International Journal of Production Research
Vol. xx, No. xx, 00 Month 2014, 1–18

RESEARCH ARTICLE

Sequencing mixed-model assembly lines operating with a heterogeneous workforce

Pâmela M. C. Cortez[1] and Alysson M. Costa[2*]

[1] *Departamento de Ciências Exatas - Universidade Estadual de Feira de Santana, Brazil*
[2] *Department of Mathematics and Statistics - University of Melbourne, Australia*

(*April - 2014*)

We study the problem of sequencing mixed-model assembly lines operating with a heterogeneous workforce. The practical motivation for this study comes from the context of managing assembly lines in sheltered work centers for the disabled. We propose a general framework in which task execution times are both worker and model dependent. Within this framework, the problem is defined and mathematical mixed-integer models and heuristic procedures are proposed. These include a set of fast constructive heuristics, two local search procedures based on approximate measures using either a solution upper bound or the solution of a linear program and a GRASP metaheuristic. Computational tests with instances adapted from commonly used literature databases are used to validate the proposed approaches. These tests give insight on the quality of the different techniques, which prove to be very efficient both in terms of computational effort and solution quality when compared to other strategies such as a random sampling or the solution of the MIP models using a commercial solver.

Keywords: mixed-model assembly lines, heterogeneous workers, disabled workers, heuristics, mixed-integer programming.

*Corresponding author. Email: alysson.costa@unimelb.edu.au

ISSN: 1478-6451 print/ISSN 1478-646X online
© 2014 Taylor & Francis
DOI: 10.1080/1478645YYxxxxxxx
http://www.informaworld.com

1. Introduction

Mixed-model assembly lines have gained importance in the last decades due to the increasing demand for mass customised products. These production systems allow for such customisation without renouncing the main efficiency benefits associated with general assembly lines. This is done with the aid of flexible resources (workers, machinery) that enable the sequential assembly of slightly different products with little or no setup times.

The different versions of a product (models) usually share a set of common tasks (possibly with different execution times per model) while each version might require some specific ones. The execution of all these tasks must obey a partial precedence order. A popular strategy used to balance mixed-model lines is to create an equivalent (average) product in which each task execution time is an average of the execution times for the different models weighted by the model demands (Thomopoulos 1970). Since average times are considered, a workstation will normally present different workloads for different models.

The actual performance of a solution obtained with such a strategy will depend on the order the different models are assembled. Indeed, if models with high loads for a given workstation are processed in sequence, this workstation will end up exceeding its allowed time (work overload) and extra work will be needed to complete the tasks in time. With a production sequence that alternates high and low demanding models, this need for extra work can be reduced to some extent. The problem of sequencing the models in order to minimize some objective such as work overload is known as the *mixed-model sequencing problem.*

Boysen *et al.* (2009) survey the literature for different mixed-model sequencing approaches. The authors classify the research in the area in three main families: mixed-model sequencing, car sequencing and level scheduling. Both mixed-model and car sequencing concern the minimization of sequence dependent work overload but car sequencing does so in an implicit manner by looking for sequences which follow some desired general rules. Level scheduling, in turn, seeks solutions that aligns with Just-in-time philosophy and looks for ideal part demand rates.

In this paper, we are interested in mixed-model sequencing. For this family of problems, Boysen *et al.* (2009) present a tuple-based taxonomy that focus on three main elements: stations operational characteristics, assembly line characteristics and objectives to be followed. The authors' literature survey lists 35 articles dealing with mixed-model sequencing including the introductory paper by Wester and Kilbridge (1963). Along with the survey, the authors list the main literature gaps, mentioning the little amount of research that has been devoted to mixed-model sequencing while including a) stochastic task times b) special layouts or c) heterogeneity on station characteristics, ($\alpha_3 = p^{sto}$), ($\beta_5 \neq o$) and ($\beta_2 = div$) in the authors' taxonomy, respectively. The literature has since then tried to fill some of these gaps. Indeed, a number of articles dealing with mixed-model sequencing with stochastic execution times (Agrawal and Tiwari 2008, Özcan *et al.* 2011, Dong *et al.* 2014) and with different layouts such as parallel lines (Özcan *et al.* 2010, Öztürk *et al.* 2013, Kucukkoc and Zhang 2014) and U-shaped lines (Kim *et al.* 2006, Hwang and Katayama 2010, Li *et al.* 2012, Lian *et al.* 2012, Hamzadayi and Yildiz 2013) have been published recently. Due to the difficulty of solving large instances to optimality in reasonable computational times, the majority of the papers present a metaheuristic approach, the most pervasive being genetic algorithms (Akgündüz and Tunalı 2011).

As for the third gap, the heterogeneity on station characteristics, to the best of our knowledge, there is still a lack of research in the literature. In this paper, we focus on this lacuna by introducing a sequencing problem in which stations are highly heterogeneous. More specifically, we consider that the amount of extra work (utility work, in the literature jargon) needed to complete the processing in an overloaded station depends not only on the tasks assigned to the station, but on the station itself.

The practical motivation for this study comes from the operation of assembly lines in sheltered work centers for the disabled. The topic of balancing assembly lines in such a context has received a considerable amount of attention in recent years. Indeed, since the introductory paper by Miralles *et al.* (2007), a number of articles have been devoted to the *assembly line worker assignment and balancing problem* (ALWABP) (Blum and Miralles 2011, Moreira *et al.* 2012, Mutlu *et al.* 2013, Vilà and Pereira 2014, Borba and Ritt 2014) and to variants of it including job rotation objectives (Costa and Miralles 2009, Moreira and Costa 2013), different line layouts (Araújo *et al.* 2012, 2014) and more general industrial settings that extrapolate that of the sheltered work centers for the disabled (Moreira *et al.* 2014).

This article aims at modelling and solving mixed-model assembly lines with heterogeneous workers. This is the first time this problem is being tackled and this fact guides this research in the direction of finding a useful general framework and simple solution methods, instead of in the development of problem-tailored metaheuristic strategies. The idea is to present methods that are flexible enough to incorporate additional problem characteristics that might be important in practical contexts.

Within the above framework, the problem described in this paper consists of a worker/task assignment balancing stage that is followed by a model production sequencing stage. The more tactical balancing problem is solved using an average model approximation and generates the data for the operational sequencing problem. This is a common approach in the literature and is often justified by the fact that the two problems, although interconnected, frequently occur at different time frames (Emde *et al.* 2010). We then focus on solving the sequencing problem that arises after the line has been balanced. The first challenge is to conceive a formal definition that is aligned to what happens in practical situations. This is done by introducing two mixed-integer formulations in the following section. Then, in Section 3, heuristic solution procedures that rely on priority rules, local search and linear programming are discussed. Results for the proposed methods are presented and analyzed in Section 4. General conclusions and hints for future research end this paper in Section 5.

2. Problem definition and mathematical models

In this paper, we focus our attention on the situation faced by sheltered work centers for the disabled. This seems to be a natural choice for the study of sequencing problems with a heterogeneous workforce since this is the context most used by the literature when dealing with heterogeneity in the balancing problem. The main assumptions used in this case are:

- The assembly line is paced, with the conveyor belt speed being constant and much lower than the walking speed of workers;
- Workers cannot cross their stations' boundaries;
- As soon as a worker finishes his tasks he instantaneously returns to the left-hand border of his station or until he finds a new product;
- There are worker/tasks incompatibilities, meaning that a worker might not be able to execute some tasks. Nevertheless, if a worker is able to execute a task, he can perform

this task in all models;
- Task execution times are worker-dependent.

This paper introduces the question of mixed-model production in this context. In accordance with the common practice of many centers of using more experienced employees as utility (extra) workers helping the regular (line) workers, we also assume:

- Utility workers are able to execute all tasks;
- Utility workers execute a task as quickly as the most skilled regular worker for that task;
- A regular worker and a utility worker can simultaneously act on the same workpiece, without interfering with each other (side-by-side policy).

The optimisation of mixed-model lines has two important components: the balancing of the line itself which, in this case, requires the assignment of tasks and workers to stations; and the sequencing of the models to be produced. In this paper, we use a successive planning approach which first balances the line and only then sequences the models to be produced. As mentioned earlier, the main justification for this strategy is the fact that balancing and sequencing usually occurs in different time frames, with line balancing occurring in a more tactical level while product sequencing must be effected in an operational basis. The focus of this paper is the sequencing problem, for which we propose two mixed-integer models in Subsection 2.2. For the sake of completeness, we first briefly introduce the balancing problem in the following subsection along with an illustrative simplified example.

2.1. *Balancing problem*

In this paper, we deal with the balancing problem by using the strategy of Thomopoulos (1970), which consists in applying a single-model solution strategy in an *average product*. The task execution times of this idealized product are averages of the different models task execution times weighted by the expected product demands.

In our case, the resulting problem is a single-model assembly line balancing problem with worker-dependent task execution times or, in other words, an instance of the ALWABP. The solution of this problem (which, in the case of this paper is obtained by solving its mixed-integer model with a commercial branch-and-cut package) originates the input for the sequencing of the models. Namely, for each station k, the balancing problem provides both the worker w_k and the set of tasks N_k assigned to it.

2.1.1. Balancing problem example

Consider an assembly line with two stations and two tasks, where task 1 precedes task 2. If no heterogeneity between workers is considered the solution is trivial. Considering, however, that the efficiency depends on which worker executes which task, we must assign workers to stations.

Table 1. Execution times of each worker in each model.

Task	Model 1		Model 2		Model 3	
	worker 1	worker 2	worker 1	worker 2	worker 1	worker 2
1	20	16	10	16	20	10
2	15	14	13	12	14	15

Table 1 shows the execution times of each worker. A feasible solution is to assign worker

1 to station 1 and worker 2 to station 2. In this solution, assuming model demands are equal, the cycle time is computed as $max\{\frac{20+10+20}{3}, \frac{14+12+15}{3}\} = 16,7$ (the average time among all models of the station with higher workload). The optimal solution, nevertheless, is to keep the task assignment but change worker's stations, which yields the minimum cycle time of 14 unities of time: $max\{\frac{16+16+10}{3}, \frac{15+13+14}{3}\} = 14$.

From this very simple example (only two tasks), we can infer that changes in the assignment of tasks may imply changes in the assignment of workers and, therefore, both decisions have to be made simultaneously.

2.2. *Sequencing problem*

The goal of the sequencing problem is to obtain a processing order for the items to be produced in a way to minimize external intervention, i.e., the necessity of additional utility work in the stations. In the highly heterogeneous situation faced by sheltered work centers, regular workers are not necessarily faster or slower than each other (they can be faster or slower depending on the task being executed). Therefore, the real amount of utility work needed to complete the tasks on a given station ultimately depends on the actual tasks the utility worker executes in the place of the regular worker (and not only on the station overload, as it is the case in lines with homogeneous workers). Our first model considers exactly this situation by defining variables:

x_{mi} :	binary variables equal to one if the i^{th} product is of model m;
s_{ki} :	continuous variables indicating the start processing position of the i^{th} product in station k;
y_{mij} :	continuous variables indicating the percentage of task j of model m that is executed by a utility worker on the i^{th} product.

Table 2. Notation

N	set of tasks,
K	set of stations,
W	set of workers,
M	set of models,
d_m	demand for model m,
I	set of positions in the processing sequence, $I = \{1, \ldots, \sum_{m \in M} d_m\}$,
C	cycle time,
l_k	"length" of station k given in time units (l_k = length of station k/conveyor speed $\geq C$),
w_k	worker assigned to workstation k,
N_k	set of tasks assigned to workstation k,
t_{jmw}	processing time of task j in model m when executed by worker w,
t^u_{jm}	processing time of task j in model m when executed by a utility worker,
$\hat{t}_{mk}$	time required by worker w_k to finish tasks N_k on model m,
$\hat{t}^u_{mk}$	time required by a utility worker to finish tasks N_k on model m.

These variables define the order of production (x_{mi}), the eventual cumulative delays in the initial processing of products in stations (s_{ki}) and the tasks executed by utility workers (y_{mij}). With the notation presented in Table 2, a first optimisation model can be written as:

$$\text{Min} \sum_{m\in M}\sum_{i\in I}\sum_{j\in N} t^u_{jm}\cdot y_{mij} \tag{1}$$

subject to

$$\sum_{m\in M} x_{mi} = 1, \qquad \forall i\in I, \tag{2}$$

$$\sum_{i\in I} x_{mi} = d_m, \qquad \forall m\in M, \tag{3}$$

$$y_{mij} \le x_{mi}, \qquad \forall m\in M, \forall i\in I, \forall j\in N, \tag{4}$$

$$s_{ki} + \sum_{m\in M}\hat{t}_{mk}\cdot x_{mi} - C - \sum_{m\in M}\sum_{j\in N_k} t_{jmw_k}\cdot y_{mij} \le s_{k,i+1}, \qquad \forall i\in I, \forall k\in K, \tag{5}$$

$$s_{ki} + \sum_{m\in M}\hat{t}_{mk}\cdot x_{mi} - \sum_{m\in M}\sum_{j\in N_k} t_{jmw_k}\cdot y_{mij} \le l_k, \qquad \forall i\in I, \forall k\in K, \tag{6}$$

$$s_{k1} = 0, s_{k,|I|+1} = 0, \qquad \forall k\in K, \tag{7}$$

$$s_{ki} \ge 0, \qquad \forall i\in I, \forall k\in K, \tag{8}$$

$$x_{mi} \in \{0,1\} \qquad \forall m\in M, \forall i\in I, \tag{9}$$

$$0 \le y_{mij} \le 1 \qquad \forall m\in M, \forall i\in I, \forall j\in N. \tag{10}$$

The objective function minimizes the amount of utility work (in time unities) needed and, therefore, consider the appropriate task execution times. Constraints (2) and (3) ensure that one model is assigned to each sequencing position and that these assignments meet the total demand, respectively. Constraints (4) link variables y_{mij} and x_{mi} by stating that a utility worker can only process a task on a model in a given sequencing position if that position is indeed occupied by such a model. Constraints (5) state that a workpiece can only be started after its predecessor has finished while constraints (6) force every item to be finished within the station's boundaries. Note that, differently from the objective function, these two set of constraints consider execution times associated with the regular worker assigned to the station. Constraints (7) state that before production begins ($i = 0$) and after it ends ($i = |I| + 1$), regular workers are on the leftmost position of their stations.

Once more, we highlight that the objective function minimizes the time spent by the utility workers (by weighting the percentage of task j executed in model m by the time needed by the utility worker to execute that task: t^u_{jm}). Nevertheless, due to the heterogeneity between workers and utility workers, this additional work has different effects on constraints (5) and (6), with the time discounted being now weighted by the execution times of the actual worker regularly operating at the station (t_{jmw_k}).

The model above introduces a high level of detail, indicating which tasks are effectively executed by utility workers at each station. In real world situations, this information is rarely relevant, since one is more interested in the general trend and since other model simplifications (such as assuming side-by-side policy) are often more important. With this in mind, we define the following parameter, which measures the relative efficiency ratio between the regular worker in station k and a utility worker:

$$\Delta\hat{t}_{mk} = \hat{t}_{mk}/\hat{t}^{u}_{mk}, \; m = 1, \cdots, |M|, \; k = 1, \cdots, |K| \tag{11}$$

With this parameter, a simplified model is proposed. This model uses variables y_{ki} representing the amount of time a utility worker acts in station k when the i^{th} product is being processed. The new model reads:

$$\text{Min} \sum_{i \in I} \sum_{k \in K} y_{ki} \tag{12}$$

subject to

$$\begin{aligned}
&(2)\text{–}(3) && && (13)\\
s_{ki} + \hat{t}_{mk} \cdot x_{mi} - C - \Delta\hat{t}_{mk} \cdot y_{ki} &\le s_{k,i+1}, && \forall m \in M, \forall i \in I, \forall k \in K, && (14)\\
s_{ki} + \hat{t}_{mk} \cdot x_{mi} - \Delta\hat{t}_{mk} \cdot y_{ki} &\le l_k, && \forall m \in M, \forall i \in I, \forall k \in K, && (15)\\
&(7)\text{–}(9) && && (16)\\
y_{ki} &\ge 0, && \forall k \in K, \forall i \in I. && (17)
\end{aligned}$$

Constraints (5) and (6) are replaced by constraints (14) and (15). These new constraints approximate the net effect of the additional work by the averaged parameter $\Delta\hat{t}_{mk}$. In these equations, task times are measured with respect to the regular worker reference. Therefore, when a utility worker spends y_{ki} units of time on station k and item i, he actually reduces the load of the station to be executed by worker w_k by $\Delta\hat{t}_{mk} \cdot y_{ki}$ units of time in average. As shown in Table 2, this average is computed considering tasks N_k present in the station. Note that when $x_{mi} = 0$, these constraints are trivially satisfied with $y_{ki} = 0$, so in each station k, the work overload y_{ki} is defined by the time needed to build model m in that station ($\hat{t}_{mk}$), by the relative efficiency ratio ($\Delta\hat{t}_{mk}$) and by the starting position of the regular worker on the station (s_{ki}). It is to be expected that if two heavy models are next to each other in the production sequence, the optimization will prefer to assign utility work mainly to help finishing the model in which the regular worker is less efficient (that with higher $\Delta\hat{t}_{mk}$). The following example illustrates this situation.

2.2.1. *Sequencing problem example*

In order to illustrate the problem and show the effect of considering a heterogeneous workforce, consider a mixed-model assembly line with two stations and three models. We assume the line has been balanced (see Section 2.1.1) and that the execution times of the tasks in the stations are given in Table 3, which shows the figures for each station and model, both for the regular worker currently assigned to the station and for a utility worker.

Consider, for example, the case of station 1: the worker assigned to this station takes 16, 16 and 10 unities of time to execute the tasks (assigned to that station) associated with models 1, 2 and 3, respectively. A utility worker executing the same tasks would be faster in model 2, in which he would only take 10 unities of time (instead of 16). In other

Table 3. Regular and utility worker execution times, and the relative efficiency ratio.

Model	Station 1			Station 2		
	w_1	utility worker	$\Delta\hat{t}_{m1}$	w_2	utility worker	$\Delta\hat{t}_{m2}$
1	16	16	1	15	14	1.07
2	16	10	1.60	13	12	1.08
3	10	10	1	14	14	1

words, each unit of time the utility worker spends at station 1 while executing tasks for model 2 will contribute 1.6 units of time to the completion of those tasks (in the scale of the actual worker assigned to that station). If choice is available, it is therefore expected that a solution minimizing total extra work will prioritize assigning utility workers during the execution of model 2.

This can be seen with a simple example with a unitary demand for each of the models, cycle time of 14 and station length of 15.4 (1.1 $\times$ the cycle time). The optimal solution for the problem is represented in the Gantt chart of Figure 1 and corresponds to the model sequence 1, 2, 3. Station 2 does not need extra work, since the extra time used at model 1 is compensated by the shorter model 2 that comes just after. In station 1, however, there is the need of extra work during the processing of both models 1 and 2. The minimum amount of extra work (0.375 unit – in order to finish tasks inside the station's boundary) is assigned during the processing of model 1 (represented by a thicker line). Therefore, the processing starts late for model 2, even if its execution time is also 2 unities higher than the cycle time. In order to compensate these 2 unities, only 1.25 units of extra work is needed, since each unit of extra work in that model is worth 1.6 units of processing time.

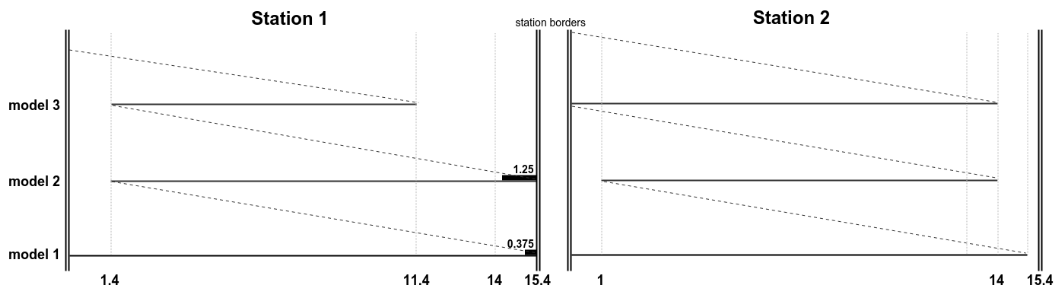

Figure 1. Gant chart for stations 1 and 2 of the example

2.3. *Lower and Upper Bounds*

In this subsection, we present simple lower and upper bounds for the optimal solution of model (12)-(17). It is known that when the workload of station k is greater than its size in units of time (l_k), utility worker intervention is mandatory. According to the regular worker speed, a utility worker should help $\hat{t}_{mk} - l_k$ units of time. But utility workers are in average $\Delta\hat{t}_{mk}$ faster than worker w_k, so actually they have to spend $(\hat{t}_{mk} - l_k)/\Delta\hat{t}_{mk}$ units of time to complete the tasks before the right-hand border of station is reached. As a result, a lower bound on the work overload in all stations and planning horizon is given by equation (18).

$$LB = \sum_{m \in M} \sum_{k \in K} d_m \cdot max\{\hat{t}_{mk} - l_k, 0\} / \Delta\hat{t}_{mk} \tag{18}$$

Analogously, an upper bound can be obtained by summing, in all stations, all workload that is larger than the cycle time despite the fact that some part of it could be compensated by assigning lighter models around it. Because of the aforementioned reason, this amount of time must also be divided by the relative efficiency ratio. This upper bound is expressed below:

$$UB = \sum_{m \in M} \sum_{k \in K} d_m \cdot max\{\hat{t}_{mk} - C, 0\} / \Delta\hat{t}_{mk} \tag{19}$$

3. Solution methods

In order to solve the mixed-model sequencing problem with heterogeneous workers, we propose a constructive heuristic method based on three different criteria and two local search procedures, one using a simplified version of model (12)-(17) and the other using an approximation of such model. We also combine these procedures into a GRASP metaheuristic. The proposed methods are detailed in the following subsections.

3.1. *Constructive Heuristics*

The main rationale behind the proposed heuristics is to alternate models with high and low time requirements in the stations (Scholl *et al.* 1998). The solution is build in a constructive fashion, by fixing a model in a given position of the sequence and moving on to the next position. Let us define $wo(mk)$ as the difference between the time needed in station k when processing model m and the cycle time:

$$wo(mk) = \hat{t}_{mk} - C \tag{20}$$

Clearly, $wo(mk)$ is positive if the station requires more than the cycle time for the given model and negative otherwise. For simplicity of presentation, let us define the positive and negative values of $wo(mk)$ as $wo^+(mk) = max\{wo(mk), 0\}$ and $wo^-(mk) = max\{-wo(mk), 0\}$, respectively.

Assume m_{i-1} is the model selected for position $i-1$ of the production schedule. We propose two criteria for the decision on the next model to be produced:

$$\text{arg min}_m \left(g^1_{im} = \sum_{k \in K} max\{wo^+(m_{i-1}k) - wo^-(m_i k), 0\} \right), \tag{21}$$

and

$$\text{arg min}_m \left(g_{im}^2 = \sum_{k \in K} |wo(m_{i-1}k) + wo(m_ik)| \right). \qquad (22)$$

The model to be produced in the i^{th} position of the production sequence should be the one (among the available models, i.e., those whose demand has not yet been fulfilled) that minimizes g_{im}^1 or g_{im}^2. The rationale behind g_{im}^1 is to sum, for all the stations, the amount of work overload associated with the model in the $(i-1)^{th}$ productive cycle that could not be compensated by idle time in the i^{th} cycle. Criterium g_{im}^1 tends to greedly assign models with large idle times very early on the productive sequence, resulting in large work overloads for the last products to be sequenced. To reduce this effect, criterium g_{im}^2 is penalized not only by non-compensated work overload but also by excessive idle time.

To complete the methods, it suffices to decide on the assignment of the first model. In both cases, the model with lowest idle time is the first to be produced (the rationale being that the idle time of the first item cannot be used to compensate for other items work overloads).

With respect to criterium g_{im}^2, the following modification has also been proposed:

$$\text{arg min}_m \left(g_{im}^3 = \sum_{k \in K} |wo(m_ik) + wo(m_{i+1}k)| \right). \qquad (23)$$

In this case, the first model to be assigned (in the last position of the sequence) is the one with the lowest work overload. The algorithm then works backwards to decide the models to be produced in the remaining positions.

Table 4 summarizes the characteristics of each greedy heuristic proposed. The table also includes a version that works by simultaneously fixing models at the beginning and end of the production sequence (HC_3), which is described in Algorithm 1.

Table 4. Constructive heuristics

	First model to fix		Direction	Measure to
	position	characteristic		be minimized
HC_1	$\|I\|$	lowest work overload	backward	g_{im}^3
HC_2	1	lowest idle time	forward	g_{im}^2
HC_3	1	lowest idle time and	forward and	g_{im}^2 and
	$\|I\|$	lowest work overload	backward	g_{im}^3
HC_4	1	lowest idle time	forward	g_{im}^1

In this algorithm, S is an array containing the solution ($S[i]$ contains the model assigned to the i^{th} position in the production sequence) and d_m^* is the residual demand of model m. The algorithm works by fixing the first and last models (lines 1 and 2) and then sequentially fixing the models simultaneously in forward (lines 6-11) and backward (lines 14-19) orders. This algorithm can be easily adapted to HC_1, HC_2 or HC_4 simply by changing the criteria used, depending on the processing direction (forward or backward) adopted by the specific heuristic (see Table 4), and doing only one of the loops in lines 6-11 or 14-19 (now for all the sequencing positions and not only for half of them in each loop).

Algorithm 1: Constructive Heuristic HC_3

```
1  $S[1] = \arg\min_m \sum_{k \in K} wo^-(mk)$ (lowest idle time)
2  $S[|I|] = \arg\min_m \sum_{k \in K} wo^+(mk)$ (lowest work overload)
3  for $i \in \{2, \ldots, \lfloor |I|/2 \rfloor\}$ do
4      /* step forward */
5      bestvalue = $\infty$
6      for $m \in M$ do
7          if $d^*_m > 0$ and $g^2_{im} <$ bestvalue then
8              bestm = $m$
9              bestvalue = $g^2_{im}$
10     $S[i]$ = bestm
11     $d^*_{bestm} = d^*_{bestm} - 1$ (update residual demand)
12     /* step backward */
13     bestvalue = $\infty$
14     for $m \in M$ do
15         if $d^*_m > 0$ and $g^3_{im} <$ bestvalue then
16             bestm = $m$
17             bestvalue = $g^3_{im}$
18     $S[|I| - i + 2]$ = bestm
19     $d^*_{bestm} = d^*_{bestm} - 1$ (update residual demand)
20 if $|I|$ is odd then
21     $S[\lceil |I|/2 \rceil] = m \mid d^*_m > 0$
22 return $S$
```

3.2. ***Local Search***

Any sequence of models is a feasible solution to the mixed-model sequencing problem as long as it respects the demands for each model. Therefore, starting with a feasible solution, one can look for new solutions with *swap movements* changing the position of two models in the production sequence. The evaluation of the new solution requires the computation of the work overloads in all stations for all $|I|$ productive cycles. An exact and an approximate way of computing this measure are proposed and give origin to the two local search procedures that will be explained in the following.

In both cases, to speed up the search, not all swap movements were tested, but only those around the production position i' containing the highest work overload: $i' = \arg\max_i \sum_{k \in K} wo_{ik}$, the idea being that the total work overload can be reduced if good compensation is made around positions with high work overloads. Randomness can be introduced in this choice by using a roulette wheel based selection: positions with higher work overloads are more likely to be selected (Goldberg 1989). In our tests, the roulette wheel selection was able to reduce premature convergence when using the exact work overload measure but had little effect when using the approximate one. Both algorithms are detailed in the following.

3.2.1. Exact Local Search

In this local search procedure (LS_E) the computation of the work overload for each new tentative solution is made by solving model (12)-(17) with x_{mi} variables fixed at the current solution values, reducing the model to a linear program. This local search procedure is described in Algorithm 2.

Algorithm 2: Local Search LS_E

```
1  $S^* = S$
2  improved = true
3  while improved and time $\leq$ time limit do
4      improved = false
5      for $i' \in$ roulette and !improved do
6          for $j \in \{i'-1, i'+1\}$ and !improved do
7              for $i \in I$ and !improved do
8                  if $m_i \neq m_j$ then
9                      swap$(S, i, j)$
10                     if value$_E(S) <$ value$_E(S^*)$ then (evaluated by model (12)-(17))
11                         improved = true
12                         $S^* = S$
13                         update(roulette)
14                     else
15                         swap$(S, i, j)$(undo swap)
16 return $S^*$
```

In this algorithm, S^* is the best solution found so far and S is the working solution. Both S^* and S are initially set at the original solution (obtained with a constructive heuristic, e.g.). Procedure *roulette* returns one sequence position using a roulette scheme weighted with the position work overloads serving as fitness values (Goldberg 1989), m_i is the model in the i^{th} position in the production sequence and $value_E(S)$ is the total work overload of solution S when evaluated by model (12)-(17) with variables x_{mi} fixed at the values indicated by solution S.

The procedure starts with an incumbent solution and uses a roulette procedure in order to select the 'pivot element in the sequence' (line 5). Then, the algorithm tries to swap the neighbors of this pivot element with other models in the sequence and evaluates the resulting work overload with the reduced version of model (12)-(17) (lines 6-9). If a better solution is found, the procedure is restarted and the process continues until some time limit has been reached or no improvement can be made.

3.2.2. Approximate Local Search

Alternatively, we can try to avoid the burden of solving multiple linear programs by proposing the following approximate measure for the work overload on station k in the i^{th} productive cycle:

$$awo(ik) = \begin{cases} \Delta wo^+(m_ik), & \text{if } i = |I| \\ max\{\Delta wo^+(m_ik) - \Delta wo^-(m_{i+1}k), 0\}, & \text{if } i < |I| \end{cases} \tag{24}$$

where m_i is the model assigned to position i, $\Delta wo^+(m_ik) = min\{wo^+(m_ik), l_k - C\}$ and $\Delta wo^-(m_ik) = min\{wo^-(m_ik), l_k - C\}$.

This measure ignores the dependence between non-adjacent items in the sequence and only discount from the total work overload the amount that is compensated by the following item.

A simplified pseudo-code for procedure LS_A can be seen in Algorithm 3. The method chooses the position in the sequence with the t^{th} higher work overload (starting with t=1 and with ties broken arbitrarily) as its pivot element (line 5), and then tries to

Algorithm 3: Approximate Local Search LS_A

```
1  min = value_A(S) (evaluated by equation (24))
2  repeat
3      best_i = −1
4      for t ∈ I and best_i < 0 do
5          i' = t^th arg max_i ∑_{k∈K} awo(ik)
6          for j ∈ {i' − 1, i' + 1} do
7              for i ∈ I do
8                  if m_i ≠ m_j then
9                      swap(S, i, j)
10                     if value_A(S) < min then
11                         min = value_A(S)
12                         best_i = i
13                         index = j
14                     swap(S, i, j) (undo swap)
15     if best_i > 0 then (new incumbent solution found)
16         swap(S, best_i, index)
17 until best_i < 0
```

swap the models in its neighboring positions to other elements in the sequence with the goal of reducing the work overload computed by criteria (24). If no improving movement is obtained, the algorithm chooses the element with second highest work overload and repeats the procedure until all positions have been tried.

In summary, we propose two local search heuristics. Both look for swap movements in a reduced all-pairs neighborhood that favours production positions with high work overload. In the first case, that we call LS_E, the real work overload is computed using model (12)-(17) with x_{mi} variables fixed and the pivot position (the position whose neighbors will be swapped) is chosen with a roulette-based selection. In the second strategy, that we call LS_A, approximate measure (24) is used and the pivot position is the non-tested position with the highest work overload (i').

Although LS_A gives a rough estimation of the real work overload, our preliminary tests showed that there was a good correlation between the ranking of the solutions in a swap neighborhood when using the exact solution of reduced model (12)-(17) and when using the approximate measure (24). This has motivated us to use this latter measure as a quick guide for finding promising regions in the search space. In Section 4 these methods are evaluated through a series of computational experiments.

3.3. *GRASP*

GRASP (Greedy Randomized Adaptive Search Procedure) is a metaheuristic procedure that relies on a randomized greedy constructive phase followed by local search. The uncertainty is introduced by means of a semi-random selection in the constructive phase: instead of selecting the candidate with the best greedy criterium, one chooses randomly within a list of good candidates (Resende and Ribeiro 2003).

In each step of the construction phase, the choice to be made is which model we should assign to the next position in the sequence. At each stage, there are at most $|M|$ different options and the model to be assigned is randomly chosen within the m' models with best greedy criterium. After the obtention of this semi-greedy solution, local search procedure

LS_A is run. The whole method can be repeated in a multi-start strategy to obtain new solutions.

Algorithm 4 details the implemented version of the metaheuristic. In this algorithm, *rcl* is a list containing the models with positive residual demand. The construction phase is executed along the lines of HC_2: lines 6 to 8 evaluate criterium (22) for each model whose demand has not yet been fulfilled. But instead of choosing the best model, line 10 randomly selects a model from the top m' candidates. Finally, LS_A is called to improve the semi-greedy solution obtained. This process is repeated during *time limit* seconds.

Algorithm 4: GRASP

```
1  $S^* = \text{NULL}$
2  while $time \leq time\ limit$ do
3      /* construction phase */
4      $S[1] = \arg\min_m \sum_{k \in K} wo^-(mk)$ (lowest idle time)
5      for $i \in \{2, \ldots, |I|\}$ do
6          for $m \in M$ do
7              if $d^*_m > 0$ then
8                  rcl.add(m, $g^2_{im}$)(builds restricted candidate list)
9          sort(rcl) (decreasing and according to the second element of the pair (model,
           measure))
10         m = rand($rcl(1), \ldots, rcl(m')$)
11         $S[i] =$ m
12         $d^*_m = d^*_m - 1$
13     /* local search phase */
14     $LS_A(S)$
15     if value$_A(S) <$ value$_A(S^*)$ then
16         $S^* = S$
17 return $S^*$
```

4. Computational results

In order to assess the quality of the solutions obtained by the proposed methods, computational tests were executed with two problem sets: Heskia and Roszieg – each one with 8 groups of 10 instances. These instances have been proposed by Chaves *et al.* (2007) and are widely used for testing planning methods for the ALWABP. Table 5 lists the problem set characteristics.

Table 5. Instances characteristics

Family	Number of tasks ($\|N\|$)	Number of workers ($\|W\|$)	Order Strength
Heskia	28	4 (groups 1–4) or 7 (groups 5–8)	22, 49
Roszieg	25	4 (groups 1–4) or 6 (groups 5–8)	71, 67

In each proposed instance, the original ALWABP instance task times were used as task times data for the first model. To generate task times for the remaining models, we multiplied the original task times by a random value in the range $\{0, 4\}$ (zero being equivalent to a task not required by the model). The same value was applied for all workers, but a new random value was taken for each task and model. We defined 25 different models and a total demand of 500 items. The expected demand for each model is a random value in $\lceil [0, 1] * 500 \rceil$.

The tests were run in an Intel® Core™ i5-2300 2.8 GHz, 4 GB of RAM and operating system Ubuntu Linux 10.04. The algorithms were implemented using language C++, g++ 4.4 compiler, and CPLEX 12.4.

The constructive heuristics and local search LS_A could run in a few seconds (less than 6, in our tests), so no time limit was imposed. The time limit for heuristics LS_E and GRASP was set as 1800 seconds, the same amount of time given to a random procedure, which randomly distributes the models in sequences while respecting their demands, generating feasible solutions and returning the best one found within the time limit. We also ran model (12)–(17) with CPLEX for two days and obtained the best feasible solution within this generous time limit.

Tables 6 and 7 show the gap between each procedure and the best feasible solution found by CPLEX in two days of computing time for families Heskia and Roszieg, respectively. As four constructive heuristics were proposed, we implemented four independent threads to evaluate the heuristics in parallel: for each thread i, LS_A starts with the solution build by HC_i and LS_E refines the solution given by LS_A. So columns a), b) and c) report the best result found by each procedure among the four threads.

Table 6. Gaps (%) between each procedure and the best feasible solution found by CPLEX in two days of computing time for family Heskia.

group	a) HC_s	b) LS_A	c) LS_E	d) GRASP	e) Random
1	-1.65	-2.44	**-3.13**	-2.79	7.73
2	-3.13	-4.58	**-5.12**	-4.85	6.21
3	-3.63	-4.46	**-5.07**	-4.90	5.27
4	-4.56	-5.25	**-5.58**	-5.47	4.23
5	-3.58	-4.76	**-5.22**	-5.03	3.58
6	-3.01	-3.95	**-4.47**	-4.19	3.52
7	-2.22	-3.01	**-3.30**	-3.07	4.25
8	-2.70	-3.59	**-3.96**	-3.80	3.63
average	-3.06	-4.00	**-4.48**	-4.26	4.80

Table 7. Gaps (%) between each procedure and the best feasible solution found by CPLEX in two days of computing time for family Roszieg.

group	a) HC_s	b) LS_A	c) LS_E	d) GRASP	e) Random
1	-2.83	-4.40	**-4.98**	-4.69	9.02
2	0.09	-1.27	-1.69	**-1.73**	12.97
3	-3.47	-4.26	**-4.73**	-4.55	8.83
4	-4.68	-6.54	**-7.43**	-7.11	6.88
5	-4.69	-5.77	**-6.21**	-6.08	5.39
6	-4.31	-5.47	**-5.82**	-5.66	5.54
7	-4.58	-5.62	**-6.05**	-5.89	4.67
8	-3.58	-5.08	**-5.43**	-5.26	5.45
average	-3.50	-4.80	**-5.29**	-5.12	7.34

In only 1 out of 16 groups of instances (group 2 of the Roszieg family), CPLEX found average better results than the constructive heuristics (with a gap of 0.09%). In general,

HC_3 performed better with an average improvement of 2.97% over the best feasible solution found by CPLEX. HC_1 and HC_2 were almost as efficient as HC_3, both presenting improvements of 2.90% over CPLEX. Although HC_4 was significantly worse in terms of average gap (improvements of 2.18%), this constructive heuristic found the best solution in 29% of the instances, showing that the proposed criteria are somehow complementary. Taking the best solution among the four methods yielded an improvement of 3.06% and 3.50% over CPLEX for families Heskia and Roszieg, respectively, as shown in Tables 6 and 7.

In order to verify that the good results obtained by the constructive heuristics were not simply because the problem had multiple (and easily found) local optima, we compared their results to a blind-search procedure that just evaluated sequentially randomly generated solutions (Random). The heuristic solution was 10% better in average (with a maximum improvement of 25%).

As shown in column b) from Tables 6 and 7, heuristic LS_A was not outperformed by CPLEX in any group of instances. CPLEX could find a better result in only 6.25% of the instances (gap of 0.62% when compared to LS_A for those instances). If all instances are considered, LS_A was consistently better, with an overall improvement of 4.40% over the CPLEX solutions, with improvements reaching 14%. Additionally, LS_A was able to quickly improve random generated solutions, motivating the study of a GRASP algorithm which used a semi-greedy version of the HC_2 and the LS_A.

The implemented GRASP algorithm had a static value for the restricted candidate list size, m'. The value $m' = 7$ was chosen after some preliminary tests. Columns b) and d) from Tables 6 and 7 show that the GRASP procedure obtained results just slightly better than LS_A. We also implemented a *random GRASP* in which the constructive phase is replaced by a random procedure. The performance of this method was only 0.5% inferior to the original GRASP, in average, indicating that the driving force behind the algorithm is indeed the quality of the LS_A solutions. This was further confirmed by the good quality solutions obtained when using the available computational time to further explore the solution found by LS_A with the LS_E procedure. Indeed, this was the most successful strategy, yielding improvements over the CPLEX feasible solutions of around 4.5% and 5.3% for the Heskia and Roszieg families, respectively.

5. Conclusions

We study the problem of planning the production sequence in mixed-model assembly lines with hetereogeneous workforce. The problem is motivated by the situation found in sheltered work centers for the disabled but can properly describe other planning situations in assembly lines with different characteristics. We formally define the problem with two mixed integer problems and propose a variety of heuristic methods including constructive heuristics, two local search procedures (one with an approximate metric and the other relying on a linear program) and a GRASP metaheuristic. Computational tests on instances adapted from commonly used databases show that the methods were fast and presented results consistently better than those CPLEX could obtain in two days of computing time. The computational study also highlights the complementarity of the heuristic criteria and the efficiency of the proposed local search procedures. Further study in this area could explore situations in which the practical context allows for both balancing and scheduling problems to be dealt with simultaneously. We believe the methods and models developed here can be of help in that research path.

Acknowledgements

The authors thank two anonymous referees and the Associated Editor for their thoughtful comments which have helped improve both the presentation and the content of this article. The authors also thank FAPESP - Brazil, for the financial support.

References

Agrawal, S. and Tiwari, M.K., 2008. A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, 46, 1405–1429.

Akgündüz, O.S. and Tunalı, S., 2011. A review of the current applications of genetic algorithms in mixed-model assembly line sequencing. *International Journal of Production Research*, 49, 4483–4503.

Araújo, F.F.B., Costa, A.M., and Miralles, C., 2012. Two extensions for the assembly line worker assignment and balancing problem: parallel stations and collaborative approach. *International Journal of Production Economics*, 140, 483–495.

Araújo, F.F.B., Costa, A.M., and Miralles, C., 2014. Balancing parallel assembly lines with disabled workers. *European Journal of Industrial Engineering*, [In press].

Blum, C. and Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. *Computers & Operations Research*, 38, 328–339.

Borba, L. and Ritt, M., 2014. A heuristic and a branch-and-bound algorithm for the Assembly Line Worker Assignment and Balancing Problem. *Computers & Operations Research*, 45, 87–96.

Boysen, N., Fliedner, M., and Scholl, A., 2009. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192, 349–373.

Chaves, A., Miralles, C., and Lorena, L., 2007. Clustering search approach for the assembly line worker assignment and balancing problem. *In*: *Proceedings of 37th ICC&IE*, Alexandria, Egypt, 1469–1478.

Costa, A.M. and Miralles, C., 2009. Job rotation in assembly lines employing disabled workers. *International Journal of Production Economics*, 120, 625–632.

Dong, J., *et al.*, 2014. Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time. *International Journal of Production Research*, DOI: 10.1080/00207543.2014.944280 [In press].

Emde, S., Boysen, N., and Scholl, A., 2010. Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload. *International Journal of Production Research*, 48, 3173–3191.

Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1st ed. Reading, MA: Addison-Wesley Professional.

Hamzadayi, A. and Yildiz, G., 2013. A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. *Computers & Industrial Engineering*, 66, 1070–1084.

Hwang, R. and Katayama, H., 2010. Integrated procedure of balancing and sequencing for mixed-model assembly lines: a multi-objective evolutionary approach. *International Journal of Production Research*, 48, 6417–6441.

Kim, Y.K., Kim, J.Y., and Kim, Y., 2006. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. *European Journal* *of Operational Research*, 168, 838–852.

Kucukkoc, I. and Zhang, D.Z., 2014. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Research*, 52, 3665–3687.

Li, J., Gao, J., and Sun, L., 2012. Sequencing minimum product sets on mixed-model U-lines to minimise work overload. *International Journal of Production Research*, 50, 4977–4993.

Lian, K., *et al.*, 2012. A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem. *International Journal of Production Research*, 50, 5117–5131.

Miralles, C., *et al.*, 2007. Advantages of assembly lines in Sheltered Work Centres for Disabled. A case study. *International Journal of Production Economics*, 110, 187–197.

Moreira, M.C.O. and Costa, A.M., 2013. Hybrid Heuristics for planning job rotation in Assembly Lines with disabled workers. *International Journal of Production Economics*, 141, 552–560.

Moreira, M.C.O., Costa, A.M., and Miralles, C., 2014. Assembly Line Worker Integration and Balancing Problem. *Computers & Operations Research*, DOI: 10.1016/j.cor.2014.08.021 [In press].

Moreira, M.C.O., *et al.*, 2012. Simple heuristics for the assembly line worker assignment and balancing problem. *Journal of Heuristics*, 18, 505–524.

Mutlu, O., Polat, O., and Supciller, A.A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. *Computers & Operations Research*, 40, 418–426.

Özcan, U., *et al.*, 2010. Balancing and sequencing of parallel mixed-model assembly lines. *International Journal of Production Research*, 48, 5089–5113.

Özcan, U., Kellegöz, T., and Toklu, B., 2011. A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. *International Journal of Production Research*, 49, 1605–1626.

Öztürk, C., *et al.*, 2013. Balancing and scheduling of flexible mixed model assembly lines with parallel stations. *The International Journal of Advanced Manufacturing Technology*, 67, 2577–2591.

Resende, M.G.C. and Ribeiro, C.C., 2003. Greedy randomized adaptative search procedures. *In*: F. Glover and G. Kochenberger, eds. *Handbook of Metaheuristics, International Series in Operations Research & Management Science*. Dordrecht: Kluwer Academic Publishers, 219–249.

Scholl, A., Klein, R., and Domschke, W., 1998. Pattern Based Vocabulary Building for Effectively Sequencing Mixed-Model Assembly Lines. *Journal of Heuristics*, 4, 359–381.

Thomopoulos, N.T., 1970. Mixed model line balancing with smoothed station assignments. *Management Science*, 16, 593–603.

Vilà, M. and Pereira, J., 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research*, 44, 105–114.

Wester, L. and Kilbridge, M.D., 1963. The assembly line model-mix sequencing problem. *In*: *Proceedings of the 3rd International Conference on Operations Research*, Oslo, 247–269.