UMA ABORDAGEM MATHEURÍSTICA PARA O PROBLEMA DE SEQUENCIAMENTO E BALANCEAMENTO DE LINHAS DE MONTAGEM COM TEMPOS DE *SETUP* DEPENDENTES DA SEQUÊNCIA

Camilo José Bornia-Poulsen
Karen Juliana Weigner de Bastos
Denise Lindstrom Bandeira

Universidade Federal do Rio Grande do Sul – Programa de Pós-Graduação em Administração
Rua Washington Luiz, 855 – CEP 90010-460 – Porto Alegre/ RS – Brasil
camilo.bornia@ufrgs.br, karendebastos@hotmail.com, dlbandeira@ea.ufrgs.br

RESUMO

O problema de balanceamento e sequenciamento de linhas de montagem com tempos de *setup* dependentes da sequência (SUALBSP em inglês, *Setup Assembly Line Balancing and Scheduling Problem*) envolve a atribuição de tarefas às estações de trabalho e o sequenciamento destas tarefas dentro da estação à qual foi atribuída. Trabalhos anteriores propuseram soluções heurísticas com excelentes resultados, porém o uso de métodos exatos, por meio de algum *MIP solver*, tem apresentado desempenhos decepcionantes, pois contém um subproblema *NP-hard* em todas as estações. Enquanto o modelo de Scholl, Boysen e Fliedner (2013) minimiza prioritariamente o número de estações, o modelo proposto neste trabalho parte da premissa de que este é um dado definido. A partir de uma estimativa inicial de número de estações, processa-se o modelo com o objetivo de distribuir as tarefas e minimizar o tempo total de estação, que é o segundo objetivo do modelo original. Se este processamento for infactível, incrementa-se o número de estações em uma unidade e reprocessa-se o modelo até se encontrar um resultado factível limitado a 100 segundos. Experimentos computacionais em 101 instâncias de dados confirmam o bom desempenho da abordagem proposta, sem qualquer prejuízo à qualidade da solução. Portanto, os resultados apresentados demonstram que há espaço para estudos futuros a partir do uso de matheurísticas.
PALAVRAS-CHAVE. Balanceamento e sequenciamento de linhas de montagem, Otimização combinatória, Programação.
Área Principal: PO na Administração & Gestão da Produção (AD & GP), Otimização Combinatória (OC), Programação Matemática (PM).

ABSTRACT

The setup assembly line balancing and scheduling problem (SUALBSP) involves the assigning of tasks to workstations and the sequencing of these tasks within the station to which they are assigned. Previous work has proposed heuristic solutions with excellent results, but the use of exact methods, by some MIP solver, has shown disappointing performance, because it contains an NP-hard sub problem in every station. While the model proposed by Scholl et al. (2013) primarily minimizes the numbers of stations, our model assumes this as a parameter. From an initial estimate of the number of stations, we process the model allocating tasks and minimizing station times, which is the second objective of the original model. If this processing is infeasible, we increase the number of stations by one unit and we reprocess the model to find a feasible result limited to 100 seconds. Computational experiments in 101 instances of data set confirm the good performance of the proposed approach, without harming the quality of the solution. Therefore, the results show that there are opportunities for future studies based on the use of matheuristics.
KEYWORDS. Assembly line balancing, Combinatorial optimization, Scheduling.
Main Area: OR in Administration & Production Management (AD & GP), Combinatorial Optimization (OC), Mathematical Programming (PM).

1 Introdução

As linhas de montagem visam, principalmente, à eficiência em custo na produção em massa de produtos padronizados. Elas consistem em estações de trabalho ligadas por um sistema de manuseio de material. Esse sistema fornece o material necessário para produzir o produto final e também move o produto pela linha (Yolmeh e Kianfar, 2012). Basicamente, o funcionamento de uma linha de montagem envolve a movimentação dos produtos de uma estação a outra. Em cada estação, um conjunto de tarefas é repetidamente executado obedecendo a um tempo de ciclo.

O problema de balancear otimamente o trabalho de montagem entre as estações de trabalho com respeito a algum objetivo é conhecido como Problema de Balanceamento da Linha de Montagem (ALBP em inglês, *Assembly Line Balancing Problem*). Ele objetiva atribuir as tarefas às estações de trabalho de modo que as relações de precedência não sejam violadas e que alguma medida de desempenho, por exemplo, tempo de ciclo, número de estações de trabalho, eficiência da linha ou tempo ocioso, seja otimizada (Erel e Sarin, 1998). Os problemas de balanceamento da linha de montagem são considerados tarefas de médio e longo prazo e requerem uma grande quantia de capital inicial e, por isso, o planejamento de sua configuração é de grande relevância e atrai a atenção de muitos pesquisadores, que tentam apoiar o planejamento da configuração através de modelos de otimização adequados (Boysen, Fliedner e Scholl, 2007).

O ALBP é dividido em duas classes: Problema de Balanceamento da Linha de Montagem Simples (SALBP em inglês, *Simply Assembly Line Balancing Problem*) e Problema de Balanceamento da Linha de Montagem Geral (GALBP em inglês, *General Assembly Line Balancing Problem*). O SALBP consiste na atribuição de um conjunto de tarefas a um conjunto de estações, com a finalidade de minimizar o número de estações ou o tempo de ciclo da linha. É conhecido como o problema mais estudado no campo do balanceamento da linha de montagem. Já o GALBP resolve as situações nas linhas de montagem de maneira mais realista considerando, por exemplo, as linhas de montagem com *layout* em U e com estações de trabalho paralelas.

Os SALBP ainda são classificados em quatro tipos quanto ao objetivo a ser otimizado (Scholl e Becker, 2006): o SALBP-1 objetiva minimizar o número de estações para um dado tempo de ciclo, o que é equivalente a minimizar o tempo ocioso total. O SALBP-2 visa minimizar o tempo de ciclo para um dado número de estações de trabalho. O SALBP-E envolve a maximização da eficiência da linha por meio da minimização do tempo de ciclo e do número de estações de trabalho simultaneamente. Já o SALBP-F é um problema de factibilidade. Dado o número de estações e o tempo de ciclo, o problema consiste em determinar se ele é factível.

Tipicamente, a literatura sobre o SALBP foca no problema num sentido puro como se, uma vez atribuídas as tarefas às estações, nenhuma definição a mais de parâmetros torna-se necessária (Andrés, Miralles e Pastor, 2008; Martino e Pastor, 2010). Entretanto, a sequência na qual as tarefas são desenvolvidas dentro da estação importa, uma vez que o tempo de *setup* exerce uma influência considerável sobre o tempo de ciclo da linha (Seyed-Alagheband, Ghomi e Zandieh, 2011) e que o sequenciamento das tarefas em cada estação influencia no tempo de *setup*, pois, se a ordem da tarefa numa estação de trabalho muda, o tempo de *setup* e, subsequentemente, o tempo de operação geral da estação se alteram, afetando também o tempo de ciclo da linha.

O Problema de Balanceamento e Sequenciamento da Linha de Montagem com tempos de *Setup* dependentes da sequência das tarefas (SUALBSP em inglês, *Setup Assembly Line Balancing and Scheduling Problem*) explora justamente essa situação, demonstrando que a linha de montagem não exige somente balanceamento, mas também o sequenciamento das tarefas atribuídas a cada estação em virtude da existência de tempos de *setup* dependentes da sequência das tarefas, culminando na resolução simultânea do problema de balanceamento e do problema de sequenciamento.

Em termos de complexidade computacional, o SUALBSP é classificado como *NP-hard* (Scholl et al., 2013; Andrés et al., 2008; Martino e Pastor, 2010; Seyed-Alagheband et al., 2011), o que implica uma exigência de grande quantidade de recursos computacionais. Além disso, existem

poucos estudos que tratam e apresentam soluções para o problema (Andrés et al., 2008; Martino e Pastor, 2010; Scholl et al., 2013). Pesquisas realizadas no Web of Science (2015) encontraram apenas cinco trabalhos que abordaram o SUALBSP.

Somente em 2008, Andrés et al. (2008) trataram do problema ao formularem um modelo matemático de Programação Linear Inteira (PLI) e proporem regras de prioridades e procedimentos GRASP para sua resolução. Em 2013, Scholl et al. (2013) modificaram o SUALBSP ao modelar os tempos de *setups* mais realisticamente, formulando um modelo matemático de programação linear binária mista mais compacto e menor que o modelo de Andrés et al. (2008). Scholl et al. (2013) propuseram uma solução heurística com excelentes resultados, porém relataram que o uso de um *MIP solver* (FICO© XPress Optimizer 20.00.05) apresentou desempenho decepcionante, pois contém um subproblema *NP-hard* em todas as estações. Neste sentido, este trabalho apresenta uma alternativa ao modelo matemático de PLI proposto por Scholl et al. (2013), desenvolvendo um novo modelo e uma matheurística para resolvê-lo. Ambos os modelos foram implementados no *MIP solver* IBM© ILOG CPLEX Optimization Studio V12.4 e comparados em um mesmo ambiente computacional, a fim de verificar a eficiência da abordagem proposta.

2 Revisão de literatura

O primeiro trabalho que tratou do SUALBSP foi o de Andrés et al. (2008). O problema apresentado por eles adicionou considerações do tempo de *setup* dependente da sequência ao clássico SALBP da seguinte maneira: sempre que uma tarefa j é atribuída ao lado de outra tarefa i na mesma estação de trabalho, um tempo de *setup* τ_{ij} deve ser adicionado para computar o tempo global da estação de trabalho, fornecendo assim a sequência de tarefas dentro de cada estação de trabalho. Da mesma forma, se a tarefa p é a última atribuída à estação de trabalho em que a tarefa i foi a primeira tarefa atribuída, então um tempo de *setup* τ_{pi} deve ser considerado também. Logo, no SUALBSP ambas as questões de balanceamento entre as estações e o sequenciamento das tarefas intraestação são resolvidas simultaneamente. O modelo foi implementado no *MIP solver* IBM© ILOG CPLEX V9 e testado em 160 instâncias com diferentes combinações de tamanho, ordem e variabilidade de *setups*. Para resolução do problema, Andrés et al. (2008) desenvolveram oito regras heurísticas simples e um método heurístico GRASP específico. As oito regras heurísticas testadas serviam para atribuir as tarefas às estações de trabalho. Já o método GRASP foi utilizado na sequenciação das tarefas. Das 160 instâncias testadas, 143 soluções não foram factíveis, 5 soluções factíveis subótimas e 12 ótimas.

Martino e Pastor (2010), por sua vez, propuseram procedimentos heurísticos baseados em regras de prioridade para resolver o problema, apresentando melhorias em relação aos procedimentos heurísticos elaborados por Andrés et al. (2008), uma vez que os procedimentos deles forneciam soluções ótimas apenas para instâncias muito pequenas. Os resultados da pesquisa de Martino e Pastor (2010) demonstraram que alguns dos procedimentos heurísticos baseados nas regras de prioridade propostas por eles apresentaram melhor desempenho que a meta-heurística GRASP e as oito regras heurísticas elaboradas por Andrés et al. (2008).

Já Seyed-Alagheband et al. (2011) adaptaram o modelo matemático de Andrés et al. (2008) e desenvolveram um algoritmo baseado na meta-heurística *Simulated Annealing* (SA) para resolver o problema. Os autores efetuaram várias modificações no clássico SA com a finalidade de trabalhar com instâncias de grandes dimensões e melhorar a estrutura de pesquisa da vizinhança do algoritmo.

Yolmeh e Kianfar (2012) propuseram um algoritmo genético para resolver o SUALBSP, utilizando o modelo matemático elaborado por Scholl et al. (2013). Os autores usaram uma permutação simples para determinar a sequência das tarefas. Para determinar a atribuição das tarefas às estações, o algoritmo genético foi hibridizado usando um procedimento de programação dinâmica.

Finalmente, Scholl et al. (2013) modificaram o SUALBSP ao modelar os tempos de *setups* mais realisticamente ao considerar os *setups forward* e *backward*, dando uma nova e mais compacta

formulação ao modelo matemático. O novo modelo foi formulado como um modelo linear binário misto e contém $2.n^2 + n.\overline{m}$ variáveis binárias e n variáveis contínuas. Devido a $\overline{m} \leq n$, o número de variáveis binárias é limitado por $O(n^2)$. Além disso, o modelo contém $7.n^2 + n.\overline{m} + 5.n + 1$ restrições, limitado por $O(n^2)$. Por outro lado, o modelo de Andrés et al. (2008) contém $2.n^2.\overline{m} + n^2 + \overline{m}$ variáveis binárias e $2.n^3.\overline{m} + n^2.(\overline{m}+1) + 2.(n+\overline{m})$, ou seja, $O(n^4)$ restrições, o que comprova que o modelo de Scholl et al. (2013) é muito mais compacto e menor que o modelo de Andrés et al. (2008). Em relação aos métodos de solução elaborados para o novo modelo, eles propuseram uma nova composição heurística que faz uso de algumas contribuições iniciais do SUALBSP e também de novas ideias. Dentre as novas ideias do novo procedimento estão as direções de planejamento diferentes, o *fathoming*, o *grouping graph*, a reotimização, e dentre as contribuições iniciais constam os procedimentos baseados em regras de prioridade com esquema de programação orientado à estação de Martino e Pastor (2010), o método GRASP desenvolvido por Andrés et al. (2008) e o procedimento Avalanche desenvolvido por Boysen e Fliedner (2008).

3 O problema

O SUALBSP, de acordo com as modificações empreendidas por Scholl et al. (2013), ao distinguir os tempos de *setup* em *forward* e *backward*, pode ser assim definido: dado um tempo de ciclo c, um grafo de precedência e os seus respectivos tempos de *setup backward* e *forward*, deve-se obter o desenho de uma linha de montagem que apresente o menor número de estações de trabalho possível, sendo que cada tarefa é atribuída a exatamente uma estação de trabalho e sequenciada nela, e as restrições de precedência e o tempo de ciclo não são violados.

Segundo Scholl et al. (2013), o termo *setup forward* refere-se a uma situação onde a tarefa j é executada diretamente depois da tarefa i no mesmo ciclo, por exemplo, no mesmo produto, observando um tempo de *setup* (*forward*) $\tau_{ij} \geq 0$. Já um *setup backward* ocorre se a tarefa i é a última executada no produto de um ciclo p e o trabalhador tem que se mover ao próximo produto, o qual tem que ser montado no ciclo $p+1$. Essa transferência causa um tempo de *setup* (*backward*) $\mu_{ij} \geq 0$ que deve ser finalizado até o final do ciclo p para iniciar a execução da tarefa j apenas quando o ciclo $p+1$ inicia. Uma vez que as estações de trabalho são supostas independentes e exclusivamente operadas por um único time de trabalhadores, os *setups forward* e *backward* são somente considerados entre tarefas na mesma estação, não entre estações adjacentes.

A Figura 1, apresentada por Scholl et al. (2013), exemplifica como os tempos de *setup backward* e *forward* estão presentes na execução das tarefas numa estação de trabalho. Pode-se observar que há uma estação de trabalho com carga ordenada i, j e h. As tarefas i, j e h são executadas de maneira cíclica em produtos consecutivos. Cada ciclo p inicia no tempo 0 com a execução da tarefa i que consome o tempo t_i. Depois, por exemplo, o operador tem que caminhar até uma caixa (1) em que deve buscar um material necessário para a tarefa j (2). Partindo desta caixa, ele tem de se mover para a posição de montagem da tarefa j (3). Somar esses tempos resulta num tempo de setup τ_{ij} para mudar da tarefa i para a j que inicia no tempo $t_i + \tau_{ij}$ e acaba na unidade de tempo t_j. Em seguida, o operador se move para a posição de montagem da tarefa h e busca uma ferramenta que toma o tempo τ_{ij}. Depois de executar h em t_h unidades de tempo, ele tem de ir para o produto seguinte no ciclo $p+1$. Isto leva um tempo de *setup backward* μ_{hi} e o ciclo atual é finalizado. Ao considerar uma outra carga ordenada $<j, h, i]$, observa-se um *setup forward* de h a i com tempo τ_{hi} (tracejado), que é diferente de μ_{hi}. Essa distinção considerada por Scholl et al. (2013) tornou o problema mais aplicável a casos reais.

Para compreender o modelo matemático formulado por Scholl et al. (2013), apresenta-se a notação já introduzida e definem-se os conjuntos, parâmetros e variáveis adicionais:

- V : conjunto de tarefas
- E : conjunto de pares de tarefas (i, j) com precedência direta

t_i : tempo de execução da tarefa $i \in V$

τ_{ij} : tempo de *setup forward* da tarefa i a j

μ_{ij} : tempo de *setup backward* da tarefa i a j

$P_i(F_i)$: conjunto de predecessores (sucessores) diretos da tarefa $i \in V$

K : conjunto de possíveis estações de trabalho, onde $K = \{1, 2, ..., \overline{m}\}$

FS_i : conjunto de estações nas quais a tarefa $i \in V$é factível de atribuição,
: onde $FS_i \subset K$

$F_i^{\tau}(P_i^{\tau})$: conjunto de tarefas que podem diretamente suceder (preceder) a tarefa $i \in V$ na carga da estação na direção *forward*

$F_i^{\mu}(P_i^{\mu})$: conjunto de tarefas que podem diretamente suceder (preceder) a tarefa $i \in V$ na carga da estação na direção *backward*

$M(\varepsilon)$: número suficientemente grande (pequeno)

x_{ik} : variável binária com valor 1, caso a tarefa $i \in V$ é atribuída à estação $k \in FS_i$; 0, caso contrário

y_{ij} : variável binária com valor 1, caso a tarefa $i \in V$ é predecessora direta de $j \in F_i^{\tau}$ na carga da estação; 0, caso contrário

w_{ij} : variável binária com valor 1, caso a tarefa $i \in V$ é a última e $j \in F_i^{\mu}$ é a primeira tarefa na carga da estação; 0, caso contrário

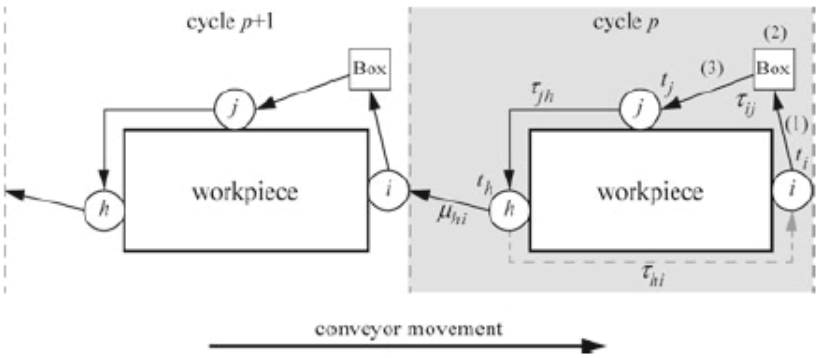
z_i : variável contínua para codificação do número da estação à qual a tarefa $i \in V$ é atribuída

f_i : variável contínua de tempo de início da tarefa $i \in V$(relativo ao tempo de lançamento da primeira estação)

T_k : variável contínua de tempo da estação $k \in K$

c : tempo de ciclo

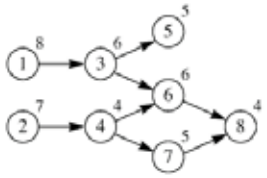
Figura 1: Ciclos consecutivos e suas conexões numa única estação



O problema e algumas das notações definidas são ilustradas por Scholl et al. (2013) por meio de um exemplo, com o tempo de ciclo definido para $c = 20$ e o gráfico de precedência, bem como os tempos de *setup forward* e *backward* apresentados na Figura 2. Os pesos dos nós denotam os tempos das tarefas. As matrizes de tempo de *setup forward* (τ) e *backward* (μ) representam os respectivos tempos do nó da linha em direção ao nó da coluna. Os valores omitidos na matriz τ

indicam que o *setup forward* somente pode representar a passagem de uma tarefa i para outra j, caso a tarefa j não esteja relacionada à i por precedência.

Figura 2: Dados do problema exemplo (tempo de ciclo $c = 20$)



τ	1	2	3	4	5	6	7	8
1	-	1	2	1	-	-	-	-
2	1	-	3	2	1	-	-	-
3	-	3	-	1	2	3	1	-
4	1	-	1	-	1	2	0	-
5	-	1	-	1	-	1	1	2
6	-	-	-	-	1	-	2	3
7	-	-	-	-	1	2	-	1
8	-	-	-	-	2	-	-	-

μ	1	2	3	4	5	6	7	8
1	4	5	-	3	-	-	-	-
2	3	4	1	-	3	-	-	-
3	6	7	4	5	-	-	5	-
4	5	6	3	4	5	-	-	-
5	4	5	2	3	4	5	3	2
6	3	4	1	2	3	4	2	-
7	5	6	3	4	5	6	4	-
8	6	7	4	5	6	7	5	4

Dada a notação, apresenta-se o modelo matemático completo:

$$\textbf{minimizar} \quad m(x,y,z) = z_n + \varepsilon.\sum_{k\in K}(T_k) \quad \text{sujeito a} \tag{1}$$

$$\sum_{k\in FS_i} x_{ik} = 1 \qquad \forall i \in V \tag{2}$$

$$z_i = \sum_{k\in FS_i} k.x_{ik} \qquad \forall i \in V \tag{3}$$

$$\sum_{j\in F_i^\tau} y_{ij} + \sum_{j\in F_i^\mu} w_{ij} = 1 \qquad \forall i \in V \tag{4}$$

$$\sum_{i\in P_j^\tau} y_{ij} + \sum_{i\in P_j^\mu} w_{ij} = 1 \qquad \forall j \in V \tag{5}$$

$$\sum_{i\in V}\sum_{j\in F_i^\mu} w_{ij} = z_n \tag{6}$$

$$z_i + M.(1-y_{ij}) \geq Z_j \qquad \forall i \in V, j \in F_i^\tau \tag{7}$$

$$z_j + M.(1-y_{ij}) \geq Z_i \qquad \forall i \in V, j \in F_i^\tau \tag{8}$$

$$z_i + M.(1-w_{ij}) \geq Z_j \qquad \forall i \in V, j \in F_i^\mu \tag{9}$$

$$z_j + M.(1-w_{ij}) \geq Z_i \qquad \forall i \in V, j \in F_i^\mu \tag{10}$$

$$f_j \geq f_i + t_i \qquad \forall (i,j) \in E \tag{11}$$

$$f_j \geq f_i + t_i + \tau_{ij} + M.(y_{ij}-1) \qquad \forall i \in V, j \in F_i^\tau \tag{12}$$

$$f_i \geq c.(z_i - 1) \qquad \forall i \in V \tag{13}$$

$$f_i + t_i + \mu_{ij} \leq c.z_i + M.(1-w_{ij}) \qquad \forall i \in V, j \in F_i^\mu \tag{14}$$

$$T_k + c.(z_i - 1) \geq f_i + t_i + \sum_{j\in F_i^\mu} \mu_{ij}.w_{ij} - M.(1-x_{ik}) \qquad \forall i \in V, k \in FS_i \tag{15}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, k \in FS_i \tag{16}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in V, j \in F_i^\tau \tag{17}$$

$$w_{ij} \in \{0,1\} \qquad \forall i \in V, j \in F_i^\mu \tag{18}$$

$$f_i, z_i, T_k \geq 0 \qquad \forall i \in V, k \in K \tag{19}$$

O objetivo do modelo 1 é minimizar o índice da estação para o qual o único nó sorvedouro n é atribuído e, assim, minimizar o número total de estações de trabalho. Como um objetivo se-

cundário os valores das variáveis T_k também são minimizados visando à obtenção de tempos de estações com *setups* mínimos.

4 Abordagem matheurística

De acordo com Della Croce e Salassa (2014), as melhorias implementadas nos *MIP solvers* nos últimos anos fizeram-nos mais competitivos em relação às abordagens meta-heurísticas na busca de soluções subótimas. A ideia central das matheurísticas explora a força dos algoritmos meta-heurísticos e dos métodos exatos. Até o presente momento, esta abordagem híbrida não tem uma classificação única, nem um quadro consolidado de trabalhos na área, sendo difícil estabelecer uma pura e nítida definição desses métodos. Uma característica distintiva é a exploração de ferramentas de programação matemática não triviais, como parte do processo de solução (Della Croce e Salassa, 2014; Raidl e Puchinger, 2008; Della Croce, Grosso e Salassa, 2014).

Neste mesmo sentido, para solucionar o SUALBSP, propõe-se uma heurística combinada com o uso de um *MIP solver*. Enquanto o modelo matemático de Scholl et al. (2013) minimiza o número de estações de trabalho, o modelo proposto parte da premissa de que o número de estações é um dado definido. Assim, a partir de uma estimativa inicial de número de estações (*lower bound*), processa-se o modelo com o objetivo de distribuir as tarefas entre as estações pré-estabelecidas, minimizando o tempo total de estação (que é o segundo objetivo do modelo proposto por Scholl et al. (2013)). Caso este processamento resulte como infactível, incrementa-se o número de estações em uma unidade e reprocessa-se o modelo. Este processo iterativo será finalizado assim que o modelo apresentar uma solução factível. Este algoritmo foi chamado de *math*-SUALBSP.

Empregou-se a mesma notação proposta por Scholl et al. (2013) e apresentada na Seção 3 com um pequeno ajuste: nas variáveis de decisão y_{ij} e w_{ij} adicionou-se a dimensão k, correspondente ao número de estações de trabalho. Assim, o modelo proposto tem y_{ijk} e w_{ijk} como variáveis de decisão.

Algoritmo 1: *math*-SUALBSP

Entrada: tarefas, tempos (tarefa, *forward* e *backward*) e relações de precedência
Saída: vetores de solução x_{ik}, y_{ijk} e w_{ijk}
$k \leftarrow$ Estimativa inicial de número de estações (cálculo de *lower bound*)
$status \leftarrow$ "Infactível"
enquanto $status =$ *"Infactível"* **faça**
  executa o modelo SUALBSP-with-fixed-stations
  se *modelo é factível* **então**
    $status \leftarrow$ "Factível"
  senão
    $status \leftarrow$ "Infactível"
    $k \leftarrow k+1$
  fim se
fim enqto
retorna vetores de solução

Para estimar o número inicial de estações utilizou-se o mesmo cálculo de *lower bound* proposto por Andrés et al. (2008), que consiste em tomar o somatório de tempo de todas as tarefas e dividir pelo tempo de ciclo:

$$k = \left\lceil \frac{t_{sum}}{c} \right\rceil \quad \text{onde} \quad t_{sum} = \sum_{j \in V} t_j \tag{20}$$

O modelo matemático "SUALBSP-with-fixed-stations" é apresentado a seguir:

$$\textbf{minimizar} \quad \sum_{k\in K} T_k \quad \text{sujeito a} \tag{21}$$

$$\sum_{k\in K} x_{ik} = 1 \qquad \forall i \in V \tag{22}$$

$$z_i = \sum_{k\in K} k \times x_{ik} \qquad \forall i \in V \tag{23}$$

$$x_{ik} \geq \sum_{j\in V} y_{ijk} \quad \text{e} \quad x_{ik} \geq \sum_{j\in V} y_{jik} \qquad \forall k \in K, i \in V \tag{24}$$

$$x_{ik} \geq \sum_{j\in V} w_{ijk} \quad \text{e} \quad x_{ik} \geq \sum_{j\in V} w_{jik} \qquad \forall k \in K, i \in V \tag{25}$$

$$\sum_{k\in K}\sum_{j\in F_i^\tau} y_{ijk} + \sum_{k\in K}\sum_{j\in F_i^\mu} w_{ijk} = 1 \qquad \forall i \in V \tag{26}$$

$$\sum_{k\in K}\sum_{i\in P_i^\tau} y_{ijk} + \sum_{k\in K}\sum_{i\in P_i^\mu} w_{ijk} = 1 \qquad \forall j \in V \tag{27}$$

$$\sum_{i\in V}\sum_{j\in V} w_{ijk} = 1 \qquad \forall k \in K \tag{28}$$

$$\sum_{k\in K} y_{ijk} \leq 1 \qquad \forall i \in V, j \in V \tag{29}$$

$$\sum_{k\in K}\sum_{i\in V} y_{iik} = 0 \tag{30}$$

$$f_j \geq f_i + t_i \qquad \forall (i,j) \in E \tag{31}$$

$$f_j \geq f_i + t_i + \tau_{ij} + M.(y_{ijk} - 1) \qquad \forall k \in K, i \in V, j \in F_i^\tau \tag{32}$$

$$f_i \geq c.(z_i - 1) \qquad \forall i \in V \tag{33}$$

$$f_i + t_i + \mu_{ij} \leq c.z_i + M.(1 - w_{ijk}) \qquad \forall k \in K, i \in V, j \in F_i^\mu \tag{34}$$

$$T_k + c.(z_i - 1) \geq f_i + t_i + \sum_{j\in F_i^\mu} \mu_{ij}.w_{ijk} - M.(1 - x_{ik}) \qquad \forall i \in V, k \in FS_i \tag{35}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, k \in K \tag{36}$$

$$y_{ijk} \in \{0,1\} \qquad \forall i \in V, j \in F_i^\tau, k \in K \tag{37}$$

$$w_{ijk} \in \{0,1\} \qquad \forall i \in V, j \in F_i^\mu, k \in K \tag{38}$$

$$f_i, z_i, T_k \geq 0 \qquad \forall i \in V, k \in K \tag{39}$$

Ao incorporar a dimensão k, correspondente às estações de trabalho K, adicionaram-se restrições adicionais ao modelo original de Scholl et al. (2013). A restrição (28) garante que uma estação $k \in K$ esteja associada, obrigatoriamente, a uma sequência de tarefas ($i \in V$, $j \in V$). Já a restrição (29) limita a sequência de tarefas $i \in V$ para $j \in V$ a no máximo uma estação. Quando uma estação $k \in K$ é carregada com apenas uma tarefa $i \in V$, tem-se $w_{iik} = 1$. Deste modo, a variável y_{iik} deve ser igual a zero, o que é garantido pela restrição (30).

5 Experimentos computacionais

O modelo matemático de Scholl et al. (2013) e a abordagem proposta foram implementados em C++ a partir do ambiente de programação do *Microsoft© Visual Studio 2010 Professional* combinado com o *MIP solver* IBM© ILOG CPLEX Optimization Studio V12.5.1 com as configurações *default*. O equipamento utilizado foi um notebook com processador Intel© Core™ i7-3520M CPU 2.9GHz, 6GB de memória RAM e sistema operacional *Microsoft© Windows 8.1 Pro* 64 bits.

O experimento tomou como base algumas instâncias de dados SBF-1 disponíveis no *website* do *Assembly Line Balancing Research Group* (`http://alb.mansci.de/`). Como o objetivo

deste estudo é apresentar uma alternativa ao modelo matemático de PLI proposto por Scholl et al. (2013), tomou-se um conjunto de instâncias possível de se instanciar e resolver no ambiente computacional disponível para esta pesquisa. A seleção destas instâncias e os resultados computacionais estão disponíveis em `https://dl.dropbox.com/u/84981398/mathSUALBSP.zip`.

As notações e medidas utilizadas para avaliar os resultados são as seguintes:

ID	conjunto de instâncias com a mesma quantidade de variáveis y_{ijk}
#var	número máximo de variáveis de y_{ijk}, em que as dimensões de $i \in V$ e $j \in V$ são dadas pela quantidade de tarefas de cada instância e $k \in K$ é dada pela estimativa de *upper bound* calculada por Scholl et al. (2013)
#inst	quantidade de instâncias de dados de cada conjunto de instâncias ID
LB	média de *lower bound* de estações encontradas no conjunto de instâncias ID
PLI	resultados do processamento do modelo de Programação Linear Inteira de Scholl et al. (2013)
MATH	resultados do processamento modelo matheurístico proposto por este estudo
time	média dos tempos (em segundos) de processamento do conjunto de instâncias ID
k	média de estações encontradas no conjunto de instâncias ID
#opt	número de instâncias que atingiram a otimalidade
%time	ganho percentual de tempo no processamento do modelo MATH em relação ao modelo PLI
#best	número de instâncias que encontraram um número menor de estações no modelo MATH em relação ao modelo PLI

Os modelos PLI e MATH foram processados no *MIP solver* limitados a 100 segundos, seguindo o exemplo do trabalho de Scholl et al. (2013). A Tabela 1 apresenta as 101 instâncias selecionadas para este experimento, devidamente agrupadas em 21 conjuntos de instâncias de acordo com o número máximo de variáveis y_{ijk} (#var).

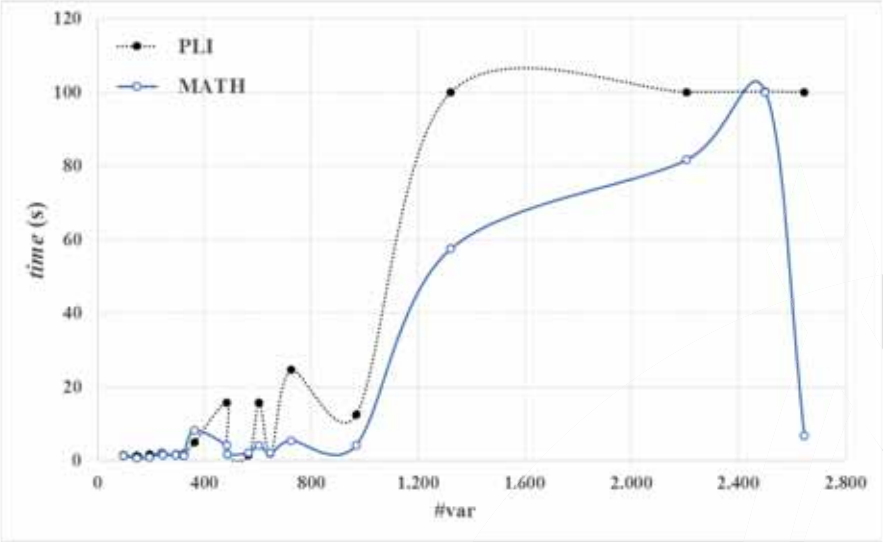
Tabela 1: Conjunto de instâncias e seus resultados

				PLI			MATH				
ID	#var	#inst	LB	*time*	k	#opt	*time*	k	#opt	*%time*	#best
1	98	6	2,0	1,496	2,0	6	1,400	2,0	6	6,41	0
2	147	4	2,5	1,444	2,5	4	0,762	2,5	4	47,24	0
3	196	2	3,0	1,773	3,0	2	0,868	3,0	2	51,07	0
4	242	4	2,0	1,857	2,0	4	2,174	2,0	4	-17,05	0
5	243	4	3,0	1,881	3,0	4	1,874	3,0	4	0,39	0
6	245	8	4,5	1,843	5,0	8	1,524	5,0	8	17,31	0
7	294	4	5,0	1,798	6,0	4	1,604	6,0	4	10,80	0
8	320	4	4,0	1,780	5,0	4	1,539	5,0	4	13,54	0
9	324	4	4,0	1,822	4,0	4	1,401	4,0	4	23,12	0
10	363	7	3,0	5,077	3,0	7	8,217	3,0	7	-61,87	0
11	484	9	3,9	15,742	3,9	9	4,169	3,9	9	73,52	0
12	486	4	5,0	1,735	6,0	4	1,731	6,0	4	0,19	0
13	567	4	6,0	1,605	7,0	4	2,205	7,0	4	-37,40	0
14	605	8	4,5	15,752	4,5	8	4,149	4,5	8	73,66	0
15	648	4	7,0	1,891	8,0	4	2,118	8,0	4	-12,00	0
16	726	4	6,0	24,607	6,0	4	5,508	6,0	4	77,62	0
17	968	4	7,0	12,535	8,0	4	4,203	8,0	4	66,47	0
18	1.323	8	3,0	100,095	3,4	0	57,531	3,0	4	42,52	3
19	2.205	5	5,0	100,114	5,2	0	81,694	5,0	1	18,40	1
20	2.500	1	4,0	100,215	5,0	0	100,091	4,0	0	0,12	1
21	2.646	3	5,0	100,094	6,0	0	6,761	5,0	3	93,25	3
Totalizações:		101		2.244,145		84	1.240,120		92	44,74	8

Considerando o tempo total de processamento de todas as instâncias analisadas, observa-se que o modelo MATH foi 44,74% mais rápido que o modelo PLI. Enquanto o modelo PLI atingiu a otimalidade em 84 instâncias (83,17%), o modelo MATH atingiu em 92 (91,09%). Esta otimalidade não se deu apenas em um menor tempo de estação T_k, que é o objetivo secundário, mas resultou em um número de estações k menor. Logo, além do ganho de desempenho computacional, o modelo MATH apresentou resultados mais satisfatórios, encontrando soluções que demandam um número inferior de estações de trabalho se comparado o modelo PLI.

A Figura 3 ilustra graficamente o desempenho computacional dos dois modelos. Claramente, o modelo proposto neste artigo (MATH) atinge resultados iguais ou melhores em um tempo de processamento inferior ao apresentado pelo modelo PLI de Scholl et al. (2013).

Figura 3: Gráfico de confronto de desempenho dos modelos PLI e MATH



À medida que se analisam instâncias com maior número de variáveis de decisão (#*var*), a diferença de desempenho computacional entre os modelos fica mais nítida. Dentre as maiores instâncias (#$var \geq 700$), evidenciadas pelo aumento da distância entre as curvas, uma única instância (#$var = 2.500$) obteve desempenho praticamente idêntico para os dois modelos. Ocorre que o limite computacional de 100 segundos sugerido por Scholl et al. (2013) foi atingido por ambos, razão pela qual nenhum deles atingiu a otimalidade.

6 Discussões, limitações e conclusões

Neste artigo foi desenvolvido um modelo matemático alternativo ao proposto por Scholl et al. (2013). Os autores relataram que os resultados obtidos, a partir do processamento do modelo em um *MIP solver*, haviam sido decepcionantes. Assim, a ideia central da proposta deste artigo foi retirar do modelo matemático de Scholl et al. (2013) a responsabilidade de determinar o número ótimo de estações de trabalho, presumindo-se que este número fora previamente conhecido. Assim, por meio de um cálculo de *lower bound*, processou-se o modelo, fixando-se o número de estações.

Havendo infactibilidade, incrementa-se em uma unidade o *lower bound* e reprocessa-se o modelo. Assim, em cada interação desta heurística, utilizou-se o CPLEX, caracterizando este algoritmo como sendo matheurístico.

Os trabalhos anteriores sugerem o uso de outros métodos para pesquisas futuras para este problema, que desafia, fundamentalmente, os responsáveis por complexas linhas industriais de

montagem. Qualquer tipo de ganho, principalmente na qualidade do balanceamento e sequenciamento da linha de montagem, pode representar uma expressiva economia que impactará em ganhos de produtividade e competitividade.

Por esta razão, este estudo centrou seus esforços em atingir resultados ótimos. As heurísticas e meta-heurísticas apresentadas em trabalhos anteriores apresentaram bons resultados e excelentes desempenhos. Mas, nos últimos anos, em virtude das melhorias implementadas nos *MIP solvers* disponíveis no mercado (Della Croce e Salassa, 2014), abriu-se oportunidade para explorar a potencialidade dos métodos exatos combinados, ou não, com heurísticas.

Este trabalho mostra que há espaço para que se evolua dentro deste campo metodológico. Estudos futuros devem necessariamente, no entendimento dos autores desta pesquisa, propor modelos ou métodos que sejam capazes de tratar instâncias maiores de dados.

Referências

Andrés, C., Miralles, C. e Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, *187*(3), 1212–1223. doi: http://dx.doi.org/10.1016/j.ejor.2006.07.044

Boysen, N. e Fliedner, M. (2008). A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, *184*, 39–56. doi: 10.1016/j.ejor.2006.11.006

Boysen, N., Fliedner, M. e Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, *183*(2), 674-693.

Della Croce, F., Grosso, A. e Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, *213*(1), 67–78. doi: 10.1007/s10479-011-0928-x

Della Croce, F. e Salassa, F. (2014). A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, *218*(1), 185-199. doi: 10.1007/s10479-012-1235-x

Erel, E. e Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, *9*, 414–434. doi: 10.1080/095372898233902

Martino, L. e Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, *48*, 1787–1804. doi: 10.1080/00207540802577979

Raidl, G. e Puchinger, J. (2008). Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In C. Blum, M. Aguilera, A. Roli e M. Sampels (Eds.), *Hybrid metaheuristics* (Vol. 114, p. 31-62). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-78295-7_2

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. doi: http://dx.doi.org/10.1016/j.ejor.2004.07.022

Scholl, A., Boysen, N. e Fliedner, M. (2013). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*, *35*(1), 291–320. doi: 10.1007/s00291-011-0265-0

Seyed-Alagheband, S. A., Ghomi, S. M. T. F. e Zandieh, M. (2011). A simulated annealing algorithm for balancing the assembly line type ii problem with sequence-dependent setup times between tasks. *International Journal of Production Research*, *49*(3), 805–825. doi: 10.1080/00207540903471486

Web of Science. (2015). *Web of Science.* Disponível em `https://apps.webofknowledge.com`

Yolmeh, A. e Kianfar, F. (2012). An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Comput. Ind. Eng.*, *62*(4), 936–945. doi: 10.1016/j.cie.2011.12.017