Implementing Changes in the Balancing of Assembly Lines: a Sequencing Problem Addressed by MILP

Celso Gustavo Stall Sikora
Federal University of Technology - Paraná (UTFPR)
Graduate Program in Electrical and Computer Engineering (CPGEI)
Av. Sete de Setembro, 3165, Curitiba-PR, Brasil, 80230-901
sikora@alunos.utfpr.edu.br

Thiago Cantos Lopes
Federal University of Technology - Paraná (UTFPR)
Graduate Program in Electrical and Computer Engineering (CPGEI)
Av. Sete de Setembro, 3165, Curitiba-PR, Brasil, 80230-901
thiagolopes@alunos.utfpr.edu.br

Leandro Magatão
Federal University of Technology - Paraná (UTFPR)
Graduate Program in Electrical and Computer Engineering (CPGEI)
Av. Sete de Setembro, 3165, Curitiba-PR, Brasil, 80230-901
magatao@utfpr.edu.br

ABSTRACT

The productivity levels of an assembly line are intimately tied to the balancing quality of the workstations. Several optimization methods focus on finding the best balancing configuration for productive lines, but very few is published about implementing these solutions. When a line's actual configuration greatly differs from the new project, a scheduling of stepwise interventions can increase the implementation viability. In this paper, a mixed-integer linear programming model for dividing the implementation in feasible steps is presented. An integrated balancing and sequencing model is used to minimize the effort or cost of implementations. Case studies are presented to illustrate industrial circumstances in which implementations have to be performed by steps and the effect on the multi-period planning. A stepwise implementation divides the changes in minimal packages, increasing the viability and applicability of (re)balancing projects.

KEYWORDS. Assembly Line Balancing; Implementation Sequencing; Implementation Planning

Main Area: AD&GP - OR in Administration & Production Management, IND - OR in Industry

1. Introduction

Assembly lines consist of a series of workstations designed to produce a high quantity of standardized products. Operations or tasks define the amount of work or time assigned to each station. Some tasks must obey precedence relations, that is, some operations depend on the conclusion of other tasks. Once the most loaded workstation is the bottleneck of the line, the task allocation is of great importance for the productivity of a line.

The Assembly Line Balancing Problem (ALBP), firstly defined by Salveson [1955], consists in finding the optimal allocation of tasks within the stations of an assembly line. A set of simplification hypothesis is often applied to balancing problems, resulting in the Simple Assembly Line Balancing Problem (SALBP). The production of a single product, deterministic processing times for the tasks, and the capability of any station to perform any task are examples of the simplifications described by Baybars [1986]. The SALBP simplifications hardly represent real-world assembly lines. Their solution methods, however, are usually integrated in other extensions of the problem [Scholl e Becker, 2006].

Recent surveys present reasons for the gap between the application of optimization methods addressed to assembly lines in the industry. Becker e Scholl [2006] focus on General Assembly Line Balancing Problems (GALBP). According to the survey, several real-world conditions (such as equipment selection, station paralleling, U-shaped lines and mixed-model lines) have been identified and modeled. However, sophisticated solution methods for GALBP are not in the same pace to the ones of SALBP.

The lack of easy-to-use software containing state-of-art optimization methods is pointed in a survey by Boysen et al. [2008]. According to the authors, only 15 out of more than 300 considered articles present results of a real-world assembly line. Even though the work of Arcus [1965] and Bartholdi [1993] presented a software interface for their heuristic methods, real case scenarios [Agnetis et al., 2007; Battini et al., 2007; Bautista e Pereira, 2007; Lapierre et al., 2006] are mostly used for testing the solution method.

The more recent survey of Battaïa e Dolgui [2013] shows that the modeling and solution methods for GALBP are evolving: the balancing problem is often treated along with related problems. The survey showed a recent focus on robust optimization due to the uncertainty of the input data.

Although the great part of papers treat the modeling and solving of balancing problems, implementation conditions may also be considered in the optimization process. Otto e Otto [2014] present a method of constructing the precedence relations of the tasks of an assembly line gathering data from multiple sources. Their algorithm uses historic data of assembly lines to build an approximative precedence diagram. The simplification reduces the warm-up period of study and data acquisition of projects.

Assembly lines usually work uninterruptedly during weekdays, reducing the time available for interventions. The redesign of lines is often limited to scheduled maintenance stops at collective vacations. During production periods, however, a project that requires significant modifications in the line might be put on hold. The interventions on assembly lines take time and incur on moving, installing, and training costs. While models focus in the final configuration of the balancing, little is published about implementing the necessary changes.

In this paper, we point to another implementation characteristic that, as the best knowledge of the authors, was not yet treated in the literature: the sequencing of implementation of the balancing in an assembly line, presented in Section 2. In Section 3, a Mixed Integer Linear Programming Model is proposed to optimally divide implementations in stepwise interventions. Two study cases inspired on real-world implementation cases are presented and discussed on Section 4. Concluding remarks are presented in Section 5.

2. The Assembly Line Balancing Implementation Sequencing Problem

Variations in the market demand and alterations in the product are common reasons to adapt an assembly line to a new configuration. Whenever we reconfigure a flow line, the assignment of tasks change from the actual or initial conditions to the goal configuration. Moving a task has a cost that can be measured either monetarily or temporally: pieces of equipment have to be moved and installed, machines and robots have to be reprogrammed, workers have to be instructed and trained, and process files have to be updated.

When the cost of a change implementation is low, the reassignments can be performed in a single step. On the other hand, when the initial and final conditions significantly differ, a high cost of implementation is expected. In those cases, instead of rebuilding an assembly line entirely, a series of smaller changes can be performed to achieve the goal configuration. Stepwise implementations can increase the viability of projects that would otherwise be impossible due to the long time of line stoppage they would require.

Sequencing and planning changes is not trivial though. The production between each step of the implementation has to be within accepted levels. All tasks must be performed, the precedence relations must be obeyed and the cycle time must be restricted by production requirements. The Assembly Line Balancing Implementation Sequencing Problem (ALBISP) consists of sequencing the assignment changes in the implementation of a new line while respecting its balancing restrictions. The ALBISP's objective is to minimize the cost of a stepwise implementation that only proposes valid balancing allocations. The result of such problem would answer in which transaction each of the tasks should be moved and the resulting balancing for each of the implementation's steps.

For SALBP, Baybars [1986] defined variations based on the parameters given and the objective of the optimization, namely, SALBP-1 and SALBP-2. These problems have a dual relation [Scholl e Becker, 2006]. In the type-1 problem, the cycle time of the assembly line is fixed, while the objective is to determine the line's minimal number of workstations. For the second variation, we minimize the cycle time of the line for a given number of workstations.

Similarly to the SALBP classification, a dual relation on fixed parameters and optimizing objectives is also observed in ALBISP. When a given amount of resources is given, such as the size of a crew or the maximal value of cost for each implementation step, the ALBISP-1 is defined. This problem is solved for the minimal number of periods necessary for the implementation. In the ALBISP-2, the cost of the implementation steps is to be minimized for a given number of steps. In other words, the second variation is the sizing of a crew or the resources needed to divide the implementation in a given schedule.

For this work, the second version of the problem is treated. The model presented in Section 3 and the case studies in Section 4 are therefore related to ALBISP-2.

3. Mathematical Model

In this section we present a mixed integer linear programming model for the Assembly Line Balancing Implementation Sequencing Problem. The sequencing can be seen as several parallel balancing problems linked by the amount of changes between each stage. For every stage, a feasibility problem in spite of the balancing must be solved: the cycle time of the line must be within a given bound. At the same time, the model seeks the minimal amount of effort needed to implement the intermediary solutions.

The balancing part of the problem consists of tasks (t) and workstations (s), while the sequencing problem presents stages or periods (p) and transitions or changes (c) between stages. These components are used as indexes for the sets and variables. The necessary parameters for the model are given in Table 1. The sets in Table 2 are used to define variables and restrictions in the model. They are function of the user given parameters.

Figure 1 illustrates how the problem is defined. The parameter NP defines the number of stages in which the model has to find an allocation, represented by white squares. The initial and final conditions (fixed stages), in gray, are parameters of the problem. We consider one transition

Table 1: Parameters used in the model.

Parameter	Element	Description
NT	t	number of tasks in the line
NS	s	number of workstations in the line
NP	p	number of periods considered in the problem (excluding initial and final conditions)
TD	TD_t	the duration time of each task t
CT_{max}	-	limit cycle time for the considered stages (periods)
$Prec$	t_p, t_s	set of precedence relations between a precedent (t_p) and a successor task (t_s)
FTA	t, s	set of the Feasible Task Allocations. The construction of this set is further explained in Subsection 3.3
$InitAlloc$	t, s, A	set of the initial allocation. A is a binary value (parameter) that represents whether task t is assigned to station s
$FinalAlloc$	t, s, A	set of the final allocation. A is a binary value (parameter) that represents whether task t is assigned to station s

Table 2: Sets used in the model.

Set	Element	Description
$Tasks$	t	set of the tasks that must be performed: $1...NT$
$Stations$	s	set of workstations present in the line: $1...NS$
$Periods$	p	set of the stages or periods considered: $1...NP$
$Transitions$	c	set of the transitions or changes between stages: $1...NP+1$
$InterTran$	c	set of transitions between two intermediary stages: $2...NP$
TSP	t, s, p	set of possible allocations of a task t in station s in period p equal to FTA for every value of p
TSC	t, s, c	set of possible changes in the allocation of task t in station s at transition c equal to FTA for every value of c

between every two stages. Once there are $NP + 2$ stages along with the fixed stages, $NP + 1$ transitions are necessary to define the problem. Furthermore, an intermediary transition is defined when it relates to two non-fixed stages ($NP - 1$ intermediary transitions).

Based on these definitions, the variables used in the model are:

Model's variables:

$Effort$: effort or cost of implementing a transition

$$x_{t,s,p} : \begin{cases} 1, \text{ if task } t \text{ is assigned to station } s \text{ at stage } p \\ 0, \text{ otherwise} \end{cases} \qquad \forall (t, s, p) \in TSP$$

$$y_{t,s,c} : \begin{cases} 1, \text{ if the assignment of task } t \text{ in station } s \\ \quad \text{becames true at transition } c \\ 0, \text{ otherwise} \end{cases} \qquad \forall (t, s, c) \in TSC$$

The model's objective function is to minimize the variable $Effort$. This value can be seen as a cost in hours, manpower or financial resources needed to perform the implementation of the transitions between stages. In this model, the number of stages (NP) is given while the amount of work needed in each transition is minimized (ALBISP-2).

3.1. The Balancing Problem Formulation

The model's restrictions can be divided into two groups: balancing and sequencing restrictions. The balancing model is based on the SALBP model of Patterson e Albracht [1975], with the difference that every restriction must also hold for each stage p.

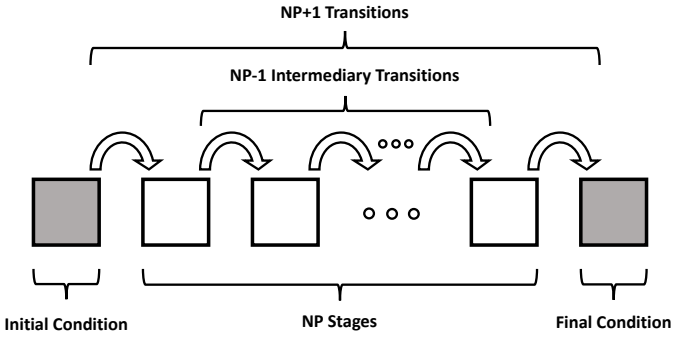

Figure 1: Parameters identification in the proposed model. A period consists in any intermediary stage between the initial and the final conditions. The set $InterTran$ is related only to intermediary stages while the set $Transitions$ also contains transitions from the initial and final stages.

$$\sum_{(t,s)\in FTA} x_{t,s,p} \cdot TD_t \leq CT_{max} \qquad \forall\, s \in Stations,\ p \in Periods \qquad (1)$$

$$\sum_{(t,s)\in FTA} x_{t,s,p} = 1 \qquad \forall\, t \in Tasks,\ p \in Periods \qquad (2)$$

$$\sum_{(t_s,k)\in FTA} k \cdot x_{t_s,k,p} \geq \sum_{(t_p,k)\in FTA} k \cdot x_{t_p,k,p} \qquad \forall\, (t_p, t_s) \in Prec,\ p \in Periods \qquad (3)$$

$$x_{t,s,p} = \{0, 1\} \qquad \forall\, (t, s, p) \in TSP \qquad (4)$$

The most loaded station is the bottleneck, determining the cycle time of the line. In the balancing problem, every workstation cannot be loaded more than the limit CT_{max} (Constraint 1). The balancing is performed assigning all tasks to workstations. The Occurrence Constraint 2 along with the Integrality Constraint 4 assure each task is allocated to a single station for each period. Finally, the precedence relations are implemented through Constraint 3, which forces followers to be allocated not before their predecessors.

Another formulation for the precedence relations is due to Ritt e Costa [2015]. Inequality 5 produces a tighter linear relaxation at the cost of more restrictions. For the results showed in Section 4, Patterson e Albracht's restrictions were used.

$$\sum_{\substack{(t_s,k)\in FTA \\ k\leq s}} x_{t_s,k,p} \leq \sum_{\substack{(t_p,k)\in FTA \\ k\leq s}} x_{t_p,k,p} \qquad \forall (t_p, t_s) \in Prec, (t_p, s, p)\ \&\ (t_s, s, p) \in TSP \qquad (5)$$

3.2. The Implementation Sequencing Problem Formulation

The balancing constraints are defined for each considered period. The sequencing restrictions, on the other hand, are applied to the transitions between each stage. The variable $y_{t,s,c}$ is responsible to track the changes between each period.

When a task is assigned to a station it wasn't assigned to on the previous period, it both leaves the old station and joins the new one. Given that we only want to track whether or not the task has changed its station, to track whether or not the task has entered ($x_{t,s,p}$ changing from 0 to 1) the station is enough (one doesn't need to also track when $x_{t,s,p}$ changes from 1 to 0, as it is bound to happen at the same transition). With this reasoning, we define the variable $y_{t,s,c}$ as 1 when task t entered station s at transition c, and 0 otherwise.

$$y_{t,s,1} \geq x_{t,s,1} - A \qquad \forall\,(t,s,A) \in InitAlloc \qquad (6)$$

$$y_{t,s,p} \geq x_{t,s,p} - x_{t,s,(p-1)} \qquad \forall\,(t,s,p) \in TSC,\ p \in InterTran \qquad (7)$$

$$y_{t,s,(NP+1)} \geq A - x_{t,s,NP} \qquad \forall\,(t,s,A) \in FinAlloc \qquad (8)$$

$$\sum_{(t,s)\in FTA} y_{t,s,c} \cdot c_t \leq Effort \qquad \forall c \in Transitions \qquad (9)$$

Constraints 6 - 8 link the variable $y_{t,s,c}$ with the balancing in each period. Once the defined sequencing variables assume 1 when a new assignment is made, a simple way to implement that is to make $y_{t,s,c}$ greater or equal to the difference of the assignment variables at subsequent periods. In constraint 7, the only combination that forces $y_{t,s,c}$ to be 1 is when $x_{t,s,p} = 1$ and $x_{t,s,p-1} = 0$. For the other possible values, the restriction 7 implies $y_{t,s,c} \geq 0$ or $y_{t,s,c} \geq -1$, allowing the variable to be null. A restriction for an upper bound for the variable is not used: the model already minimizes the sum of $y_{t,s,c}$. Inequality 6 is used for the first transition. The initial allocation is a parameter (A), so only one balancing variable is considered ($x_{t,s,1}$). For the intermediary transitions, inequality 7 is applied. Finally, constraint 8 is used for the final transition.

The variable $Effort$ is defined in inequality 9. For every transition, this variable must be greater or equal to the cost of every modification that occurred. Once the objective function is to minimize $Effort$, the optimization results in an evenly balanced implementation workload between the transitions.

In this paper, for simplicity, the balancing equations are modeled for a Simple Assembly Line Balancing Problem (SALBP). For other applications, however, other extensions of the problem can be used.

3.3. Domain Reduction

In this section we define the set of the feasible task allocations FTA used in the model. Even though assembly lines could allow a task to be assigned in any workstation, precedence constraints usually reduce these assignment possibilities. One task with several followers, for instance, can not be assigned to the latest workstation without inducing a work overload. Patterson e Albracht [1975] presented a simple way to use the precedence graph to calculate bounds for the interval in which a task can be assigned.

The earliest bound for a task (E_t) is the minimal number of workstations necessary to fit the task along with all its predecessors (P_t^*). It is determined as the sum of the duration time of the tasks divided by the cycle time, as in equation 10. The latest bound for each task (L_t) uses the duration time of the followers (F_t^*) to calculate the latest station a task can be assigned to (equation 11). The set FTA gathers all the possible task-station assignments (equation 12), reducing the number of variables created in the model.

$$E_t = \left\lceil \frac{TD_t + \sum_{i\in P_t^*} TD_i}{CT_{max}} \right\rceil \qquad \forall\, t = 1, ..., NT \qquad (10)$$

$$L_t = NS - 1 + \left\lceil \frac{TD_t + \sum_{i\in F_t^*} TD_i}{CT_{max}} \right\rceil \qquad \forall\, t = 1, ..., NT \qquad (11)$$

$$FTA = \left\{(t,s) : s = E_t, ..., L_t\right\} \qquad \forall\, t = 1, ..., NT \qquad (12)$$

Further developments on the calculations of bounds for SALBP can be seen in the works of Scholl [1999] and Fleszar e Hindi [2003]. For this paper, however, only the reduction described here is applied.

4. Results and Discussions

In this section, we propose two case studies that exemplify the use of the model in sequencing the implementation of a new balancing. The cases represent structural changes that need several allocation changes to reach the new desired configuration. Both cases are based on the precedence diagram presented by Buxey [1974]. With 29 tasks, the problem presents enough elements to allow model insights. At the same time, the number of tasks is still adequate to represent the allocations in simple figures.

4.1. Augmenting the number of stations

For this case study we suppose a fictional case based on a real industry situation. In this first situation, a company is producing pieces according to the precedence diagram of Buxey [1974] using 10 workstations. The optimal answer of the balancing problem with 10 stations is equal to 34 time units. Due to an increasing demand of the market, the production sector of the company intends to add a workstation to reduce the cycle time. The engineers solved the new problem and obtained the optimal answer for the cycle time of 32 time units, enough to attend the new demand.

When engineers compared the initial implementation with the new goal, they found that 22 allocations have changed, out of 29 total possible allocations. They estimated that reassigning a task would require 2 hours of work, consisting in moving and installing equipment and updating process files. The line runs intermittently during weekdays, so that the changes have to be performed at weekends.

The manager calculates that a single pass implementation would require 44 hours of overtime. This change requires the assignment of a team of 6 employees working on Saturday or 3 employees working the whole weekend. Since this is not the only project being scheduled for the month, the manager evaluates whether the change can be implemented in periods with less employees being needed in each intervention. After each pass, however, the line must be able to operate at the initial cycle time of 34. That is, the intermediary changes must not negatively effect the output of the line.

Using the model of Section 3 implemented in a state-of-the art MILP solver, the results of Table 3 are obtained in less than 1 minute for $NP \leq 10$ and up to 10 minutes for more periods. The cost of moving each task (c_t) is equal to 2 hours of work. As a result, the value of the variable $Effort$ represents the number of hours necessary to implement the changes for the most loaded transition. This value can be used to determinate the size of the team needed for the project.

Table 3: Results for the model for the first case study. The value of $Effort$ is diluted between stages of implementation. As stated in Figure 1, Stages are intermediary configurations of the line.

Stages	0	1	2	3	4	5	6	7	8	9	10	11	...	21	..	24
Transitions	1	2	3	4	5	6	7	8	9	10	11	12	...	22	..	25
Effort (hours)	44	22	16	12	10	8	8	6	6	6	6	4	...	4	...	2

According to the results of Table 3, the size of the employed team can significantly reduce by allowing intermediary stages. With one intermediary period, the amount of hours is reduced to half: 22 hours of work in 2 implementations. Although the effort needed in the implementation reduces with a greater number of stages, some solutions are clearly dominated. Dividing the implementation in 6 stages, for instance, has at least one transition that needs a team for 8 hours of work. A similar result can be obtained using only 5 stages. Note that to reduce the effort to only 2 hours (one change) per transition, 24 periods are necessary. This means that when only one reallocation at a time is allowed, it is necessary to perform extra reassignments: some tasks will be moved more than once between stations (25 changes for the 22 allocation differences).

The model does not only calculates the amount of effort needed for the implementation, but also finds every intermediary configuration that must be applied. Figure 2 exemplifies one possible planning option for 1 stage, where each task is represented by a different color. Note that

in the initial condition, the station 11 is empty. At Stage 1, some tasks are relocated, while the cycle time of the line remains unchanged (34 Time Units). For the final condition, an assignment with 32 Time Units for the cycle time is reached.

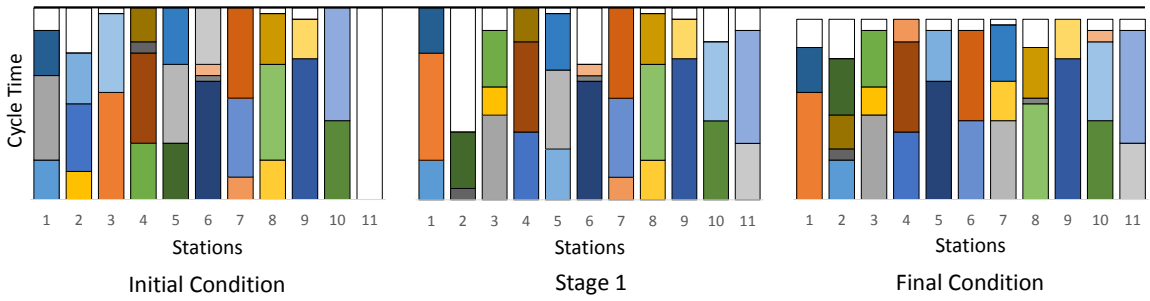

Figure 2: Implementation planning with one intermediary period. Each task is represented by a block with a size proportional to its duration time. The assignments are changed so that the line's configuration go from the initial to final condition with two transitions. Note that the cycle time reduces from 34 to 32 Time Units.

The number of variables of the model is proportional to the size of the set FTA and the number of periods (NP) considered. For each period, the binary variables $x_{t,s,p}$ are defined. At the same time, each transition uses one variable $y_{t,s,c}$ for each element of FTA. At a given instance, the number of periods is defined as NP, resulting in $NP+1$ transitions. Finally, a single variable ($Effort$) is used in the objective function. For this case study, the number of possible allocations (FTA) is equal to 185. This way, we can calculate the number of variables used by: $185 \cdot (2 \cdot NP + 1) + 1$.

4.2. Adding a new task

For the second case study, a second instance based on a real industry is presented. We suppose that after the line was adapted to support 11 workstations, the marketing department recognized a new trend in the market and required a product modification. The project department developed the extra feature, which would require a new task to be performed. The production engineers concluded that the new task (number 30) would require 18 time units to be processed, resulting in the precedence diagram showed in Figure 3.

The final condition showed in Figure 2 (or initial condition in Figure 4) contains several workstation with idle time (namely stations 1, 2, 3, 5, 6, 7, 8, 10 and 11). Individually, no station has an idle time of 18 time units, but the sum of the idle time for the line is 28. By rearranging tasks, it is possible to add Task 30 in the line without impacting the cycle time and the number of stations. The production engineers chose a new balancing that, however, required 18 task changes or 36 hours of overtime. Once again, the manager wondered if the implementation could be done in smaller steps.

Table 4 shows the results for an interval of the number of stages. With one intermediary stage, it is possible to divide the 18 changes in exactly 2 implementations of 9 changes (18 hours of overtime). This second case proved to be more difficult to find solutions with only one change at a time. Even if we allow 36 transitions, two for each difference in initial and final allocations, we still need at least one transition with 2 changes.

Table 4: Results for the model for the second case study. The value of $Effort$ is diluted between stages of implementation.

Stages	0	1	2	3	4	5	6	7	8	9	10	...	35
Transitions	1	2	3	4	5	6	7	8	9	10	11	...	36
Effort (hours)	36	18	14	10	8	8	6	6	6	6	4	...	4

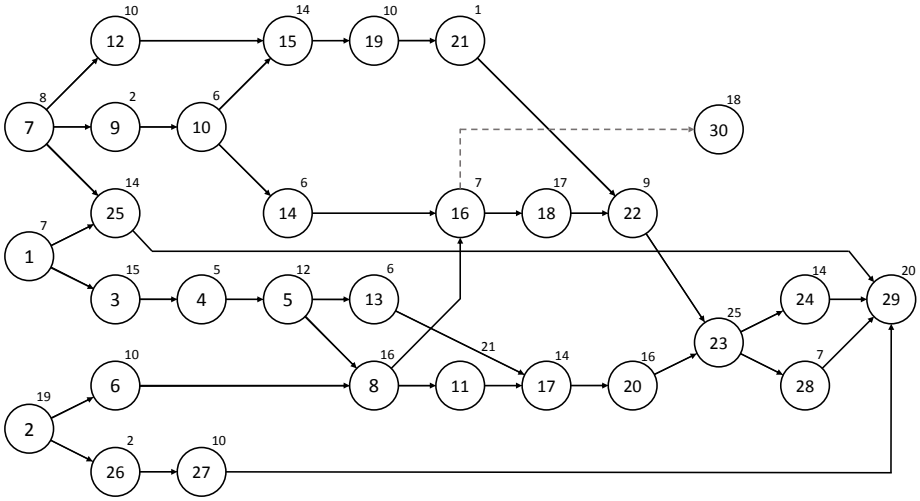

Figure 3: Adapted precedence diagram from Buxey. The values inside circles show the number of the task while the duration time of each task is represented at above each task. Task 30 (18 time units) is added to the problem for this case study along with the precedence relation between tasks 16 and 30.

Figure 4 shows a diagram with the planning solution for ten intermediary stages. In the figure, the initial condition, the last intermediary stage and the final assignment are represented. Note that the new task (colored in black) is only added in the final condition. That is, the occurrence and precedence restrictions for Task 30 are dismissed for the intermediary stages. The first transitions are used to create space for the new task, whose assignment happens in the last transition.

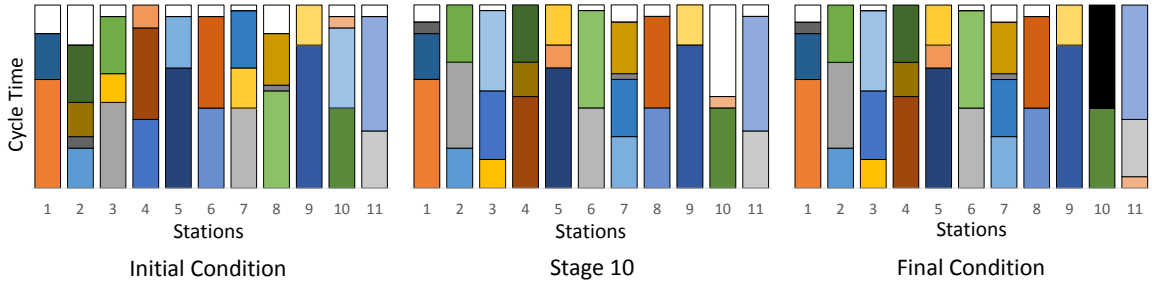

Figure 4: Implementation planning with ten intermediary periods. The initial and final conditions as well as the last intermediary stage are showed. Each task is represented by a block with a size proportional to its duration time. Task 30 is represented only in the final condition, in black, at station 10. The cycle time of all configurations is 32 Time Units.

With these results, the manager has more options to schedule and implement projects in assembly lines. The model's result can be interpreted as the size of the team needed to implement the changes, which varies according to the number of stages allowed.

5. Conclusions

We presented a model for sequencing a balancing implementation. Instead of implementing a new configuration at once, a project scheduling can allow smaller stepwise interventions. The Assembly Line Balancing Implementation Sequencing Problem-Type 2 (ALBISP-2) is defined as finding the best sequence of alterations in a line's balancing given a number of intermediary periods. The ALBISP can be seen as several parallel balancing problems linked together by the sequencing

restrictions: the changes in the implementations between the configurations are minimized while the cycle time restrictions are valid for every intermediary configuration.

The sequencing problem is important when the implementation of a new configuration is work or time intensive. Projects with significant differences between the initial and final configurations may require long intervention times. To test the model under these circumstances, two case studies are presented to illustrate situations in which assembly lines may require several assignment changes in their reconfiguration. The fact of changing the number of stations or adding a task in the line can require the reassignment of almost every task. In the first case study, for instance, 22 out of the 29 tasks were reassigned. In case study 2, the extra task required 18 changes in their allocations. As a result, the model is able to plan the implementation step by step and can be used in sizing the intervention crew.

Once assembly lines usually work in multiple shifts, the modifications and redesigning of these lines are restricted to weekends or programmed maintenance stops. Drastic changes, however, are usually too time-consuming to be implemented in only a weekend. Dividing the work in steps, so that in every step the line is not negatively effected by the changes, is not a trivial problem.

The Implementation Sequencing Problem's objective is not only dividing the implementation in steps, but also minimize the necessary effort for the implementation. The ALBISP solution can aid in the application of the optimization of balancing of assembly lines. By dividing the implementation in smaller interventions, the use of balancing techniques becomes more viable in industrial scenarios.

Acknowledgments

To the financial support from Fundação Araucária (Agreement 141/2015 FA – UTFPR) and CNPq (Grant 305405/2012-8).

References

Agnetis, A., Ciancimino, A., Lucertini, M., e Pizzichella, M. (2007). Balancing flexible lines for car components assembly. *International Journal of Production Research*, 33(2):333–350. ISSN 0020-7543.

Arcus, A. L. (1965). A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, 4(4):259–277. ISSN 0020-7543.

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study. *International Journal of Production Research*, 31(10):2447–2461.

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

Battini, D., Faccio, M., Ferrari, E., Persona, A., e Sgarbossa, F. (2007). Design configuration for a mixed-model assembly system in case of low product demand. *The International Journal of Advanced Manufacturing Technology*, 34(1-2):188–200. ISSN 0268-3768.

Bautista, J. e Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, 177(3):2016–2032. ISSN 03772217.

Baybars, I. (1986). Survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32(8):909–932. ISSN 00251909.

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

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

Buxey, G. M. (1974). Assembly line balancing with multiple stations. *Management Science*, 20(6): 1010–1021. ISSN 00251909.

Fleszar, K. e Hindi, K. S. (2003). An enumerative heuristic and reduction methods for the assembly line balancing problem. *European Journal of Operational Research*, 145(3):606–620. ISSN 03772217.

Lapierre, S. D., Ruiz, A., e Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168(3):826–837. ISSN 03772217.

Otto, C. e Otto, A. (2014). Multiple-source learning precedence graph concept for the automotive industry. *European Journal of Operational Research*, 234:253–265.

Patterson, J. H. e Albracht, J. J. (1975). Assembly-Line Balancing: Zero-One Programming with Fibonacci Search. *Operations Research*, 23(1):166–172.

Ritt, M. e Costa, A. M. (2015). Improved integer programming models for simple assembly line balancing and related problems. *International Transactions in Operational Research*, online. ISSN 1475-3995.

Salveson, M. (1955). The assembly line balancing problem. *Journal of Industrial Engineering*, 6 (3):18–25.

Scholl, A. (1999). *Balancing and Sequencing of Assembly Lines*. Physica, Heidelberg, second edition.

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