Balancing and Cyclical Scheduling of Asynchronous Mixed-Model Assembly Lines with Parallel Stations

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

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

Abstract

This paper considers two optimization problems commonly associated to mixed-model assembly lines: balancing task-station assignments and sequencing/scheduling different product models in a cyclical manner. Cyclical scheduling for this particular problem variant is challenging, and multiple approaches have been previously employed by different authors. This paper presents a new mixed-integer linear programming formulation to optimize the steady-state of these lines. Tests on a 36-instance benchmark demonstrated that the proposed model significantly outperformed the previous literature formulation. Furthermore, it is shown that common scheduling rules (often used in simulators) do not necessarily converge to optimal cyclical schedules even when the optimal launch order is used. Tests have also demonstrated that parallelism can allow a marginally increasing value for workstations: doubling (tripling) stations in a line with parallelism can often offer more than double (triple) the optimal throughput of lines without parallelism.

Keywords: Assembly Line Balancing, Mixed-Model Sequencing, Cyclical Scheduling, Parallel Stations, Asynchronous Unpaced Line, Mixed-Integer Linear Programing

Journal of Manufacturing Systems, Volume 50, Pages 193-200
DOI: 10.1016/j.jmsy.2019.01.001

1. Introduction

Assembly lines are product oriented manufacturing layouts commonly employed for large scale production of similar products [1]. The optimization of such lines has been subject of a substantial body of literature, including balancing task distributions [2] and scheduling product models [3]. While these inter-related optimization problems are usually treated in a sequential (hierarchical) manner, multiple recent works have combined them [4, 5, 6, 7].

Balancing (and balancing-sequencing) works range from studies on simple single product deterministic lines [8] to ones that incorporate diverse realistic features [9]. Some of the commonly

*Corresponding author: magatao@utfpr.edu.br

considered aspects are: set-up times between tasks [10, 11], ergonomic constraints [12, 13], space constraints [14, 15, 16, 17], worker movement between stations [18], among others. Extensive classifications of these problems are provided by references [9, 19]. In this paper, parallel stations and product diversity (mixed-model) under stable demands are the main considered features.

There are multiple manners assembly lines employ parallelism [2]: Some authors consider parallel lines with crossover workers [4, 20], others consider parallel work within a (multi-manned) station [21, 22]. This paper considers parallel stations [23] under an unpaced asynchronous line control [19]. Parallel stations allow higher station-wise processing times, more balancing flexibility, and also offer better recovery from failure capacity as the line is not necessarily stopped when one of the stations needs maintenance [24, 25, 26]. Asynchronous line control means that products can move downstream when processing at the current station is completed and the next stage has an empty station. Blockages occur when a product's processing is completed and all stations of the next stage are busy, and starvations occur when a station is empty and no product from the previous stage has been yet fully processed [19].

Product diversity can also affect assembly lines in multiple manners [9]: in multi-model lines set-up times exist between models, hence, lot-sizing becomes an important part of the optimization process [19]. This paper focuses on less restrictive mixed-model lines, in which there is no such model-wise set-up times. Furthermore, demand rates are considered stable enough, so that cyclically scheduling [27] the line is viable. The mixed-model aspect of the problem has also been treated differently by multiple authors: some consider that total processing times (or averages weighted by the demand rates) should be equalized [13, 14, 28]. This consideration can be too optimistic without a sufficient number of buffers [29]. A more common approach is to impose a cycle time constraint on the processing times of all product models [10, 22]. The minimization of such cycle time has also been proposed as a goal function [30]. However, this approach can be pessimistic as the differences between product models can often compensate higher processing times for one model, especially if its demand rate is low [29]. Other goal functions such as workload smoothing [20], vertical balancing [31, 32] and total deviations from average [33, 34] have also been employed, but, as references [35, 36, 37] point out, these are not goals in themselves, but rather supposed means to achieve a high and stable throughput. Tiacci [36] proposes a simulation-based evaluation process to assess the quality of solutions. These simulation-based works employ an assembly line simulator [38] that uses common and intuitive priority and scheduling rules, such as: moving products forward as soon as possible and giving priority to pieces based on when they were completed. These rules have been shown to converge towards accurate steady-state assessments for straight lines without parallelism [29]. However, it is not entirely clear if simulations based on such rules necessarily converge towards the optimal cyclical schedule [27] given a launch order in lines with parallel stations.

Cyclical scheduling [39] is a research field closely related to the studied problem. It consists on providing schedules that can be infinitely repeated after a certain cycle time. A review on cyclical scheduling is presented by [27]. A usual assumption on this area is that machine-wise

processing times of products are known, which in the assembly line context means line balancing (task allocation) is a parameter. Cyclical scheduling is applied to project scheduling [40], robotic cell scheduling [41, 42], to job-shops [43, 44], and flow-shops [45, 46]. The latter category is the problem class which most closely resembles the studied assembly lines.

A recent work [23] has presented a constraint logic programming model to optimize asynchronous mixed-model lines with parallel stations. Its core rationale is to combine balancing and cyclical scheduling to minimize the makespan of multiple replications of the Minimal Part Set [27] (MPS) and, indirectly, minimize the cycle time. A study [29] has shown that this approach does converge towards steady-state optimization as more MPS replications are considered, and presented a formulation to assess the performance of (single station) asynchronous lines with a single MPS replication. An extension of this formulation [47] incorporates sequencing variables for lines without parallelism. However, generalizing [29]'s formulation to lines with parallel workstations is challenging due to the fact that the order products enter a stage can differ from the order products depart from it. Lopes *et al.* [48] describes a simplifying ordering hypothesis (entry order equals departure order) and present the mathematical model to approximately describe the line. This work seeks to remove such simplification and exactly extend the mixed-integer linear programming formulation presented by [29, 47] to incorporate sequencing and cyclical scheduling for asynchronous mixed-model lines with parallel stations. Illustrative examples are given to demonstrate some of the developed features.

The dataset presented by [23] is employed as a benchmark, consequently, all benchmark features are incorporated: Mixed-model balancing, parallel stations, and cyclical scheduling are the main features and this paper's focus. However, [23] employs space constraints, different precedence relations per product model, and model-wise flexible task assignments. These are added to the proposed model so that direct comparisons are possible.

This paper is structured as follows: Section 2 describes the optimization problem. Section 2.1 discusses the proposed cyclical scheduling concept (Extension of the formulation presented by [29, 47]) and how it compares to previous ones. Section 3 presents the developed mathematical model. Section 4 reports the results of computational tests performed on the dataset provided by [23]. Last, Section 5 summarizes the findings and contributions of the paper.

2. Problem Statement

Each product **M**odel m (job) in the set M has a specific set T_m of **T**asks t which must be assigned to one of the **S**tages s in the set S. Tasks can be assigned to different stations for each model, and the union set of all tasks for all models is denoted as T. Each product model m has a set of **P**recedence relations P_m for task pairs (t_1, t_2), which require task t_1 to be performed before task t_2 for the model m. Each stage s has a given amount of **A**vailable space A_s, which is consumed by the assignment of tasks to them: When task t is assigned to the stage s for any model m, it **R**equires a given amount of space $R_{(t,s)}$. The **D**uration of each task t at each stage s for each model m is given by the parameter $D_{(t,m,s)}$. Each stage s has a parallelism degree k_s,

meaning it has k_s stations (workers, machines) and can process up to k_s products simultaneously. Each of the k_s products processed simultaneously in stage s are independent: they can depart the stage s at any time provided that they have completed processing at stage s and stage $s+1$ is not full (i.e. has less than $k_{(s+1)}$ products currently at it). The problem's major physical layout concepts are illustrated by Figure 1.

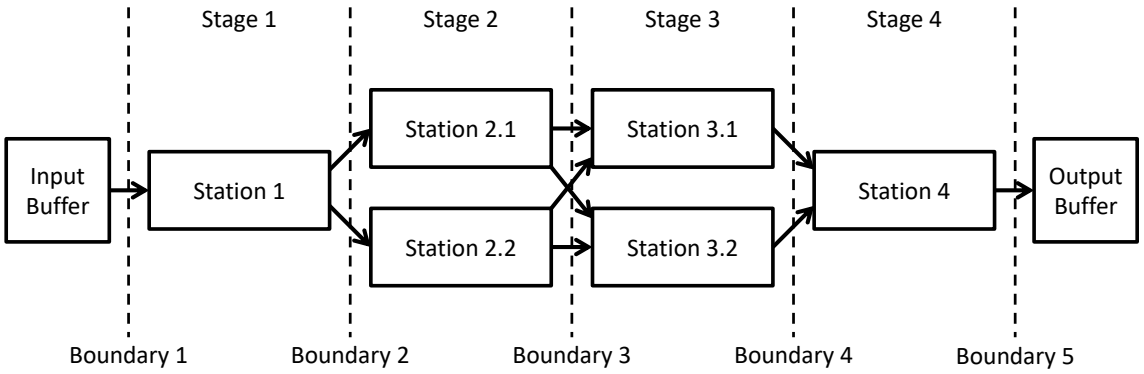


Figure 1: Line layout example with four stages, $k_1 = k_4 = 1$, and $k_2 = k_3 = 2$

The problem consists of providing a line balancing (distributions of tasks to stages for each product model) and cyclical schedule for the Minimal Part Set (MPS) that minimize cycle time (maximize throughput) such that: precedence relations and space constraints are respected; No buffer exists between stages; Infinite buffers exist before the first stage and after the last one.

2.1. Cyclical Scheduling with Parallelism

The standard definition of a cyclical schedule is one that repeats every CT units of time. This period is called the cycle time [27] and it can be interpreted as: if an event happens at a given time t, it happens again for the next MPS at $t + CT$. When there are no parallel machines or workstations, this definition leaves little margin for interpretation, as discussed in Section 1. With parallelism, the most common interpretation [49] for this definition is: if at time t the model m is processed at the k^{th} position (station, machine) of the stage s then at time $t + CT$ the equivalent model of the next MPS is processed at the k^{th} position of the stage s. These positions are often referred to as stations, workers or machines. This cyclical scheduling concept is hereafter referred to as machine-wise cyclical concept. This interpretation implies on a designation of pieces within an MPS to machines (workers, or stations) in each stage.

However, in the context of parallel identical machines, it is not necessary to assign models to specific machines. It is sufficient to ensure that at most k_s models are within stage s at the same time. By defining entry and departure boundaries for each stage, the cyclical schedule can be redefined in terms of when these boundaries are crossed, i.e. when pieces enter and depart each stage. These boundaries are illustrated by Figure 1, and the cyclical nature of their schedules can be defined as follows: if at time t the model m enters (departs from) the stage s, then at time $t + CT$ the equivalent model of the next MPS enters (departs from) the stage s. Notice that no mention of the specific machine is necessary. This subtle difference in interpretation means

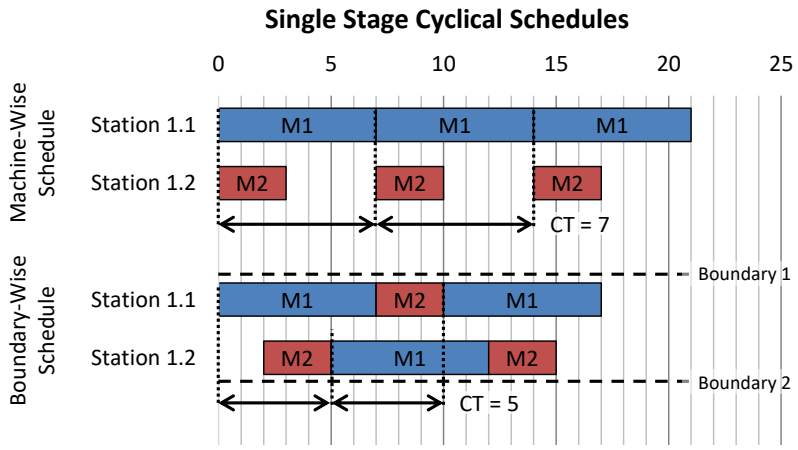


Figure 2: Comparison between machine-wise and boundary-wise optimal cyclical schedules for a single stage example.

that model-machine assignments can differ in each replication of the MPS. This cyclical concept is referred to as boundary-wise cyclical concept.

At first glance, this boundary-wise interpretation might seem to violate the purpose of cyclical scheduling: provide a stable and predictable pattern. However, the boundary-based concept does provide both these results: one full MPS is still guaranteed to be produced every CT Time Units (T.U.), and the schedule is completely defined by one MPS, although this might not be as apparent. The major difference lies in the additional flexibility, which is hereafter illustrated: consider a flowshop with a single stage equipped with two parallel identical machines and an MPS with one unit of products M1 and M2, with processing times 7 and 3 T.U., respectively. If each piece is assigned to a machine, then the minimal cycle time of a cyclical schedule is 7 T.U.. However, if the boundary concept of cyclical schedule is employed, then the minimal cycle time becomes 5 T.U.: one unit of product M1 can enter the stage whenever a unit of product M2 departs it and vice-versa. By doing this, one unit of each product type crosses entry and departure boundaries every 5 T.U.. These schedules are illustrated by Figure 2. From the perspective of the machines, a cyclical pattern is achieved with two replications of the MPS, but from the perspective of the boundaries, the schedule is cyclically defined within a single MPS.

If the flowshop has a single station at each stage, then the second model can only enter it after the first one has departed it. This generalizes to the $(n+1)^{\text{th}}$ model and the n^{th} one. The cycle time is bounded by the time difference between the entry of the first model and the departure of the last one [29]. In the case without parallelism, the boundary and the machine perspectives of cyclical scheduling are essentially the same. However, if the flowshop has multiple parallel stations, these concepts differ: Consider the case with a parallelism degree of $k = 2$ and an MPS with more than two models. The first two (k) models to enter a stage can do so independently. The third ($k+1$) model, however, can only enter the stage after the first model to depart from the stage

departs from it[1]. This generalizes to the $(n+k)^{\text{th}}$ model and the n^{th} one. Now consider the next MPS entering the line. The first model of the next MPS is able to enter the stage after the second last (k^{th} last) model of the first MPS departs it, and the second model of the next MPS is able to enter the stage after the last model of the first MPS departs it. In this case, there are two ways that cycle time is bounded per stage. Contrary to single lines, however, the n^{th} model to enter a stage is not necessarily the n^{th} one to depart it. This can happen due to processing times differences of the models in each stage. Consequently, the order products cross the entrance boundary of a stage is not necessarily the same order they cross the departure one, and hence the cyclical nature of the schedule (while fully described by one MPS) might only become fully apparent in a machine-wise sense when multiple MPSs are considered.

Table 1: Illustrative case data: model-stage processing times (Time Units - T.U.)

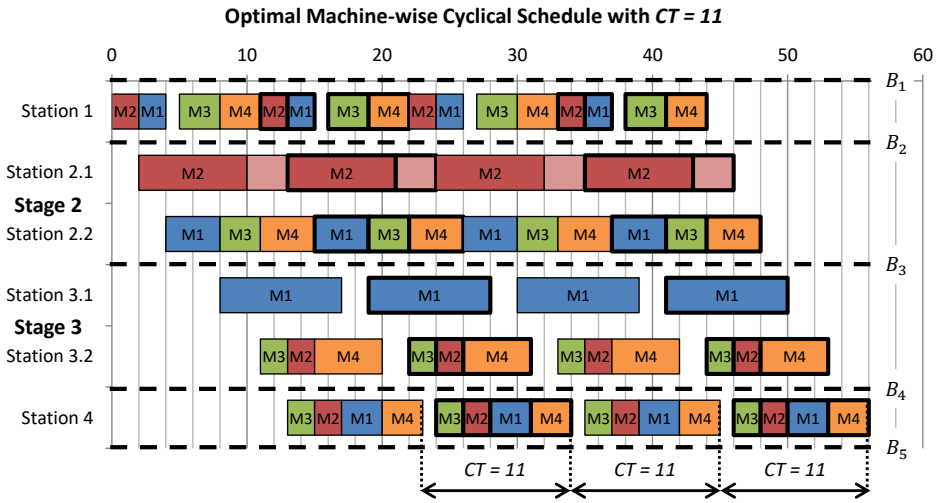
Stage	S1	S2	S3	S4
Model 1	2	4	9	3
Model 2	2	8	2	2
Model 3	3	3	2	2
Model 4	3	4	5	3

Consider an illustrative case with the processing time data from Table 1 and an MPS with one unit of each product model. The case's line layout is presented in Figure 1 (with $k_1 = k_4 = 1$ and $k_2 = k_3 = 2$). Given the processing times at the second stage, it is clear that a machine-wise cyclical schedule cannot arrange models in two groups with a cycle time smaller than 11 T.U.. Indeed, Figure 3a presents one such optimal schedule. However, the boundary-wise optimal shown by Figure 3b cyclic schedule allows a cycle time of 10 T.U.. The boundaries are indicated by the dashed lines labeled $B_1, ..., B_5$. Cycle time measurements are presented bellow the last stage in each schedule.

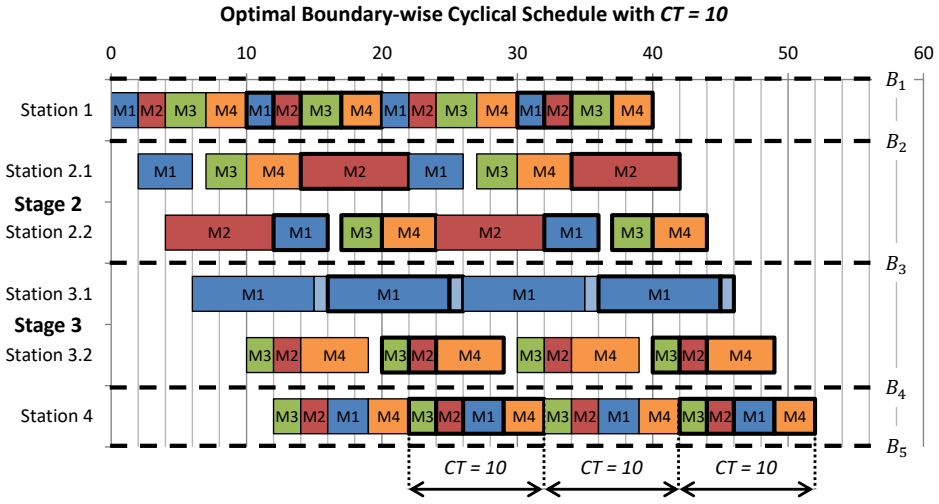
Figure 3b further exemplifies two different blockage concepts that is employed in Section 3, namely regular and cyclical blockage. These concepts are based on the MPS concept: regular blockages occur between pieces of the same MPS, while cyclical blockages occur between pieces of different MPS. While both are tied to scheduling variables, the latter is tied to the steady-state cycle time, hence the label cyclical blockage. Notice that in Figure 3, even MPSs are highlighted with thicker borders. Some examples of these blockage types are hereby provided: In Figure 3b, model M2 "follows" model M1 at the first stage, i.e. it is blocked by it. Between minimal part sets, model M1 "cyclically follows" model M4, i.e. it is cyclically blocked by it. Due to the difference of entry orders and departure orders at each stage, different sets of blockages and cyclical blockages occur per stage.

Notice that the boundary-wise cyclical concept allows more flexible scheduling patterns. The mathematical model presented in Section 3 is defined based on the boundary-wise cyclical schedule

[1]The first model to depart the stage is not necessarily the first one to enter it. This is illustrated by Figure 3b, where model M1 is the first to enter Stage 3, but the third to depart from it.



(a) Optimal machine-wise solution for the illustrative case



(b) Optimal boundary-wise solution for the illustrative case

Figure 3: Machine-wise and Boundary-wise cyclical Schedules

concept.

3. Mathematical Model

The following mixed-integer linear programming model is proposed to describe the problem at hand. In order to do so, the following decision variables are introduced: the continuous variable CT measures the line's cycle time; the binary variable $x_{(t,m,s)}$ is set to 1 if task t is assigned to stage s for product model m; the continuous variables $Tin_{m,s}$, $Tx_{m,s}$, $Tout_{m,s}$ measure entry, processing, and departure times of each model m at, in, and from each stage s. The binary ordering variables $y_{m,n,s}$ are set to 1 if the model m is the n^{th} one to pass through the s^{th} boundary, meaning that it is the n^{th} piece to leave stage $s-1$ and enter stage s. For instance, in Figure 3b, model 1 ($m = 1$) is the first ($n = 1$) model to enter stage 1 ($s = 1$), therefore, $y_{1,1,1} = 1$. Similarly, model 3 is the first

model to enter stage 4, hence $y_{3,1,4} = 1$. The binary $f_{m_2,m_1,s}$ is set to 1 when model m_2 is blocked by model m_1 at stage s. For instance, in Figure 3b, model 4 ($m_2 = 4$) is blocked by model 3 ($m_1 = 3$) at stage 2 ($s = 2$), meaning that model 3 must depart stage 2 so that model 4 can enter it, therefore, $f_{4,3,2} = 1$. Similarly, model 4 is blocked by model 2 at stage 3, hence, $f_{4,2,3} = 1$. Lastly, the binary $cf_{m_2,m_1,s}$ is set to 1 when model m_2 is cyclically blocked by model m_1 at stage s, meaning that m_2 can only enter stage s when model m_1 from the previous MPS departs from it. For instance, in Figure 3b, model 1 ($m_2 = 1$) is cyclically blocked by model 4 ($m_1 = 4$) at stage 1 ($s = 1$), therefore, $cf_{1,4,1} = 1$. Similarly, model 1 is cyclically blocked by itself in stage 3, hence, $cf_{1,1,3} = 1$, meaning it can only enter stage 3 when the model 1 piece of the previous MPS departs from it. The Minimal Part Set is assumed to contain one model of each type ($|M|$ pieces, in total), and different demand rates can be represented by having multiple models with the same processing times. For the sake of simplicity, the model presented in this section assumes that k_s is lower or equal to $|M|$ for all stages s, and a generalization of this model is presented in Appendix A removing such hypothesis.

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

Subject to:

$$\sum_{s \in S} x_{(t,m,s)} = 1 \qquad \forall\, m \in M,\ t \in T_m \tag{2}$$

$$\sum_{t \in T} R_{(t,s)} \cdot \max_{m \in M} x_{(t,m,s)} \leq A_s \qquad \forall\, s \in S \tag{3}$$

$$\sum_{s \in S} s \cdot x_{(t_1,m,s)} \leq \sum_{s \in S} s \cdot x_{(t_2,m,s)} \qquad \forall\, m \in M,\ (t_1, t_2) \in P_m \tag{4}$$

$$Tx_{(m,s)} = \sum_{t \in T_m} D_{(t,m,s)} \cdot x_{(t,m,s)} \qquad \forall\, m \in M,\ s \in S \tag{5}$$

$$CT \geq \sum_{m \in M} Tx_{(m,s)}/k_s \qquad \forall\, s \in S \tag{6}$$

$$\sum_{m \in M} y_{(m,n,s)} = 1 \qquad \forall\, n \in M,\ s \in S \tag{7}$$

$$\sum_{n \in M} y_{(m,n,s)} = 1 \qquad \forall\, m \in M,\ s \in S \tag{8}$$

$$y_{(m,n,s+1)} \leq \sum_{n' \in M : n' \leq n + k_s - 1} y_{(m,n',s)} \qquad \forall\, m \in M,\ n \in M,\ s, s+1 \in S \tag{9}$$

$$f_{(m_2,m_1,s)} \geq y_{(m_2,n+k_s,s)} + y_{(m_1,n,s+1)} - 1 \qquad \forall\, s, s+1 \in S,\ m_1,\ m_2,\ n \in M : n + k_s \leq |M| \tag{10}$$

$$cf_{(m_2,m_1,s)} \geq y_{(m_2,n-|M|+k_s,s)} + y_{(m_1,n,s+1)} - 1 \quad \forall\ s, s+1\ \in\ S,\ m_1,\ m_2,\ n\ \in\ M : n + k_s > |M| \tag{11}$$

$$Tout_{(m,s)} \geq\ Tin_{(m,s)} + Tx_{(m,s)} \qquad \forall\ m\ \in\ M,\ s\ \in\ S \tag{12}$$

$$Tout_{(m,s-1)} =\ Tin_{(m,s)} \qquad \forall\ m\ \in\ M,\ s\ \in\ S : s > 1 \tag{13}$$

$$Tin_{(m_2,s)} \geq Tout_{(m_1,s)} - BigM \cdot (1 - f_{(m_2,m_1,s)}) \quad \forall\ m_1,\ m_2\ \in\ M,\ s\ \in\ S \tag{14}$$

$$CT + Tin_{(m_2,s)} \geq Tout_{(m_1,s)} - BigM \cdot (1 - cf_{(m_2,m_1,s)}) \quad \forall\ m_1, m_2\ \in\ M,\ s\ \in\ S \tag{15}$$

Expression 1 states the problem's goal function, i.e. cycle time minimization. Equation 2 states the balancing occurrence constraint: each task must be assigned for a stage for each product model. Inequality 3 states that the space required for the tasks assigned to a stage[2] is limited by the space available in it (parameter A_s). Notice that models will not necessarily contribute equally to the space constraints for all tasks: each model m has its own set of tasks (T_m) and, therefore, the occurrence constraint (Equation 2) will not affect some balancing binary variables ($x_{t,m,s}$). Expression 4 states the precedence constraints between tasks. With the balancing variables, processing times for each model at each stage are determined by Constraint 5, and a lower bound is defined for cycle time by Expression 6: the total processing time at a stage divided by the parallelism factor. Constraints 7 and 8 control the ordering variables, demanding that each model be in a single position and each position contain a single model. Constraint 9 imposes a consistency constraint in ordering variables of neighboring stages: e.g. With a parallelism degree $k_s = 2$, if model m is the n^{th} to leave the stage s (and enter stage $s+1$) it must have been at most the $(n+1)^{\text{th}}$ to enter the stage s. Notice that if $k_s = 1$ then the order in which each model m enters in a stage is the same order it departs from the stage. Constraint 10 and 11 control the blockage and cyclical blockage binary variables sets f and cf in accordance to the boundary-wise cyclical concept described at Section 2.1. Constraint 12 requires each model's processing to be completed at each stage before departing it. Constraint 13 states that when a model leaves a stage it immediately enters the next one. Constraint 14 states that if model m_2 is blocked by model m_1 at stage s, m_2 can only enter stage s after model m_1 has departed from it, these blockages occur within an MPS. Constraint 15 controls the cycle time by tying the analogous scheduling blockages

[2]The max function in Constraint 3 is non-linear, but also easily linearized. Furthermore, the function is available and automatically linearized by OPL ILOG CPLEX.

between MPS. In Constraints 14 and 15, the relaxation factor $BigM$ can take any valid value of CT multiplied by the highest parallelism factor k_s. The fact that sequencing and blockage variables are allowed to be different for each stage s and the constraints that relate to them (Constraints 7 to 11, 14, and 15) are the key factors that extend the formulation proposed by [29, 47] by removing the ordering hypothesis (requiring the entry orders and departure orders be the same) from, and hence generalizing, the model presented by [48].

The additional formulation (three sequencing variable sets: y, f and cf) complexity required to adequately represent the additional liberties of parallel stations poses a question: is it possible to simply define sequencing variables for the first boundary (line entrance) and to simulate the line's behavior based on common and intuitive scheduling rules? While this approach is feasible [36], it is not guaranteed to lead to optimal cyclical schedules even when an optimal entry order is employed. This fact is demonstrated by an example provided in 4.2.

4. Results

The model described in Section 3 was applied to the 36-instance data set presented by [23]. The summary of most relevant instance information and results are reported by Table 2. The proposed model results are reported under the "Full" column, its previous particularized [48] form is reported under the "Part." column, and the best solution found by Ozturk's [23] model is reported under the benchmark (BMK) column. Table 2 compares the cycle time values of the best incumbent found by each method within the time limit. The proposed model was only solved to optimality for instances marked with 0% Gap (except 22 and 28). The particularized model was solved to optimality for all instances, except the largest three (33-36). The benchmark [23], however, did not report optimality in any instance. This may be due to the model's nature: [23]'s machine-wise formulation is a Constraint Logic Programming model; The approach implemented in [23] lacks a systematic lower bound and might have difficulties in fully exploring large search fields.

In order to set aside software and hardware considerations, all models were executed in the same hardware and software conditions: Tests were performed on a Core i7-3770 CPU (3.4GHz) with 16 GB RAM with a time limit of one hour; The universal solver CPLEX v12.8 was used and the original implementation files provided by [23] were employed. Cycle time comparisons are made between the upper bound of the proposed model and the average of model-wise cycle times reported by [23] implementation files. Models, data, and solution files are provided by this paper's Supplementary Material.

For instances with parallelism that were not solved to optimality, a relaxation of the proposed model was also used to strengthen the lower bounds: By removing the sequencing and scheduling variables and constraints, the reduced model (Expressions 1 to 6) states a balancing lower bound for stable cycle time. The lower bound values reported by Table 2 are the maximum of the best bound found by the proposed model and the optimal answer of the relaxed model. For instances without parallelism, the lower bounds reported by the particularized version of the model [48] are valid and, hence, were also used to define the best lower bounds (column LB of Table 2).

Table 2: Results summary for the 36-instance dataset

Instance info (Parameters)					Solution info (Cycle Time) - T.U.				
i	$\sum_m \lvert T_m \rvert$	$\lvert M \rvert$	$\lvert S \rvert$	k_s	Full	Part.	BMK	LB	Gap
1	38	5	3	1	47	47	47	47	0%
2	38	5	3	2	23.5	23.5	23.8	23.5	**0%**
3	38	5	3	3	15.7	15.7	18	15.7	**0%**
4	38	5	5	1	26	26	26	26	0%
5	38	5	5	2	**12**	13	14.8	12	**0%**
6	38	5	5	3	**8**	8.7	9.8	8	**0%**
7	52	7	3	1	57	57	58	57	0%
8	52	7	3	2	**28**	28.5	31.1	26.5	5.4%
9	52	7	3	3	**18**	19	24.7	17.7	1.7%
10	52	7	5	1	34	34	34	34	0%
11	52	7	5	2	17	17	20.4	16	5.9%
12	52	7	5	3	12	**11.3**	13.1	10.7	5.3%
13	114	5	3	1	107	107	121	107	0%
14	114	5	3	2	**52**	53.5	62.8	52	**0%**
15	114	5	3	3	**34.7**	35.7	40.4	33	4.9%
16	114	5	5	1	68	68	73.4	68	0%
17	114	5	5	2	**32**	34	36	30.5	4.7%
18	114	5	5	3	**22.3**	22.7	24.6	20.3	9%
19	156	7	3	1	142	142	149	142	0%
20	156	7	3	2	**70**	71	73.9	67.5	3.6%
21	156	7	3	3	48	**47.3**	61.4	45	4.9%
22	156	7	5	1	89	**88**	102	88	0%
23	156	7	5	2	46	**44**	61	41	6.8%
24	156	7	5	3	30	**29.3**	32.7	27.3	6.8%
25	190	5	3	1	197	197	197	197	0%
26	190	5	3	2	98.5	98.5	103.6	98.5	**0%**
27	190	5	3	3	**64.3**	65.7	80.4	64.3	**0%**
28	190	5	5	1	114	**113**	116	113	0%
29	190	5	5	2	**56**	56.5	63.2	50	10.7%
30	190	5	5	3	**36**	37.7	46.4	33.3	7.5%
31	260	7	3	1	263	263	267	263	0%
32	260	7	3	2	**131**	131.5	139	126	3.8%
33	260	7	3	3	**86**	87.7	104.1	84	2.3%
34	260	7	5	1	156	**155**	165	135	12.9%
35	260	7	5	2	**76**	77.5	87.3	67.5	11.2%
36	260	7	5	3	53	**51.7**	57.1	45	13%

For most instances (all except 1, 4, 10, and 25, in which all formulations tied), the proposed model (and its particularized version) outperformed the benchmark [23]. Out of the 36 instances, 18 were solved to optimality. However, most of them (11) are instances without parallelism ($k_s = 1$). This highlights the additional complexity tied to parallel stations: sequencing variables have more flexibility as the order models enter and depart stages is not necessarily the same.

Comparing the proposed model (Full) to its particularized version (Part.) [48], it is clear that the additional flexibility (removal of the ordering hypothesis) comes with a trade-off convergence difficulty: In 15 instances (boldfaced in column Full), removing the ordering hypothesis leads to better answers than the particularized version. All these answers are unfeasible for the particularized model due to the ordering hypothesis. However, in 8 instances (boldfaced in column Part.), the particularized version of the model found better solutions, all of which are feasible for the generalized version - these were not found due to time limit. Overall, the additional flexibility of the generalized model did produce improvements: fifteen new best answers (boldfaced in column Full), four of which are optimal ones (instances 5, 6, 14, and 27), and three new optimality proofs (instances 2, 3, and 26).

4.1. Parallelism influence

A close analysis of the input data provided by [23] revealed that instances are equal in groups of three, differing only by the parallelism degree: Instances 1-3 are identical except for k_s, this is also the case for instances 4-6, 7-9, etc. Furthermore, the k_s parameter is constant and equal for all stages of the same instance. This means that the performance of instances in the same group can be normalized if their cycle time values are multiplied by k_s. This comparison then allows the verification of whether or not station parallelism (cross-overs between parallel lines) allows better performance than to simply have parallel identical and independent lines. The results of the comparisons are summarized by Table 3, in which the best answers for each instance is used to generate the normalized cycle time values, and optimal answers are underlined.

Table 3: Normalized Cycle Time Comparisons

Instance	Normalized Cycle Times - T.U.			Improvements	
Group	$k_s = 1$	$k_s = 2$	$k_s = 3$	2 vs 1	3 vs 1
1-3	47	47	47	0%	0%
4-6	26	**24**	**24**	7.7%	7.7%
7-9	57	**56**	**54**	1.8%	5.3%
10-12	34	34	34	0%	0%
13-15	107	**104**	**104**	2.8%	2.8%
16-18	68	**64**	**67**	5.9%	1.5%
19-21	142	**140**	142	1.4%	0%
22-24	88	88	88	0%	0%
25-27	197	197	**193**	0%	2%
28-30	113	**112**	**108**	0.9%	4.4%
31-33	263	**262**	**258**	0.4%	1.9%
34-36	155	**152**	155	1.9%	0%
			Average	1.9%	2.1%
			Maximum	7.7%	7.7%

Notice that in 15 out of 24 cases with parallelism the solution found by the model is better than k_s times the solution without parallelism ($k_s = 1$). In average, this improvement rate is close to 2%, and in the best case it is 7.7%. This demonstrates that parallel workstations can offer an increasing marginal value in the following specific sense: the use of twice (three times) as many resources lead to results better than double (triple) that of those obtained by the base quantity. This is due to the fact that parallelism allows more scheduling flexibility: entry and departure orders at each stage are allowed to differ. Notice that an analogous increasing marginal value of workstations is not possible for a serial line in a Simple Assembly Line Balancing (SALB) context: doubling (tripling) the number of serial single stations can at most half (reduce to a third) the line's cycle time. This fact is hereby proved by contradiction: suppose a solution with $n \cdot k$ stations exists with cycle time $CT_{n \cdot k}$ lower than $1/k$ the optimal cycle time CT_n for the line with n stations ($CT_n > k \cdot CT_{n \cdot k}$). By merging task-stations assignments of the solution in groups of k stations ($1 \ldots k, k+1 \ldots 2k$, etc.), a feasible (precedence relations are respected) balancing solution for n stations is produced with $CT_n \leq k \cdot CT_{n \cdot k}$.

In 14 out of these 15 cases (all except instance 35), the proposed formulation outperformed the lower bound obtained by its particularized version [48] tied to the ordering hypothesis (imposing entry order equal to departure order at every stage). These results show that the proposed model is capable of taking advantage of parallelism increased flexibilities, and that the imposition of the ordering hypothesis can prune part of the valid search field.

4.2. Comparison to Completion-Based Priority Rules

Multiple recent works have employed simulation techniques [12, 35, 36, 37] to measure performance. These are based on a simulator [38] that employs common priority rules for sequencing and scheduling decisions: pieces move to the next station as soon as they can, and priority is given to the first piece to be completed in case more than one can move. These common scheduling rules are hereafter referred to as Completion-Based Priority Rules (CBPR). It might seem at first that a simulation approach based on CBPR (such as the simulator proposed by [38]) would necessarily lead to the optimal steady-state given a cyclic entry order in the flowshop. This does work for lines without parallelism as simulations gradually converge towards the steady-state [29]. However, that is not necessarily true when parallelism is present. In order to demonstrate the necessity of multiple sets of sequencing variables (one for each boundary), an illustrative example is hereby given.

Consider the cyclical scheduling problem associated to the processing time data presented by Table 4. Consider an MPS with one unit of each of the three product models and consider that both stages have a parallelism degree of two.

Assuming the cyclic entry order (M1,M2,M3) at least two different cyclical schedules are possible: one by simulating the piece flow based on the CBPR rules, i.e. by having pieces move to the next stage as soon as they can with priority to pieces that are completed first; the other by using the proposed model and allowing sequencing variables at each remaining boundary, i.e. by fixing

Table 4: CBPR vs. Model example processing time data (Time Units - T.U.)

Stage	S1	S2
Model 1	3	7
Model 2	7	6
Model 3	5	3

only those tied to the first entrance boundary. These solutions are presented by Figure 4. In it, even MPSs are highlighted with thicker borders and B1, B2, and B3 stand for Boundary 1, 2, and 3, respectively.

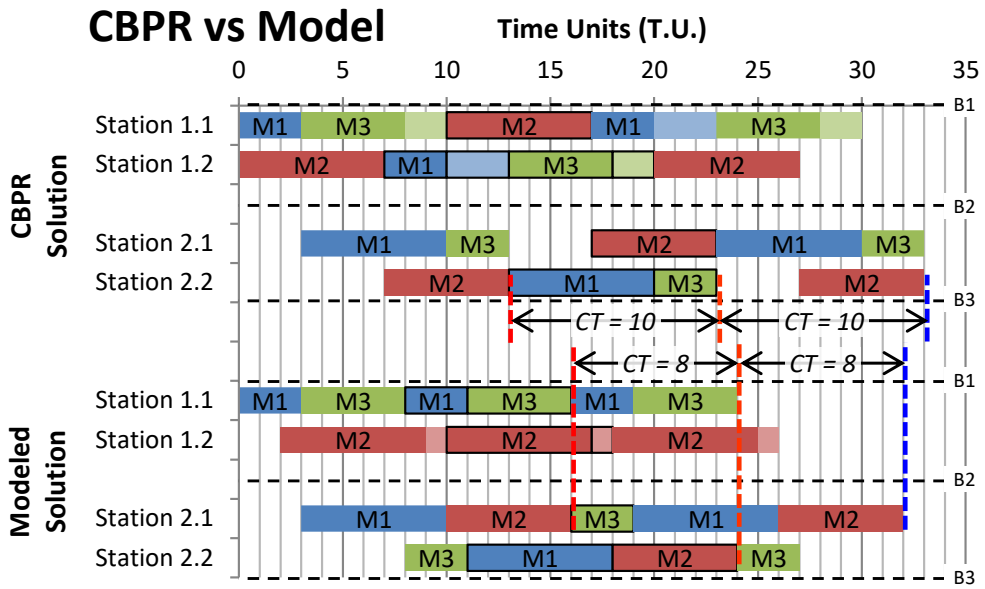


Figure 4: Comparison of CBPR vs modeled solution, imposing the same cyclical entry order at the first stage

Notice that the modeled solution has a cycle time of 8 T.U., while the CBPR one has a cycle time of 10 T.U.. The 2 T.U. waiting time before the entrance of M2 at the first MPS replication leads to a higher makespan value for the first MPS for the modeled solution (16 vs 13). However, due to the lower value of cycle time, this difference decreases steadily after each replication of the minimal part set. After the third replication, the better cyclical behavior dominates the worse transient one, meaning that the modeled solution has a better makespan as well (32 vs 33). The dashed colored lines in Figure 4 highlight these makespan differences.

This example demonstrates that a CBPR approach that only takes into account the entry order at the first stage is insufficient as a cyclical performance measurement: while CBPR simulations can generate cyclical schedules, they are not guaranteed to be able to lead to the best possible one. By induction, this argument demonstrates the importance of sequencing variables at every boundary.

5. Conclusions

Mixed-model assembly lines are often employed in industrial contexts as they balance product diversity and production efficiency. Balancing workloads (task distribution) and cyclically scheduling these lines are key to obtain a high and stable throughput. In the context of unpaced asynchronous lines in particular, blockages and starvations must be taken into account to optimize the flowshop. Parallel workstations offer further flexibility to the line by allowing stations to act as partial buffers and by enabling more sequencing and scheduling possibilities.

Authors thus far have defined cyclical schedules of flowshops in general (and assembly lines in particular) by a machine-wise cyclical concept. In that concept, the cycle time is defined as the time between equivalent events of different minimal parts sets at each machine. However, by defining boundaries between stages, as indicated by Figure 1, better and more general cyclical schedules are possible when a boundary-wise cyclical concept is employed. This formulation is introduced in this paper, both conceptually (Section 2.1) and formally (Section 3). This formulation extends a previous simple station (no parallelism) one [29, 47], and generalizes another that incorporated parallel stations [48] by removing a simplifying hypothesis. An illustrative example (Figure 3) is provided to demonstrate the higher flexibility obtained by the boundary-wise cyclical concept. While for assembly lines without parallelism the machine-wise and boundary-wise formulations are the same, this is shown not to always be the case for assembly lines with parallelism. The proposed formulation is also shown to allow better schedules than those obtained by scheduling with common priority rules (often employed by simulation-based approaches) the assembly line with a given entry order (Section 4.2).

A mixed-integer linear programing model is presented to solve the asynchronous mixed-model assembly line simultaneous balancing and cyclical scheduling problem. The proposed formulation is capable of representing cyclical schedules for any number of stages and different numbers of parallel workstations in each stage. Tests on a 36-instance literature benchmark provided new best-known solutions for 15 instances, with 4 new optimal ones, and 3 new optimality proofs, as summarized by Table 2. Furthermore, by analyzing instances that only differed by the parallelism degree it was possible to show that unpaced lines with parallel stations can outperform parallel unpaced lines with single stations (Table 3): Cycle times found for instances with parallelism are, in average, 2% better than cycle times found for the equivalent instances without parallelism divided by the parallelism degree. This difference is as high as 7.7% for one instance in particular.

The results summarized in Table 3 demonstrate that, aside from allowing higher recovery from failure capacity, parallel workstations can allow a crescent marginal value: it might be possible to more than double (triple) the production rate by doubling (tripling) the amount of resources. Further works should seek to apply the proposed boundary-wise cyclical formulation to other scheduling problems with parallelism such as job-shop and flowshop. Alternatively, parts of the formulation can be adapted and extended to other classes of line balancing problems, in which scheduling plays a major role, such as multi-manned and set-up based ones.

Acknowledgments

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

Appendix A. Formulation Extension

The MILP model presented in Section 3 is built under the hypothesis that the parallelism degree k_s is smaller or equal than the number of pieces in the minimal part set $|M|$ at every stage s. This might, however, not be the case: in some very specific environments, very long tasks might require degrees of parallelism k_s to be higher than $|M|$ for a specific stage.

The generalization of the proposed model for such cases is based on the stage-wise correction factor j_s, whose value is defined by the Equation A.1 for each stage s. This factor states the smallest integer degree to which k_s is greater than $|M|$. For instance, with 5 pieces in the MPS, j_s is defined as 0, 1, 2, respectively, when k_s belongs to the ranges $\{1, \dots, 5\}$, $\{6, \dots, 10\}$, $\{11, \dots, 15\}$.

$$j_s = \left\lfloor \frac{k_s - 1}{|M|} \right\rfloor \qquad \forall\ s \in S \tag{A.1}$$

With the correcting factor j_s, four expressions must be revised: Constraints A.2 and A.3 replace Constraints 10 and 11; and Constraints A.4 and A.5 replace Constraints 14 and 15. Notice that when j_s is zero (i.e. the hypothesis $k_s \leq |M|$ holds) the reviewed constraints are the same as the ones presented in Section 3.

$$\begin{aligned} f_{(m_2,m_1,s)} \geq\ & y_{(m_2,n+k_s-j_s\cdot|M|,s)} + y_{(m_1,n,s+1)} - 1 \\ & \forall\ s, s+1 \in S,\ m_1,\ m_2,\ n \in M : n + k_s - j_s \cdot |M| \leq |M| \end{aligned} \tag{A.2}$$

$$\begin{aligned} cf_{(m_2,m_1,s)} \geq\ & y_{(m_2,n-(j_s+1)\cdot|M|+k_s,s)} + y_{(m_1,n,s+1)} - 1 \\ & \forall\ s, s+1 \in S,\ m_1,\ m_2,\ n \in M : n + k_s - j_s \cdot |M| > |M| \end{aligned} \tag{A.3}$$

$$\begin{aligned} CT \cdot j_s + Tin_{(m_2,s)} \geq & Tout_{(m_1,s)} - BigM \cdot (1 - f_{(m_2,m_1,s)}) \\ & \forall\ m_1, m_2 \in M,\ s \in S \end{aligned} \tag{A.4}$$

$$\begin{aligned} CT \cdot (j_s + 1) + Tin_{(m_2,s)} \geq & Tout_{(m_1,s)} - BigM \cdot (1 - cf_{(m_2,m_1,s)}) \\ & \forall\ m_1, m_2 \in M,\ s \in S \end{aligned} \tag{A.5}$$

References

[1] A. Scholl, Balancing and sequencing assembly lines, 2nd Edition, Physica, Heidelberg, 1999.

[2] N. Boysen, M. Fliedner, A. Scholl, A classification of assembly line balancing problems, European Journal of Operational Research 183 (2) (2007) 674–693. doi:10.1016/j.ejor.2006.10.010.

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

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

[5] A. Hamzadayi, G. Yildiz, 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) (2012) 206–215. doi:10.1016/j.cie.2011.09.008.

[6] C. Öztürk, S. Tunali, B. Hnich, M. A. Örnek, Balancing and scheduling of flexible mixed model assembly lines, Constraints 18 (3) (2013) 434–469. doi:10.1007/s10601-013-9142-6.

[7] I. Kucukkoc, D. Z. Zhang, Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines, International Journal of Advanced Manufacturing Technology 82 (2016) 265–285. doi:10.1007/s00170-015-7320-y.

[8] A. Scholl, C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, European Journal of Operational Research 168 (3) (2006) 666–693. doi:10.1016/j.ejor.2004.07.022.

[9] O. Battaïa, A. Dolgui, A taxonomy of line balancing problems and their solution approaches, International Journal of Production Economics 142 (2) (2013) 259–277. doi:10.1016/j.ijpe.2012.10.020.

[10] S. Akpinar, A. Baykasoglu, Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model, Journal of Manufacturing Systems 33 (2014) 177–187. doi:10.1016/j.jmsy.2013.11.004.

[11] S. Akpinar, A. Baykasoglu, Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm, Journal of Manufacturing Systems 33 (4) (2014) 445–461. doi:10.1016/j.jmsy.2014.04.001.

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

[13] A. Alghazi, M. E. Kurz, Mixed model line balancing with parallel stations, zoning constraints, and ergonomics, Constraints 23 (2018) 123–153. doi:10.1007/s10601-017-9279-9.

[14] T. Sawik, Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line, European Journal of Operational Research 143 (1) (2002) 115–124. doi:10.1016/S0377-2217(01)00328-9.

[15] M. Chica, Ó. Cordón, S. Damas, J. Bautista, Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search, Information Sciences 180 (18) (2010) 3465–3487. doi:10.1016/j.ins.2010.05.033.

[16] C. Öztürk, S. Tunali, B. Hnich, A. Örnek, Balancing and scheduling of flexible mixed model assembly lines with parallel stations, International Journal of Advanced Manufacturing Technology 67 (9-12) (2012) 255–2591. doi:10.1007/s00170-012-4675-1.

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

[18] C. G. S. Sikora, T. C. Lopes, L. Magatão, Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies, European Journal of Operational Research 259 (3) (2017) 949–971. doi:10.1016/j.ejor.2016.11.027.

[19] N. Boysen, M. Fliedner, A. Scholl, Assembly line balancing: Which model to use when?, International Journal of Production Economics 111 (2) (2008) 509–528. doi:10.1016/j.ijpe.2007.02.026.

[20] I. Kucukkoc, D. Z. Zhang, Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines, International Journal of Production Economics 158 (2014) 314–333. doi:10.1016/j.ijpe.2014.08.010.

[21] T. Kellegöz, Assembly line balancing problems with multi-manned stations: a new mathematical formulation and Gantt based heuristic method, Annals of Operations Research 253 (1) (2017) 377–404. doi:10.1007/s10479-016-2156-x.

[22] A. Roshani, F. G. Nezami, Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach, Assembly Automation 37 (1) (2017) 34–50. doi:10.1108/AA-02-2016-016.

[23] C. Öztürk, S. Tunali, B. Hnich, A. Örnek, Cyclic scheduling of flexible mixed model assembly lines with parallel stations, Journal of Manufacturing Systems 36 (3) (2015) 147–158. doi:10.1007/s00170-012-4675-1.

[24] B. Rekiek, A. Dolgui, A. Delchambre, A. Bratcu, State of art of optimization methods for assembly line design, Annual Reviews in Control 26 (2) (2002) 163–174. doi:10.1016/S1367-5788(02)00027-5.

[25] J. Bukchin, J. Rubinovitz, A weighted approach for assembly line design with station paralleling and equipment selection, IIE Transactions 35 (1) (2003) 73–85. doi:10.1080/07408170304429.

[26] A. Lusa, A survey of the literature on the multiple or parallel assembly line balancing problem, European Journal of Industrial Engineering 2 (1) (2008) 50–72. doi:10.1504/EJIE.2008.016329.

[27] E. Levner, V. Kats, D. A. L. de Pablo, T. C. E. Cheng, Complexity of cyclic scheduling problems: A state-of-the-art survey, Computers and Industrial Engineering 59 (2) (2010) 352–361. doi:10.1016/j.cie.2010.03.013.

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

[29] T. C. Lopes, C. G. S. Sikora, A. S. Michels, L. Magatão, Mixed-Model Assembly Line Balancing with Given Buffers and Product Sequence: Model, Formulation Comparisons and Case Study, Annals of Operations Research 1 (1) (2018) 1–26. doi:10.1007/s10479-017-2711-0.

[30] S. Karabati, S. Sayin, Assembly line balancing in a mixed-model sequencing environment with synchronous transfers, European Journal of Operational Research 149 (2) (2003) 417–429. doi:10.1016/S0377-2217(02)00764-6.

[31] U. Saif, Z. Guan, W. Liu, B. Wang, C. Zhang, Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line, International Journal of Advanced Manufacturing Technology 75 (2014) 1809–1827. doi:10.1007/s00170-014-6153-4.

[32] Y.-g. Zhong, Hull mixed-model assembly line balancing using a multi-objective genetic algorithm simulated annealing optimization approach, Concurrent Engineering: Research and Applications 25 (1) (2017) 30–40. doi:10.1177/1063293X16666204.

[33] Y. K. Kim, S. J. Kim, J. Y. Kim, Balancing and sequencing mixed-model U-lines with a coevolutionary algorithm, Production Planning and Control: The Management of Operations 11 (8) (2000) 754–764. doi:10.1080/095372800750038355.

[34] Y. K. Kim, J. Y. Kim, Y. K. Kim, An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines, European Journal of Operational Research 168 (3) (2006) 838–852. doi:10.1016/j.ejor.2004.07.032.

[35] L. Tiacci, 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 (2015) 201–215. doi:10.1016/j.ijpe.2015.01.022.

[36] L. Tiacci, 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 (2015) 319–333. doi:10.1016/j.ijpe.2014.05.005.

[37] L. Tiacci, Mixed-model U-shaped assembly lines: Balancing and comparing with straight lines with buffers and parallel workstations, Journal of Manufacturing Systems 45 (2017) 286–305. doi:10.1016/j.jmsy.2017.07.005.

[38] L. Tiacci, Event and object oriented simulation to fast evaluate operational objectives of mixed model assembly lines problems, Simulation Modelling Practice and Theory 24 (2012) 35–48. doi:10.1016/j.simpat.2012.01.004.

[39] C. Hanen, A. Munier, Cyclic scheduling on parallel processors: an overview, in: P. Chretienne, E. G. Coffman, J. K. Lenstra, Z. Liu (Eds.), Scheduling Theory and Its Applications, John Wiley & Sons LTd, 1994, Ch. 4. doi:10.2298/SOS1102133M.

[40] B. D. de Dinechin, A. M. Kordon, Converging to periodic schedules for cyclic scheduling problems with resources and deadlines, Computers and Operations Research 51 (2014) 227–236. doi:10.1016/j.cor.2014.03.004.

[41] N. Brauner, Identical part production in cyclic robotic cells: Concepts, overview and open questions, Discrete Applied Mathematics 156 (13) (2008) 2480–2492. doi:10.1016/j.dam.2008.03.021.

[42] A. Elmi, S. Topaloglu, Cyclic job shop robotic cell scheduling problem: Ant colony optimization, Computers and Industrial Engineering 111 (2017) 417–432. doi:10.1016/j.cie.2017.08.005.

[43] P. Brucker, T. Kampmeyer, A general model for cyclic machine scheduling problems, Discrete Applied Mathematics 156 (13) (2008) 2561–2572. doi:10.1016/j.dam.2008.03.029.

[44] F. Quinton, I. Hamaz, L. Houssin, Algorithms for the Flexible Cyclic Jobshop Problem, in: 14th IEEE International Conference on Automation Science and Engineering (CASE 2018), Munich, Germany, 2018, p. 5.

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

[46] W. Bozejko, M. Uchroński, M. Wodecki, Parallel metaheuristics for the cyclic flow shop scheduling problem, Computers and Industrial Engineering 95 (2016) 156–163. doi:10.1016/j.cie.2016.03.008.

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

[48] T. C. Lopes, C. G. S. Sikora, A. S. Michels, L. Magatão, A New Model for Simultaneous Balancing and Cyclical Sequencing of Asynchronous Mixed-Model Assembly Lines with Parallel

Stations, in: Annals of the XLIX Brazilian Symposium of Operations Research, Blumenau, 2017, pp. 3570–3581.

[49] M. L. Pinedo, Scheduling: Theory, algorithms, and systems, 3rd Edition, Springer, 2008. doi:10.1007/978-0-387-78935-4.