VARIABLE SETS REDUCTION FOR ASSEMBLY LINE BALANCING PROBLEM: MILP MODEL AND CASE STUDIES

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

Line balancing problems consist on assigning tasks to stations and/or workers in a restricted way. Most models deal with simple versions of this problem not taking into account real world restrictions. In this paper we present a new re-balancing model that takes into account assignment restrictions and builds the variable sets in a more efficient way. The model also takes into consideration worker displacement for problems with fewer workers than stations. To measure how constrained a line is in terms of task-stations allocations possibilities, two coefficients are proposed. The model was tested with variations of thee-hundred literature problems, each variation with a different degree of assignment restrictions. The elimination of allocation possibilities, a consequence of real world restrictions, leads to a reduction on processing time. The reduction of the variable sets might be of use for both exact and heuristic algorithms.

KEYWORDS. Assembly-Line Balancing Problem, Re-balancing, Variable Set Reduction.

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

1. Introduction

Assembly lines concentrate much of the world's large scale and high-productivity manufacturing, consequently, efforts to increase outputs and reduce costs are also reflected on how to plan and balance the assembly lines. The basic ALB (Assembly Line Balancing problem) is how to distribute tasks to stations along the conveyor belt to the machines and workers to perform, subject to restrictions such as precedence and incompatibility of tasks. This problem and its many variations have motivated a wide field of research, however there are still large gaps between academic research and real world implementations (Boysen et al., 2008). Scholl and Klein (1999) indicated that only a minority of articles explicitly address real world assembly lines systems.

As many other mathematical models and problems, an assembly line balancing problem's size impacts directly on its computing cost. Given that these problems are NP-Hard (Wee and Magazine, 1982), this often motivated the search for good solutions via heuristics and meta-heuristics rather that exact solutions. Its worth mentioning that some efforts have been made on the sense of reducing the size of the problem (Gokcen and Erel, 1998), by regarding the precedence diagram, and that such work can be beneficial for both exact and approximated methods. However just as real world problems usually have a large number of variables, they also offer a large amount of restrictions as for how those tasks can be performed and where. In particular when the assembly line has already been deployed making a re-balancing is necessary (Gamberini et al. (2006), Falkenauer (2005), Makssoud et al. (2015)) for new models of products, changes in performance of certain activities, or a reduced cycle time, there are heavy equipments that might be too expensive to move and other real world restrictions beyond those already present on the base problem (of an unimplemented assembly line). Real world restrictions might be associated to the shape of the assembly line, to whether or not tasks are performed on both sides of the assembly lines, to whether or not a task is fixed at a certain station or can only be executed at a subset of all stations (such as a certain type of station).

In assembly lines with a greater number of stations than workers, their allocation might be of particular interest and many restrictions might also apply to how those are to be distributed along the lines (such as the two sided assembly lines studied by Tuncel and Aydin (2014)). In those cases, the worker's displacement time might have important impacts on cycle time, and finally the decision variables for allocations worker-task-station tend to be much bigger that the simpler task-station decision variables set.

It is known that the processing time of a linear problem tends to grow with the number of restrictions (Hillier and Lieberman, 2001), but when it comes to mixed-integer linear programming, some restrictions, such as fixing a task to a station, might very well reduce the search-space and make execution faster (Williams, 1999). However, restrictions such as fixing a task to a set of stations make some variables redundant (such as the binary variable associated with the assignment of that task to a station outside the set), suggesting that it is possible to build the variable sets in a way that some restrictions are automatically incorporated on them, as part of the preprocessing of a problem, allowing a reduction of the final computing cost. The process for the definition of such sets is not intuitive and requires a series of auxiliary sets and set operations as shown in the following section.

In this paper we present a strategy to reduce the number of variables on a assembly line problem based on real life restrictions, generating variables such that the final allocation will automatically satisfy them, reducing substantially the size of the problem and the time required to solve it. A Mixed-Integer Linear Programing model (MILP) for the worker-task-station assembly line balancing problem is presented based on the generated variable sets, along with the case studies for the developed model and the results of its application to variations of three hundred ALB problems from open databases.

2. Mathematical Model

Restrictions on a problem make many of the variables generated by standard variable matrices (such as a task-station or worker-task-station matrices) unnecessary: If a task is fixed to a station or restricted to a few, it is pointless to create decision variables tying said tasks to the other stations. The general strategy employed here is to, during the preprocessing phase of the model, generate sets of vectors with integer coefficients, each vector will correspond to a variable and each position on the vector to either tasks, stations or workers. During preprocessing those vector sets will be defined so that all feasible possibilities are allowed, at the same time that the number of vectors (and thus variables) is kept at minimal.

2.1. Ranges, Sets and Parameters

In order to define the variables needed for the proposed model, some sets are defined. These sets are lists of integers and vectors of integers associated with the assembly line balancing problem. In essence, the ALB is described by parameters like the number of tasks (NT) the number or workstations or simply stations, for short (NS), the number of workers (NW) and in some cases the number of models (NM).

These numbers define ranges to list all tasks, stations, workers and models, namely: $Task = (1, 2, ..., NT)$, $Stations = (1, 2, ..., NS)$, $Workers = (1, 2, ..., NW)$ and $Models = (1, 2, ..., NM)$.

2.1.1. Parameter Sets: Workers, Tasks and Station

In order to efficiently generate the variables associated with the problem, namely the decision variables that relate to worker-station-task allocation, some auxiliary sets must be defined. These sets help to impose restrictions on the variable structure, allowing a model with fewer variables and restrictions than to simply create one decision variable for each worker-station-task combination. The sorted sets generated will also help build the restrictions the model is to be subject to. Those sets are sorted sets of integers that relate to task, workers and stations. The capital letters T, W and S stand for task, worker and station.

The first kind of parameter set is the task-task type, namely the precedence relations. The set TT_{pre} states the precedence relations between tasks, each element ($t1$, $t2$) states that $t1$ precedes $t2$.

There are two sorted sets of the worker-station type. WS_{fix} lists workers that are fixed to a station, this means they will perform tasks at that station and only at that station, here (w, s) means worker w only works at the station s. The set WS_{feas} refers to the feasible allocation on workers to stations, if the worker w only appears at WS_{feas} in $(w, s1)$ and $(w, s2)$ then he will only be allowed to perform tasks at the stations $s1$ and $s2$, this might be particularly useful to describe long assembly lines or yet assembly lines where tasks can be performed at both sides. All workers that are not listed on WS_{fix} and WS_{feas} are allowed to perform tasks at any station.

There are two sorted sets of the task-station type. TS_{fix}, much like WS_{fix} relates tasks fixed to stations, namely to represent conditions in which a task cannot be moved. For example, task that requires a heavy machine to be performed, in a circumstance where to move the machine is difficult and expensive. TS_{feas}, much like WS_{feas}, relates tasks that can only be performed on a few stations, so if the task t appears twice on the set at $(t, s1)$ and $(t, s2)$, then t can only be performed at either station $s1$ or station $s2$.

These are sets that describe the problem from the perspective of the factory, but are not directly employed to generate variables, as some set operations are necessary firstly. In order to avoid redundant variables and unnecessary constraints, some set operations are needed as part of the preprocessing.

2.1.2. Auxiliary Parameter Sets and Set Operations

Based on the sets previously presented, it is important to define lists of workers, tasks and stations of interest. The goal here is to define sets that generate the smallest set of variables that

satisfies automatically the restrictions described (such as a task being fixed to a station) and still allows all viable possibilities of a task-worker-station matrix of variables.

Sets of workers are defined based on WS_{fix} and WS_{feas}, namely workers that belong to each set. $W_{fix} = w|(\exists(w,s) \in WS_{fix})$ lists workers fixed to stations, $W_{feas} = w|(\exists(w,s) \in WS_{feas})$ lists workers limited to a few stations, $\overline{W_{fix}} = w|(w \in Workers, w \notin W_{fix})$ lists workers that are not fixed to stations, $\overline{W_{feas}} = w|(w \in Workers, w \notin W_{feas})$ lists workers that are not limited to a few stations.

Sets of stations are defined based on WS_{fix}, namely stations where the allocated worker is already predefined. $S_{fix} = s|(\exists(w,s) \in WS_{fix})$ lists stations to which a worker is fixed, $\overline{S_{fix}} = s|(s \in Stations, s \notin S_{fix})$ lists stations to which no worker is fixed.

Sets of task are defined based on TS_{fix} and TS_{feas}. $T_{fix} = t|(\exists(t,s) \in TS_{fix})$ lists tasks fixed to a station, $T_{feas} = t|(\exists(t,s) \in TS_{feas})$ lists tasks restricted to a few stations, $\overline{T_{fix}} = t|(t \in Tasks, t \notin TS_{fix})$ lists tasks not fixed to a station, $\overline{T_{feas}} = t|(t \in Tasks, t \notin T_{feas})$ lists tasks not restricted to a few stations.

The purpose of those sets is to assist on the creation of the sets of variables as part of the preprocessing. Notice that the over-line here means all other elements of such kind, the elements being tasks, stations or workers.

These sets allow us to generate sets of possible worker-station and task-stations sets. Firstly a set is defined for the unrestricted pairs worker-stations, we call this set WS_{free}. This set relates workers that are not restricted to stations that are not fixed to a specific worker.

$WS_{free} = (w,s)|(w \in (\overline{W_{fix}} \cap \overline{W_{feas}}), s \in \overline{S_{fix}})$

Combining this set to the restricted sets for worker-station allocations we get to a first full set of possible allocations of workers and stations, which is called WS_i. This final set will be generate from the combination of WS_{free} set and the two others.

$WS_i = WS_{free} \cup WS_{fix} \cup WS_{feas}$

The set TS_{free} is defined in a similar manner to the set WS_{free}, combining all unrestricted tasks with all stations.

$TS_{free} = ((t,s)|(t \in (\overline{T_{fix}} \cap \overline{T_{feas}}), s \in Stations))$

Combining the free set with the restricted sets for the task-station allocations, we reach the full initial set TS_i, just like WS_i.

$TS_i = TS_{free} \cup TS_{fix} \cup TS_{feas}$

The set TW_i is generated by combining all possible tasks to all possible workers, here some restrictions as workers disabilities - that is, incompatibility between a worker and a task - might be added, but those were not considered for this model.

$TW_i = (t,w)|(t \in tasks, w \in workers)$

2.1.3. Main variable sets

Finally these sets are combined to generate all interesting combinations of task-worker-station. This intersection reduces significantly the number of variables the model will have to process. The elimination of variables performed are, in essence, a way to construct the restrictions (such as task restricted to a few stations), the which only affects the problem's preprocessing and speeds up the actual processing.

$TWS = ((t,w,s)|(t,w) \in TWi, (t,s) \in TSi, (w,s) \in WS_i)$

Depending on the restrictions, redefining the sets of possible task-worker, task-station and worker-station might lead to smaller sets for the model to process, thus these sets are reintroduced in the model, this time named TW, TS, and WS.

$TW = ((t,w)|(t,w,s) \in TWS)$

$TS = ((t,s)|(t,w,s) \in TWS)$

$WS = ((w,s)|(t,w,s) \in TWS)$

These four sets are the main sets employed to generate the decision variables in an efficient manner and will be employed on the model's restrictions. This model was designed for scenarios

with fewer workers than stations and thus should take in account the required displacement time by the worker. This is achieved with a last variable set WSS, which is defined for whether or not the worker w moves between stations $s1$ and $s2$, and is based on the set WS.

$$WSS = ((w, s1, s2)|((w, s1) \in WS, (w, s2) \in WS, s2 > s1))$$

2.1.4. Other parameters and sets

In order to allow the model to work with more than one model of product produced on the assembly line a set is defined for the occupation rate. Its clear that the occupation rates should add up to one, that is: $\sum_{i=1}^{NM} OR(i) = 1$.

In multi-model assembly lines, the duration of each task is considered by the model. Thus a set model-task-time is defined as a parameter. $DTm = ((m, t, ti))$, the time ti required to perform the task t of model m.

The movement between stations also require a parameter set. Be $MT = ((s1, s2, tm))$, the time tm required to move between stations $s1$ and $s2$.

2.2. Model Variables

Based on the sets of tasks, stations and workers (associated with vectors of integers), vectors of variables are defined to represent the valid decision variables (that is binary) necessary to describe the problem.

A set $TWS_v(TWS)$ is defined as a general decision variable set for which task will be performed by which worker on which station, having a *yes* or *no* answer to each feasible possibility on the set TWS.

Other vectors are defined to specifically describe the worker-station, task-worker and task-station decisions. The problem constraints tie these sets to the broader TWS_v vector. For the worker-station variables: $WS_v(WS)$, for the task-worker variables: $TW_v(TW)$, for the task-station variables: $TS_v(TS)$. The set $WSS_v(WSS)$ is employed to describe worker movements between stations and is used to help determine each worker's workload.

Those vectors of variables are tied to the sets of integers previously described, and due to the previously described restrictions they tend to be smaller than a simple 2-D and 3-D matrix of boolean variables. This reduction in variable amount doesn't compromise the model's capacity to describe problems, but substantially increase its performance. The more restricted the workshop, the smaller the decision variable vectors will be.

The other variables are simpler than the previous ones, for instance: $W_{time}(Workers)$ and $S_{time}(Stations)$ are variables vectors that count the total workload (measured in time units) for each worker and at each station. These are naturally not integer, but real number variables. W_{time} and S_{time} are used to determine the cycle time and can also be employed to minimize the difference of workload between workers and stations. The last variable is the cycle time, to be minimized, CT.

2.3. Model Constraints

The following model presents the different constraints to provide the basic functions of assembly line balancing problems models with the developed variable sets. The equation (1) states that all tasks must be allocated to a station :

$$\sum_{(t,s)\in TS} TS_v((t, s)) = 1 \qquad \forall t \in Tasks \tag{1}$$

The precedence constraints between tasks, expressed by the inequality (2), must be respected. Here, tp stands for preceding task and ts stands for succeeding task.

$$\sum_{(tp,s)\in TS} s \cdot TS_v((tp, s)) \leq \sum_{(ts,s)\in TS} s \cdot TS_v((ts, s)) \qquad \forall (tp, ts) \in TT_{pre} \tag{2}$$

The cycle time can be determined by either looking at the bottleneck worker or the bottleneck station. The constraints on the inequalities (3) and (4) consider both these possibilities:

$$S_{time}(s) \leq CT \qquad \forall s \in Stations \tag{3}$$
$$W_{time}(w) \leq CT \qquad \forall w \in Workers \tag{4}$$

The occupation of each station is defined by the tasks assigned to it, as stated by the equation (5), where DTm stands for a parameter set of vectors containing the duration tp of the task t of the model m.

$$S_{time}(s) = \sum_{\substack{(t,s)\in TS \\ (m,t,tp)\in DTm}} OR(m) \cdot tp \cdot TS_v((t,s)) \qquad \forall s \in Stations \tag{5}$$

The workload of each worker can be defined by the task he performs and by the movements required for said tasks, as stated by the equation (6). The first term refers to the tasks performed by the worker. The second term refers to movement between stations, here estimated as twice the time the worker takes to move between each pair of stations he is being assigned to.

$$\begin{aligned} Wtime(w) = & \sum_{\substack{(t,w,s)\in TWS \\ (m,t,tp)\in DTm}} OR(m) \cdot tp \cdot TWS_v((t,w,s)) + \\ & +2 \cdot \sum_{\substack{(w,sp,ss)\in WSS \\ (sp,ss,tm)\in MT}} WSS_v((w,sp,ss)) \cdot tm \qquad \forall w \in Workers \end{aligned} \tag{6}$$

In order to consider the time required for workers to move between stations the variable set WSS_v must be tied to the variable set WS_v. Namely a worker has to move between the stations $s1$ and $s2$ when he is assigned to both of them. Such logical and correspondence is described by the inequality set (7).

$$\begin{aligned} WSS_v((w,s1,s2)) &\leq WS_v((w,s1)) \quad \forall (w,s1,s2) \in WSS, (w,s1) \in WS \\ WSS_v((w,s1,s2)) &\leq WS_v((w,s2)) \quad \forall (w,s1,s2) \in WSS, (w,s2) \in WS \\ WSS_v((w,s1,s2)) &\geq WS_v((w,s1)) + WS_v((w,s2)) - 1 \\ & \forall (w,s1,s2) \in WSS, (w,s1) \in WS, (w,s2) \in WS \end{aligned} \tag{7}$$

As part of the assignment constraints, each task is performed by a worker, as stated by the equation (8):

$$\sum_{(t,w)\in TW} TW_v((t,w)) = 1 \qquad \forall t \in Tasks \tag{8}$$

The variable sets TS_v, TW_v and TWS_v are not independent. In order to ensure coherence between the decision variables on each variable set, it is stated that TWS_v is equal to the logical and operation between the sets TS_v and TW_v. These restrictions are expressed by the inequality set (9).

$$\begin{aligned} TWS_v((t,w,s)) &\leq TW_v((t,w)) \qquad \forall (t,w,s) \in TWS \\ TWS_v((t,w,s)) &\leq TS_v((t,s)) \qquad \forall (t,w,s) \in TWS \\ TWS_v((t,w,s)) &\geq TW_v((t,w)) + TS_v((t,s)) - 1 \qquad \forall (t,w,s) \in TWS \end{aligned} \tag{9}$$

The restrictions of the inequality set (9) ensure that TWS_v follows TS_v and TW_v. The reverse is achieved by the constraints on inequality set (10).

$$\begin{aligned} \sum_{(t,w,s)\in TWS} TWS_v((t,w,s)) \geq TW_v((t,w)) \qquad \forall (t,w) \in TW \\ \sum_{(t,w,s)\in TWS} TWS_v((t,w,s)) \geq TS_v((t,s)) \qquad \forall (t,s) \in TS \end{aligned} \tag{10}$$

The constraints required to tie workers to a station employ a *big-M* strategy (Wolsey, 1998). If the worker doesn't perform a single task at the station then he is not assigned to it, on the other hand if he is assigned to a station he can perform many tasks there. These two ideas are described by the following restrictions, where NT, the number of tasks, is the *big-M* factor. Naturally the second constraint is only significant when $WS_v((w,s))$ is zero for a pair (w,s). This means that if a worker is not assigned to a station he can't perform any task on it. These ideas are described by the inequality set (11).

$$\begin{aligned} WS_v((w,s)) \leq \sum_{(t,w,s)\in TWS} TWS_v((t,w,s)) \qquad \forall (w,s) \in WS \\ WS_v((w,s)) \cdot NT \geq \sum_{(t,w,s)\in TWS} TWS_v((t,w,s)) \qquad \forall (w,s) \in WS \end{aligned} \tag{11}$$

The last assignment restriction states that only one worker is assigned to a station and is presented by the inequality (12).

$$\sum_{(w,s)\in WS} WS_v((w,s)) \leq 1 \qquad \forall s \in Stations \tag{12}$$

3. Results and Discussions

In this section, the effect of the variables elimination in the preprocessing phase is investigated. Assignment restrictions such as a task that is fixed to a station or limited to a group of feasible options reduce the number of total possibilities of allocations. The amount of such restrictions is related to the assembly line degrees of freedom.

A similar relation can be seem in the precedence diagram. In (Bhattacharjee and Sahu, 1990), the authors define the *order strength* of a graph (G) as the ratio of the sum of the number of task followers and the maximal possible number of precedence relationships. The *order strength* varies from 0 to 1. When $OS(G) = 1$ the graph leads to a single task sequence, while $OS(G) = 0$ means the tasks don't have any precedence relation and can be ordered in any sequence.

Once in real world assembly lines the assumption that every task can be performed in any station is often not verified, a ratio inspired in the *order strength* can be useful to measure how constrained the assembly line is by these restrictions. A *task-station restriction factor* or *TSr*, for an abbreviation, can be defined as in Equation (13), where PA stands for *possible allocations*. An assembly line balancing problem without such restrictions has $PA = NT \cdot NS$ possible allocations, resulting in $TSr = 0$. On the other hand, a fixed line with every task fixed to a particularly station has $PA = NT$ possibilities, one only for each task ($TSr = 1$). Real world assembly lines present a behavior within these limits.

$$TSr = \frac{NT \cdot NS - PA}{NT \cdot (NS - 1)} \qquad for\ NS > 1, NT \leq PA \leq NT \cdot NS \tag{13}$$

Depending on the nature of tasks and the necessary equipment, a line can be soft or hardly restricted. For instance, taking just some of the papers of ALB from SBPO last years publications, a varied range of TSr values can be found. The assembly line treated by Donnini et al. (2010) has $TSr = 0.19$, while Rodrigues et al. (2013) and Magatão et al. (2011) presented more heavily restricted lines with factor of 0.73 and 0.97, respectively. These value were obtained by analyzing the restrictions of tasks in each problem.

3.1. Case Study 1: The effect of the Task-Station Restriction Factor

To evaluate how the TSr effects the processing time of the model, the testbed from Scholl and Klein (1999) was used. The data set is a collection of case studies in the literature. The data is available for download at the website www.assembly-line-balancing.de. For the test, the SALBP-2 cases were chosen. The set contains a total of 302 instances, each one presenting from 29 to 297 tasks. A SALBP case contains task duration and precedence information.

The cases were solved firstly for the SALBP data without any assignment restriction ($TSr = 0$), that is, all allocations that obeyed the precedence graph were considered feasible. The best allocation found was stored and used to create the cases with TSr bigger then 0. In this process, a list of NT allocations for each instance was stored. Nine new data sets were created randomly, from $TSr = 0.1$ to 0.9 with increments of 0.1. These datasets have the same duration time and precedence data as the SALBP cases, but some allocation restrictions were added. This restrictions correspond to the deletion of some task-station allocations. Once the restrictions were randomly created, optimal allocations could be restricted, resulting in a ALBP with a different optimum cycle time answer. To preserve the possibility of reaching the same $cycletime$ in each case, the SALBP allocations were protected, thus, the task-station allocation that leads to the SALBP answer was not deleted from the set of possibilities. That is, from the $NT \cdot NS$ possible allocations, the NT SALBP allocations are set aside and TSr of the other ones were deleted.

All datasets were solved using the Solver IBM ILOG CPLEX 12.51 using a Core i7 (2.3 GHz) with 16.0 GB RAM. It was established a maximal time of 2 minutes for all cases, except Scholl (297 tasks). For these 297 task-problems, it was given up to 20 minutes for the SALBP instances, 10 minutes for the cases with $TSr = 0.1$, and 5 minutes for $TSr = 0.2$ and 0.3. Different time limits were used to assure a solution close to the optimum, once the answer of the SALBP was used in the process of creating restrictions.

The obtained results can be seen in Table 1 . The resulting allocations present similar distance from the optimum answer ($TSr = 0.00$), in average. The solutions do not improve in relation to the SALBP answer, once some optimal allocation option might be excluded in the restriction creating process. The case with $TSr = 0.9$, for instance, has very low flexibility, resulting in a answer very similar to the SALBP dataset. It can be observed that to obtain a similar response, a more restricted assembly line problem is solved in less time. For every increment of the TSr, the solution is found faster.

A more restricted problem can be solved quickly, but it might not converge to the unrestrained optimal response. Some of the restrictions can make the best solution infeasible. In real world problems, balancing lines are usually made every time there is a change in the mix of production. Depending of the frequency of the changes the redesigning of a assembly line might be more costly than the advantages of eliminating such restrictions. In (Falkenauer, 2005), the author argues that more commonly assembly lines need to be *re*-balanced. Workstations cannot be considered identical and, therefore, the elimination of stations might not be a possibility. Hence, assignment restrictions are important to represent the real line characteristics, as emulated within the developed case study.

3.2. Case Study 2: Assembly line with more stations than workers

In real world lines it is possible for an assembly line to have more workstations than workers in a shift. Low production phases or tasks that require very different machinery and/or

Table 1: Results of the SALBP cases ($TSr = 0$) and cases with $TSr > 0$. The first left column refers to the TSr ratio. Each line of the Table summarizes results obtained from 302 executed instances.

TSr	AT (s)	SDT (s)	$MaxT$ (s)	DO	SDO	$\#OF$	$\#BSALBP$
.00	109.42	184.91	1202.86	0.50%	2.12%	196	
.10	68.89	101.11	603.04	0.70%	3.37%	195	242
.20	54.39	67.82	306.35	0.54%	1.58%	194	259
.30	42.70	57.34	301.40	0.41%	0.88%	196	263
.40	39.21	53.72	121.85	0.46%	1.30%	196	269
.50	32.55	50.78	121.18	0.50%	1.75%	196	273
.60	26.76	47.06	121.79	0.49%	1.84%	198	285
.70	21.24	43.67	120.92	0.51%	1.85%	197	283
.80	9.59	29.58	120.57	0.50%	1.86%	197	293
.90	0.35	0.26	1.51	0.48%	1.81%	196	302

AT stands for Average processing time, SDT and $MaxT$ are the standard deviation and maximal value of the processing time. DO means Distance from Optimum followed by the standard deviation SDO. $\#OF$ and $\#BSALBP$ are the number of optima found and the number of answers better or equal than the SALBP answer found.

equipment are some reasons for that. A model that treats differences between the number of workers and stations have to consider that a worker can be allocated in more than one workstation as well as the movement time must be accounted.

In (Magatão et al., 2011), a MILP model is used to solve a ALB with 70 tasks, 15 stations and 6 workers. Although the task-station allocation was heavily restricted, the workers were free to be allocated in any station, except in the case of a robot, which was fixed in a station.

When one considers the worker-station allocation in the model, the problem assumes one more dimension. Instead of finding the best pair task-station allocations, the new model has much more variables. For every task-station possibility in a two dimensional problem, there is a combination with each one of the workers.

We can also define a restricting factor for a three-dimensional ALBP. The *task-worker-station restriction factor*, or $TWSr$, is equivalent to the TSr but also considers the station-worker allocations. This factor can be defined as in Equation (14). Once again, PA stands for *possible allocations*. The minimal number for PA is NT, when the line is fixed to a single sequence of tasks, with determined workers and stations. In this case $TWSr$ assumes the value 1, meaning there is no flexibility in the line. On the other hand, the maximal value for PA is $NT \cdot NW \cdot NS$, it is, all the possible allocations. This upper value leads to $TWSr = 0$, implying in a assembly line with all possible degrees of freedom.

$$TWSr = \frac{NT \cdot NW \cdot NS - PA}{NT \cdot NW \cdot NS - NT} \qquad for\, NS > 1, NT \leq PA \leq NT \cdot NW \cdot NS \qquad (14)$$

In (Magatão et al., 2011), the problem restrictions allow the existence of 465 possible task-worker-station allocations. While the TSr factor is 0.97, the flexible options of allocating the 5 non-robots workers results in a $TWSr$ factor of 0.94.

To point out how the TSr and $TWSr$ factors can greatly differ, we pick the Buxey SALBP, the smallest problem from the Scholl and Klein (1999) SALBP-2 dataset. This problem has 29 tasks and the variation chosen presents 7 workstations. Without any task-station restrictions, $TSr = 0$. Once the workers in each station are not considered in this case, we could see this problem as a task-worker-station problem, but with each worker fixed to a station. Although in the set task-station all allocation combinations are feasible, the set worker-station is heavily restricted. This results in a $TWSr$ of 0.88, since only 203 (29 tasks and 7 stations) out of the 1421 combinations (29 tasks, 7 workers and 7 stations) are feasible.

In order to exemplify the model's capability to solve problems with a surplus of workstations, in this second case study a slight variation of the Buxey problem is proposed. Table 2 shows

the results of the Buxey problem with 7 stations and 7 workers (instance 1) and one variation with 8 stations and 7 workers (instance 2). For the case with one extra station, we fixed every worker to a station and let one of them responsible for 2 stations. The movement time between stations is set to 0, so the optimum cycle time of the two cases must coincide. Both instances have a cycle time of 47. The last worker in instance 2 is allocated in stations 7 and 8, having a combined workload of 44.

Table 2: Results of the smallest case of Scholl and Klein (1999) with 7 workers for 7 stations (instance 1) and 8 stations (instance 2). The first line shows how the tasks are allocated and the second one shows the workload. One worker is allocated in stations 7 and 8 in instance 2.

Instance	Station	1	2	3	4	5	6	7	8
1	Tasks allocated	1, 3, 4 5, 7	2, 6 8, 9	10, 11, 13 14, 16	17, 18 20	12, 15, 19 21, 22	23, 24 26	25, 26 27, 29	-
	Workload	47	47	47	47	44	46	46	-
2	Tasks allocated	1, 3, 4 5, 7	2, 6 8, 9	10, 11, 13 14, 16	17, 18 20	12, 15, 19 21, 22, 26	23, 25 28	24 27	29
	Workload	47	47	47	47	46	46	24	20

4. Conclusions

This paper presented an MILP model aided by a preprocessing method that reduces the variable sets. The combined approach is used to solve ALBP with two (task-station) or three (task-worker-station) dimensions. The model supports the movement of workers between stations, which may be presented in real world assembly lines. Furthermore, the preprocessing phase described allows the model simplification by eliminating allocation possibilities that are physically unfeasible.

The effect of this simplification is measured with the use of a testbed of 302 instances from the literature (Scholl and Klein, 1999). Two coefficients are proposed to measure how constrained a line is in terms of task-station / task-worker-station allocation possibilities. The obtained results from the proposed approach are summarized in Table 1 and indicate that a more restricted assembly line problem is solved in less time. The experiments showed that the total processing time diminishes as the number of variables in more restrained problems is smaller. Although the constrained problem might not reach the same optimum cycle time as a theoretical SALBP, practical restrictions involving task-station / task-worker-station are seem in real world assembly lines.

This paper contributes shrinking the gap between the literature and the real applications for ALBP. Although there are several efficient algorithms for solving SALBP, simple problems with no practical restrictions are hardly seem in the industry and focus on re-balancing problems with practical singularities is a relevant issue.

For further studies and researches, improvements in the MILP model can be made, in a way to reduce the processing time. More information can be used to restrain even more the domain of variables, such as logical inferences from the precedence diagram. According to Falkenauer (2005), models have to contempt features such as respecting unmovable operations or stations, zoning constraints, multiple operator tasks, and ergonomic constraints. Some of them are treated in the present model, but there is still no single model that treats every need of real world problems.

Acknowledgments

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

References

Bhattacharjee, T. and Sahu, S. (1990). Complexity of single model assembly line balancing problems, *Engineering Costs and Production Economics* **18**(3): 203–214.

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

Donnini, N., Magatão, L. and Rodrigues, L. C. A. (2010). Balanceamento de uma Linha de Montagem de Bancos de Automóveis com Buffer Intermediário Usando Programação Inteira Mista, *Proc. of XLII SBPO*, Bento Gonçalves, pp. 72–83.

Falkenauer, E. (2005). Line Balancing in the Real World, *Proceedings of the International Conference on Product Lifecycle Management PLM 05*, Lyon, pp. 360–370.

Gamberini, R., Grassi, A. and Rimini, B. (2006). A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem, *International Journal of Production Economics* **102**(2): 226–243.

Gokcen, H. and Erel, E. (1998). Binary Integer Formulation for Mixed-Model Assembly Line Balancing Problem, *Computers & Industrial Engineering* **34**(2): 451–461.

Hillier, F. S. and Lieberman, G. J. (2001). *Introduction to operations research*, McGraw-Hill series in industrial engineering and management science, McGraw-Hill.

Magatão, L., Rodrigues, L. C. A., Marcilio, I. and Skraba, M. (2011). Otimização do Balanceamento de uma Linha de Montagem de Cabines de Caminhões por meio de Programação Inteira Mista, *Proc. of XLIII SBPO*, Ubatuba, pp. 1–12.

Makssoud, F., Battaïa, O., Dolgui, A., Mpofu, K. and Olabanji, O. (2015). Re-balancing problem for assembly lines: new mathematical model and exact solution method, *Assembly Automation* **35**: 16–21.

Rodrigues, L. C. A., Mibach, F. A. R., Campos, L. A. M. and Magatão, L. (2013). Balanceamento da Produção de uma Linha de Usinagem em uma Empresa de Autopeças de Curitiba, *Proc. of XLV SBPO*, Natal, pp. 10–19.

Scholl, A. and Klein, R. (1999). Balancing assembly lines effectively - A computational comparison, *European Journal of Operational Research* **114**(1): 50–58.

Tuncel, G. and Aydin, D. (2014). Two–sided assembly line balancing using teaching–learning based optimization algorithm, *Computers & Industrial Engineering* **74**(1): 291–299.

Wee, T. and Magazine, M. (1982). Assembly line balancing as generalized bin packing, *Operations Research Letters* **1**(2): 56 – 58.

Williams, H. (1999). *Model Building in Mathematical Programming*, A Wiley-Interscience publication, Wiley.

Wolsey, L. (1998). *Integer Programming*, Wiley Series in Discrete Mathematics and Optimization, Wiley.