UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO

Karen Juliana Weigner de Bastos

Uma abordagem Matheurística para o Problema de Sequenciamento de tarefas e Balanceamento de Linhas de Montagem de modelo único com Tempos de *Setup* dependentes da Sequência

Porto Alegre
2015

Karen Juliana Weigner de Bastos

Uma abordagem Matheurística para o Problema de Sequenciamento de tarefas e Balanceamento de Linhas de Montagem de modelo único com Tempos de *Setup* dependentes da Sequência

Dissertação submetida ao Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul, como requisito parcial para obtenção do título de Mestre em Administração.

Orientadora: Profª. Dra. Denise Lindstrom Bandeira

Porto Alegre

2015

CIP - Catalogação na Publicação

Bastos, Karen Juliana Weigner de
Uma abordagem Matheurística para o Problema de Sequenciamento de tarefas e Balanceamento de Linhas de Montagem de modelo único com Tempos de Setup dependentes da Sequência / Karen Juliana Weigner de Bastos. -- 2015.
91 f.

Orientadora: Denise Lindstrom Bandeira.

Dissertação (Mestrado) -- Universidade Federal do Rio Grande do Sul, Escola de Administração, Programa de Pós-Graduação em Administração, Porto Alegre, BR-RS, 2015.

1. Balanceamento de linha de montagem. 2. Sequenciamento de linha de montagem. 3. Tempo de setup. 4. Matheurística. 5. MIP solver. I. Bandeira, Denise Lindstrom, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Karen Juliana Weigner de Bastos

Uma abordagem Matheurística para o Problema de Sequenciamento de tarefas e Balanceamento de Linhas de Montagem de modelo único com Tempos de *Setup* dependentes da Sequência

Dissertação submetida ao Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul, como requisito parcial para obtenção do título de Mestre em Administração.

Aprovada em: 09 de julho de 2015.

BANCA EXAMINADORA:

_____

Professor Doutor Michel José Anzanello – PPGEP/UFRGS

_____

Professor Doutor Denis Borenstein – PPGA/UFRGS

_____

Professor Doutor João Luiz Becker – PPGA/UFRGS

Porto Alegre

2015

AGRADECIMENTOS

- A Deus pela proteção, pelas bênçãos derramadas e pela presença constante.

- Aos meus pais por ensinarem a valorizar o que realmente importa na vida, por acreditarem em mim e sempre fornecerem o suporte necessário para que eu lutasse pelos meus sonhos. Vocês são exemplos de vida!

- Ao meu noivo José Landim que sempre esteve ao meu lado ao longo desta caminhada, me incentivando e auxiliando para que conquistasse mais esse objetivo. Agradeço imensamente pelo seu carinho e cumplicidade.

- À minha família, principalmente, ao meu irmão, minha cunhada e minha sobrinha Ana Luiza, que nos dá trabalho, mas também muita felicidade.

- À Universidade Federal do Rio Grande do Sul por fornecer um ensino gratuito e de qualidade.

- À minha orientadora, Profª Denise Lindstrom Bandeira, pelos valiosos ensinamentos e por acreditar no meu potencial.

- Aos professores da banca avaliadora deste trabalho que são ótimos profissionais e incentivadores da busca pelo conhecimento.

- Ao Camilo Poulsen pela disponibilidade em sempre me auxiliar, pela boa vontade e por compartilhar seu conhecimento. Será um ótimo professor!

- Aos colegas de Mestrado, em especial, à Luciane e à Evelyn pelo companheirismo, pelas angústias compartilhadas e pela ajuda mútua.

- A CAPES pelo auxílio financeiro.

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*) 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 resolvedor de Programação Inteira Mista, 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 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. 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 linha de montagem, tempo de *setup*, matheurística, MIP *solver*.

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 Mixed-Integer Programming solver, has shown disappointing performance, because it contains an NP-hard sub problems in every station. While the model proposed by Scholl, Boysen and Fliedner (2013) primarily minimizes the numbers of stations, our model assumes it as a parameter. From an initial estimate of the number of stations, we process the model for allocating tasks and minimize 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. 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 and scheduling, setup time, matheuristic, MIP solver.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALB	Assembly Line Balancing (em português, Balanceamento da Linha de Montagem)
ALBP	Assembly Line Balancing Problem (em português, Problema de Balanceamento da Linha de Montagem)
CPMP	Capacitated P-median Problem (em português, Problema de P-median Capacitado)
DBCMND	Design-Balanced Capacitated Multicommodity Network Design Problem (em português, Problema de Desenho de Rede Multicommodity Capacitada Balanceada)
GALBP	General Assembly Line Balancing Problem (em português, Problema de Balanceamento da Linha de Montagem Geral)
GALBPS	General Assembly Line Balancing Problem with Setups (em português, Problema de Balanceamento da Linha de Montagem Geral com *Setups*)
GL	Giant Leap
GRASP	Greedy Randomized Adaptive Search Procedure
HRH	High-level Relay Hybrid (em português, Hibridização Relay de Alto nível)
HTH	High-level Teamwork Hybrid (em português, Hibridização Teamwork de Alto nível)
LRH	Low-level Relay Hybrid (em português, Hibridização Relay de Baixo nível)
LSNDP	Liner Shipping Network Design Problem (em português, Problema Linear de Desenho de Rede de Transporte Marítimo)
LTH	Low-level Teamwork Hybrid (em português, Hibridização Teamwork de Baixo nível)
MIP	Mixed-Integer Programming (em português, Programação Inteira Mista)
MLCLSP	Multi-level, Multi-item Capacitated Lot-Sizing Problem (em português, Problema de Dimensionamento de Lotes Capacitado Multi-produto e Multi-nível)
PLB	Programação Linear Binária
PLBM	Programação Linear Binária Mista
PLI	Programação Linear Inteira (em inglês, Integer Linear Programming)
PO	Pesquisa Operacional
SA	Simulated Annealing
SALBP	Simply Assembly Line Balancing Problem (em português, Problema de Balanceamento da Linha de Montagem Simples)
SBF1	Conjunto de dados n° 1 de Scholl, Boysen e Fliedner
SBF2	Conjunto de dados n° 2 de Scholl, Boysen e Fliedner
SH	Station oriented strategy Heuristic (em português, estratégia Heurística orientada à Estação)
SUALBSP	Setup Assembly Line Balancing and Scheduling Problem (em português, Problema de Sequenciamento e Balanceamento da Linha de Montagem com tempos de *Setup* dependentes da sequência)
TH	Task oriented strategy Heuristic (em português, estratégia Heurística orientada à Tarefa)

TQDP	Total Quantity Discount Problem (em português, Problema de Desconto de Quantidade Total)
TSP	Travelling Salesman Problem (em português, Problema do Caixeiro Viajante)
VNS	Variable Neighborhood Search (em português, Pesquisa na Vizinhança Variável)
VRP	Vechicle Routing Problem (em português, Problema de Roteamento de Veículos)

SUMÁRIO

1 INTRODUÇÃO

Originalmente, as linhas de montagem foram desenvolvidas visando à eficiência em custo na produção em massa de produtos padronizados, à exploração da alta especialização do trabalho e aos efeitos benéficos do aprendizado. Contudo, as exigências dos sistemas de produção têm mudado drasticamente (BOYSEN; FLIEDNER; SCHOLL, 2007), fazendo com que as linhas de montagem tenham importância até mesmo na produção de baixo volume dos produtos customizados.

Conforme Seyed-Alagheband, Ghomi e Zandieh (2011), uma linha de montagem consiste em uma sequência de estações de trabalho executando um conjunto finito de tarefas, cada uma com o seu respectivo tempo de processamento da operação e com um conjunto de relações de precedência, as quais especificam as ordens de permissão das tarefas. Becker e Scholl (2006) e Boysen, Fliedner e Scholl (2007) acrescentam que uma linha de montagem consiste de estações de trabalho $k = 1, \dots, m$ arranjadas ao longo de uma esteira transportadora ou um material mecânico similar. Os produtos são consecutivamente lançados para a linha e movidos de uma estação a outra estação. Em cada estação, certas operações são repetidamente executadas respeitando o tempo de ciclo, que é o tempo máximo para cada ciclo de trabalho.

Posto isso, o problema de decisão de otimamente particionar (balancear) o trabalho de montagem entre as estações com respeito a algum objetivo é conhecido como o Problema de Balanceamento da Linha de Montagem (ALBP, em inglês *Assembly Line Balancing Problem*) e esse balanceamento apresenta efeitos significativos sobre o desempenho e a produtividade dos sistemas de manufatura (HAMTA *et al*., 2013).

O objetivo do problema de balanceamento da linha de montagem é 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, sejam otimizadas (EREL; SARIN, 1998). É um problema que vem sendo estudado desde 1955 e que desde então está sendo abordado de diferentes maneiras com a finalidade de amenizar a lacuna existente entre a discussão acadêmica e as aplicações práticas, uma vez que muitas situações reais ainda estão inexploradas, além de apresentar uma vasta classificação baseada nas suas características e particularidades.

Considerando essa lacuna, o Problema de Sequenciamento e Balanceamento da Linha de Montagem com Tempos de *Setup* dependentes da sequência das tarefas (SUALBSP, em

inglês *Setup Assembly Line Balancing and Scheduling Problem*) vem sendo recentemente abordado na literatura, tornando o problema de balanceamento da linha de montagem mais real, uma vez que em muitos casos o tempo de *setup* exerce uma influência considerável sobre o tempo de ciclo da linha (SEYED-ALAGHEBAND; GHOMI; 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 mudará, afetando também o tempo de ciclo da linha.

Além disso, segundo Scholl, Boysen e Fliedner (2013), existem muitas situações práticas em que os tempos de *setups* incorridos são baseados na sequência de tarefas atribuídas aos operadores da linha: por exemplo, nas linhas de montagens automotivas o trabalho pode ser realizado em diferentes posições dentro e fora do corpo do carro tornando as distâncias de caminhada relevantes e dependentes das posições onde duas tarefas consecutivas são realizadas. Outro exemplo são as ferramentas exigidas para realizar as tarefas. Se tarefas consecutivas exigem diferentes ferramentas, é necessário mudá-las ocasionando tempo de *setup* ou quando os componentes dos produtos são colocados em contêineres distantes das linhas de montagem e as distâncias das caminhadas para buscar partes do material no contêiner são consideráveis (ANDRÉS; MIRALLES; PASTOR, 2008).

Em virtude de o SUALBSP ter sido estudado somente a partir de 2008 por Andrés, Miralles e Pastor (2008), a literatura apresenta escassas abordagens para resolver o problema, gerando oportunidades para o desenvolvimento de técnicas de solução para a sua resolução (SEYED-ALAGHEBAND; GHOMI; ZANDIEH, 2011). Além disso, as soluções heurísticas que foram propostas na literatura do problema apresentaram excelentes resultados, porém o uso de um MIP *solver* apresentou desempenho decepcionante, pois o SUALBSP apresenta um subproblema *NP-hard* em todas as estações. Sendo assim, o presente trabalho objetiva elaborar uma solução para o SUALBSP baseado na abordagem matheurística. Dessa forma, o resultado dessa dissertação pretende contribuir na resolução desse problema de grande relevância e no enriquecimento da literatura que trata do problema em questão.

A presente dissertação está organizada da seguinte forma: no Capítulo 1, são apresentados a justificativa e os objetivos deste estudo; no Capítulo 2, é feita uma revisão da literatura, apresentando definição de conceitos e assuntos relacionados com o SUALBSP, tais como linha de montagem, o problema de balanceamento da linha de montagem, a classificação dos ALBP e o conjunto de artigos que tratam sobre o SUALBSP; no Capítulo 3, são apresentados os procedimentos metodológicos empregados e uma explanação acerca das técnicas de solução do SUALBSP e da abordagem matheurística. No Capítulo 4, é definido o

problema abordado e é apresentado o modelo matheurístico proposto no presente trabalho. O Capítulo 5 aborda como foram selecionados os dados e os experimentos computacionais realizados no estudo; e, finalmente, no Capítulo 6, as considerações finais são feitas, apresentando-se as contribuições deste trabalho, suas limitações e sugestões de estudos futuros.

1.1 JUSTIFICATIVA

Apesar do tempo de *setup* ser muito relevante nos sistemas de montagem do mundo real, o Problema de Sequenciamento e Balanceamento da Linha de Montagem com o tempo de *Setup* dependente da sequência das tarefas (SUALBSP) tem sido considerado por poucos trabalhos (ANDRÉS; MIRALLES; PASTOR, 2008; MARTINO; PASTOR, 2010; SCHOLL; BOYSEN; FLIEDNER, 2013), e os procedimentos de solução do problema apenas começaram a ser estudados, conforme está explicitado no capítulo dedicado à revisão da literatura sobre o problema. Adicionalmente, as primeiras pesquisas consideraram o tempo de *setup* somente indiretamente via restrições de atribuição, por exemplo, Boysen, Fliedner e Scholl (2007) e Scholl, Fliedner e Boysen (2010). Apenas em 2008 Andrés, Miralles e Pastor (2008), trataram do problema ao formular uma programação linear binária e propor regras de prioridades e procedimentos GRASP para sua resolução.

Dessa forma, ainda há oportunidade para que o problema seja mais estudado. Além disso, ao considerar a relevância prática do problema percebe-se a necessidade do desenvolvimento de heurísticas e soluções rápidas e de fácil implementação desenvolvendo procedimentos hábeis para resolver até mesmo grandes instâncias do problema rapidamente enquanto alcança uma alta qualidade de solução (SCHOLL; BOYSEN; FLIEDNER, 2013). Com o desenvolvimento de soluções rápidas e de qualidade, os planejadores/gestores possuem à disposição ferramentas que os auxiliam a examinar as alternativas possíveis de sequenciamento e balanceamento da linha de montagem de forma mais acurada. De outra forma, os planejadores/gestores tendem a esperar que as soluções propostas sejam computadas quase instantaneamente, então os tempos computacionais são de crucial importância em relação à aceitabilidade de uma ferramenta de planejamento.

Sendo assim, o desenvolvimento de soluções que atendam a essa necessidade é de fundamental importância. Corrobora com essa proposta a recomendação de que uma

promissora direção para futuras pesquisas acerca do problema, segundo Scholl, Boysen e Fliedner (2013), Seyed-Alagheband, Ghomi e Zandieh (2011) e Andrés, Miralles e Pastor (2008), é a adaptação ou a criação de heurísticas e metaheurísticas para buscar um melhor *trade-off* entre qualidade da solução e tempo computacional.

Ao pesquisar os artigos disponíveis sobre o SUALBSP, encontrou-se apenas um trabalho que adota o modelo matemático estendido do problema desenvolvido por Scholl, Boysen e Fliedner (2013). Assim sendo, essa dissertação parece ser um dos primeiros trabalhos a adotá-lo utilizando uma técnica de solução ainda não empregada no SUALBSP para sua resolução, contribuindo, assim, com o desenvolvimento de problema tão relevante para a Pesquisa Operacional (PO) e para os planejadores/gestores.

1.2 OBJETIVOS

São objetivos deste trabalho:

a) Geral:

Propor uma abordagem alternativa para a resolução do SUALBSP, baseado em Scholl, Boysen e Fliedner (2013), que minimize o tempo de estação.

b) Específicos:

- Desenvolver um modelo matemático alternativo ao modelo proposto por Scholl, Boysen e Fliedner (2013).
- Selecionar uma técnica de solução para o SUALBSP.
- Comparar e avaliar os resultados obtidos oriundos da abordagem proposta com o modelo matemático de Programação Linear Inteira (PLI) de Scholl, Boysen e Fliedner (2013).

2 REVISÃO TEÓRICA

Este capítulo visa fundamentar teoricamente o presente trabalho, a partir de uma revisão da literatura científica sobre o tema proposto.

Primeiramente, na Seção 2.1 são apresentadas algumas definições dos termos empregados no problema objeto de estudo. A Seção 2.2 apresenta o conceito de linha de montagem e sua importância. Já o problema de balanceamento da linha de montagem é abordado na Seção 2.3. A Seção 2.4 traz a classificação do ALBP elaborada por Boysen, Fliedner e Scholl (2007). Os artigos que tratam do SUALBSP e as técnicas de solução empregadas por eles são descritas na Seção 2.5.

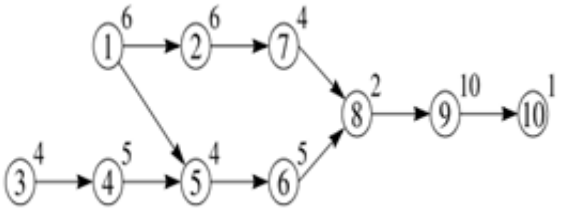
2.1 CONCEITOS

Ao longo desse trabalho, alguns conceitos são empregados, determinando a necessidade de defini-los devidamente com a finalidade de compreender o contexto em que estão inseridos.

Estação de trabalho: é uma localização física onde um conjunto particular de tarefas é executado (GAITHER; FRAZIER, 2005).

Grafo de precedência: consiste de nós que representam as tarefas do processo de produção, de um conjunto de arcos que representam as relações de precedência entre as tarefas e de pesos nos nós que são os tempos das tarefas (BOYSEN; FLIEDNER; SCHOLL, 2007). Na Figura 1, apresenta-se um grafo de precedência onde é possível visualizar que existem 10 tarefas (n = 10) e que os tempos das tarefas variam entre 1 e 10 unidades de tempo. Além disso, apresenta também as restrições de precedência, por exemplo, a tarefa 5 só pode ser iniciada depois que as tarefas 1 e 4 (predecessores diretos) e 3 (predecessor indireto) estejam completas. De outro modo, é possível interpretar que a tarefa 5 deve ser finalizada antes que seus sucessores (diretos e indiretos) 6, 8, 9 e 10 possam ser iniciados.

Figura 1 - Grafo de precedência



Fonte: Boysen, Fliedner e Scholl (2007, p. 676)

Linha de modelo único: um produto homogêneo é manufaturado na linha.

Linha de modelo misto: muitos modelos de uma família de produto são manufaturados simultaneamente.

Linha de multimodelos: muitos produtos são manufaturados em lotes separados.

Linha em forma de U: linhas que possuem a entrada e a saída no mesmo lugar (BATTAIA; DOLGUI, 2013).

Linha de montagem compassada: o tempo de estação de cada estação é limitado a um tempo de ciclo *c* como um valor máximo para cada produto.

Restrições de precedência: o conjunto de restrições que determina a sequência segundo a qual as tarefas podem ser executadas. As restrições de precedência são definidas por aspectos tecnológicos ou de concepção do produto (SIMARIA, 2001).

Tarefa: é uma unidade de trabalho produtiva e indivisível, ou seja, é um elemento de trabalho que acrescenta valor e que não pode ser subdividido sem que isso crie trabalho adicional (SIMARIA, 2001).

Tempo de ciclo: tempo máximo disponível para cada ciclo de trabalho (BECKER; SCHOLL, 2006).

Tempo ocioso: é a diferença entre o tempo de ciclo e o tempo da estação de trabalho (YOLMEH; KIANFAR, 2012).

Tempo de processamento da tarefa: o tempo requerido para executar uma dada tarefa (SIMARIA, 2001).

2.2 LINHA DE MONTAGEM

Originalmente, as linhas de montagem foram desenvolvidas visando à eficiência em custo na produção em massa de produtos padronizados, à exploração da alta especialização do

trabalho e aos efeitos benéficos do aprendizado; contudo, as exigências dos sistemas de produção têm mudado drasticamente (BOYSEN; FLIEDNER; SCHOLL, 2007). Adicionalmente, para responder às diversas necessidades dos clientes, as empresas permitem a individualização dos seus produtos e adotam máquinas e ferramentas que tornam as linhas de montagem importantes na montagem de produtos de baixo volume, além de possibilitar o desenvolvimento da estratégia de produção de customização em massa.

Segundo Simaria (2001) uma linha de montagem é um conjunto de postos de trabalho dispostos sequencialmente e interligados por um sistema de transporte de materiais. Em cada posto de trabalho é executado um conjunto de tarefas (ou operações) pré-especificadas num processo de montagem, no qual são definidos o tempo de processamento da tarefa e as restrições de precedência. As restrições de precedência são representadas por meio de um grafo de precedência composto por nós numerados, que representam as tarefas, e por arcos, que mostram quais são as relações de precedência entre as tarefas.

A decisão sobre a instalação de uma linha de montagem é de longo prazo e requer 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 real através de modelos de otimização adequados. Esse planejamento inclui a definição da capacidade do sistema (tempo de ciclo, número de estações e equipamentos), bem como a atribuição do conteúdo do trabalho às unidades produtivas (atribuição de tarefas e sequência das operações) (BOYSEN; FLIEDNER; SCHOLL, 2007).

2.3 PROBLEMA DE BALANCEAMENTO DE LINHAS DE MONTAGEM

O problema de balanceamento da linha de montagem tem sido estudado há mais de 50 anos e diferentes enfoques com diversas considerações têm sido investigadas durante esse período (HAMTA *et al.*, 2013). A proposta do problema de balanceamento de linha de montagem é encontrar um balanceamento de linha factível (BECKER; SCHOLL, 2006) e para isso atribui as tarefas às estações de trabalho de tal maneira que as relações de precedência e outras restrições não sejam violadas e que alguma medida de efetividade seja otimizada (EREL; SARIN, 1998).

A primeira formulação conhecida de um problema de balanceamento de linha de montagem foi feita por Salveson (1955 *apud* DOLGUI; BATTAIA, 2013). Desde então,

diferentes abordagens foram propostas enriquecendo o estudo sobre o ALBP, além de muitas tentativas para reduzir a lacuna entre a discussão acadêmica e as situações reais. A formulação de Salveson consiste em, primeiramente, atribuir um conjunto de tarefas $I = \{1, 2, \dots, i, \dots, |I|\}$ a estações de trabalho linearmente ordenadas $M = \{1, 2, \dots, k, \dots, m\}$. A ordem das relações entre as tarefas é dada por um grafo de precedência *G*, onde um arco (*i*, *j*) existe se a tarefa *j* não puder ser iniciada antes do término da tarefa *i*. As tarefas atribuídas à estação de trabalho *k*, definida como I_k, são executadas sequencialmente, o tempo de processamento da estação de trabalho $T_k = \sum_{i \in I_k} t_i$ onde t_i é o tempo de processamento da tarefa i. A restrição de tempo de ciclo exige que o tempo de processamento da estação de trabalho não exceda um dado valor *c* definido como o tempo de ciclo da linha. O objetivo é atribuir todas as tarefas respeitando as restrições de precedência e de tempo de ciclo enquanto minimiza o número de estações de trabalho (BATTAIA; DOLGUI, 2013). A modelagem matemática do ALBP descrita na presente seção geralmente é formulada como um problema de programação binária, conforme o modelo apresentado nas Figura 2 e 3 por Boysen e Fliedner (2008).

Figura 2 - Notação ALBP

n	número de tarefas;
M^{St}	número máximo de estações;
c	tempo de ciclo;
t_i	tempo de processamento da tarefa *i*;
V	conjunto de atividades;
E	conjunto de tarefas que só podem ser executadas depois da tarefa *i* estar concluída (tarefas sucessoras de *i*);
x_{ik}	variável binária com valor 1, caso a tarefa $i \in V$ seja atribuída à estação $k \in M^{St}$

Fonte: Adaptado de Boysen e Fliedner (2008, p. 43)

Figura 3 - Modelagem matemática do ALBP

$$Min\ z = \sum_{k \in M^{St}} x_{nk} \times k \qquad (1)$$

$$s.t. \sum_{k \in M^{St}} x_{ik} = 1 \qquad \forall i \in V \qquad (2)$$

$$\sum_{k \in M^{St}} K \times x_{ik} \leq \sum_{k \in M^{St}} k \times x_{jk} \qquad \forall (i,j) \in E \qquad (3)$$

$$max_{k \in M^{St}} \left\{ \sum_{i \in V} x_{ik} \times t_i \right\} \leq c \qquad (4)$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V; k \in M^{St} \qquad (5)$$

Fonte: Adaptado de Boysen e Fliedner (2008, p. 43)

Na Figura 3, a função objetivo (1) visa à minimização do número de estações de trabalho utilizadas. A restrição (2) estabelece que cada tarefa seja atribuída a uma única estação. Já a restrição (3) é uma restrição de precedência. A restrição (4) impõe que a soma dos tempos t_i das atividades alocadas para cada estação não ultrapasse o tempo de ciclo *c* em nenhum posto de trabalho e a restrição (5) é de integralidade.

Devido à natureza combinatória do ALBP, ele é *NP-hard* mesmo na sua versão mais simples (GUTJAHR; NEMHAUSER, 1964) e uma grande quantidade de pesquisas têm sido conduzidas para resolver o problema em tempo computacional razoável (um exemplo de esforço nesse sentido é o trabalho de Sewell e Jacobson, 2012). Então, em virtude da complexidade do problema, os procedimentos heurísticos e metaheurísticas são os métodos de abordagem mais aceitáveis para o ALBP, uma vez que são mais hábeis para lidar com problemas complexos, grandes e que apresentam numerosas restrições (RASHID; HUTABARAT; TIWARI, 2011).

2.4 CLASSIFICAÇÃO DOS PROBLEMAS DE ALB

Uma classificação inicial dos ALBP foi proposta por Baybars em 1986 que os dividiu 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 problema de balanceamento da linha de montagem simples trata-se do problema que foi formulado por Salveson e 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 e possui as seguintes propriedades (BAYBARS, 1986; SCHOLL; BECKER, 2006):

- produção em massa de um produto homogêneo;

- todas as tarefas são processadas de um modo pré-determinado;

- linha compassada com tempo de ciclo *c* fixo de acordo com uma desejada quantidade de resultado;

- tempos de operação (de tarefa) determinísticos e inteiros;

- nenhuma restrição de atribuição além das restrições de precedência;

- *layout* da linha em série com *m* estações somente de um lado (*one sided*) e sem linha de alimentação;

- a sequência do processamento das tarefas está sujeita a restrições de precedência;

- uma tarefa não pode ser dividida entre duas ou mais estações;

- todas as estações são igualmente equipadas com respeito a máquinas e trabalhadores; e

- maximização da eficiência da linha: $E_{ff} = \frac{t_{sum}}{m \times c}$, no qual *m* é o número de estações e $t_{sum} = \sum_{j=1}^{N} t_j$ é a soma do tempo de processamento de todas as tarefas.

Em relação à função objetivo considerada, os SALBP ainda são distinguidos (SCHOLL; BECKER, 2006) em quatro tipos: o SALBP – 1 objetiva minimizar o número de estações para um dado tempo de ciclo. O SALBP – 2 visa minimizar o tempo de ciclo para um dado número de estações de trabalho. O SALBP – E é uma versão mais geral; ele 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 consiste em um problema de factibilidade. Dado o número de estações *m* e o tempo de ciclo *c*, o problema consiste em determinar se ele é factível ou não e, se o mesmo for factível, o objetivo passa a ser encontrar essa solução. As versões do SALBP descritas podem ser resumidas no Quadro 1:

Quadro 1 - Versões do SALBP

	Tempo de ciclo *c*	
	Dado	Minimizar
Nº de estações *m*		
Dado	SALBP-F	SALBP-2
Minimizar	SALBP-1	SALBP-E

Fonte: Adaptado de Becker e Scholl (2006, p. 698)

Como se pode perceber, as propriedades do SALBP são muito restritas com respeito aos sistemas de linhas de montagem do mundo real. Portanto, os pesquisadores têm focado

em identificar, modelar e resolver as situações nas linhas de montagem de maneira mais realista em relação às propriedades descritas. Os problemas resultantes das considerações do mundo real são chamados de Problemas de Balanceamento de Linhas de Montagem Gerais (GALBP) e eles vêm cada vez mais ganhando o interesse dos pesquisadores (MARTINO; PASTOR, 2010).

Alguns exemplos de GALBP são os trabalhos que consideram estações de trabalho paralelas (GÖKÇEN; AGPAK; BENZER, 2006), balanceamento de linhas de montagem em forma de U (AVIKAL *et al*., 2013) e linhas de montagem de dois lados (KHORASANIAN; HEJAZI; MOSLEHI, 2013). Uma visão mais abrangente do GALBP é fornecida pela *survey* de Becker e Scholl (2006).

Apesar de o SALBP ser intensivamente estudado, a quantidade de publicações recentes mostra que ele ainda é um tópico desafiador para os pesquisadores (vide os trabalhos de Bautista e Pereira, 2009; Kilincci, 2010; Sewell e Jacobson, 2012). Além disso, o crescente número de trabalhos que tratam do SALBP também se justifica pelo fato de que todos os demais problemas de balanceamento de linha de montagem derivarem dele.

De acordo com Boysen, Fliedner e Scholl (2007), embora exista uma grande variedade de ALBP na literatura, há cinco pressupostos presentes em qualquer tipo de ALBP, tanto no SALBP quanto no GALBP. São eles:

H-1: Os produtos a serem manufaturados (um ou mais) são conhecidos com certeza.

H-2: Um conjunto de processamentos alternativos é dado.

H-3: A linha deve ser configurada de tal modo que as quantidades alvo de produção sejam satisfeitas considerando certo horizonte de planejamento. Isso pode ser realizado ao definir o tempo de ciclo e, assim, a taxa de produção, ou, se o número de vendas não for um fator limitante, por buscar produzir tanto quanto possível.

H-4: O fluxo da linha é unidirecional.

H-5: A sequência de processamento das tarefas está sujeita às restrições de precedência.

O esquema de classificação dos ALBP proposto por Boysen, Fliedner e Scholl (2007) baseia-se nessas cinco hipóteses para julgar se determinado problema trata-se de ALBP (SALBP ou GALBP). Tal classificação será adotada nessa dissertação. Preponderantemente, o esquema de classificação de Boysen, Fliedner e Scholl (2007) se fundamenta na constatação de que qualquer problema ALB consiste de, no mínimo, três elementos básicos: um *grafo de precedência*, o qual compreende todas as tarefas e recursos a serem atribuídos, as *estações*, que formam a linha e algum tipo de *objetivo* a ser otimizado. Sendo assim, a classificação de

Boysen, Fliedner e Scholl (2007), baseia-se nesses três elementos, os quais são representados pela notação tupla [α | β | γ], onde:

α – refere-se às características do grafo de precedência;

β – representa as características da estação e da linha;

γ – representa os objetivos esperados com o modelo.

Para cada um dos elementos que faz parte da notação tupla, descreve-se na próxima seção uma série de possibilidades de restrições que são encontrados em problemas reais e que são retratados pela literatura.

2.4.1 Características do grafo de precedência (α)

As características do grafo de precedência (α) são representadas pelos seis atributos seguintes:

a) α_1: Grafos de precedência para produtos específicos;

b) α_2: Estrutura do grafo de precedência;

c) α_3: Tempo de processamento;

d) α_4: Incrementos dos tempos de tarefas dependentes da sequência;

e) α_5: Restrições de atribuição; e

f) α_6: Alternativas de processamento.

Além dessa divisão em relação aos atributos dos grafos de precedência, existe uma subdivisão em cada uma das características dos atributos, conforme se verifica no Quadro 2.

Quadro 2 - Características do grafo de precedência (α)

α_1	mix	Modelos mix de produtos
	mult	Modelos multi produtos
	o	Modelo único de produto
α_2	spec	Estruturas especiais do grafo de precedência
	o	Grafo de precedência pode assumir qualquer estrutura acíclica
α_3	t^{sto}	Tempos de processamento estocásticos
	t^{dy}	Tempos de processamento dinâmicos (efeitos de aprendizado)
	o	Tempos de processamento constantes e determinísticos
α_4	Δt^{dir}	Incrementos do tempo da tarefa devido a sucessão direta entre tarefas (troca de ferramentas)
	Δt^{indir}	Incrementos do tempo da tarefa devido a sucessão de tarefas (atividades atrapalham umas as outras)
	o	Incrementos do tempo da tarefa não considerados
α_5	link	Tarefas interligadas devem ser endereçadas para mesma estação
	inc	Tarefas incompatíveis não podem compartilhar a mesma estação
	cum	Restrições cumulativas de tarefas e estações
	fix	Tarefa só pode ser endereçada para uma estação específica
	excl	Tarefa não pode ser endereçada para uma estação específica
	type	Tarefa deve ser endereçada para determinado tipo de estação
	min	Mínima distância entre tarefas deve ser observada
	max	Máxima distância entre tarefas deve ser observada
	o	Restrições de endereçamento não são consideradas
α_6	pa^{o}	Tempos de processamento e custos são alterados
	pa^{predec}	Restrições de precedência são alteradas
	$pa^{subgraph}$	Subgrafos de precedência são adicionados
	o	Alternativas de processamento não são consideradas

Fonte: Cristo (2010, p. 19)

O atributo α_1 objetiva determinar se o problema trata de apenas um único produto ou de muitos produtos. Caso trate de apenas um produto, somente um grafo de precedência deve ser levado em consideração. Caso contrário, muitos grafos de precedência devem ser considerados simultaneamente. Salienta-se que o importante para o ALBP não é o número de produtos diferentes, mas sim o nível de homogeneidade entre os grafos de precedência. Seguindo a lógica do atributo α_1, ele pode ser classificado em:

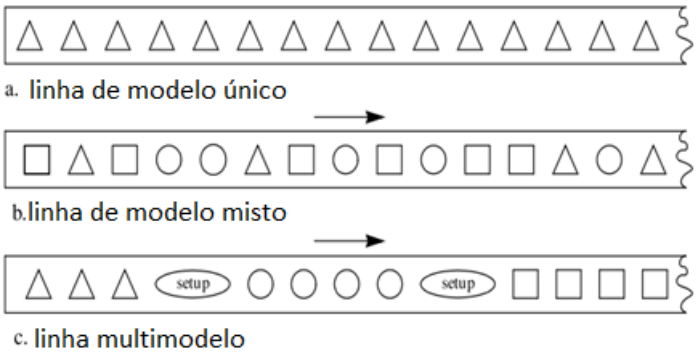
- Linha de modelo misto (α_1= mix): uma linha de modelo misto produz unidades de diferentes modelos de um produto numa sequência misturada arbitrariamente (BUKCHIN; DAR-EL; RUBINOVITZ, 2002).

- Linha multimodelo (α_1= mult): uma linha de multimodelos produz uma sequência de lotes (cada lote contém unidades de um modelo somente ou um grupo de modelos similares) com *setup* entre as operações. Na linha multimodelo, o balanceamento e o sequenciamento estão conectados ao problema de tamanho do lote.

- Linha de modelo único (α_1= o): um único produto é manufaturado ou o processo de produção de múltiplos produtos são (quase) idênticos não sendo necessário distingui-los (BOYSEN; FLIEDNER; SCHOLL, 2007).

Na Figura 4 estão caracterizados os três tipos de linhas abordados, onde os modelos diferentes de produtos são simbolizados por formas geométricas diversas.

Figura 4 - Linhas de montagem para produtos únicos, mistos e múltiplos



Fonte: Adaptada de Becker e Scholl (2006, p. 696)

Em relação ao atributo da estrutura do grafo de precedência, ele pode ser:

- especial (α_2= spec): quando o grafo possui uma estrutura especial como o grafo linear, divergente ou convergente.

- α_2= o: quando o grafo de precedência pode ter qualquer estrutura acíclica.

Já o atributo tempo de processamento diz respeito à variabilidade dos tempos das tarefas e se divide em:

- tempo de processamento estocástico (α_3= t^{sto}): neste caso é considerada a variabilidade dos tempos de processamento (SIMARIA, 2001) e se emprega uma função distribuição. As variações consideráveis de tempo devem-se, principalmente, à instabilidade dos humanos em relação à taxa de trabalho, habilidade e motivação, bem como, à possibilidade de falhas dos processos complexos.

- tempo de processamento dinâmico (α_3= t^{dy}): são as variações no tempo de processamento decorrentes dos efeitos da aprendizagem dos operadores.

- tempo de processamento estático e determinístico (α_3= o): ocorre quando as variações esperadas dos tempos da tarefa são suficientemente pequenas, como no caso de tarefas simples.

Outro atributo a ser observado é o que se refere aos incrementos no tempo das tarefas em virtude da ordem em que são realizadas. Este incremento pode ser causado pela sucessão direta de tarefas (α_4= Δt^{dir}). Por exemplo, caso duas tarefas sejam executadas na estação, uma

após a outra, um tempo incremental pode ser exigido para as operações de *setup* ou para a mudança de ferramentas (BOYSEN; FLIEDNER; SCHOLL, 2007). Ou pode ser causado pela sucessão indireta ($\alpha_4 = \Delta t^{indir}$) que ocorre caso o estado alcançado ao completar tarefas particulares tenha um efeito sobre o tempo de processamento de outras tarefas que são executadas por último na mesma ou em outra estação (SCHOLL; BECKER, 2006). Além disso, existe a possibilidade desses incrementos não serem considerados ($\alpha_4 =$ o).

O atributo α_5 trata das restrições de atribuição das tarefas às estações de trabalho. Em determinadas situações, algumas tarefas devem ser atribuídas à mesma estação, por exemplo, quando as estações empregam o mesmo recurso que não pode ser duplicado. A essa restrição dá-se o nome de *link* ($\alpha_{5=}$ link). Outra restrição é a de incompatibilidade entre as tarefas ($\alpha_{5=}$ inc). Ela determina que um conjunto de tarefas são incompatíveis e não devem ser atribuídas à mesma estação. É o que ocorre quando atividades incompatíveis como solda e abastecimento de combustível, por exemplo, não podem ser alocadas a uma mesma estação. Outra limitação possível na atividade de atribuição das tarefas se refere aos recursos de uma estação, como, por exemplo, o espaço disponível para peças, que pode ser limitado, restringindo o número de atividades a serem alocadas para ela ($\alpha_{5=}$ cum). Existe também a restrição que atribui algumas tarefas somente a estações específicas ($\alpha_{5=}$ fix) devido à dificuldade de realocar ou mover determinado recurso, principalmente, no caso de instalações antigas. Também é possível ocorrer o caso de algumas tarefas não poderem ser atribuídas a determinadas estações ($\alpha_5 =$ excl). Este evento pode ocorrer caso uma estação tenha alto índice de especialização e, para utilizar melhor tais recursos, atividades que não exigem tal tecnologia para sua realização são encaminhadas para outros postos de trabalho (CRISTO, 2010).

De outra forma, a especialização do posto de trabalho pode determinar que certas atividades sejam realizadas nas posições em que se apresentam disponíveis os recursos necessários ($\alpha_{5=}$ type). Finalmente, existem as restrições relativas às distâncias mínimas ($\alpha_5 =$ min) ou máximas ($\alpha_5 =$ max) entre as tarefas, podendo, essas distâncias, serem representadas em tempo, número de estações ou distância física. Uma distância mínima de tempo, por exemplo, é exigida quando se trata da atividade de secagem da tinta antes de realizar outras atividades que utilizem o objeto que foi pintado. Já uma situação de distância máxima é observada entre a atividade de aplicação de cola e o posicionamento do objeto a ser colado. Também existe a possibilidade de não levar em consideração as restrições de endereçamento ($\alpha_{5=}$ 0).

O último atributo da classificação, segundo as características dos grafos de precedências, refere-se às alternativas de processamento (α_6= pa^{λ}), as quais podem ser divididas segundo os efeitos que tem sobre o grafo de precedência em:

- λ = o: quando as alternativas de processamento afetam somente os tempos e os custos.

- λ = preced: ocorre quando o processamento alternativo afeta os tempos, os custos e as relações de precedência entre as tarefas. Boysen Fliedner e Scholl (2007) exemplificam esta situação com a condição de, ao ser necessária a inclusão de um componente no produto, as relações de precedência de outro componente podem sofrer alterações.

- λ = subgraph: os processamentos alternativos alteram grandes partes do processo produtivo, de tal forma que subgrafos inteiros podem ser substituídos.

- 0 = nessa situação não se considera as alternativas de processamento.

2.4.2 Características das estações de trabalho e das linhas (β)

As características das estações de trabalho e do arranjo das linhas são classificadas da seguinte forma:

a) movimento dos produtos;
b) *layout* da linha;
c) paralelismo;
d) atribuição de recursos;
e) estação sujeita a incrementos no tempo de processamento; e
f) aspectos adicionais da configuração da linha.

Cada uma das classificações citadas acima possui ainda pontos específicos, conforme se verifica no Quadro 3.

Quadro 3 - Características das estações de trabalho e das linhas (β)

β_1	oλυ	λ=o	Linha compassada e atividades restritas pelo tempo de ciclo (na média)
		λ=each	Linha compassada e todos os modelos devem respeitar o tempo de ciclo
		λ=prob	Linha compassada e o tempo de ciclo é respeitado com certa probabilidade
		υ=o	Linha compassada e tempo de ciclo único para todas as estações
		υ=div	Linha compassada e tempos de ciclo locais
	$unpac^{o}$		Linha descompassada e dessincronizada
	$unpac^{syn}$		Linha descompassada e sincronizada
β_2	o		Linha seriada
	u^{o}		Linha em U
	u^{n}		Linha em U formando "n" Us
β_3	$pline^{\lambda}$		Linhas em paralelo
	$pstat^{\lambda}$		Estações em paralelo
	$ptask^{\lambda}$		Atividades em paralelo
	$pwork^{\lambda}$		Trabalhos paralelos em uma mesma estação
	o		Nenhum paralelismo é considerado
β_4	equip		Problema de seleção de equipamentos
	res^{01}		Se duas atividades dividem um recurso, os investimentos são reduzidos na estação
	res^{max}		A tarefa mais complexa define o nível de qualificação requerido para os recursos
	res^{o}		Outros tipos de sinergia ou dependência
β_5	Δt^{imp}		Atividades improdutivas na estação são consideradas
	o		Não considerados incrementos de tempo na estação
β_6	buffer		Buffers devem ser alocados e dimensionados
	feeder		Linhas secundárias devem ser balanceadas simultaneamente
	mat		Embalagens dos materiais devem ser dimensionadas e posicionadas
	change		Necessárias máquinas para trocar os produtos de posição
	o		Aspectos adicionais não são considerados

Fonte: Cristo (2010, p. 25)

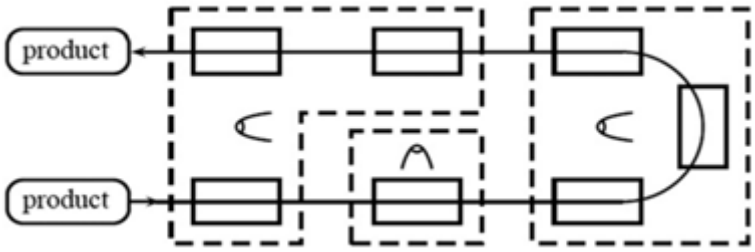
Os produtos podem mover-se pelas linhas de duas formas, segundo Boysen, Fliedner e Scholl (2007): de maneira compassada (β_1= oλv) ou descompassada (β_1= $unpac^{\lambda}$). Num sistema de produção com linha compassada se tem um tempo de ciclo, o qual restringe o tempo do processo em todas as estações (λ= o). O compasso (ritmo) é mantido por meio de uma esteira transportadora que força os operadores a finalizarem suas operações antes que o produto alcance o fim da estação de trabalho ou por meio de um transporte intermitente, onde os produtos chegam a um ponto final em cada estação, mas são automaticamente transferidos assim que decorre um determinado intervalo de tempo (BOYSEN; FLIEDNER; SCHOLL, 2008). Nas linhas compassadas, o objetivo é que todas as estações tenham um volume de atividades que permita respeitar o tempo de ciclo em todas elas, adequadas para os casos de linha que produz um único produto (*single line*). Contudo, quando se trata de linhas mistas (*mix line*) ou multimodelo considera-se que todos os modelos devem respeitar o tempo de ciclo (λ= each) ou que ele deve ser respeitado com certa probabilidade ou proporção (λ= prob). Além disso, existem as linhas nas quais todas as estações e modelos têm que respeitar o

mesmo tempo de ciclo global (ν= o) e aquelas nas quais o tempo de ciclo diverge entre as estações e os modelos (ν= div), que é o caso de estações teste.

Um sistema de produção com linha descompassada (β_1= unpac$^{\lambda}$), por sua vez, não é estritamente restrito por um tempo de ciclo, ou seja, os produtos são transferidos para as etapas seguintes assim que as operações necessárias em determinada estação foram terminadas, em vez de ser vinculadas a um dado intervalo de tempo. As linhas descompassadas se subdividem em sincronizadas (λ= syn) e em dessincronizadas (λ= o). Uma linha sincronizada transfere o produto simultaneamente assim que todas as estações terminam as suas atividades e a linha dessincronizada transfere o produto para a próxima etapa assim que uma estação termina suas tarefas.

Quanto ao *layout* da linha (β_2) ela pode ser em série (β_2= o) ou em forma de U (β_2 = u$^{\lambda}$). O *layout* da linha em forma de U surgiu com a implantação dos princípios *just-in-time*. Neste tipo de *layout* as duas extremidades da linha encontram-se próximas uma da outra formando um ”U”, possibilitando que os trabalhadores operem nos dois segmentos da linha, conforme se observa na Figura 5.

Figura 5 - Linha em formato U



Fonte: Battaia e Dolgui (2013, p. 263)

Dentre os benefícios advindos da utilização de uma linha de montagem em forma de U cita-se o aprimoramento da habilidade e experiência dos operadores, uma vez que eles estão envolvidos em diferentes partes do processo produtivo. Também a melhoria da visibilidade do processo e da comunicação entre os operadores, visto que os postos estão mais próximos. Ademais, no caso das linhas em U, uma determinada tarefa pode ser atribuída a postos anteriores, o que aumenta a eficiência da linha de montagem, equilibrando melhor a carga de trabalho entre os postos (SIMARIA, 2001).

De acordo com Boysen, Fliedner e Scholl, (2007) o *layout* de linha em U se classifica em:

- λ = o: a linha forma um único U.

- λ = n: são múltiplos Us formando uma linha com n-Us.

Outro atributo presente nos problemas de ALB diz respeito ao paralelismo (β_3) entre atividades e estações. Este atributo está subdivido em quatro categorias: paralelismo de linhas (β_3= pline$^{\lambda}$), paralelismo de estações (β_3= pstat$^{\lambda}$), paralelismo de tarefas (β_3= ptask$^{\lambda}$) e paralelismo entre operadores (β_3= pwork$^{\lambda}$). Também existe a possibilidade de desconsiderar qualquer tipo de paralelismo (β_3= 0).

O paralelismo entre linhas (β_3= pline$^{\lambda}$) permite aumentar a flexibilidade e o tempo de ciclo, diminuir a sensibilidade a falhas do sistema produtivo e possibilita uma resposta rápida a variações na demanda, pois o número de linhas em funcionamento pode ser alterado. Com o aumento do tempo de ciclo torna-se possível obter melhores balanceamentos, devido ao maior número de combinações possíveis entre as tarefas. Nesse contexto, surge o problema de determinar o número ótimo de linhas de montagem e os problemas de atribuição dos produtos e da mão-de-obra às linhas instaladas.

A situação de estações em paralelo (β_3= pstat$^{\lambda}$) é abordada na segunda categoria. A utilização de estações de trabalho em paralelo é necessária quando, numa determinada linha, a taxa de produção exigida é tão elevada que os tempos de processamento de algumas tarefas excedem o tempo de ciclo. Assim sendo, a linha de montagem passa a ser composta por estações de trabalho em que cada uma delas contém um certo número de postos de trabalho idênticos que executam simultaneamente o mesmo conjunto de tarefas, o que permite reduzir o tempo de ciclo (SIMARIA, 2001).

No contexto das tarefas paralelas (β_3= ptask$^{\lambda}$), a realização de tarefas similares ou idênticas em estações distintas é possível. Muitas vezes essa situação é evitada, pois pode gerar desperdícios. Contudo, o mais correto é fazer uma avaliação custo *versus* benefício para saber se o melhor é atribuir tarefas similares a uma única estação ou a estações distintas (BUKCHIN; RABINOWITCH, 2006).

Por último, a quarta categoria mostra o paralelismo entre os operadores (β_3= pwork$^{\lambda}$). Um caso especial de operadores trabalhando no mesmo produto simultaneamente são as linhas que possuem dois lados (*two-sided line*).

A característica β_4 refere-se à necessidade de alocação de recursos nas estações de trabalho. Segundo Boysen, Fliedner e Scholl (2007), a atribuição desses recursos se dá de dois modos. No primeiro (β_4= equip), apenas um equipamento de um conjunto de equipamentos pré-especificados pode ser escolhido para cada estação, conectando, assim, o problema de balanceamento ao problema de seleção de equipamento. No segundo (β_4= res$^{\lambda}$), os equipamentos são selecionados à medida que as tarefas são atribuídas às estações. Dessa forma, as atribuições de recursos no segundo caso se dão de três formas. A primeira consiste

em atribuir o recurso se, no mínimo, uma atividade da estação precisar dele (β_4= res^{01}), configurando-se em um problema de investimento 0-1. A segunda consiste em definir que a tarefa com maior nível de exigência impõe quais os recursos a serem utilizados (β_4= res^{max}). Já a terceira (β_4= res^{0}) consiste em alocar os recursos conforme outras situações de sinergia e dependência que não foram descritas por Boysen, Fliedner e Scholl (2007).

Uma outra variável influente nas linhas de montagem que pode ser considerada é o tempo de realização de atividades por condições que não agregam valor (β_5= Δ^{inp}), tais como os tempos de transporte de peças, tempos de deslocamento do operador ou tempos de trocas dos equipamentos. Outra possibilidade é desconsiderar os incrementos de tempo nas estações (β_5= 0).

Dependendo do sistema produtivo, exigências técnicas adicionais, como *buffers* e alimentadores, podem ser necessárias para balancear a linha de montagem (β_6). Muitas vezes, para minimizar os tempos de espera, *buffers* podem ser instalados entre as estações, os quais podem estocar os produtos temporariamente (β_6= buffer) (BOYSEN; FLIEDNER; SCHOLL, 2008). Além disso, alimentadores (β_6= feeder) podem ser empregados quando a linha de montagem tem linhas secundárias que se unem a uma linha principal. Nessa situação, as linhas secundárias devem ser balanceadas juntamente com a linha principal. As caixas que contém os materiais necessários (β_6= mat) devem estar disponíveis no momento de execução da tarefa e bem localizadas para evitar os custos de deslocamento. Algumas vezes também a tarefa exige que os produtos fiquem em determinadas posições (β_6= change), sendo necessário avaliar a necessidade de instalar máquinas para movimentar o produto. Em outras situações os aspectos adicionais nem são considerados (β_6= 0).

2.4.3 Características dos objetivos dos modelos (γ)

Os objetivos dos modelos dos problemas ALB também fazem parte da sua caracterização. Os principais objetivos estão enumerados no Quadro 4.

Quadro 4 - Objetivos (γ)

γ	m	Minimizar o número de estações
	c	Minimizar o tempo de ciclo
	E	Maximizar a eficiência da linha
	Co	Minimizar custo
	Pr	Maximizar rentabilidade
	SSL^{stat}	Os tempos das estações devem ser equalizados em uma estação (balanceamento horizontal)
	SSL^{line}	Os tempos das estações devem ser equalizados entre estações (balanceamento vertical)
	score	Minimizar ou maximizar algum indicador composto
	o	Somente soluções viáveis são testadas

Fonte: Cristo (2010, p. 30)

Os objetivos básicos de um problema ALB são de minimizar o número de estações *m* (γ= m), dado um tempo de ciclo, e o outro é de minimizar o tempo de ciclo *c* (γ= c), dado um determinado número de estações.

Além desses dois objetivos, outro objetivo clássico do problema ALB é a maximização da eficiência da linha (γ= E), minimizando simultaneamente o tempo de ciclo e o número de estações de trabalho. Outros objetivos dizem respeito a variáveis financeiras, como a minimização dos custos operacionais (γ= Co) ou a maximização da rentabilidade (γ= Pr).

Adicionalmente aos objetivos mencionados acima, há um objetivo complementar, que consiste no nivelamento da carga de trabalho entre as estações. Esse nivelamento pode ocorrer através do balanceamento horizontal (γ= SSL^{stat}), o qual ameniza as diferenças na carga de trabalho entre os produtos de uma linha mista ou multimodelos, ou por meio do balanceamento vertical (γ= SSL^{line}), que torna todas as estações da linha iguais no que tange à taxa de ocupação.

Por fim, Boysen, Fliedner e Scholl (2007), apresentam a notação (γ= score) caso seja necessária outra função objetivo (maximização ou minimização) para atender uma situação específica. Ainda, há a possibilidade de testar somente soluções viáveis (γ= 0).

2.5 PROBLEMA DE BALANCEAMENTO E SEQUENCIAMENTO DA LINHA DE MONTAGEM COM *SETUP*

O SUALBSP se constitui em uma abordagem mais enriquecida do clássico SALBP e foi denominado inicialmente de GALBPS (em inglês, *General Assembly Line Balancing Problem with Setups*). Ele é um dos problemas constituintes do ALB (em inglês, *Assembly Line Balancing*), que objetiva a redução da distância entre a teoria acadêmica e a realidade industrial e tem sido escassamente abordado na literatura científica (ANDRÉS; MIRALLES; PASTOR, 2008; MARTINO; PASTOR, 2010; SCHOLL; BOYSEN; FLIEDNER, 2013). Esse problema e suas outras versões são *NP-hard* (SCHOLL; BOYSEN; FLIEDNER, 2013; ANDRÉS; MIRALLES; PASTOR, 2008; MARTINO; PASTOR, 2010; SEYED-ALAGHEBAND; GHOMI; ZANDIEH, 2011).

Segundo Andrés, Miralles e Pastor (2008) e Martino e Pastor (2010), a literatura sobre balanceamento da linha de montagem tipicamente foca no problema num sentido puro como se, uma vez atribuídas as tarefas às estações, nenhuma definição de parâmetros a mais torna-se necessária. Em muitas linhas de produção reais, contudo, a sequência em que as tarefas são desenvolvidas dentro da estação importa, uma vez que os tempos de *setup* dependentes da sequência entre as tarefas estão presentes. O problema elaborado pelos autores explora justamente essa situação, mostrando 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. Essa resolução simultânea dos problemas reflete em um cenário mais realista para muitas linhas de montagem, especialmente para as linhas de montagem da indústria eletrônica ou setores com características similares que possuem tempos de ciclos baixos.

Além disso, a resolução conjunta dos problemas de balanceamento e sequenciamento representou uma inovação, uma vez que até aquele momento nos casos onde existiam tempos de *setup*, a programação da tarefa para cada estação era usualmente definida somente depois do balanceamento da linha e o problema de sequenciamento dentro de cada estação de trabalho poderia ser resolvido aproximadamente ou como um *Travelling Salesman Problem* (TSP), em que os tempos de *setup* representam as distâncias entre as cidades, embora existissem alguns caminhos proibidos devido à existência de relações de precedência.

Entretanto, essa abordagem de resolução do problema em dois estágios separados, fornecia normalmente somente soluções subótimas.

De acordo com Andrés, Miralles e Pastor (2008), na maioria das linhas de montagem industriais, os tempos de *setup* existem, mas não são considerados, pois são muito baixos comparados aos tempos de operação. Além disso, eles são considerados independentemente, ou seja, são executados apenas antes ou depois das tarefas (adicionando-se o tempo de *setup* ao tempo das tarefas). Contudo, são vários os exemplos práticos em que o tempo de *setup* é influenciado pelo sequenciamento das tarefas nas linhas de montagem compassadas (ritmadas). Um exemplo é a utilização de diferentes ferramentas sendo manuseadas pelos trabalhadores para desenvolver diferentes tarefas. Nessa situação, o importante é definir a melhor sequência de trabalho para o trabalhador, objetivando minimizar o tempo global da estação de trabalho, incluindo os tempos de *setup*. Essa é uma questão que se torna ainda mais relevante quando o tempo de ciclo é baixo, já que o tempo de *setup* pode representar um alto percentual desse tempo.

Outra situação real em que é possível encontrar os tempos de *setup* dependente do sequenciamento das tarefas é dentro das estações de linha de montagem com robôs. Nessas linhas, o robô deve retirar a ferramenta atual, selecionar a nova ferramenta de um conjunto e fazer ajustes antes de iniciar a próxima tarefa já atribuída. O tempo para esse ajuste de ferramenta pode variar e depende da sequência das tarefas. Outro exemplo está nas linhas de montagens automotivas, nas quais o trabalho pode ser realizado em diferentes posições dentro e fora do corpo do carro, tornando as distâncias de caminhada relevantes e dependentes das posições onde duas tarefas consecutivas são realizadas (SCHOLL; BOYSEN; FLIEDNER, 2013).

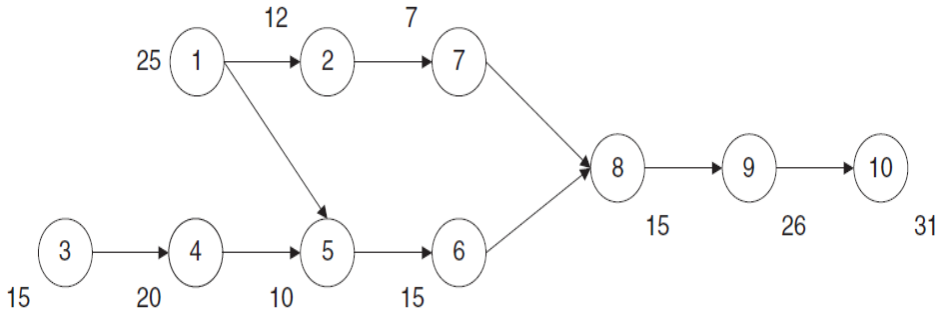
Sendo assim, o SUALBSP consiste em atribuir e sequenciar um conjunto de tarefas a estações de trabalho, tal que as restrições de precedência entre tarefas sejam mantidas, os tempos de *setup* entre as tarefas sejam considerados e uma dada medida de eficiência seja otimizada (ANDRÉS; MIRALLES; PASTOR, 2008; MARTINO; PASTOR, 2010).

Um exemplo de uma linha de montagem com tempos de *setup* dependentes da sequência é fornecido por Seyed-Alagheband, Ghomi e Zandieh (2011). O objetivo deles é atribuir as tarefas às estações de trabalho de tal maneira que nenhuma relação de precedência seja violada, minimizando o tempo de ciclo para um dado número de estações de trabalho. Destaca-se que esse problema lida com a maximização da taxa de produção numa linha de montagem existente. Ele pode surgir quando mudanças no processo produtivo ou na estrutura da procura levam a uma alteração do sistema de produção. Nesta linha de montagem (e

também na linha de montagem considerada no presente trabalho) um único modelo de produto é montado numa linha compassada, os tempos de processamento das tarefas e de *setups* são determinísticos e independentes das estações de trabalho em que são processados, cada tarefa é atribuída e processada somente em uma estação, todas as estações são igualmente equipadas e podem processar somente uma tarefa por vez.

Seyed-Alagheband, Ghomi e Zandieh (2011) apresentam o grafo de precedência na Figura 6, no qual os tempos de processamento da tarefa correspondem aos números ao lado dos círculos que contêm o número da tarefa. Por exemplo, a tarefa 1 possui um tempo de processamento de 25. Uma matriz de tempo de *setup* foi gerada aleatoriamente, conforme a Figura 7. A matriz de tempo de *setup* é interpretada da seguinte maneira: o tempo de *setup* tsu_{58} representa que, caso a tarefa 8 seja atribuída exatamente depois da tarefa 5, um tempo adicional $tsu_{58} = 2$ necessita ser considerado (por exemplo, para a troca de ferramenta) antes de executar a tarefa 8.

Figura 6 - Grafo de precedência exemplo



Fonte: Seyed-Alagheband, Ghomi e Zandieh (2011, p. 808)

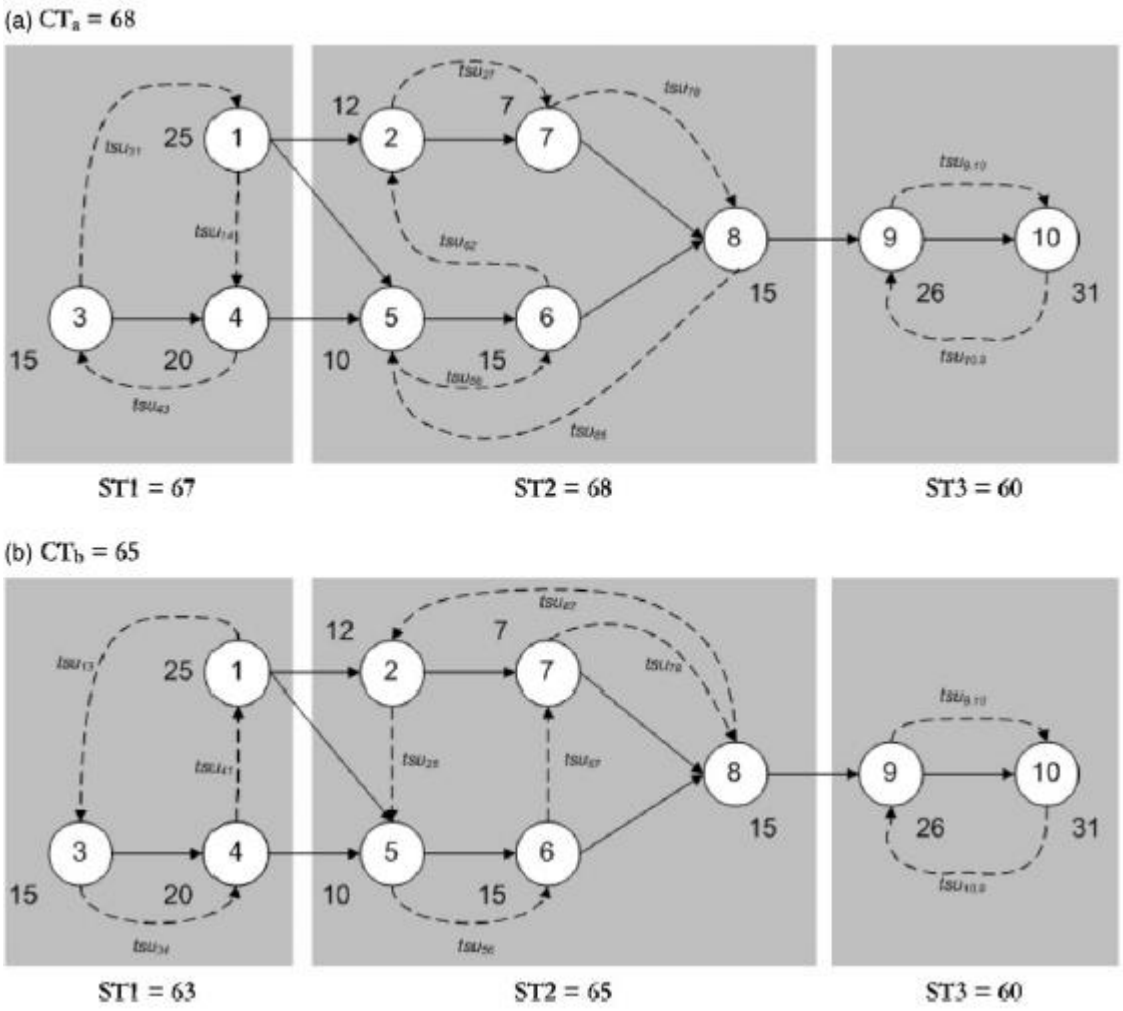
Figura 7 - Matriz de tempos de *setup*

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	2	3	0	2	2	1	0
2	0	0	3	0	2	0	3	2	0	1
3	3	0	0	0	2	1	1	0	1	0
4	2	2	2	0	2	3	1	1	0	0
5	0	2	3	0	0	3	2	2	1	0
6	1	0	2	0	1	0	0	0	0	0
7	2	3	2	3	2	1	0	0	1	3
8	0	1	2	2	3	2	2	0	3	3
9	1	0	2	3	3	0	2	1	0	1
10	1	2	1	1	3	1	2	0	2	0

Fonte: Seyed-Alagheband, Ghomi e Zandieh (2011, p. 808)

Considerando que o número de estações é dado e igual a 3, duas soluções são obtidas, vide Figura 8. Nessa figura, os sequenciamentos do trabalho dentro de cada estação são representados pelas linhas pontilhadas. Embora a atribuição das tarefas às estações nas duas soluções seja a mesma, o sequenciamento intraestação das tarefas resultou em dois tempos de ciclo diferentes para cada solução (tempo de ciclo (a) = 68 e tempo de ciclo (b) = 65). Este exemplo demonstra a importância de levar em consideração os tempos de *setup* dependentes da sequência entre as tarefas.

Figura 8 - Tempos de ciclo diferentes devido ao sequenciamento



Fonte: Seyed-Alagheband, Ghomi e Zandieh (2011, p. 809)

2.5.1 Trabalhos Anteriores

O SUALBSP tem sido escassamente abordado na literatura (ANDRÉS; MIRALLES; PASTOR, 2008; MARTINO; PASTOR, 2010; SCHOLL; BOYSEN; FLIEDNER, 2013), conforme demonstra o Quadro 5 que apresenta as principais características do problema abordadas nos artigos no que tange à classificação do problema, tipo de linha, o método de solução adotado, os tamanhos das instâncias testadas, o tipo de modelo adotado e se foram encontradas soluções ótimas para todas as instâncias testadas.

Quadro 5 - Resumo dos artigos sobre SUALBSP

Ano	Autores	Tipo de Problema	Tipo de linha	Técnica de Solução	Tamanho da Instância	Modelo	Solução ótima
2008	Andrés, Miralles e Pastor	GALBPS-1	linha de modelo único compassada	Regras heurísticas e GRASP	160	PLB*	Não
2010	Martino e Pastor	GALBPS-1	linha de modelo único compassada	Regras heurísticas	640	PLB	Não
2011	Seyed-Alagheband, Ghomi e Zandieh	GALBPS-2	linha de modelo único compassada	*Simulated Annealing*	40	PLB	Não
2012	Yolmeh e Kianfar	SUALBPS-2	linha de modelo único compassada	Algoritmo Genético híbrido	269	PLBM**	Não
2013	Scholl, Boysen e Fliedner	SUALBPS-1	linha de modelo único compassada	Regras heurísticas e GRASP	2.152	PLBM	Não

Fonte: Elaborado pela autora

*PLB = Programação Linear Binária
**PLBM = Programação Linear Binária Mista

O primeiro trabalho que tratou do SUALBSP foi o de Andrés, Miralles e Pastor (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* $\tau_{i,j}$ deve ser adicionado para computar o tempo global da estação de trabalho, fornecendo assim a sequência de tarefas dentro de cada estação de trabalho. Outrossim, 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* $\tau_{p,i}$ deve ser considerado também. Isso porque as tarefas são repetidas ciclicamente. A última tarefa em um ciclo da estação de trabalho é executada somente antes da primeira tarefa no próximo ciclo. Eles abordaram essa constatação provando que a linha de montagem não requer somente balanceamento, mas também a definição da sequência das tarefas atribuídas a cada estação em virtude, principalmente, da existência do tempo de *setup* dependente da sequência. Sendo assim, ambas as questões de balanceamento entre as estações e o sequenciamento das tarefas intraestação são resolvidas simultaneamente pelos autores.

Para formular o problema, Andrés, Miralles e Pastor (2008) consideraram um cenário em que diferentes ferramentas devem ser manuseadas pelos trabalhadores para desenvolver diferentes tarefas e, portanto, sempre que mais de uma tarefa é atribuída à mesma estação de trabalho os tempos de *setup* intraestação entre tarefas aparecem. O objetivo nessa situação é definir a melhor sequência de trabalho para o trabalhador visando à minimização do tempo global da estação de trabalho, incluindo os tempos de *setup*. As instâncias foram geradas a partir do exemplo que aparece em Scholl e Becker (2006) e que estão disponíveis na *homepage* da pesquisa de balanceamento de linha de montagem (http://alb.mansci.de/), sendo que os tempos de *setup* foram gerados aleatoriamente, já que se tratava de um trabalho inédito.

O modelo matemático formulado pelos autores é um modelo de programação linear binária e objetiva atribuir as tarefas às estações de trabalho e sequenciá-las de tal maneira que as relações de precedência não sejam violadas e que o tempo global, incluindo os tempos de *setup*, seja menor que o tempo de ciclo. Experimentalmente, eles trataram do GALBPS-1, cuja função objetivo é minimizar o número de estações para um dado tempo de ciclo. Os pressupostos básicos do modelo são os seguintes:

1- os tempos de processamento das tarefas, a matriz de tempos de *setup* e as relações de precedência são conhecidas deterministicamente;

2- os tempos de processamento e de *setup* são independentes das estações de trabalho nas quais as tarefas são processadas;

3- um único modelo de um produto é montado na linha;

4- trata-se de uma linha em série compassada onde os *buffers* não são considerados;

5- cada tarefa é atribuída a somente uma estação de trabalho;

6- as tarefas devem ser processadas somente uma vez;

7- todas as estações de trabalho são igualmente equipadas e qualquer tarefa pode ser atribuída a qualquer estação de trabalho;

8- nenhum dos tempos de processamento de tarefas pode ser maior que o tempo de ciclo; e

9- as estações de trabalho podem processar somente um produto de cada vez.

O modelo de André, Miralles e Pastor (2008) é composto pelos seguintes conjuntos, variáveis de decisão, parâmetros, função objetivo e restrições.

Conjuntos:

T_j = conjunto de tarefas que podem ser atribuídas à estação *j*

P = conjunto de duplas de tarefas (i, k) onde i é o predecessor imediato de k

PT_i = conjunto de todas as tarefas predecessoras da tarefa i

Variáveis de decisão:

$$y_j = \begin{cases} 1, \text{caso qualquer tarefa seja atribuída à estação } j\ (j = m_{min} + 1, \dots, m_{max}) \\ 0, \text{caso contrário} \end{cases}$$

$$x_{ijs} = \begin{cases} 1, \text{caso a tarefa } i \text{ seja atribuída à estação } j \text{ na posição } s \\ \text{do sequenciamento } (\mathrm{i} = 1, \dots, \mathrm{N}; \mathrm{j} = E_i, \dots, L_i; s = 1, \dots, N_{mj}) \\ 0, \text{caso contrário} \end{cases}$$

$$z_{ikj} = \begin{cases} 1, \text{caso a tarefa i seja executada imediatamente antes da} \\ \text{tarefa k na estação j no mesmo ou no próximo} \\ \text{ciclo } \left(\forall j;\ \forall (i,k) \middle| (i \neq k) \wedge \left(i, k \in T_j\right)\right) \\ 0, \text{caso contrário} \end{cases}$$

$$\mathrm{w_{ij}} = \begin{cases} 1, \text{caso a tarefa i seja a última na sequência das tarefas} \\ \text{atribuídas à estação } j\ (\forall i; j = E_i, \dots, L_i) \\ 0, \text{caso contrário} \end{cases}$$

Parâmetros:

i, k = tarefa

j = estação de trabalho

s = posição no interior do sequenciamento da estação

m_{min} = *lower bound* do número de estações

m_{max} = *upper bound* do número de estações

t_i = duração da tarefa i

TC = *upper bound* para o tempo de ciclo

E_i, L_i = primeira e última estação onde a tarefa i pode ser atribuída

Nm_j = número máximo de tarefas que podem ser atribuídas à estação *j*

NT_m = número máximo de tarefas que podem ser atribuídas a qualquer estação $NT_m = max_j \{Nm_j\}$

tsu_{ik} = tempo de *setup* quando a tarefa k é executada somente após a tarefa i na mesma estação

A expressão da função objetivo deste modelo é dada por:

$$Min\ Z = \sum_{j=m_{min}+1}^{m_{max}} j \times y_j \tag{6}$$

A função objetivo (6) visa minimizar o número de estações de trabalho.

$$\sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} x_{ijs} = 1 \qquad \forall i \tag{7}$$

A restrição (7) impõe que cada tarefa deve ser atribuída a somente uma posição no interior de uma estação de trabalho.

$$\sum_{\forall i \in T_j} x_{ijs} \leq 1 \qquad \forall j; = 1, \dots, Nm_j \tag{8}$$

A restrição (8) garante que cada posição no interior de cada estação de trabalho não terá mais do que uma tarefa atribuída.

$$\sum_{\forall i \in T_j} x_{ij,s+1} \ \leq \ \sum_{\forall i \in T_j} x_{ijs} \qquad \forall j; s = 1, \dots, Nm_j - 1 \tag{9}$$

A restrição (9) garante que as tarefas têm que ser atribuídas em posições crescentes no sequenciamento de cada estação de trabalho.

$$\sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} (NTm \times (j-1) + s) \times x_{ijs} \leq \sum_{j=E_k}^{L_k} \sum_{s=1}^{Nm_j} (NTm \times (j-1) + s) \times x_{kjs} \qquad \forall (i,k) \in P \tag{10}$$

A restrição (10) garante que as restrições de precedência não são violadas em relação à atribuição a diferentes estações e à posição no interior da mesma estação.

$$\sum_{\forall i \in T_j} \sum_{s=1}^{Nm_j} t_i \ \times \ x_{ijs} + \sum_{\forall (i,k|i \neq k) \wedge (i,k \in T_j)} tsu_{ik} \times z_{ikj} \leq TC \qquad j = 1, \dots, m_{min} \tag{11}$$

$$\sum_{\forall i \in T_j} \sum_{s=1}^{Nm_j} t_i \times x_{ijs} + \sum_{\forall (i,k|i \neq k) \wedge (i,k \in T_j)} tsu_{ik} \times z_{ikj} \ \leq TC \times y_j \qquad j = m_{min} + 1, \dots, m_{max} \tag{12}$$

Já as restrições (11) e (12) garantem que o tempo global em cada estação, incluindo o tempo de duração da tarefa e os tempos de *setup*, estão abaixo do tempo de ciclo.

$$x_{ijs} \ + \ x_{kj,s+1} \ \leq 1 \ + \ z_{ikj} \quad \forall j; s = 1, \dots, Nm_j - 1; \forall (i,k) | (i \neq k) \wedge (i,k \in T_j) \wedge (k \notin PT_i) \tag{13}$$

Na restrição (13) a variável z_{ikj} é 1 sempre que a tarefa *i* é atribuída à posição *s* e a tarefa *k* é atribuída à posição $s \ + 1$ na programação da estação de trabalho *j*.

$$x_{ijs} - \sum_{\forall k \in T_j \,|\, (i \neq k) \wedge (k \notin PT_i)} x_{kj,s+1} \leq w_{ij} \qquad \forall j; s = 1, \ldots, Nm_j - 1; \forall i \in T_j \qquad (14)$$

Na restrição (14) a variável w_{ij} é 1 sempre que a tarefa *i* é atribuída à última posição na programação da estação de trabalho *j*.

$$w_{ij} + x_{kj1} \leq 1 + z_{ikj} \qquad \forall j; \forall (i,k) | (i \neq k) \wedge (i,k \in T_j) \wedge (i \notin PT_k) \qquad (15)$$

A restrição (15) impõe que a variável z_{ikj} é 1 sempre que a tarefa *i* é atribuída à última posição e a tarefa *k* à primeira na programação da estação de trabalho *j*.

O modelo foi implementado no CPLEX 9 e testado em 160 instâncias com diferentes combinações de tamanho, ordem e variabilidade de *setups*. Para resolução do problema, Andrés, Miralles e Pastor (2008) desenvolveram oito regras heurísticas simples e um método heurístico GRASP específico para o problema. As oito regras heurísticas testadas servem para atribuir as tarefas às estações de trabalho e são baseadas na combinação entre a estratégia heurística orientada à estação (SH, em inglês *Station oriented strategy Heuristic*) e a estratégia heurística orientada à tarefa (TH, em inglês *Task oriented strategy Heuristic*) e nos seguintes critérios de ordem de seleção das tarefas: máximo tempo de *setup* mais tempos de processamento (max_ts), mínimo tempo de *setup* mais tempos de processamento (min_ts), máximo tempo de *setup* (max_s) e mínimo tempo de *setup* (min_s). Já o método GRASP foi utilizado devido ao fato de que o ALBP pode ser visto como um *packing problem* especial e ele produz soluções de boa qualidade para problemas de otimização combinatória *hard*, particularmente os *set packing problems*. 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 e os valores das soluções obtidas foram usadas como *lower bound* para avaliar a eficiência dos procedimentos heurísticos. As heurísticas com melhores desempenhos foram a GRASP com 10 iterações e a estratégia heurística orientada a estação com máximo tempo de *setup* mais tempos de processamento (max_ts SH).

Martino e Pastor (2010), por sua vez, propõem procedimentos heurísticos baseados em regras de prioridade para resolver o GALBPS-1, apresentando melhorias em relação aos procedimentos heurísticos elaborados por Andrés, Miralles e Pastor (2008), uma vez que os procedimentos deles forneciam soluções ótimas apenas para instâncias muito pequenas.

Os procedimentos heurísticos elaborados por Martino e Pastor (2010) são os procedimentos orientados à estação baseado nas regras de prioridades não ponderadas; os procedimentos orientados à tarefa com diversas regras de prioridades; os procedimentos

orientados à estação baseado em regras de prioridade ponderadas, ou seja, são aqueles procedimentos que foram aperfeiçoados por meio do algoritmo Nelder e Mead e os esquemas melhorados de atribuição de tarefas dentro de uma estação de trabalho. Para melhorar as atribuições de tarefas, esses esquemas basearam-se no estudo de todas as posições que uma tarefa candidata pode assumir, na execução de uma otimização local das tarefas atribuídas às estações de trabalho e na otimização local das tarefas atribuídas à estação de trabalho cada vez que uma nova tarefa é atribuída a ela.

Martino e Pastor (2010), da mesma forma que Andrés, Miralles e Pastor (2008), geraram as instâncias do problema baseando-se no conjunto de problemas disponíveis na *homepage* da pesquisa de balanceamento de linha de montagem e também geraram os tempos de *setup* aleatoriamente. Ao todo foram analisados 640 casos.

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 metaheurística GRASP e as oito regras heurísticas elaboradas por Andrés, Millares e Pastor (2008), sendo que o procedimento heurístico que obteve melhor desempenho apresentou um erro médio máximo de 14,96% em relação à solução ótima.

Já Seyed-Alagheband, Ghomi e Zandieh (2011), abordaram o GALBPS-2, cujo objetivo é encontrar o tempo de ciclo mínimo para um número pré-definido de estações de trabalho. Para resolver o problema, os autores adaptaram o modelo matemático de Andrés, Miralles e Pastor (2008) e desenvolveram um novo algoritmo *Simulated Annealing* (SA), já que os métodos de solução propostos por Andrés, Miralles e Pastor (2008) e por Martino e Pastor (2010) para o GALBPS-1 não forneceram bons resultados para o GALBPS-2.

Eles optaram em adotar *Simulated Anealinng* porque os experimentos mostraram que utilizar algoritmos evolucionários no caso do GALBPS-2 resultaria numa convergência prematura ao ótimo local. Contudo, como a versão clássica do SA é ineficiente para a resolução do GALBPS-2 com instâncias de tamanho grande e, além disso, a estrutura de pesquisa da vizinhança do SA também é fraca. Seyed-Alagheband, Ghomi e Zandieh (2011) efetuaram então várias modificações no clássico SA.

Seyed-Alagheband, Ghomi e Zandieh (2011) empregaram o método de Taguchi para aperfeiçoar os parâmetros do algoritmo SA modificado, o que o tornou mais eficiente. Dentre as melhorias adotadas no SA elaborado pelos autores consta a definição de novos operadores para a estrutura de pesquisa da vizinhança e a combinação desses operadores novos e os antigos, já que nenhum dos operadores sozinhos evita cair numa solução ótima local. Assim, se uma estrutura não pode fornecer melhores soluções depois de algumas iterações, a outra é

usada. Também usaram um mecanismo como estratégia de migração que é hábil em diversificar o espaço solução e aumentar a possibilidade de aceitar uma região inferior ou não visitada, chamada de "*giant leap*" (GL).

O modelo foi formulado em CPLEX 10.1 e testado em um conjunto de quatro problemas retirados da *homepage* de pesquisa sobre o tema de balanceamento de linha de montagem com seus respectivos tempos de *setup* gerados aleatoriamente. O *benchmark* construído e testado continha 40 instâncias com diferentes combinações de tamanho e tempos de *setup*.

Por fim, o algoritmo desenvolvido por Seyed-Alagheband, Ghomi e Zandieh (2011) obteve soluções ótimas para um dos conjuntos de problemas, retratando a qualidade aceitável do algoritmo SA proposto. Em termos de tempo de CPU, a abordagem exata levou 912.2 segundos para resolver o mesmo conjunto de problemas, enquanto que o SA proposto levou 0.4 segundos, mostrando o desempenho superior do SA.

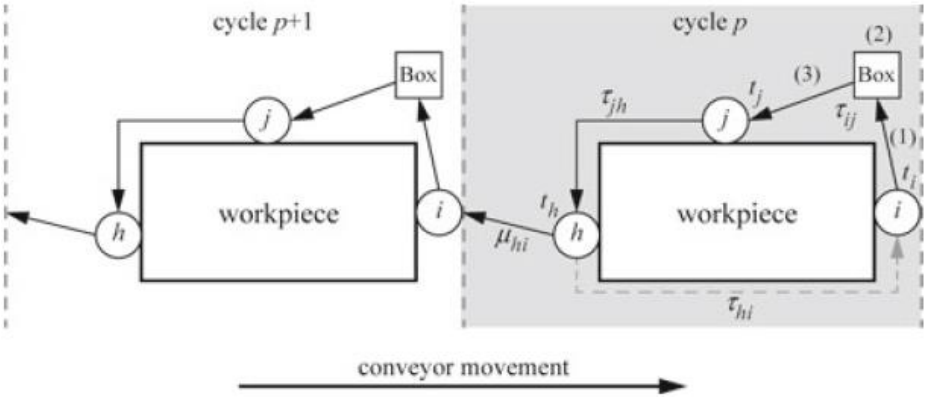
Scholl, Boysen e Fliedner (2013), por sua vez, modificaram o problema ao modelar os tempos de *setups* mais realisticamente, dando uma nova e mais compacta formulação ao modelo matemático. Além disso, desenvolveram procedimentos de solução heurísticas efetivos e renomearam o problema de GALBPS para SUALBSP.

A principal modificação empreendida por eles no SUALBSP foi a extensão do modelo de Andrés, Miralles e Pastor (2008), ao distinguirem entre *setups forward e backward*. Segundo Scholl, Boysen e Fliedner (2013), o termo *setup forward* se refere 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.

Scholl, Boysen e Fliedner (2013) mostram através da Figura 9 que os tempos de *setup forward* e *backward* geralmente são desiguais no mundo real. Na Figura 9 tem-se uma estação de trabalho com carga ordenada i, j, h. As tarefas i, j e h são executadas de maneira cíclica sobre produtos consecutivos. Cada ciclo *p* inicia no tempo 0 com a execução da tarefa *i* que

consome o tempo t_i. Depois, o operador tem que caminhar até uma caixa (1) onde ele tem que buscar uma parte necessária para a tarefa *j* (2). Partindo desta caixa, ele tem que 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_{jh}. Depois de executar *h* em t_h unidades de tempo, ele tem que ir para o produto sucessor no ciclo $p + 1$. Isto leva um tempo de *setup backward* μ_{hi} e o ciclo atual é finalizado. Para conseguir gerenciar todos os passos produtivos e improdutivos dentro de cada ciclo, respeita-se a seguinte restrição de tempo de ciclo: $t_i + \tau_{ij} + t_j + \tau_{jh} + t_h + \mu_{hi} \leq c$.

Figura 9 - Ciclos consecutivos e suas conexões numa única estação



Fonte: Scholl, Boysen e Fliedner (2013, p. 295)

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}. O modelo de Andrés, Miralles e Pastor (2008) não distinguia entre as cargas i, j, h e j, h, i, porque eles assumiam que os mesmos tempos de *setup* eram válidos em ambas as direções e isso restringe a aplicabilidade do modelo deles na prática a casos onde os *setups* não estão relacionados ao trabalho ou ao movimento de produtos.

No modelo com *setups forward* e *backward*, os autores assumiram que a desigualdade triangular (modificada) é válida. Por exemplo, assume-se que o tempo da estação cresça monotonicamente quando uma tarefa adicional é incluída. Essa condição é verdadeira caso as seguintes desigualdades se mantenham para todas as tarefas triplas *i, j, h* ∈*V*:

$$\tau_{ih} + t_h + \tau_{hj} \geq \tau_{ij} \text{ e } \tau_{ih} + t_h + \mu_{hj} \geq \mu_{ij} \; \forall i, j, h \in V \qquad (16)$$

Scholl, Boysen e Fliedner (2013), definiram que o novo problema SUALBSP consiste em construir um número mínimo de cargas ordenadas da estação de tal forma que a tarefa seja

atribuída a exatamente uma estação, as restrições de precedência sejam observadas e o tempo de ciclo não seja ultrapassado em qualquer estação quando se considera a execução da tarefa cíclica com tempos de *setups* (*backward e forward*) dependentes da sequência. Na classificação elaborada por Boysen, Fliedner e Scholl (2007), o SUALBSP é representado pela tupla [Δt_{dir} | | m].

O novo modelo do SUALBSP foi formulado por Scholl, Boysen e Fliedner (2013) como um modelo linear binário misto e ele contém $2 \times n^2 + n \times \bar{m}$ variáveis binárias e n variáveis contínuas. Devido $\bar{m} \leq$ n, o número de variáveis binárias é limitado por $O(n^2)$. Além disso, o modelo contém $7 \times n^2 + n \times \bar{m} + 5 \times n + 1$ restrições, limitado por $O(n^2)$. Por outro lado, o modelo de Andrés, Miralles e Pastor (2008) contêm $2 \times n^2 \times \bar{m} + n^2 + \bar{m}$ variáveis binárias e $2 \times n^3 \times \bar{m} + n^2 \times (\bar{m} + 1) + 2 \times (n + \bar{m}) + n$, ou seja, $O(n^4)$ restrições. O que comprova que o modelo de Scholl, Boysen e Fliedner (2013) é muito mais compacto e menor que o modelo de Andrés, Miralles e Pastor (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* e a reotimização. 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, Miralles e Pastor (2008) e o procedimento Avalanche desenvolvido por Boysen e Fliedner (2008).

Apenas para ilustrar o que Scholl, Fliedner e Boysen (2013) fizeram: eles modificaram o procedimento GRASP de Andrés, Miralles e Pastor (2008), pois a abordagem de seleção era bastante arbitrária já que somente uma única regra de prioridade era considerada e a média parecia ser bem ampla. Então, os autores definiram uma abordagem GRASP modificada, chamada Regra-GRASP, porque nela não são as tarefas que são selecionadas aleatoriamente, mas as regras de prioridade. Usando a regra escolhida, a tarefa com mais alta prioridade é selecionada e atribuída, em vez de qualquer uma com prioridade aceitável.

Para avaliar as capacidades das soluções dos novos procedimentos heurísticos elaborados para o SUALBSP, eles transformaram as 160 instâncias testadas no trabalho de Andrés, Miralles e Pastor (2008) na versão SUALBSP com *setups forward e backward*. Da mesma maneira, Scholl, Fliedner e Boysen (2013), testaram as 640 instâncias contidas no conjunto de dados de Martino e Pastor (2010).

Em decorrência do conjunto de dados de Martino e Pastor (2010) apresentar muitas limitações, tais como: (1) as instâncias terem sido selecionadas do conjunto de dados *benchmark* SALBP-1 muito arbitrariamente, (2) não ter sido feita uma distinção entre tempos de *setup forward e backward*, (3) a desigualdade triangular não ter sido observada, (4) o aperfeiçoamento de heurísticas apenas para esse conjunto de dados e (5) as soluções ótimas serem conhecidas somente para uma parte das instâncias tal que desvios do (também fraco) *lower bound* necessitarem serem considerados para medir a qualidade da solução, Scholl, Fliedner e Boysen (2013) elaboraram os conjuntos de dados SBF1 (para remover as limitações 1 a 4) e SBF2 (para remover a limitação 5), para os testes.

O SBF1 consiste de quatro conjuntos de dados. Cada conjunto de dados contém 269 instâncias (tendo 7 – 297 tarefas) com todos os demais dados do SALBP-1 como, os tempos de ciclo, os tempos das tarefas e as restrições de precedência. Como na maioria dos casos existem tempos de tarefas mínimos muito pequenos e, além disso, na prática os tempos de *setup* podem ser bem maiores que o tempo mínimo dessas tarefas, Scholl, Fliedner e Boysen (2013) relacionaram os tempos de *setup* ao tempo médio de tarefa t_{av}. Eles definiram $\alpha \times t_{av}$ como *upper bound* para os tempos de *setup forward* gerados, obtendo, assim, quatro conjuntos de dados diferentes ao definir α em 0,25, 0,50, 0,75 e 1,00. Isto é, os tempos de *setup* médios são 12,5, 25, 37,5 e 50% do tempo médio da tarefa. O segundo conjunto SBF2 é composto por quatro conjuntos de dados e é gerado como o SBF1, mas é tomado cuidado para que pelo menos uma solução SALBP-1 ótima permaneça factível e, assim, constitua uma solução ótima também para o SUALBSP.

Os algoritmos foram codificados e compilados usando Borland Delphi 7.0 como aplicativos de console e implementaram o modelo no MIP *solver* Fico XPress Optimizer 20.00.05.

Para examinar quais dos componentes desenvolvidos eram úteis, Scholl, Boysen e Fliedner (2013) testaram um número de combinações. Cada uma contendo a estrutura de solução Regra-GRASP e mais as seguintes:

F10 – 10 passos na direção *forward*,

FB10 – 10 passos na direção *forward* e 10 passos na direção *backward*,

FBC10 – FB10 mais abordagem de *grouping graph*,

FBR10 – FB10 mais a reotimização,

FBCR10 – FB10 mais *grouping graph* e reotimização,

FBCI10 – FB10 mais *grouping graph* e *fathoming*,

FBRI10 – FB10 mais reotimização e *fathoming*,

FBCRI10 – FB10 mais *grouping graph*, reotimização e *fathoming*,

FBCRI100 – FBCRI com 100 passos em vez de 10,

FBLS10 – FB10 mais pesquisa local de Andrés, Miralles e Pastor (2008) com primeira adaptação e movimento de aceitação estendido.

Nos testes realizados com os conjuntos de dados de Andrés, Miralles e Pastor (2008), os melhores desvios relativos foram obtidos pela estrutura FBCRI100, a qual combina todos os componentes desenvolvidos por Scholl, Boysen e Fliedner (2013). Essa heurística encontrou as melhores soluções para 157 das 160 instâncias com tempos computacionais moderados de cerca 1,5 s por instância na média.

Baseando-se nesse resultado, Scholl, Boysen e Fliedner (2013) concluíram que os novos componentes regra – GRASP, planejamento dos *backward* e a reotimização tem mais influência no resultado satisfatório, enquanto que o *grouping graph* é bem sucedido somente em poucos casos, mas possui baixo tempo computacional. Ademais, o *fathoming* reduziu significativamente o tempo computacional. O Avalanche apresentou qualidade de solução aceitável, mas exige muito tempo e o desempenho do *solver* XPress foi decepcionante.

Os testes realizados com os conjuntos de dados de Martino e Pastor (2008) obtiveram resultados similares aos conjuntos de dados de Andrés, Miralles e Pastor (2008).

Por outro lado, os testes realizados no conjunto de dados SBF demonstraram que esses novos conjuntos são mais desafiadores que os outros dois. No conjunto de dados SBF1, os procedimentos heurísticos FBRI10 e FBCRI10, superaram os outros procedimentos apresentando um pequeno tempo computacional e FBCRI100 forneceu mais melhorias nos resultados. Já no conjunto de dados SBF2, os achados são muito similares àqueles do SBF1. Quando aplicado ao conjunto de dados SBF, os resultados do *solver* Fico XPress mostraram-se decepcionantes e não competitivos.

Na Tabela 1 são mostrados os desempenhos do *solver* Fico XPress e da heurística FBCRI100, que apresentou os melhores resultados, aplicados ao conjunto de dados SBF2. As medidas de desempenho adotadas foram rel.best: desvio médio relativo do melhor *upper bound* encontrado (em %); rel.LB: desvio médio relativo do melhor *lower bound* conhecido (em %); rel.opt: desvio médio relativo do número mínimo de estações m (em %); cpu: tempo computacional médio em segundos; #opt: o número comprovado ideal, por exemplo, UB=LB; #found: número ótimo encontrado e #best: número de melhores soluções encontradas.

Tabela 1- Resultados *solver* Fico XPress e heurística FBCRI100 no conjunto SBF2

	Fico XPress				FBCRI100			
	α = 0,25	α = 0,50	α = 0,75	α = 1,00	α = 0,25	α = 0,50	α = 0,75	α = 1,00
Nº Instâncias	269	269	269	269	269	269	269	269
rel.best	4,03	6,92	9,94	11,61	0,02	0,06	0,1	0,14
rel.opt	7,16	10,74	14,4	16,39	3,06	3,67	4,18	5,32
rel.LB	12,39	16,23	20,01	22,04	8,2	8,99	9,52	10,7
cpu	97,13	97,35	97,87	97,54	9,8	10,68	10,99	3,52
#opt	37	22	19	21	67	64	61	56
#found	79	50	37	36	144	130	124	112
#best	116	67	44	43	267	264	261	209

Fonte: Elaborada pela autora, baseada em Scholl, Boysen e Fliedner, 2013

Observa-se que o desempenho do *solver* Fico XPress foi realmente decepcionante, principalmente, em relação ao tempo computacional médio (cpu). Esse tempo foi muito superior ao tempo da heurística FBCRI100, cerca de 1.014,29% a mais. Outros desempenhos insatisfatórios do *solver* foram relativos ao número de instâncias que encontraram melhores soluções (#best), pois ele encontrava, em média, 183 soluções a menos que a heurística FBCRI100. Além disso, o número de instâncias que atingiram a otimalidade (#opt) no Fico XPress é cerca de 39,92% menor do que o número da heurística com melhor desempenho.

Após, Scholl, Boysen e Fliedner (2013) avaliaram os resultados obtidos com o novo modelo e com os procedimentos heurísticos propostos comparando-os com os modelos que consideram os tempos de *setups* de uma maneira relaxada. Eles compararam com as seguintes abordagens de lidar com tempos de *setup* sem usar o SUALBSP: ignorar completamente os tempos de *setup* (NoS), aumentar os tempos de tarefas através de um tempo de *setup* mínimo (MiS), aumentar o tempo de tarefa através do tempo de *setup* médio (MeIS), aumentar o tempo de tarefa por meio da média de todos os tempos de *setup* (MeS), aumentar o tempo de tarefa pelo (tempo) distância a partir da posição média de montagem (MeP) e aumentar o tempo de tarefa considerando o tempo de *setup* máximo (MxS).

Essas abordagens foram aplicadas ao conjunto de dados SBF1 e somente MxS foi hábil em garantir soluções factíveis. Contudo, os resultados apresentaram um excesso de capacidade média de pelo menos 46%, ou seja, o número de estações exigidas para alcançar cargas de estação factíveis é 46% maior que o melhor número conhecido de estações computado pelas heurísticas SUALBSP de Scholl, Boysen e Fliedner (2013) (cerca de 78% quando comparados com os *lower bounds*). Todas as outras abordagens na realidade são ruins e esse mau resultado é devido ao relaxamento do aspecto mais importante do problema.

Resumidamente, Scholl, Boysen e Fliedner (2013) propuseram uma ferramenta que compreende muitos componentes heurísticos, que foram combinados para render soluções mais rápidas e com melhor qualidade para as instâncias de tamanho do mundo real. Os resultados computacionais demonstraram que as novas heurísticas são melhores que as primeiras abordagens de Andrés, Miralles e Pastor (2008) e de Martino e Pastor (2010) com relação à qualidade da solução e aos tempos computacionais. Outrossim, os experimentos computacionais mostraram que existe a necessidade de procedimentos de solução melhorados, em particular, nos casos de tempos de *setup* grandes (SCHOLL; BOYSEN; FLIEDNER, 2013).

Já Yolmeh e Kianfar (2012), propuseram um algoritmo genético híbrido para resolver o SUALBSP-2, cuja função objetivo é minimizar o tempo de ciclo para um dado número de estações. Eles utilizaram o modelo matemático elaborado por Scholl, Boysen e Fliedner (2013), que leva em consideração os tempos de *setup forward* e *backward* entre as tarefas. Além disso, os autores usaram uma permutação simples para determinar a sequência das tarefas e para determinar a atribuição das tarefas às estações, o algoritmo genético foi hibridizado usando um procedimento de programação dinâmica.

Yolmeh e Kianfar (2012) calibraram os parâmetros e os operadores do algoritmo usando o método de desenho dos experimentos e utilizaram o conjunto de dados SBF1 para ser resolvido pelo algoritmo genético elaborado. Dessa maneira, os resultados do algoritmo genético foram comparados com os resultados apresentados pelas heurísticas FBRI10, FBCRI10, FBCRI100 e FBLS10 elaborados por Scholl, Boysen e Fliedner (2013). Comprovou-se que os resultados do algoritmo genético híbrido são significativamente melhores que as heurísticas mais eficientes apresentadas para a resolução do SUALBSP.

3 PROCEDIMENTOS METODOLÓGICOS

Este capítulo é dividido em três seções: a Seção 3.1 especifica todas as fases do método empregado pelo estudo, a Seção 3.2 apresenta as técnicas de resolução do SUALBSP e na Seção 3.3 explana-se de forma mais aprofundada a técnica de solução utilizada neste trabalho, a abordagem matheurística.

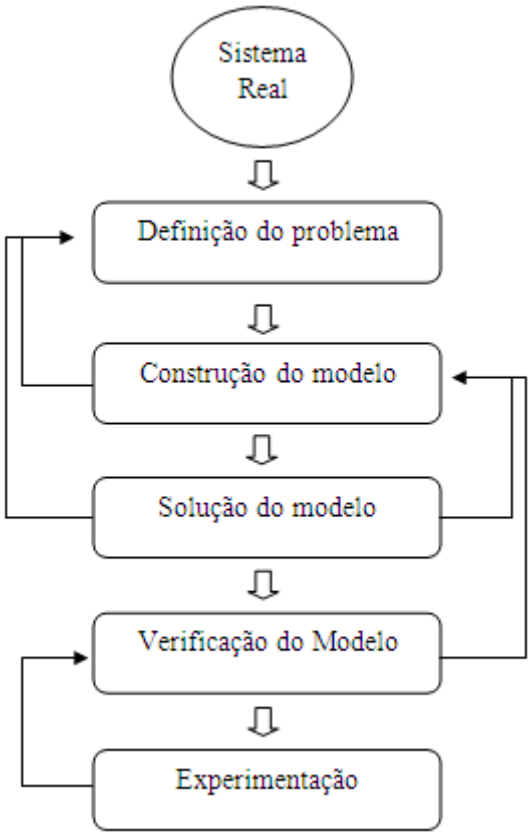
3.1 MÉTODO

Este trabalho foi desenvolvido com base nos métodos de modelagem de problemas da Pesquisa Operacional (PO).

De acordo com Arenales *et al*., (2007) fazer ciência é a capacidade de observar e descrever fenômenos naturais, sociais, econômicos, entre outros, sendo a matemática de importância fundamental nessas descrições. A partir da observação de fenômenos, processos ou sistemas, que podem ser físicos, químicos, biológicos ou econômicos, buscam-se leis que os regem. Essas leis, se passíveis de serem descritas por relações matemáticas, dão origem aos modelos matemáticos. Para Arenales *et al*., (2007) um modelo é um objeto abstrato que procura imitar as principais características de um objeto real para fins de representação do objeto real. Sendo assim, o modelo matemático é uma representação simplificada (abstração) do problema real. Ele deve ser suficientemente detalhado para captar os elementos essenciais do problema, mas suficientemente tratável por métodos de resolução.

Segundo os autores, a abordagem de resolução de um problema por meio de Pesquisa Operacional envolve as seguintes fases: (i) definição do problema; (ii) construção do modelo; (iii) solução do modelo; (iv) validação do modelo; e (v) implementação da solução. Neste trabalho, as etapas (iv) e (v) foram substituídas pelas etapas de verificação e de experimentação, respectivamente, de acordo com a Figura 10.

Figura 10 - Fases para resolução de um problema de PO



Fonte: Elaborada pela autora, baseada em Belfiore e Fávero, 2012.

Salienta-se que, diante de resultados insatisfatórios, cada fase do procedimento metodológico adotado pode obrigar que se retorne às fases anteriores para que ajustes sejam promovidos.

Para atingir os objetivos propostos neste trabalho, baseado nas fases apresentadas por Arenales *et al.*, (2007), descrevem-se os procedimentos metodológicos apresentados a partir da Seção 3.1.1.

3.1.1 Fase (i) – Definição do problema

O problema que este trabalho pretende resolver é o de balanceamento e sequenciamento de linha de montagem com tempos de *setup* dependentes da sequência (SUALBSP).

Para o SUALBSP tem-se: dado o número de estações de trabalho *k*, um grafo de precedência e os seus respectivos tempos de *setup backward* e *forward*, busca-se oferecer ao

tomador de decisão o desenho de uma linha de montagem com tempo total de estação minimizado, sabendo-se que cada tarefa será atribuída a exatamente uma estação de trabalho e sequenciada nela e que as restrições de precedência, o tempo de ciclo e o número de estações de trabalho não serão violados. A Seção 4.1 apresenta detalhadamente o problema objeto deste trabalho.

3.1.2 Fase (ii) – Modelagem

De acordo com Arenales *et al.*, (2007) essa fase traduz a fase de definição do problema em relações matemáticas ou lógicas de simulação, ou uma combinação delas. O desenvolvimento desta fase neste trabalho está exposto na Seção 4.2 que trata do modelo matemático e matheurístico.

3.1.3 Fase (iii) – Solução do modelo

Após a construção do modelo, parte-se para a fase (iii). A fase (iii) utiliza métodos de solução e algoritmos conhecidos para solucionar o modelo (ARENALES *et al*., 2007). Esta fase está contemplada no Capítulo 5, que trata da experimentação computacional do modelo, a partir de uma série de instâncias de dados. A fim de dirimir quaisquer dúvidas, apresenta-se na Seção 3.3 uma explanação sobre a abordagem de solução utilizada neste trabalho.

3.1.4 Fase (iv) – Verificação do modelo

O modelo foi verificado mediante a execução de algumas instâncias de dados SBF2 disponíveis no *website* do *Assembly Line Balancing Research Group* (http://alb.mansci.de/). Para selecionar essas instâncias processou-se o modelo PLI de Scholl, Boysen e Fliedner (2013) no MIP *solver* IBM© ILOG CPLEX Optimization Studio V12.5.1 com as configurações *default*, utilizando todo o conjunto de instâncias SBF2 limitado a 100

segundos. As instâncias que apresentaram resultados factíveis dentro dos 100 segundos foram selecionadas e processadas também no modelo MATH limitado igualmente a 100 segundos. Dessa forma, tornou-se possível comparar os resultados de ambos os modelos aplicados a mesma base de dados. Baseado nesse esquema, as instâncias selecionadas e utilizadas neste trabalho constam no Anexo A.

3.1.5 Fase (v) – Experimentação Computacional

Após a verificação do modelo, foram realizados experimentos computacionais com o modelo matemático de Scholl, Boysen e Fliedner (2013) e com a abordagem proposta. Eles 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 101 instâncias de dados SBF2, conforme será explicado no Capítulo 5.

Os resultados obtidos e suas avaliações encontram-se no Capítulo 5.

3.2 TÉCNICAS DE RESOLUÇÃO DO SUALBSP

De acordo com Caserta e Voβ (2009), a grande maioria dos problemas de decisão complexos do mundo real, quando modelados como problemas de otimização, pertencem à classe dos problemas *NP-hard*, classe na qual o SUALBSP está enquadrado. O fato de um problema pertencer à classe dos problemas *NP-hard* significa que não se conhece um algoritmo capaz de resolver o problema na otimalidade em tempo polinomial (CASERTA; VOβ, 2009). Além disso, segundo Goldberg e Luna (2005), o cerne da dificuldade da abordagem dos problemas denominados *NP-hard* está na explosão combinatória dos métodos enumerativos.

Dada essa realidade, nos últimos anos verificou-se o surgimento de técnicas de resolução de problemas *NP-hard* e *NP-completo* que garantem boas soluções, mas não soluções ótimas. Essas técnicas são a heurística e a metaheurística. Uma heurística é uma técnica (composta de uma regra ou um conjunto de regras) que busca boas soluções com um custo computacional razoável. Uma heurística é um método aproximado no sentido de que ele fornece uma solução boa com relativamente pouco esforço, mas não garante a otimalidade.

Já a metaheurística é um processo de geração iterativo que guia uma heurística subordinada ao combinar inteligentemente diferentes conceitos para explorar os espaços de pesquisa usando estratégias de aprendizado para estruturar a informação para encontrar soluções próximas ao ótimo eficientemente. A metaheurística apresenta um propósito geral, ou seja, ela não requer conhecimento específico sobre o problema e pode prontamente ser aplicada a um amplo espectro de classes de problemas, ao contrário das heurísticas.

O Quadro 5, apresentado na Seção 2.5.1, relaciona os trabalhos que trataram do SUALBSP juntamente com a devida técnica empregada. A partir desse quadro é possível verificar que as técnicas heurísticas e as metaheurísticas (GRASP, Algoritmo Genético e *Simulated Annealing*) já foram utilizadas pela comunidade acadêmica. Buscando o ineditismo na resolução do SUALBSP, o presente trabalho adotou a técnica matheurística, que será descrita detalhadamente na Seção 3.3.

A matheurística é plenamente justificável neste trabalho uma vez que, segundo Ribeiro e Maniezzo (2015, p. 1):

> Os algoritmos e estruturas metaheurísticas, tais como *Tabu Search*, *Simulated Annealing*, GRASP, VNS, algoritmos genéticos e colônia de formigas, entre outros, foram originalmente propostos e desenvolvidos quando os algoritmos e *softwares* de Programação Inteira Mista (MIP) não eram uma alternativa eficiente ou mesmo factível para resolver instâncias de problemas de larga escala do mundo real. Contudo, a pesquisa sobre programação matemática e, em particular, sobre otimização discreta, tem levado ao estado da arte onde MIP *solvers* ou códigos MIP customizados podem ser efetivos mesmo num contexto heurístico, primeiramente com os *solvers* ou com procedimentos para resolver subproblemas.

3.3 ABORDAGEM MATHEURÍSTICA

O interesse sobre métodos de otimização híbridos tem crescido nos últimos anos (JOURDAN; BASSEUR; TALBI, 2009). Híbrido pode indicar a combinação de diferentes meta(heurísticas) ou o uso de meta(heurísticas) com técnicas de programação matemática (CASERTA; VOβ, 2009). No presente trabalho, híbrido foi empregado no sentido de

combinação de meta(heurísticas) com técnicas de métodos exatos.

Tradicionalmente, os métodos de solução para alguns problemas de otimização combinatória são classificados em três grupos: métodos exatos, heurísticas e metaheurísticas. Nos últimos anos uma nova geração de métodos de otimização híbridos conhecidos como matheurísticas tem emergido e apareceram como uma quarta alternativa promissora (VILLEGAS *et al*., 2013). As matheurísticas vêm atraindo a atenção da comunidade de pesquisadores, o que originou uma quantidade abundante de trabalhos sobre o tema em poucos anos (DELLA CROCCE; GROSSO; SALASSA, 2014). Contudo, até o presente momento, esta abordagem híbrida não possui uma classificação única, nem um quadro de trabalhos consolidados na área, sendo difícil estabelecer uma pura e nítida definição desses métodos.

Segundo Della Crocce, Grosso e Salassa (2014), matheurística é um método que explora tanto a força dos algoritmos meta(heurísticos) quanto dos métodos exatos, levando a uma abordagem híbrida. Além disso, dentre os métodos híbridos, as matheurísticas estão recebendo uma atenção maior da comunidade de pesquisa, graças também ao grande avanço e melhoria nos *solvers* de programação inteira (STEFANELLO; ARAÚJO; MULLER, 2015). Destaca-se que nos últimos anos, os melhores resultados encontrados para muitos problemas de otimização acadêmicos ou práticos foram obtidos usando matheurísticas (MANERBA; MANSINI, 2014).

Uma característica distintiva das matheurísticas é a exploração de ferramentas de programação matemática não triviais, como parte do processo de solução (DELLA CROCCE; GROSSO; SALASSA, 2014). A questão crucial é que a estrutura desses métodos não é definida *a priori* e uma abordagem de solução pode ser construída de muitas maneiras diferentes. Como um exemplo geral, se pode construir um algoritmo matheurístico baseado numa metaheurística como a Pesquisa na Vizinhança Variável (VNS em inglês, *Variable Neighborhood Search*), por exemplo, com fases de pesquisa realizadas por um algoritmo exato, bem como por um MIP *solver*. Uma outra abordagem pode ser um procedimento de dois estágios: primeiro um procedimento heurístico é aplicado ao problema para gerar uma solução inicial e, então, um procedimento posterior de “refinamento” é aplicado explorando, por exemplo, alguma propriedade particular da formulação matemática do problema sob análise. Do mesmo modo, algumas matheurísticas usam informações da relaxação dos problemas de programação inteira para guiar a pesquisa local.

Ainda, de acordo com Boschetti *et al*., (2009), a matheurística pode ser feita de duas maneiras: uma usando programação matemática para melhorar ou desenhar meta(heurísticas)

outra usando meta(heurísticas) para melhorar conhecidas técnicas de programação matemática. A primeira é a mais estudada e é empregada, por exemplo, para melhorar a pesquisa local.

Concordemente com Boschetti *et al*., (2009), Caserta e Voβ (2009) explicam que, de um modo geral, algoritmos híbridos apresentam uma estrutura chamada "*master-slave*" para guiar um processo e sua aplicação. Ou (1) a meta(heurística) atua em alto nível e controla as chamadas à abordagem exata ou (2) a técnica exata atua como o mestre chamando e controlando o uso do esquema meta(heurístico).

Algoritmos híbridos do tipo 1 são aqueles em que a *definição* da vizinhança segue a lógica de uma meta(heurística), enquanto que a *exploração* da própria vizinhança é guiada pela abordagem exata. Dessa perspectiva, a meta(heurística) atua como o mestre ao definir o tamanho e os limites da vizinhança e por controlar as repetidas chamadas do método exato, o qual, por sua vez, atua explorando cada vizinhança de modo exato (CASERTA; VOβ, 2009).

Já na abordagem do tipo 2, o esquema metaheurístico está incorporado dentro do *solver*. Um exemplo são os *Solvers Branch and Cut* modernos que exploram os potenciais das (meta)heurísticas que rapidamente fornecem boas soluções, especialmente nos estágios iniciais da árvore de exploração. Os limites (*bounds*) relacionados são empregados para podar ramos (*branchs*) da árvore e, consequentemente, contribuir na rapidez do processo de pesquisa e reduzir os esforços computacionais totais. Desde que foi observado que em alguns importantes casos práticos os MIP *solvers* gastavam uma grande quantidade de tempo computacional antes de encontrar a primeira solução factível, foi introduzido um esquema heurístico chamado *feasibility pump*, que objetivou encontrar rapidamente soluções iniciais de boa qualidade (CASERTA; VOβ, 2009). Sendo assim, esses esquemas heurísticos podem ser usados para encontrar rapidamente soluções iniciais que alimentam os algoritmos híbridos do tipo 1.

Em Puchinger e Raidl (2005), os autores fornecem uma classificação do método matheurístico em duas classes. A primeira trata de combinações colaborativas, na qual nenhum algoritmo está contido em outro, apenas trocam informações. A segunda se refere a combinações integrativas, na qual ou o algoritmo exato está incorporado nas metaheurísticas ou as metaheurísticas estão incorporadas nos algoritmos exatos.

Além disso, a primeira classe pode ser executada sequencialmente ou paralelamente. A implementação sequencial é a mais usada e executa o método exato como um tipo de pré-processamento antes da metaheurística, ou vice-versa. A implementação paralela, por sua vez, é pouco usada e quando empregada, geralmente, é implementada como pequenos *clusters*.

Para Jourdan, Basseur e Talbi (2009), a hibridização envolve dois componentes principais: o desenho e a implementação. O desenho se refere ao próprio algoritmo híbrido, envolvendo questões como funcionalidade e arquitetura. A implementação leva em consideração a plataforma *hardware*, o modelo do programa e o ambiente.

Em relação ao desenho, ele pode ser classificado em dois tipos:

- Baixo nível/Alto nível

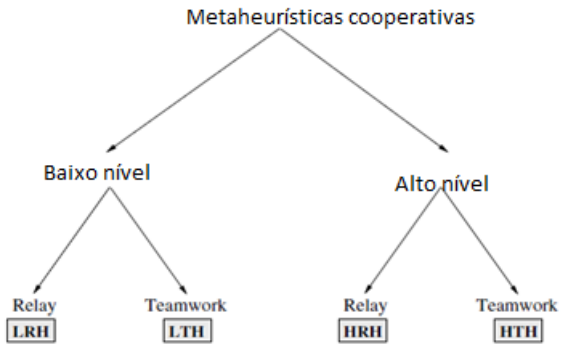
- *Relay/Teamwork*

No desenho de baixo nível tem-se a composição funcional de um único método de otimização e uma dada função de uma metaheurística é substituída por outro método. Já no de alto nível, diferentes algoritmos estão autocontidos.

No desenho da metaheurística *relay* um conjunto de métodos é aplicado um após o outro, cada um usando o resultado do anterior como *input*, atuando num modo *pipeline*. O desenho *teamwork* representa modelos de otimização hibridizados.

Da classificação acima se derivam quatro classes, conforme a Figura 11: a LRH (*Low-level Relay Hybrid*), a LTH (*Low-level Teamwork Hybrid*), a HRH (*High-level Relay Hybrid*) e a HTH (*High-level Teamwork Hybrid*).

Figura 11 - As quatro classes das metaheurísticas cooperativas



Fonte: Jourdan, Basseur e Talbi (2009, p. 622)

A classe LRH corresponde aos algoritmos em que um dado método está incorporado em outro método e o método incorporado tem que ser executado sequencialmente. Na hibridização entre métodos exatos e métodos heurísticos, a abordagem mais natural é desenhar uma heurística pra melhorar a estratégia de pesquisa do método exato.

Na classe LTH, um elemento de um dado método é substituído por outro método. Ao contrário das hibridizações LRH, a LTH consiste num método incorporado que pode ser executado em paralelo com o método global. Os dois principais tipos de abordagens são:

- Hibridização da pesquisa exata LTH: a abordagem exata constrói soluções parciais,

que são usadas para definir um espaço de pesquisa para a abordagem heurística. Em seguida, os resultados obtidos pela heurística são analisados para refinar os limites, ou colunas para gerar um algoritmo *branch and cut*.

- Hibridização de pesquisa heurística LTH: a pesquisa heurística trabalha como os algoritmos meméticos, mas neste caso, o operador genético é substituído por uma pesquisa exata dentro de um subespaço do espaço de pesquisa global.

De acordo com Jourdan, Basseur e Talbi (2009), na classe HRH os diferentes métodos estão autocontidos e são executados em sequência. Em geral, a abordagem mais natural é desenhar a execução sequencial de uma metaheurística que é lançada antes de uma abordagem exata. A metaheurística é desenhada para dar informação ao algoritmo exato. Numa pesquisa exata, a informação dada poderia ser os limites iniciais, por exemplo, o que ajuda o algoritmo exato a acelerar a pesquisa.

Já a classe HTH contém algoritmos onde métodos autocontidos estão executando uma pesquisa de maneira paralela e híbrida. Essa hibridização envolve principalmente modelos de ilhas paralelas. Tem-se dois tipos diferentes de ilhas, aquelas que são dedicadas à pesquisa exata e aquelas dedicadas à pesquisa heurística. Durante a execução, os diferentes algoritmos trocam informações, que são dependentes do tipo de ilha.

Dentre os problemas *NP-hard* pesquisados na literatura, alguns que já adotaram a técnica matheurística foram os seguintes: Problema de Desenho de Rede *Multicommodity* Capacitada Balanceada (DBCMND, em inglês *Design-Balanced Capacitated Multicommodity Network Design Problem*), Problema de *P-median* Capacitado (CPMP, em inglês *Capacitated P-median Problem*), *Set Covering Problem*, Problema Linear de Desenho de Rede de Transporte Marítimo (LSNDP, em inglês *Liner Shipping Network Design Problem*), Problema de Roteamento de Veículos (VRP, em inglês *Vechicle Routing Problem*), Problema de Dimensionamento de Lotes Capacitado Multi-produto e Multi-nível (MLCLSP, em inglês *Multi-level, Multi-item Capacitated Lot-Sizing Problem*) e Problema de Desconto de Quantidade Total (TQDP, em inglês *Total Quantity Discount Problem*).

4 O MODELO PARA O SUALBSP

Neste capítulo é apresentado o modelo matemático alternativo do SUALBSP proposto neste trabalho. Na Seção 4.1, o problema é definido. A Seção 4.2 explana sobre o modelo matheurístico desenvolvido, apresentando também o modelo matemático elaborado, juntamente com seus conjuntos de dados, parâmetros, variáveis de decisão e auxiliares, função objetivo e as suas restrições. Na Seção 4.3 é realizada uma comparação entre o modelo matemático de Scholl, Boysen e Fliedner e o modelo matemático proposto.

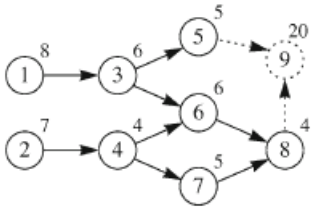
4.1 DEFINIÇÃO DO PROBLEMA

Para que se compreenda corretamente o presente trabalho é fundamental que o SUALBSP seja explicado detalhadamente.

O SUALBSP aplicado a este trabalho consiste em minimizar o tempo total de estação, dado um número de estações de trabalho. Para tanto, cada tarefa deve ser atribuída a exatamente uma estação de trabalho, as restrições de precedência devem ser observadas e o tempo de ciclo não pode ser excedido em nenhuma estação ao considerar a execução cíclica da tarefa com tempos de *setup forward* e *backward* dependentes da sequência.

Para resolver o SUALBSP alguns dados são necessários, como: as relações de precedência entre as tarefas, os tempos de processamento de cada tarefa e os tempos de *setup forward* e *backward*. Os dados referentes às relações de precedência e aos tempos de processamento das tarefas podem ser visualizados, por exemplo, por meio de um grafo de precedência, vide Figura 12.

Figura 12 - Grafo de precedência SUALBSP



Fonte: Scholl, Boysen e Fliedner (2013, p. 297)

Na Figura 12, os pesos nos nós denotam os tempos de processamento das tarefas. Por exemplo, a tarefa 1 possui um tempo de processamento de 8. Já os tempos de *setup backward* e *forward* podem ser visualizados por meio de uma matriz de tempo de *setup*, conforme a Figura 13.

Figura 13 - Matrizes de tempo *setup forward* e *backward*

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

Fonte: Scholl, Boysen e Fliedner (2013, p. 297)

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.

Outros dados necessários para a definição do problema são o tempo de ciclo e o número de estações.

4.2 MODELO MATHEURÍSTICO

Para solucionar o SUALBSP, propôs-se uma matheurística que consiste em uma heurística combinada com o uso de um resolvedor de programação linear exata (*MIP solver*). Enquanto o modelo matemático de Scholl, Boysen e Fliedner (2013), apresentado na Seção 2.5.1, 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, Boysen e Fliedner (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 findado assim que o modelo apresentar uma solução factível. Este algoritmo foi chamado de *math*-SUALBSP.

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← "Infactível"
enquanto *status* = "*Infactível*" **faça**
  executa o modelo SUALBSP-with-fixed-stations
  se *modelo é factível* **então**
    status ← "Factível"
  senão
    status ← "Infactível"
    $k \leftarrow k + 1$
  fim se
fim enquanto
retorna vetores de solução

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

$$k = \left\lceil \frac{t_{sum}}{c} \right\rceil \qquad \text{onde } t_{sum} = \sum_{j \in V} t_j \tag{17}$$

Onde:

k = número de estações de trabalho;

t_{sum} = somatório do tempo de processamento/operação de todas as tarefas;

c = tempo de ciclo;

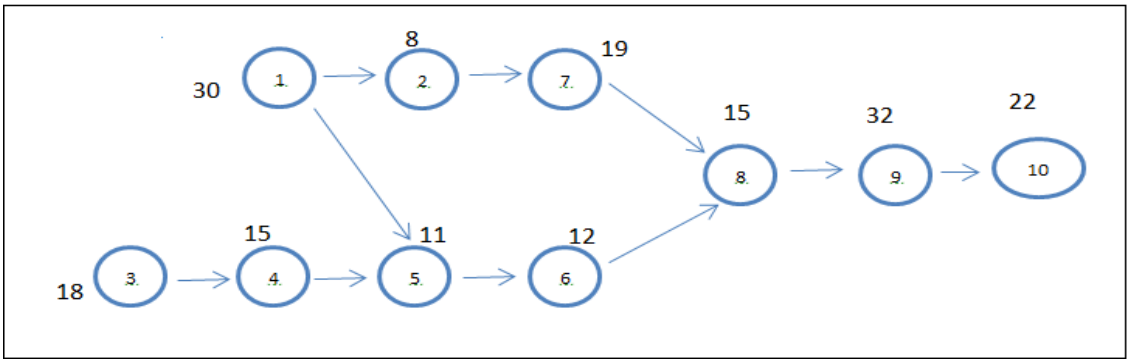
t_j = tempo de processamento/operação da tarefa j;

j = tarefa j;

V = conjunto de tarefas.

Esse *lower bound* é uma adaptação do *lower bound* mais usual no SALBP. Apenas para exemplificar seu cálculo: tem-se um grafo de precedência na Figura 14, no qual os tempos de processamento/operação de cada tarefa (10 tarefas) estão ao lado dos círculos e o tempo de ciclo é de 71. Sendo assim, o *lower bound* deste exemplo é 3, pois $k = \left\lceil \frac{18+15+11+12+15+32+22+30+8+19}{71} \right\rceil = \left\lceil \frac{182}{71} \right\rceil = \lceil 2{,}56 \rceil = 3$.

Figura 14 - Grafo de precedência exemplo *lower bound*



Fonte: Elaborada pela autora

O modelo matemático SUALBSP-*with-fixed-stations* adota os mesmos pressupostos básicos de Andrés, Miralles e Pastor (2008) e de Scholl, Boysen e Fliedner (2013), quais sejam:

1- os tempos de processamento das tarefas, a matriz de tempos de *setup* e as relações de precedência são conhecidos deterministicamente;

2- os tempos de processamento e de *setup* são independentes das estações de trabalho nas quais as tarefas são processadas;

3- um único modelo de um produto é montado na linha;

4- trata-se de uma linha em série compassada onde os *buffers* não são considerados;

5- cada tarefa é atribuída a somente uma estação de trabalho;

6- as tarefas devem ser processadas somente uma vez;

7- todas as estações de trabalho são igualmente equipadas e qualquer tarefa pode ser atribuída a qualquer estação de trabalho;

8- nenhum dos tempos de processamento de tarefas pode ser maior que o tempo de ciclo; e

9- as estações de trabalho podem processar somente um produto de cada vez.

Para uma melhor compreensão do modelo matemático SUALBSP-*with-fixed-stations* apresenta-se os seus respectivos conjuntos de dados, parâmetros, variáveis de decisão e variáveis auxiliares, função objetivo e restrições.

Conjuntos de dados:

Os conjuntos de dados são as entidades envolvidas no processo de balanceamento e sequenciamento da linha de montagem com tempos de *setup* dependentes da sequência. O modelo proposto tem seis conjuntos, conforme apresentado a seguir.

V = conjunto de tarefas, onde $V = \{1, \dots, n\}$ (C1)

E = conjunto de pares de tarefas (i, j) com precedência direta (C2)

K = conjunto de possíveis estações de trabalho, onde $K = \{1, 2, \dots, \overline{m}\}$. No modelo, $\overline{m}$ é o *upper bound* do número de estações. Seu valor pode ser, por exemplo, $\overline{m} = n$ ou o valor de uma solução heurística. (C3)

FS_i = conjunto de estações nas quais a tarefa $i \in V$ é factível de atribuição, onde $FS_i \subset K$ (C4)

F_i^{τ} $(P_i^{\tau}) =$ conjunto de tarefas que podem diretamente suceder (preceder) a tarefa $i \in V$ na carga da estação na direção *forward* (C5)

F_i^{μ} $\left(P_i^{\mu}\right)$ = conjunto de tarefas que podem diretamente suceder (preceder) a tarefa $i \in V$ na carga da estação na direção *backward* (C6)

Parâmetros:

Os parâmetros são valores fixos, responsáveis por traduzirem para o modelo a forma de relacionamento entre os conjuntos de dados, além de definirem os valores das restrições.

t_i = tempo de execução da tarefa $i \in V$, onde $t_{sum} = \sum_{i \in V} t_i$ (P1)

τ_{ij} = tempo de *setup forward* da tarefa *i* a *j* (P2)

$\mu_{ij} =$ tempo de *setup backward* da tarefa *i* a *j* (P3)

M = número suficientemente grande, por exemplo, $M = n \times c$ (P4)

c = tempo de ciclo (P5)

Variáveis de decisão:

As variáveis de decisão, por sua vez, são as incógnitas a serem determinadas pela solução do modelo. No caso do SUALBSP-*with-fixed-stations*, os valores das variáveis de decisão apontam o tempo total de estação resultante. Observa-se que no modelo aqui proposto adicionou-se a dimensão *k* nas variáveis de decisão y_{ij} e w_{ij}, o que corresponde ao número de estações de trabalho.

$$x_{ik} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é atribuída à estação } k \in FS_i \\ 0, \text{caso contrário} \end{cases} \quad \text{(V1)}$$

$$y_{ijk} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é predecessora direta de } j \in F_i^{\tau} \text{ na carga da estação } k \\ 0, \text{caso contrário} \end{cases} \quad \text{(V2)}$$

$$w_{ijk} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é a última e } j \in F_i^{\mu} \text{ é a primeira tarefa na carga da estação } k \\ 0, \text{caso contrário} \end{cases} \quad \text{(V3)}$$

T_k = variável contínua de tempo da estação $k \in K$ (V4)

Variáveis auxiliares:

A variável auxiliar também é uma variável de decisão. No entanto, chama-se de variável auxiliar, pois foi criada para apoiar o cálculo de alguma variável de decisão.

z_i = variável contínua para codificação do número da estação à qual a tarefa $i \in V$ é atribuída (A1)

f_i = variável contínua de tempo de início da tarefa $i \in V$ (relativo ao tempo de lançamento da primeira estação) (A2)

Função objetivo:

A função objetivo ou de avaliação se constitui em uma função matemática que define a qualidade da solução obtida, em função dos valores das variáveis de decisão. A função objetivo do modelo proposto neste trabalho é do tipo minimizante. Logo, a redução do valor da função objetivo significa um aumento de qualidade do balanceamento e sequenciamento da linha de montagem com tempos de *setup* dependentes da sequência.

A expressão da função objetivo deste modelo é dada por:

$$\text{Minimizar} \sum_{k \in K} T_k \quad (18)$$

A função objetivo proposta (18) visa obter tempos de estação mínimos. Observa-se que este era o segundo objetivo do modelo de Scholl, Boysen e Fliedner (2013). Primeiramente, esses autores desejavam minimizar o número de estações. No modelo proposto, o número de estações já é definido por meio do cálculo do *lower bound* e isso contribui significativamente no menor tempo de processamento do modelo proposto. Esta foi

a primeira alteração empreendida por este trabalho ao modelo de Scholl, Boysen e Fliedner (2013).

Restrições:

O modelo SUALBSP-*with-fixed-stations* é composto por dezoito restrições. As restrições do modelo proposto se configuram em uma série de equações e inequações lineares, as quais limitam as variáveis de decisão a seus valores possíveis ou viáveis.

$$\sum_{k \in K} x_{ik} = 1 \qquad \forall i \in V \quad (19)$$

A restrição (19) impõe a atribuição de cada tarefa i a exatamente uma estação $k \in FS_i$. Essa restrição cumpre um dos pressupostos do modelo (pressuposto 5), o qual prediz que cada tarefa deve ser atribuída a somente uma estação de trabalho. Exemplificando, se a tarefa $i = 5$ foi atribuída à estação $k = 2$, então a equação (19) tem o valor 1. Mas, o resultado de $i = 5$ e $k = 1$, terá valor 0 na equação (19), pois a tarefa $i = 5$ já foi atribuída à estação $k = 2$ e, portanto, ela não pode ser atribuída à outra estação.

$$z_i = \sum_{k \in K} k \times x_{ik} \qquad \forall i \in V \quad (20)$$

A restrição (20) serve apenas para transformar a restrição (19) em um índice de estação. Atenta-se para o fato que a variável z_i pode ser removida do modelo ao substituí-la nas restrições em que é utilizada via restrição (20). Por exemplo, a tarefa $i = 5$ foi atribuída à estação $k = 2$, então o índice de estação z_5 será igual a 2, pois, $k = 2$ e $x_{52} = 1$. Basicamente, a restrição (20) mostra para qual estação a tarefa i foi atribuída.

$$x_{ik} \geq \sum_{j \in V} y_{ijk} \; e \; x_{ik} \geq \sum_{j \in V} y_{jik} \qquad \forall k \in K, i \in V \quad (21)$$

As restrições (21) e (22) são adicionais ao modelo de Scholl, Boysen e Fliedner (2013).

Resumidamente, as restrições (21) e (22) impõem que uma determinada tarefa deve estar alocada em uma estação k e deve participar de alguma transição (*forward* ou *backward*). A restrição (21) garante que a tarefa i seja alocada à estação k no sentido *forward*, sendo que a segunda parte da restrição garante a atribuição da tarefa i à estação k quando essa mesma tarefa i é a última do ciclo.

Para exemplificar a restrição (21): supondo-se que a tarefa $i = 5$ e que existam três estações de trabalho $(k = \{1, 2, 3\})$, uma das variáveis x_{ik} deve ser x_{51}, x_{52} ou x_{53}. A

variável y_{ijk} é 1 quando há uma transição de tarefas entre a tarefa i e a tarefa j no sentido *forward*. Caso as estações de trabalho estivessem sequenciadas da seguinte maneira: $k = 1: 1 - 3 - 5 - 7$ e $k = 2: 2 - 4 - 6 - 8$, a variável y_{ijk} para a tarefa 5 seria y_{571}, pois ela sai da tarefa 5 para a tarefa 7 na estação de trabalho 1. Sendo assim, a primeira parte da restrição (20) ficaria assim: $x_{51} \geq y_{571}$, ou seja $1 \geq 1$, o que cumpriria a primeira parte da restrição (20).

Agora, supondo-se que para este mesmo exemplo deseja-se saber para qual estação de trabalho a tarefa 8 foi atribuída. Neste caso, como a tarefa 8 é a última do ciclo, não há variável y_{i8k} = 1 em nenhum caso. Mas, há variável y_{8ik}= 1. Logo, a segunda parte da restrição (21) garante a atribuição da tarefa i à estação k quando a tarefa é a última do ciclo, e esta segunda parte da equação ficaria $x_{82} \geq y_{862}$, ou seja $1 \geq 1$.

$$x_{ik} \geq \sum_{j \in V} w_{ijk} \; e \; x_{ik} \geq \sum_{j \in V} w_{jik} \qquad \forall k \in K, i \in V \quad (22)$$

Por outro lado, existem casos nos quais a tarefa i é a única na estação k. Portanto, ela não faz transição com nenhuma outra tarefa. Logo, a restrição (22) se aplica a esses casos e garante que x_{ik} seja 1 quando não há variável y_{ijk} disponível. Sabendo-se que a variável w_{ijk} é 1 quando há uma transição de tarefas entre a tarefa i e a tarefa j no sentido *backward* e supondo-se que a restrição (22) apresente estação $k = 2$ e esteja configurada como $k = 2: 9$, conclui-se que a configuração $k = 2: 9$ significa que a tarefa 9 é a única na estação 2. Não existe uma variável y_{ij2} = 1. Ela é 0, pois a tarefa 9 não faz transição com nenhuma outra. Contudo, a variável w_{992} é igual a 1. Sendo assim, a restrição (22) garante que x_{ik} seja 1 quando não há variável y_{ijk} disponível.

Como se pode observar existe redundância de restrições nas equações (21) e (22). Apenas a primeira parte das restrições (21) e (22) ($x_{ik} \geq \sum_{j \in V} y_{ijk}$ e$x_{ik} \geq \sum_{j \in V} w_{ijk}$) já seria o suficiente. Entretanto, existem situações nas quais colocar restrições redundantes ajudam o modelo a ser resolvido mais rapidamente. No caso do modelo proposto, as restrições redundantes ajudaram a melhorar o tempo de processamento e, por isso, foram mantidas.

$$\sum_{k \in K} \sum_{j \in F_i^{\tau}} y_{ijk} + \sum_{k \in K} \sum_{j \in F_i^{\mu}} w_{ijk} = 1 \qquad \forall i \in V \quad (23)$$

A restrição (23) garante que a tarefa i seja sucedida por outra tarefa j na direção *backward* ou *forward* na estação k.

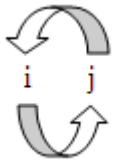
$$\sum_{k \in K} \sum_{i \in P_i^{\tau}} y_{ijk} + \sum_{k \in K} \sum_{i \in P_i^{\mu}} w_{ijk} = 1 \qquad \forall j \in V \quad (24)$$

De outra forma, a restrição (24) garante que a tarefa i seja precedida por outra tarefa j na direção *backward* ou *forward* na estação k.

Conjuntamente, as restrições (23) e (24) garantem que cada tarefa i seja sucedida e precedida por exatamente uma outra tarefa j formando uma sequência cíclica da carga da estação k.

Caso $\sum_{k \in K} \sum_{j \in F_i^{\tau}} y_{ijk}$ seja igual a 1 na restrição (23), o $\sum_{k \in K} \sum_{j \in F_i^{\mu}} w_{ijk}$ desta mesma restrição será 0, mas $\sum_{k \in K} \sum_{i \in P_i^{\mu}} w_{ijk}$ da restrição (24) será 1, pois assim a tarefa i será sucedida por uma tarefa j na direção *forward* e precedida por uma tarefa j na direção *backward*, formando uma sequência cíclica na estação k, conforme a Figura 15.

Figura 15 - Sucessor *forward* e predecessor *backward*

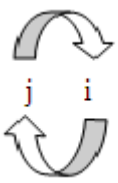


Fonte: Elaborada pela autora

Agora, caso $\sum_{k \in K} \sum_{j \in F_i^{\mu}} w_{ijk}$ seja igual a 1 na restrição (23), $\sum_{k \in K} \sum_{j \in F_i^{\tau}} y_{ijk}$ será 0.

Entretanto, $\sum_{k \in K} \sum_{i \in P_i^{\tau}} y_{ijk}$ na restrição (24) será 1. Assim, a tarefa i será sucedida por uma tarefa j na direção *backward* e precedida por uma tarefa j na direção *forward*, formando uma sequência cíclica na estação k, de acordo com a Figura 16.

Figura 16 - Sucessor *backward* e predecessor *forward*



Fonte: Elaborada pela autora

$$\sum_{i \in V} \sum_{j \in V} w_{ijk} = 1 \qquad \forall k \in K \quad (25)$$

Uma das restrições adicionais ao modelo original de Scholl, Boysen e Fliedner (2013) é a restrição (25). Essa restrição garante que uma estação $k \in K$ esteja associada, obrigatoriamente, a uma sequência de tarefas ($i \in V, j \in V$).

$$\sum_{k \in K} y_{ijk} \leq 1 \qquad \forall i \in V, j \in V \quad (26)$$

Já a restrição (26) restringe a sequência de tarefas $i \in V$ para $j \in V$ a no máximo uma estação k. Ela também é uma restrição adicional ao modelo de Scholl, Boysen e Fliedner (2013). Qualquer valor superior a 1, indicaria que a sequência de tarefas $i \in V$ para $j \in V$ estaria presente em mais de uma estação de trabalho, o que viola os pressupostos do modelo. Sendo assim, a restrição (26) assegura que a sequência de tarefas $i \in V$ para $j \in V$ não seja alocada a mais de uma estação k.

$$\sum_{k \in K} \sum_{i \in V} y_{iik} = 0 \qquad (27)$$

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 (27). Essa restrição também se constitui em uma restrição adicional ao modelo original de Scholl, Boysen e Fliedner (2013).

$$f_j \geq f_i + t_i \qquad \forall (i,j) \in E \quad (28)$$

As relações de precedência entre as tarefas são garantidas por meio da restrição (28). Essa restrição garante que o tempo de início da tarefa j (f_j) deve ser maior ou igual ao tempo de início tarefa i (f_i) mais o tempo de execução desta tarefa (t_i). Assim, garante-se que a tarefa j iniciará no momento em que a tarefa i se encerra ou a tarefa j iniciará algum tempo depois da tarefa i, mas a tarefa j só começará após o término da tarefa i.

$$f_j \geq f_i + t_i + \tau_{ij} + M \times (y_{ijk} - 1) \qquad \forall k \in K, i \in V, j \in F_i^{\tau} \quad (29)$$

Na restrição (29), garante-se a ordem *forward* das tarefas na carga da estação k. Nessa restrição, somente quando $y_{ijk} = 1$ obtém-se a ordem *forward* das tarefas na estação k, culminando com um valor de M igual a 0. Caso $y_{ijk} = 0$, o tempo de início da tarefa j (f_j) será menor do que o somatório dos tempos de início da tarefa i (f_i), tempo de execução da tarefa i (t_i), tempo de *setup forward* da tarefa i a j (τ_{ij}) e M, demonstrando que as tarefas não seguem uma ordem *forward* na estação k.

$$f_i \geq c \times (z_i - 1) \qquad \forall i \in V \quad (30)$$

A restrição (30) obriga que cada tarefa i inicie no tempo $c \times (z_i - 1)$ atribuído a estação z_i. Supondo-se que o tempo de ciclo (c) seja 30 e que o índice z_i seja equivalente a 2, uma vez que a tarefa i =1 foi atribuída à estação $k = 2$, o tempo de início da tarefa 1 (f_i) deve ser maior ou igual a 30, para que a linha de montagem mantenha um ritmo.

$$f_i + t_i + \mu_{ij} \leq c \times z_i + M \times (1 - w_{ijk}) \qquad \forall k \in K, i \in V, j \in F_i^{\mu} \quad (31)$$

Já a restrição (31) garante que cada tarefa i termine no tempo $c \times z_i$, não ultrapassando o tempo que lhe foi determinado, ou seja, ela garante que o *setup backward* final (indicado por $w_{ijk} = 1$) será finalizado dentro do intervalo de tempo determinado. A violação desta restrição implicaria em atraso nas estações subsequentes.

Entende-se que as restrições (30) e (31) determinam que a tarefa i seja completamente executada no intervalo de tempo $(c \times (z_i - 1), c \times z_i)$ atribuído à estação z_i. Esse intervalo de tempo contempla, inclusive, desde o momento inicial em que o produto foi lançado na linha.

$$T_k + c \times (z_i - 1) \geq f_i + t_i + \sum_{j \in F_i^{\mu}} \mu_{ij} \times w_{ij} - M \times (1 - x_{ik}) \qquad \forall i \in V, k \in FS_i \quad (32)$$

O tempo da estação T_k é calculado pela restrição (32). Ou seja, T_k calcula o tempo de duração de todo o ciclo completo de uma estação k.

$$x_{ik} \in \{0, 1\} \qquad \forall i \in V, k \in K \quad (33)$$

$$y_{ijk} \in \{0, 1\} \qquad \forall i \in V, j \in F_i^{\tau}, k \in K \quad (34)$$

$$w_{ijk} \in \{0, 1\} \qquad \forall i \in V, j \in F_i^{\mu}, k \in K \quad (35)$$

As restrições (33), (34) e (35) são de integralidade.

$$f_i, z_i, T_k \geq 0 \qquad \forall i \in V, k \in K \quad (36)$$

Já a restrição (36) se constitui em uma restrição de não-negatividade.

4.3 COMPARAÇÃO ENTRE O MODELO MATEMÁTICO DE SCHOLL, BOYSEN E FLIEDNER (2013) E O MODELO MATHEURÍSTICO

Primeiramente, apresenta-se a seguir o modelo matemático elaborado por Scholl, Boysen e Fliedner (2013). Este modelo é composto pelos seguintes conjuntos, variáveis de decisão variáveis auxiliares, parâmetros, função objetivo e restrições.

Conjuntos:

V = conjunto de tarefas, onde $V = \{1, \dots, n\}$

E = conjunto de pares de tarefas (i, j) com precedência direta

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

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

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

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

Variáveis de decisão:

$$x_{ik} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é atribuída à estação } k \in FS_i \\ 0, \text{caso contrário} \end{cases}$$

$$y_{ij} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é predecessora direta de } j \in F_i^{\tau} \text{ na carga da estação} \\ 0, \text{caso contrário} \end{cases}$$

$$w_{ij} = \begin{cases} 1, \text{se a tarefa } i \in V \text{ é a última e } j \in F_i^{\mu} \text{ é a primeira tarefa na carga da estação} \\ 0, \text{caso contrário} \end{cases}$$

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

Variáveis auxiliares:

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)

Parâmetros:

t_i = tempo de execução da tarefa $i \in V$, onde $t_{sum} = \sum_{i \in V} t_i$

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

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

M = número suficientemente grande

ε = número suficientemente pequeno

c = tempo de ciclo

A expressão da função objetivo deste modelo é dada por:

$$Min\ m(x, y, z) = z_n + \varepsilon \times \sum_{k \in K} T_k \tag{37}$$

A função objetivo (37) minimiza o índice da estação para o qual o único nó sorvedouro n é atribuído e, assim, minimiza o número total de estações de trabalho. Como um objetivo secundário os valores das variáveis T_k são minimizados para obter tempos de estações com *setups* mínimos. Devido ao ε ser suficientemente pequeno, o objetivo secundário não influencia na minimização do número de estações.

As restrições do modelo são as seguintes:

$$\sum_{k \in FS_i} x_{ik} = 1 \qquad \forall i \in V \tag{38}$$

A restrição (38) garante a atribuição de cada tarefa i a exatamente uma estação $k \in FS_i$.

$$z_i = \sum_{k \in FS_i} k \times x_{ik} \qquad \forall i \in V \tag{39}$$

A restrição (38) é transformada no correspondente índice de estação z_i através da restrição (39).

$$\sum_{j \in F_i^{\tau}} y_{ij} + \sum_{j \in F_i^{\mu}} w_{ij} = 1 \qquad \forall i \in V \tag{40}$$

A restrição (40) garante que a tarefa i seja sucedida por uma outra tarefa j na direção *backward* ou *forward* na carga da estação.

$$\sum_{i \in P_j^{\tau}} y_{ij} + \sum_{i \in P_j^{\mu}} w_{ij} = 1 \qquad \forall j \in V \tag{41}$$

De outra forma, a restrição (41) garante que a tarefa i seja precedida por uma outra tarefa j na direção *backward* ou *forward* na carga da estação.

Os conjuntos de restrições (40) e (41) garantem que cada tarefa i seja seguida e precedida por exatamente uma outra tarefa j numa sequência cíclica de uma carga de estação.

$$\sum_{i \in V} \sum_{j \in F_i^{\mu}} w_{ij} = z_n \quad (42)$$

Em combinação com a restrição (42), as restrições (40) e (41) demandam que em cada ciclo exatamente uma das relações seja um *setup backward*.

$$z_i + M \times (1 - y_{ij}) \geq z_j e\ z_j + M \times (1 - y_{ij}) \geq z_i \qquad \forall i \in V, j \in F_i^{\tau} \quad (43)$$

$$z_i + M \times (1 - w_{ij}) \geq z_j\ e\ z_j + M \times (1 - w_{ij}) \geq z_i \qquad \forall i \in V, j \in F_i^{\mu} \quad (44)$$

As tarefas contidas no mesmo ciclo são forçadas a ser atribuídas à mesma estação pelas restrições (43) e (44), as quais devem ser cumpridas em pares como equações $(z_i = z_j)$, se $y_{ij} = 1$ e $w_{ij} = 1$, respectivamente.

$$f_j \geq f_i + t_i \qquad \forall (i,j) \in E \quad (45)$$

As relações de precedência são cumpridas através da restrição (45).

$$f_j \geq f_i + t_i + \tau_{ij} + M \times (y_{ij} - 1) \qquad \forall i \in V, \in F_i^{\tau} \quad (46)$$

A ordem *forward* das tarefas numa carga da estação é garantida pela restrição (46).

$$f_i \geq c \times (z_i - 1) \qquad \forall i \in V \quad (47)$$

A restrição (47) obriga que cada tarefa i inicie no tempo $c \times (z_i - 1)$ atribuído a estação z_i.

$$f_i + t_i + \mu_{ij} \leq c \times z_i + M \times (1 - w_{ij}) \qquad \forall i \in V, j \in F_i^{\mu} \quad (48)$$

A restrição (48) garante que o *setup backward* final (indicado pelo $w_{ij} = 1$) será finalizado em tempo.

$$T_k + c \times (z_i - 1) \geq f_i + t_i + \sum_{j \in F_i^{\mu}} \mu_{ij} \times w_{ij} - M \times (1 - x_{ik}) \qquad \forall i \in V, k \in FS_i \quad (49)$$

O tempo da estação T_k é calculado pela restrição (49).

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, k \in FS_i \quad (50)$$

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

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

A restrição (50) é de integralidade.

$$f_i, z_i, T_k \geq 0 \qquad \forall i \in V, k \in K \quad (51)$$

As variáveis são definidas também pela restrição (51), que é uma restrição de não-negatividade.

Como se pode observar, a função objetivo do modelo proposto (18) objetiva minimizar apenas uma variável de decisão, que é o tempo de estação (T_k), enquanto que a função objetivo do modelo de Scholl, Boysen e Fliedner (2013) (37) visa minimizar o número de estações (z_i) e o tempo de estação (T_k). No modelo proposto, o número de estações é definido de maneira exógena por meio do cálculo do *lower bound*. Esta estratégia adotada contribuiu decisivamente na diminuição do tempo de processamento do modelo proposto.

Em virtude do número de estações ser definido exogenamente no modelo proposto, a restrição (42) não se faz mais necessária e foi retirada do modelo matheurístico, o que melhorou o tempo computacional. Além disso, também foram desconsideradas as restrições (43) e (44).

As restrições (21), (22), (25), (26) e (27) são adicionais ao modelo de Scholl, Boysen e Fliedner (2013) e, de alguma maneira, se relacionam com alguma estação de trabalho k. Pode-se dizer que essas restrições adicionais ao modelo de Scholl, Boysen e Fliedner (2013) também auxiliaram no desempenho do modelo proposto.

5 EXPERIMENTOS COMPUTACIONAIS

Este capítulo aborda os experimentos computacionais realizados no presente estudo. Os experimentos foram executados em 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. Os experimentos 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*.

Para verificar o modelo proposto neste trabalho, tomaram-se algumas instâncias do conjunto de dados SBF2 disponíveis no *website* do *Assembly Line Balancing Research Group* (http://alb.mansci.de/). Os resultados obtidos, comparados com os de Scholl, Boysen e Fliedner (2013), permitem verificar o modelo proposto.

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

Na Seção 5.1 é explicado como o conjunto de dados foi selecionado. Na Seção 5.2 são apresentados os resultados computacionais dos experimentos realizados, comparando-se com os resultados do modelo de PLI de Scholl, Boysen e Fliedner (2013).

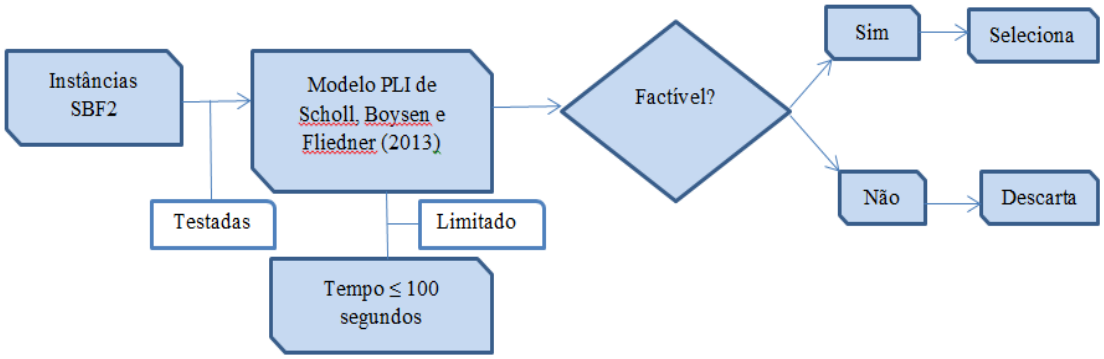
5.1 CONJUNTO DE DADOS

Os dados das instâncias utilizadas neste trabalho estão disponíveis no *website* do *Assembly Line Balancing Research Group* (http://alb.mansci.de/). Foram selecionadas 101 instâncias pertencentes ao conjunto de dados SBF2. Destaca-se que esse conjunto de dados possui um total de 1.076 instâncias.

A seleção das 101 instâncias obedeceu ao seguinte critério: todas as instâncias de dados SBF2 foram testadas no modelo PLI de Scholl, Boysen e Fliedner (2013), que foram processadas no MIP *solver* IBM© ILOG CPLEX Optimization Studio V12.5.1 com as configurações *default.* O processamento foi limitado em 100 segundos, a exemplo do trabalho

de Scholl, Boysen e Fliedner (2013). Após os 100 segundos, 101 instâncias apresentaram resultado factível. Então, elas foram selecionadas para também serem processadas no modelo proposto neste trabalho no MIP *solver* IBM© ILOG CPLEX Optimization Studio V12.5.1, limitado igualmente a 100 segundos. Dessa forma, tornou-se possível comparar os resultados dos dois modelos aplicados a uma mesma base de dados. O critério adotado está esquematizado na Figura 17.

Figura 17 - Esquema de seleção das instâncias



Fonte: Elaborada pela autora

5.2 ANÁLISE DOS RESULTADOS

Após a seleção das 101 instâncias a serem testadas, executou-se o modelo PLI e o modelo MATH no MIP *solver* limitados a 100 segundos.

Como um dos objetivos deste trabalho é comparar e avaliar os resultados obtidos oriundos da abordagem matheurística proposta com o modelo matemático de PLI de Scholl, Boysen e Fliedner (2013) foram adotadas as seguintes notações e medidas de comparação.

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, Boysen e Fliedner (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, Boysen e Fliedner (2013)

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
MATH	resultados do processamento do modelo matheurístico proposto por este estudo
%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

Na notação #var i e j representam as tarefas e k são as estações de trabalho. Logo, deve-se considerar o número máximo de tarefas e o máximo de estações. O máximo de tarefas já é dado em cada instância e o número máximo de estações é dado pela estimativa de *upper bound* de Scholl, Boysen e Fliedner (2013) que já apresentam esse dado calculado para cada instância. Assim, uma instância que possua 15 tarefas e 12 estações terá #var $=$ 15 x 15 x 12 ($|i|$ x $|j|$ x $|k|$).

Informa-se que, assim como o trabalho de Scholl, Boysen e Fliedner (2013) que avaliou os algoritmos propostos através das médias de algumas medidas, o presente estudo também adotou o mesmo padrão.

Sendo assim, a Tabela 2 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). Os resultados que apresentaram melhor desempenho estão destacados em negrito na tabela.

Tabela 2 - 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**

Fonte: Elaborada pela autora.

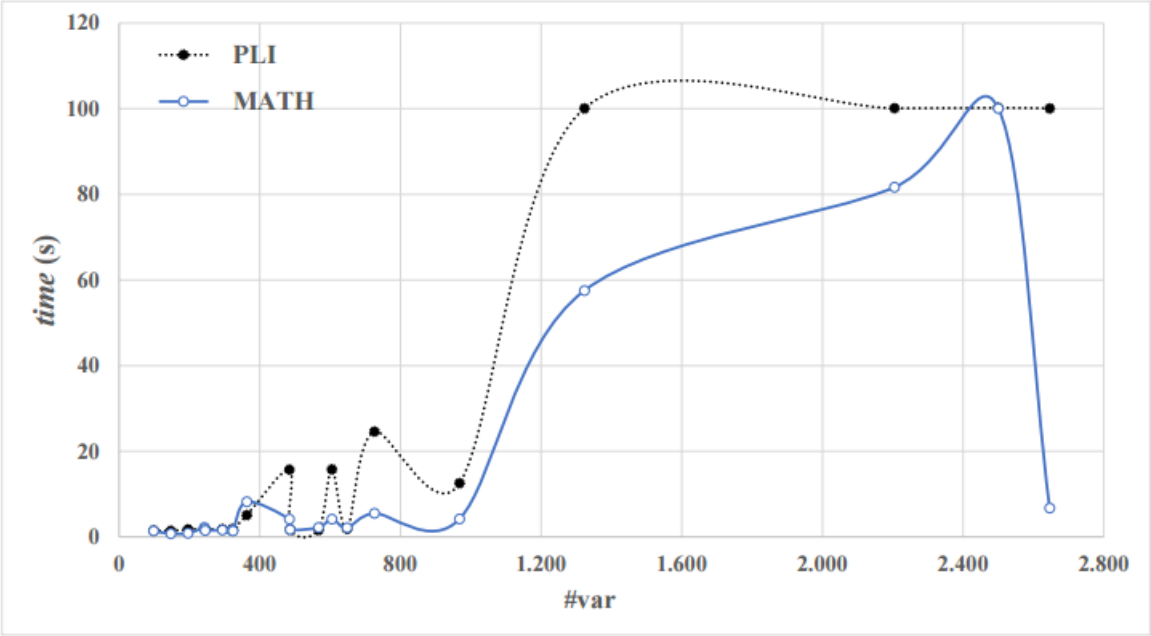
Considerando o tempo total de processamento de todas as instâncias analisadas, observa-se que o modelo MATH foi 44,74% mais rápido que o modelo PLI, o que é de suma importância visto que os planejadores de linhas de montagem tendem a esperar que as soluções propostas sejam computadas quase instantaneamente. Assim sendo, os tempos de computação são de crucial importância em relação à aceitabilidade de uma ferramenta de planejamento de linha de montagem.

Em alguns conjuntos de instâncias, como o conjunto 21, o desempenho computacional do modelo MATH é muito superior ao modelo PLI, apresentando um ganho de tempo de processamento de 93,25%. Além disso, dos 21 conjuntos de instâncias testados, em apenas 5 conjuntos o modelo MATH não obteve desempenho computacional superior ao modelo PLI. Esse ganho computacional do modelo MATH em relação ao modelo PLI se deve à simplificação empreendida pelo modelo MATH que não tem a responsabilidade de determinar o número de estações, culminando em um menor tempo de processamento.

Nota-se também que, enquanto o modelo PLI atingiu a otimalidade em 84 instâncias (83,17%), o modelo MATH atingiu em 92 (91,09%). Além disso, esta otimalidade não se deu apenas em um menor tempo de estação T_k, mas resultou em um número menor de estações k. 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 ao modelo PLI.

Observa-se na Figura 18 que à medida que se analisam instâncias com maior número de variáveis de decisão (#var), o modelo MATH apresenta melhor desempenho em relação ao modelo PLI. Presume-se que isso ocorra porque à medida que o número de variáveis de decisão (#var) aumenta, é exigido um maior esforço computacional do MIP *solver*. Enquanto no modelo MATH, o conjunto de estações é dado, no modelo PLI o número de estações é o principal componente que a função objetivo busca minimizar. Logo, o modelo PLI demanda maior esforço computacional do que o modelo MATH.

Figura 18 - Gráfico de confronto de desempenho dos modelos PLI e MATH



Fonte: Elaborada pela autora.

Ainda, verifica-se na Figura 18 que dentre as maiores instâncias (#var ≥ 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, Boysen e Fliedner (2013) foi atingido por ambos, razão pela qual nenhum deles atingiu a otimalidade.

A Figura 18 demonstrou claramente que o modelo proposto no presente trabalho (MATH) atinge resultados iguais ou melhores em um tempo de processamento inferior ao apresentado pelo modelo PLI de Scholl, Boysen e Fliedner (2013).

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvido um modelo matemático alternativo ao proposto por Scholl, Boysen e Fliedner (2013). Os autores relataram que os resultados obtidos, a partir do processamento do modelo no MIP *solver* Fico XPress Optimizer 20.00.05, haviam sido decepcionantes, visto que o SUALBSP apresenta um subproblema *NP-hard* em todas as estações. Esses resultados se referiam, principalmente, ao tempo de processamento das instâncias (cpu), ao número de instâncias que atingiram a otimalidade (#opt) e ao número de instâncias que encontraram melhores soluções (#best). Enquanto o tempo médio de processamento das instâncias na heurística que apresentou melhor desempenho foi de 8,75 segundos, no *solver* Fico XPress esse tempo foi de 97,47 segundos. Além disso, o *solver* Fico XPress encontrava, em média, 183 melhores soluções (#best) a menos que a referida heurística. Salienta-se que o número máximo de soluções era de 269. Já o número de instâncias que atingiram a otimalidade (#opt) no Fico XPress foi cerca de 39,92% menor do que o número da heurística com melhor desempenho.

Assim, a ideia central deste trabalho foi retirar do modelo matemático de Scholl, Boysen e Fliedner (2013) a responsabilidade de determinar o número ótimo de estações de trabalho, presumindo-se que este número fora previamente conhecido. Dessa forma, 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 MIP *solver* IBM© ILOG CPLEX Optimization Studio V12.5.1 com as configurações *default*, caracterizando este algoritmo como sendo matheurístico.

Os trabalhos anteriores sugeriam 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 metaheurísticas 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; GROSSO;

SALASSA, 2014), abriu-se oportunidade para explorar a potencialidade dos métodos exatos combinados, ou não, com heurísticas.

Sendo assim, na fase de experimentação computacional, a qual processou o modelo proposto neste trabalho (MATH) e o modelo PLI de Scholl, Boysen e Fliedner (2013) com algumas instâncias de dados SBF2, pôde-se constatar que o modelo MATH atingiu resultados iguais ou melhores em um tempo de processamento inferior ao apresentado pelo modelo PLI. Conforme os resultados obtidos, o modelo MATH apresentou um ganho de 44,74% de desempenho computacional em relação ao modelo PLI. Além disso, a otimalidade atingida pelo modelo MATH não se deu apenas em um menor tempo de estação, mas resultou em um número de estações menor. Portanto, além do ganho de desempenho computacional, o modelo proposto (MATH) apresentou resultados mais satisfatórios, encontrando soluções que demandam um número inferior de estações de trabalho se comparado ao modelo PLI.

Acredita-se, portanto, que os objetivos propostos neste trabalho foram atingidos. Também é possível afirmar que a abordagem proposta parece ser adequada ao compromisso de encontrar boas soluções para o SUALBSP em tempo curto como exigido nos ambientes de planejamento do mundo real.

6.1 CONTRIBUIÇÕES

Este trabalho buscou contribuir para os estudos do SUALBSP ao apresentar um modelo alternativo ao modelo de Scholl, Boysen e Fliedner (2013), o qual mostrou-se viável ao utilizar um menor tempo de processamento computacional, o que é de fundamental importância para a aceitabilidade de uma ferramenta de planejamento de linha de montagem.

Além disso, a técnica de solução proposta neste trabalho ainda não havia sido empregada na resolução do SUALBSP. Sendo assim, o presente trabalho pretendeu contribuir para o enriquecimento da pesquisa e da literatura que trata do SUALBSP. Destaca-se, ainda, que a técnica de solução proposta apresentou um bom desempenho tanto em tempo computacional, quanto em qualidade da solução.

Na prática, a abordagem proposta pode impactar na melhor qualidade do balanceamento e sequenciamento da linha de montagem, culminando em ganhos econômicos para as empresas.

6.2 LIMITAÇÕES DO TRABALHO

Como limitações do presente trabalho, pode-se destacar que o modelo MATH tratou apenas com instâncias pequenas de dados de, no máximo, 25 tarefas (vide ANEXO A). Isso ocorreu, pois o critério de seleção das instâncias que foram testadas tanto no modelo PLI de Scholl, Boysen e Fliedner (2013) quanto no modelo MATH, era a apresentação de resultado factível dentro de um tempo limite de processamento de 100 segundos. Com este limite de tempo de processamento acabaram sendo selecionadas instâncias pequenas. Talvez, ao aumentar-se o tempo limite de processamento, tornaria possível selecionar instâncias com um maior número de tarefas.

Chama-se a atenção para o fato de que é muito importante também testar instâncias grandes de dados, pois as instâncias do SUALBSP no mundo real podem ser extremamente numerosas.

Faz-se necessário registrar também que o estudo de Scholl, Boysen e Fliedner (2013) desenvolveu diversas heurísticas e forneceu muitas ferramentas para a resolução do SUALBSP. Essas heurísticas e ferramentas apresentaram resultados muito bons, o que dificultou consideravelmente a busca por lacunas no trabalho dos autores e, por conseguinte, a elaboração deste trabalho.

Outra limitação é que o modelo proposto não foi aplicado a uma instância real. Isto ocorreu pela dificuldade de encontrar os dados disponíveis.

6.3 SUGESTÕES DE TRABALHOS FUTUROS

Este trabalho mostra que há espaço para que se evolua dentro do campo das abordagens híbridas, tal como a matheurística. Desta forma, estudos futuros devem necessariamente propor modelos ou métodos que sejam capazes de tratar instâncias de dados maiores. Outra sugestão é a aplicação do método proposto neste estudo no contexto real, o que inclui, por exemplo, considerar as linhas de montagem com formato em U, as linhas de dois lados e as linhas paralelas.

Também, como sugestão nota-se a possibilidade de utilização de outros tipos de abordagens híbridas, valendo-se da combinação entre duas ou mais metaheurísticas ou

matheurísticas. Ainda, após poderia ser realizado o confronto entre os resultados dessas outras abordagens híbridas com a empregada neste trabalho. Também deve-se avaliar a necessidade de elaboração de *lower bounds* melhorados.

Sugere-se o aprimoramento no método de solução empregado para que seja possível solucionar o problema considerando aspectos ergonômicos, que cada vez mais têm recebido atenção por parte das indústrias.

Finalmente, o modelo de solução proposto pode ser adaptado a outros problemas com estrutura matemática similar.

REFERÊNCIAS

ANDRÉS C.; MIRALLES, C.; PASTOR, R. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. **European Journal of Operational Research**, v. 187, p. 1212-1223, 2008.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional**. Rio de Janeiro: Elsevier, 2007 – 6ª reimpressão.

AVIKAL, S. J. R.; MISHRA, P. K.; YADAV, H. C. A heuristic approach for U-shaped assembly line balancing to improve labor productivity. **Computers & Industrial Engineering**, v. 64, p. 895-901, 2013.

BATTAIA, O.; DOLGUI, A. A taxonomy of line balancing problems and their solution approaches. **International Journal Production Economics**, v. 142, p. 259-277, 2013.

BAUTISTA, J.; PEREIRA, J. A dynamic programming based heuristic for the assembly line balancing problem. **European Journal of Operational Research**, v. 194, n. 3, p. 787-794, 2009.

BAYBARS, I. A survey of exact algorithms for the simple assembly line balancing problem. **Management Science**, v. 32, p. 909-932, 1986.

BECKER, C.; SCHOLL, A. A survey on problems and methods in generalized assembly line balancing. **European Journal of Operational Research**, v. 168, p. 694-715, 2006.

BELFIORE, P.; FÁVERO, L. P. **Pesquisa operacional para cursos de administração, contabilidade e economia**. Rio de Janeiro: Elsevier, 2012.

BOSCHETTI, M. A.; MANIEZZO, V.; ROFFILLI, M.; RÖHLER, A. B. Matheuristics: Optimization, Simulation and Control. *In*: HYBRID METAHEURISTICS INTERNATIONAL WORKSHOP, 6th, 2009, Udine-Italy. **Proceedings**. Germany: Springer, 2009, p. 171-177.

BOYSEN, N; FLIEDNER, M. A versatile algorithm for assembly line balancing. **European Journal of Operational Research**, v. 184, p. 39-56, 2008.

BOYSEN, N.; FLIEDNER, M.; SCHOLL, A. A classification of assembly line balancing problems. **European Journal of Operational Research**, v. 183, p. 674-693, 2007.

BOYSEN, N.; FLIEDNER, M.; SCHOLL, A. Assembly line balancing: Which model to use when? **International Journal Production Economics**, v. 111, p. 509-528, 2008.

BUKCHIN, Y; RABINOWITCH, I. A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. **European Journal of Operational Research**, v. 174, p. 492-508, 2006.

BUKCHIN, J.; DAR-EL, E. M.; RUBINOVITZ, J. Mixed-model assembly line design in a make-to-order environment. **Computers & Industrial Engineering**, v. 41, p. 405-421, 2002.

CASERTA, M.; VOβ, S. Chapter 1. Metaheuristics: Intelligent problem solving. Maniezzo, V. *et al.* (eds.), Matheuristics, **Annals of Information Systems 10**, p. 1-38, 2009.

CRISTO, R. L. **Balanceamento de Linhas de Montagem com uso de Algoritmo Genético para o caso de linhas simples e extensões**. 2010. 85 f. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-graduação em Engenharia de Produção, Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis.

DELLA CROCE, F.; GROSSO, A.; SALASSA, F. A Matheuristic approach for the two-machine total completion time flow shop problem. **Annals of Operation Research**, v. 213, p. 67-78, 2014.

EREL, E.; SARIN, S. C. A survey of the assembly line balancing procedures. **Production Planning and Control**, v. 9, p. 414-434, 1998.

GAITHER, N., FRAZIER, G. **Administração da produção e operações**. 8. ed. São Paulo: Pioneira, 2005.

GÖKÇEN H.; AGPAK K.; BENZER, R. Balancing of parallel assembly lines. **International Journal Production Economics**, v. 103, p. 600-609, 2006.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear**. Rio de Janeiro: Elsevier, 2005.

GUTJAHR, A. L.; NEMHAUSER, G. L. An algorithm for the line balancing problem. **Management Science**, v. 11, n. 2, p. 308-315, 1964.

HAMTA, N.; GHOMI, S. M. T. F.; JOLAI, F.; SHIRAZI, M. A. A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. **International Journal Production Economics**, v. 141, p. 99-111, 2013.

JOURDAN, L.; BASSEUR, M.; TALBI, E.-G. Hybridizing exact methods and metaheuristics: A taxonomy. **European Journal of Operational Research**, v. 199, p. 620-629, 2009.

KHORASANIAN, D.; HEJAZI, S. R.; MOSLEHI, G. Two-sided assembly line balancing considering the relationship between tasks. **Computers & Industrial Engineering**, v. 66, p. 1096-1105, 2013.

KILINCCI, O. A Petri net-based heuristic for simple assembly line balancing problem of type 2. **International Journal of Advanced Manufacturing Technology**, v. 46, n. 1, p. 329-338, 2010.

MANERBA, D.; MANSINI, R. An effective matheuristic for the capacitated total quantity discount problem. **Computers & Operations Research**, v. 41, p. 1-11, 2014.

MARTINO, L.; PASTOR, R. Heuristic procedures for solving the general assembly line balancing problem with setups. **International Journal of Production Research**, v. 48, p. 1787-1804, 2010.

PUCHINGER, J.; RAIDL, G. R. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *In*: INTERNATIONAL WORK-CONFERENCE ON THE INTERPLAY BETWEEN NATURAL AND ARTIFICIAL COMPUTATION, IWINAC, 1, 2005, Las Palmas, Canary Islands, Spain. **Proceedings, Part II**.Germany: Springer, 2005, p. 41-53.

RASHID, M. F. F.; HUTABARAT W.; TIWARI, A. A review on Assembly Sequence Planning and Assembly Line Balancing Optimisation using soft computing approaches. **International Journal of Advanced Manufacturing Technology**, v. 59, issue 1-4, p. 335-349, 2011.

RIBEIRO, C. C.; MANIEZZO, V. Preface to the special issue on matheuristics: Model-based metaheuristics. **International Transactions in Operational Research**, v. 22, p. 1, 2015.

SCHOLL, A.; BECKER, C. State-of-art exact and heuristic solution procedures for simple assembly line balancing. **European Journal of Operations Research**, v. 168, p. 666-693, 2006.

SCHOLL, A.; FLIEDNER, M.; BOYSEN, N. Absalom: Balancing assembly lines with assignment restrictions. **European Journal of Operational Research**, v. 200, n. 3, p. 688-701, 2010.

SCHOLL, A.; BOYSEN N.; FLIEDNER, M. The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. **OR Spectrum**, v. 35, p. 291-320, 2013.

SEWELL, E. C.; JACOBSON, S. H. A Branch, bound, and remember algorithm for the Simple Assembly Line Balancing Problem. **INFORMS Journal on Computing**, v. 24, n. 3, p. 433-442, 2012.

SEYED-ALAGHEBAND, S. A.; GHOMI, S. M. T.; ZANDIEH, M. A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. **International Journal of Production Research**, v. 49, n. 3, p. 805-825, Feb. 2011.

SIMARIA, A. S. A. **Uma metodologia para o balanceamento de linha de montagem**. 2001. 103 f. Dissertação (Mestrado). Escola de Gestão do Porto, Porto.

STEFANELLO, F.; ARAÚJO, O. C. B.; MULLER, F. M. Matheuristics for the capacitated p-median problem. **International Transactions in Operational Research**, v. 22, p. 149-167, 2015.

VILLEGAS, J. G. A matheuristic for the truck and trailer routing problem. **European Journal of Operational Research**, v. 230, p. 231-244, 2013.

YOLMEH, A.; KIANFAR, F. An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. **Computers & Industrial Engineering**, v. 62, p. 936-945, 2012.

ANEXO A – DADOS SELECIONADOS DAS INSTÂNCIAS TESTADAS

Nome da Instância	α	Número de Tarefas	Tempo de ciclo
Mertens	25	7	15
		7	18
		7	10
		7	7
		7	8
		7	6
	50	7	15
		7	18
		7	10
		7	7
		7	8
		7	6
	75	7	18
		7	15
		7	10
		7	7
		7	8
		7	6
	100	7	18
		7	15
		7	7
		7	8
		7	10
		7	6
Mansoor	25	11	94
		11	62
		11	48
	50	11	94
		11	62
		11	48
	75	11	94
		11	62
		11	48
	100	11	94
		11	62
		11	48
Jaeschke	25	9	18
		9	10
		9	8
		9	7
		9	6
	50	9	18
		9	10
		9	8
		9	7
		9	6
	75	9	18
		9	10
		9	8

		9	7
		9	6
	100	9	18
		9	10
		9	8
		9	7
		9	6
Bowman	25	8	20
	50	8	20
	75	8	20
	100	8	20
Jackson	25	11	21
		11	13
		11	14
		11	10
		11	9
		11	7
	50	11	21
		11	14
		11	10
		11	13
		11	7
		11	9
	75	11	21
		11	14
		11	10
		11	13
		11	9
		11	7
	100	11	21
		11	13
		11	14
		11	10
		11	9
		11	7
Mitchell	25	21	35
		21	39
		21	21
		21	26
	50	21	35
		21	39
		21	26
		21	21
	75	21	35
		21	39
		21	26
		21	21
	100	21	35
		21	39
		21	26
		21	21
Roszieg	25	25	32