.9 | 25 | 8.0% | 0.0% | 0.0% | 96.7% | 0 | 3.3% | 3.3% |
| Large $\lvert T\rvert=50$ $\lvert S\rvert=15$ | 4 | 0.2 | 75 | 22.2% | 8.8% | 11.3% | 100.2% | 62 | 8.8% | - |
| | | 0.6 | 75 | 20.6% | 8.7% | 10.9% | 100.6% | 66 | 8.7% | - |
| | | 0.9 | 25 | 14.8% | 6.8% | 8.0% | 101.0% | 17 | 7.0% | - |
| | 6 | 0.2 | 75 | 18.0% | 7.1% | 9.5% | 100.1% | 61 | 7.1% | - |
| | | 0.6 | 75 | 16.9% | 7.1% | 9.1% | 100.5% | 66 | 7.1% | - |
| | | 0.9 | 25 | 11.4% | 4.5% | 5.1% | 100.7% | 14 | 4.7% | - |
| Average/Total | | | | 16.9% | 5.3% | 6.6% | 99.2% | 381 | 6.7% | 5.0% |

Table 5.: Average integrality gaps and lower bounds

*LBCT*. The 'Rate' column presents the rate between *LBCT* and the lower bound obtained by the monolithic model: values lower than 100% indicate that the monolithic model provided better bounds in average. Notice that for small instances, in average, the monolithic model provided better lower bounds and that the reverse happened for the larger instances. Column '$\text{N}_{\text{Imp.}}$' indicates the number of instances in which *LBCT* was higher than the bound provided by the monolithic model. Notice that this was the case for the majority of Large instances, however the difference was relatively small. For a minority of small instances, the sequential approach improved lower bounds.

Lastly, Table 5 presents a gap between the best upper bound (usually found by the proposed decomposition) and the *LBCT* lower bound. This rate is computed by dividing the difference between *LBCT* and the $\overline{CT}$ for the best method by *LBCT*. The 'All' column presents the average difference across all instances and the 'Optimal' column presents the average across the instances solved to optimality. Notice that the values in these columns are similar for small instances, suggesting that gaps to the optimal solutions are smaller than reported by the Integrality Gap columns: Instances that were solved to optimality presented average gaps to *LBCT* comparable in magnitude to those that were not. This suggests that the monolithic model and the proposed decomposition might have more difficulty in providing good lower bounds than in providing good solutions. This is expected, lower bounds for the BCSB-P tend to be difficult to establish due to both the combination of the three degrees of freedom and the difficulties in performance measure.

## 6.2. *Influence of Optimality Cuts*

In order to measure the influence of the Optimality Cuts (Inequality 16 and Equation 17) on the method's performance, the following tests were conducted: For a fifth of BCSB-P instances in each dataset (140 out of 700 total), 25 randomly generated sequences were supplied to model D1. For each product sequence, the model was executed twice, once with and once without the Optimality Cuts (7000 total Model D1 instances). Each Model D1 instance was solved with a time limit of 60 seconds, the same one employed by the decomposition step. The solutions obtained with and without the Optimality Cuts were compared both in terms of solution quality (cycle time) and solution time. The percentage of instances with different values (reductions and increases) of cycle time and solution time is reported by Table 6. Instances in which solution quality and time were better with the Cuts than without them are reported as reductions (Red.) and when the contrary happened they are reported as increases (Inc.). The average relative difference in those measures is also reported by Table 6. For multiple Model D1 instances, the difference in $CT$ between solutions was only 1 time unit (less than 0.05%) of one another. In order to restrict the analysis to more substantial differences, these were considered as ties for the purpose of counting. Similarly, time comparisons are considered ties when optimality is not reached neither with or without the optimality cuts.

| Instances | | $CT$ Differences | | | Time Differences | | |
|---|---|---|---|---|---|---|---|
| Size | OS | Nb. Red. | Nb. Inc. | Avg. Red. | Nb. Red. | Nb. Inc | Avg. Red. |
| | 0.2 | 27.6% | 11.3% | 0.16% | 87.5% | 3.2% | 44.4% |
| Small | 0.6 | 32.8% | 11.5% | 0.26% | 92.8% | 2.2% | 50.9% |
| | 0.9 | 52.0% | 18.8% | 1.2% | 98% | 2% | 47.7% |
| | 0.2 | 63.7% | 3.2% | 0.66% | 0% | 0% | 0% |
| Large | 0.6 | 55.2% | 4.4% | 0.59% | 0% | 0% | 0% |
| | 0.9 | 72.4% | 16.8% | 0.44% | 98.8% | 0% | 73.5% |

Table 6.: Influences of Optimality Cuts: Cycle Time and Solution Time comparisons

For both instance sizes and all values of OS, the number of instances with CT reductions was more than double the number of instances with increases. A large number of these instances was solved to optimality, meaning that comparing the time required to do so can be relevant. In small instances (with low, medium, and high OS), as well as large ones with high OS, the majority of iterations were solved significantly faster when the optimality cuts are employed. For small instances, this time reduction was near 50%, while for large ones reductions only occurred for high OS, averaging 73.5%. This means that more iterations can be performed within the time limit, as more cyclical product sequences and buffer allocations are fully tested. Large instances with low and medium OS displayed too broad search fields so that none of their model D1 instances were solved to optimality. This means that time comparisons were not possible in these datasets, as all instances tied at the time limit. However, in the majority of executions on these instances, the optimality cuts led to better solutions (63.7% and 55.2%, as indicated by column Nb. Red.), averaging around 0.5% reductions in cycle time. Therefore, Table 6 shows that, for most instances, the optimality cuts lead to better average performance both in terms of the goal function (cycle time minimization) and in terms of time required to solve Model D1 instances.

## 7. Conclusions

Optimizing throughput of mixed-model asynchronous lines requires the solution of three optimization problems: balancing the task distribution, sequencing product models, and allocating buffers (internal storage). While many works have focused on each of these problems, and on some combinations of them, none thus far had explicitly combined all three degrees of freedom. This paper expands a cyclical formulation capable of measuring steady-state performance of said lines when cyclical sequencing and buffer allocation are parameters, and allows these to be decision variables.

A new mixed-integer linear programming model is presented to describe and optimize the simultaneous **B**alancing, **C**yclical **S**equencing and **B**uffer allocation asynchronous assembly line **P**roblem (BCSB-P). These three degrees of freedom have not been previously combined in the literature. However, by comparing the proposed monolithic model to a sequential procedure (which mimics previous literature methods), tests on a 700-instance dataset show that combining the degrees of freedom is very important to achieve good answers: the average integer gap of the solutions obtained by the monolithic model is 60% smaller than those of the sequential approach (Table 5). However, the combined problem is computationally challenging.

Based on this computational difficulty, an iterative decomposition procedure is also presented to obtain better answers than the proposed monolithic model (Figures 3 and 4). Two mathematical models operate alternatively: a balancing model in which product sequence and buffer layouts are parameters, and a sequencing and buffer allocation model in which processing times (balancing) are parameters. These models are based on the monolithic model (Defined by Expressions 1 to 13): they retain part of the original decision variables and change the others to parameters. Optimality cuts are also incorporated by the decomposition (Expressions 16 and 17) and are shown to lead to better solution quality and time at the method's iterations (Table 6). The proposed decomposition outperforms the monolithic model in the production of upper bounds for the problem. The average integrality gap of solutions obtained by the proposed decomposition is 19% smaller than those of the monolithic model (Table 4). This difference is more significant for large instances and tends to be stronger for less restricted cases. Although the decomposition's focus is to generate good solutions rather than optimality proof, it does have the capacity to do so.

The combination of the degrees of freedom makes the attainment of strong lower bounds a challenge: in most large instances, the bounds obtained by the monolithic model were weaker than the lower-bound provided by the virtual single model balancing problem. Stronger lower bounds are, therefore, a direction for further works, which can also include the explicit combination of the degrees of freedom to other problem variants such as paced assembly lines, flexible flow-shops, parallel discrete stations, or U-Shaped lines. Further works should also apply and adapt the proposed method to larger problems, in particular in regard to sequencing and buffer allocation.

## Funding

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).

## References

Akpinar, Sener, and Adil Baykasoglu. 2014. "Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model." *Journal of Manufacturing Systems* 33: 177–187.

Akpinar, Sener, Atabak Elmi, and Tolga Bektas. 2017. "Combinatorial Benders cuts for assembly line balancing problems with setups." *European Journal of Operational Research* 259 (2): 527–537.

Alghazi, Anas, and Mary E Kurz. 2018. "Mixed model line balancing with parallel stations, zoning constraints, and ergonomics." *Constraints* 23: 123–153.

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

Battini, Daria, Maurizio Faccio, Alessandro Persona, and Fabio Sgarbossa. 2009. "Balancing - sequencing procedure for a mixed model assembly system in case of finite buffer capacity." *International Journal of Advanced Manufacturing Technology* 44 (3-4): 345–359.

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

Biele, Alexander, and Lars Mönch. 2018. “Hybrid approaches to optimize mixed-model assembly lines in low-volume manufacturing.” *Journal of Heuristics* 24 (1): 49–81.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2007. “A classification of assembly line balancing problems.” *European Journal of Operational Research* 183 (2): 674–693.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2008. “Assembly line balancing: Which model to use when?” *International Journal of Production Economics* 111 (2): 509–528.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2009a. “Assembly line balancing: Joint precedence graphs under high product variety.” *IIE Transactions* 41 (3): 183–193.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2009b. “Production planning of mixed-model assembly lines: Overview and extensions.” *Production Planning and Control: The Management of Operations* 20 (5): 455–471.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2009c. “Sequencing mixed-model assembly lines: Survey, classification and model critique.” *European Journal of Operational Research* 192 (2): 349–373.

Codato, Gianni, and Matteo Fischetti. 2006. “Combinatorial Benders’ Cuts for Mixed-Integer Linear Programming.” *Operations Research* 54 (4): 756–766.

Defersha, Fantahun M, and Fatemeh Mohebalizadehgashti. 2018. “Simultaneous balancing, sequencing, and workstation planning for a mixed model manual assembly line using hybrid genetic algorithm.” *Computers & Industrial Engineering* 119: 370–387.

Delice, Yılmaz, Emel Kizilkaya Aydogan, Uğur Özcan, and Mehmet Sitki Ilkay. 2017. “A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing.” *Journal of Intelligent Manufacturing* 28: 23–36.

Demir, Leyla, Semra Tunali, and Deniz Tursel Eliiyi. 2014. “The state of the art on buffer allocation problem: a comprehensive survey.” *Journal of Intelligent Manufacturing* 25 (3): 371–392.

Dong, Jietao, Linxuan Zhang, and Tianyuan Xiao. 2018. “A hybrid PSO / SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints.” *Journal of Intelligent Manufacturing* 29 (4): 737–751.

Hamzadayi, Alper, and Gokalp Yildiz. 2012. “A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints.” *Computers and Industrial Engineering* 62 (1): 206–215.

Hamzadayi, Alper, and Gokalp Yildiz. 2013. “A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines.” *Computers and Industrial Engineering* 66 (4): 1070–1084.

Helber, Stefan, and Florian Sahling. 2010. “A fix-and-optimize approach for the multi-level capacitated lot sizing problem.” *International Journal of Production Economics* 123 (2): 247–256.

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

Kim, Yeo Keun, Jae Yun Kim, and Yeongho Keun Kim. 2006. “An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines.” *European Journal of Operational Research* 168 (3): 838–852.

Kim, Yeo Keun, Sun Jin Kim, and Jae Yun Kim. 2000a. “Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm.” *Production Planning and Control: The Management of Operations* 11 (8): 754–764.

Kim, Yeongho Keun, Jae Yun Kim, and Yeongho Keun Kim. 2000b. “A Coevolutionary Algorithm for Balancing and Sequencing in Mixed Model Assembly Lines.” *Applied Intelligence* 13 (3): 247–258.

Koenigsberg, Ernest. 1959. “Production Lines and Internal Storage – A Review.” *Management Science* 5 (4): 410–433.

Kucukkoc, Ibrahim, and David Z Zhang. 2016. “Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines.” *International Journal of Advanced Manufacturing Technology* 82: 265–285.

Levner, Eugene, Vladimir Kats, David Alcaide López de Pablo, and T. C. E. Cheng. 2010. “Complexity of cyclic scheduling problems: A state-of-the-art survey.” *Computers and Industrial Engineering* 59 (2): 352–361.

Lopes, Thiago Cantos, Adalberto Sato Michels, Celso Gustavo Stall Sikora, and Leandro Magatão. 2019. “Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations.” *Journal of Manufacturing Systems* 50: 193–200.

Lopes, Thiago Cantos, Adalberto Sato Michels, Celso Gustavo Stall Sikora, Rafael Gobbi Molina, and Leandro Magatão. 2018a. “Balancing and cyclically sequencing synchronous, asynchronous, and hybrid unpaced assembly lines.” *International Journal of Production Economics* 203: 216–224.

Lopes, Thiago Cantos, Celso Gustavo Stall Sikora, Adalberto Sato Michels, and Leandro Magatão. 2018b. “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.

McCormick, S. Thomas, and U. S. Rao. 1994. “Some complexity results in cyclic scheduling.” *Mathematical and Computer Modelling* 20 (2): 107–122.

Mosadegh, H., M. Zandieh, and S. M T Fatemi Ghomi. 2012. “Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines.” *Applied Soft Computing* 12: 1359–1370.

Nilakantan, J.M., Z. Li, Q. Tang, and P. Nielsen. 2017. “MILP models and metaheuristic for balancing and sequencing of mixed-model two-sided assembly lines.” *European Journal of Industrial Engineering* 11 (3): 353–379.

Otto, Alena, Christian Otto, and Armin Scholl. 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.

Özcan, Ugur, Hakan Çerçioglu, Hadi Gökçen, and Bilal Toklu. 2010. “Balancing and sequencing of parallel mixed-model assembly lines.” *International Journal of Production Research* 48 (17): 5089–5113.

Öztürk, Cemalettin, Semra Tunali, Brahim Hnich, and Arslan Örnek. 2015. “Cyclic scheduling of flexible mixed model assembly lines with parallel stations.” *Journal of Manufacturing Systems* 36 (3): 147–158.

Öztürk, Cemalettin, Semra Tunali, Brahim Hnich, and M. Arslan Örnek. 2013. “Balancing and scheduling of flexible mixed model assembly lines.” *Constraints* 18 (3): 434–469.

Roshani, A, and Farnaz Ghazi Nezami. 2017. “Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach.” *Assembly Automation* 37 (1): 34–50.

Salveson, M E. 1955. “The assembly line balancing problem.” *The Journal of Industrial Engineering* 6: 18–25.

Sawik, Tadeusz. 2002. “Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line.” *European Journal of Operational Research* 143 (1): 115–124.

Sawik, Tadeusz. 2004. “Loading and scheduling of a flexible assembly system by mixed integer programming.” *European Journal of Operational Research* 154 (1): 1–19.

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

Scholl, Armin. 1999. *Balancing and sequencing assembly lines.* 2nd ed. Heidelberg: Physica.

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

Thomopoulos, Nick T. 1967. “Line Balancing-Sequencing for Mixed-Model Assembly.” *Management Science* 14 (2): B—-59—-B75.

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

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

Tiacci, Lorenzo. 2017. "Mixed-model U-shaped assembly lines: Balancing and comparing with straight lines with buffers and parallel workstations." *Journal of Manufacturing Systems* 45: 286–305.

Zhong, Yu-guang. 2017. "Hull mixed-model assembly line balancing using a multi-objective genetic algorithm simulated annealing optimization approach." *Concurrent Engineering: Research and Applications* 25 (1): 30–40.