UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DE SAÚDE

DEPARTAMENTO DE COMPUTAÇÃO

JOÃO VITOR ALMEIDA COSTA

UMA META-HEURÍSTICA SIMULATED ANNEALING APLICADA AO PROBLEMA DE BALANCEAMENTO DE LINHA DE MONTAGEM SIMPLES DO TIPO DOIS

ALEGRE

2019

JOÃO VITOR ALMEIDA COSTA

UMA META-HEURÍSTICA SIMULATED ANNEALING APLICADA AO PROBLEMA DE BALANCEAMENTO DE LINHA DE MONTAGEM SIMPLES DO TIPO DOIS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e de Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Dayan de Castro Bissoli.

ALEGRE

2019

2019

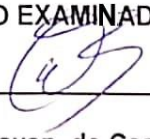
JOÃO VITOR ALMEIDA COSTA

UMA META-HEURÍSTICA SIMULATED ANNEALING APLICADA AO PROBLEMA DE BALANCEAMENTO DE LINHA DE MONTAGEM SIMPLES DO TIPO DOIS

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e de Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em 28 de junho de 2019.

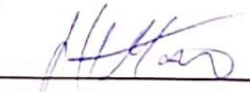
COMISSÃO EXAMINADORA



Prof. Dr. Dayan, de Castro Bissoli

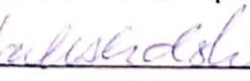
Universidade Federal do Espírito Santo

Orientador



Prof. Dr. Geraldo Regis Mauri

Universidade Federal do Espírito Santo



Prof. M.sc. Valeria Alves da Silva

Universidade Federal do Espírito Santo

“A minha família”

AGRADECIMENTOS

Agradeço primeiramente a Deus que esteve comigo em todos os momentos, os mais difíceis e os mais felizes. Obrigado Deus por tudo.

A minha família, primeiramente aos meus pais, Ironice e João, que me apoiaram a em buscar meus sonhos e sempre me aconselhando nas minhas decisões, agradeço a minha irmã Juliana, que me apoiou ao decorrer da graduação com seus ensinamentos, minha eterna melhor amiga e a agradeço o apoio de toda minha família, e agradecer a minha melhor segunda amiga, Amanda Aschauer por fazer minha vida maravilhosa e sempre cuidar de mim e me apoiar nas minhas decisões.

Aos meus amigos da República DNA, que me acolheram quando cheguei em Alegre, e os amigos de república que criei ao decorrer dos anos, que fiz uma grande amizade que alegram minhas noites na república com muitas piadas, noites jogando ps4, sociais da DNA. Aos meus amigos que fiz na UFES que tornaram meus cinco lá especiais.

A todos os amigos que eu conheci nessa faculdade, que sempre estavam dispostos a me ajudar, mesmo quando às vezes parecia impossível, e a todos que estiveram comigo ao decorrer do curso.

Ao meu orientador Dayan, por ter aceitado esse desafio de me orientar, pela paciência, pelos seus ensinamentos e pela oportunidade de participar deste trabalho. Agradecer aos membros da banca examinadora pela participação nesse trabalho de conclusão de concurso.

A equipe do STI da UFES onde pude conhecer melhor a universidade e seus funcionários. A empresa Click júnior que me proporcionou uma experiência e me deu muitos ensinamentos para vida. A Universidade Federal do Espírito Santo e a todos seus funcionários, em especial aos professores do departamento de computação por me transmitirem o conhecimento e experiências vividas.

A todos, muito obrigado.

RESUMO

Com o aumento da competitividade ao longo do tempo, indústrias baseadas em linhas de montagem precisam produzir seus produtos com objetivo de que perdas de tempos sejam minimizadas de modo que não eleve os custos de produção. Nesse contexto, surge o problema denominado Balanceamento de Linhas de Montagem Simples (SALBP), no qual o produto é fabricado executando um conjunto de tarefas que possuem restrições de precedência entre si em determinadas estações de trabalho. Dessa forma, deve-se determinar o conjunto de tarefas que cada estação de trabalho irá processar de modo a otimizar um determinado critério. O trabalho proposto visa revisar e resolver o problema de balanceamento de linha de montagem tipo 2 (SALBP-2), que é uma variação do SALBP. A função objetivo deste problema *NP-Hard* consiste em minimizar o tempo de ciclo, que é a estação de trabalho que apresenta a maior unidade de tempo. Para solucionar o SALBP-2, considerando que vem apresentando bons resultados em outros problemas de otimização combinatória, foi desenvolvida a meta-heurística Simulated Annealing (SA).

Palavras-chave: Balanceamento de linha de montagem, SALBP-2, Simulated Annealing, Otimização Combinatória, Administração e Gestão da Produção.

ABSTRACT

With increasing competitiveness over time, assembly-line-based industries need to produce their products in order to minimize time-wasting so as not to raise production costs. In this context, the problem called Simple Assembly Line Balancing (SALBP) arises, in which the product is manufactured by executing a set of tasks that have precedence constraints on certain workstations. In this way, one must determine the set of tasks that each workstation will process in order to optimize a certain criterion. The proposed work aims to review and solve the problem of assembly line balancing type 2 (SALBP-2), which is a variation of SALBP. The objective function of this NP-Hard problem is to minimize the cycle time, which is the workstation that presents the longest unit of time. To solve the problem SALBP-2, considering that it has been presenting good results in other problems of combinatorial optimization, a meta-heuristic Simulated Annealing (SA) was developed.

Keywords: Assembly line balancing, SALBP-2, Simulated Annealing, Combinatorial Optimization, Production Management.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Com a grande variedade de produtos no mercado nos últimos tempos, os consumidores têm criado gostos e preferências de diversos estilos, junto à constante e rápida evolução do mercado de vestuário, de tecnologia, automotivo, dentre outros. Com isso, as organizações devem produzir com alta qualidade e baixo custo, oferecendo produtos com máximo de valor agregado por um menor custo e tempo de resposta ao cliente (GAITHER; FRAZIER, 1999; PAIVA et al., 2004).

Em ambientes produtivos, uma linha de produção é caracterizada por uma série de estações de trabalhos de montagens mecânicas ou manuais, onde um ou vários produtos finais são gerados seguindo uma determinada ordem de tarefas**.** Contudo, o problema de balanceamento da linha de montagem consiste em distribuir a carga de trabalho total entre as estações de trabalho para a fabricação de qualquer unidade do produto a ser montado ao longo da linha (CARNAHAN et al., 2001).

As categorias conhecidas como problemas de balanceamento de linha de montagem (*Assembly Line Balancing Problem* - ALBP) dizem respeito à otimização de processos relacionados à fabricação de produtos através de linhas de montagem com estações de trabalhos. A importância no mundo industrial é mostrada pelo fato de que muitos esforços de pesquisa foram realizados e dedicados a muitos tipos diferentes de ALBPs durante os últimos 50-60 anos (SALVESON, 1995).

O problema abordado neste trabalho é denominado de problema de balanceamento de linha de montagem simples (SALBP). Uma linha de montagem é composta por um conjunto de estações de trabalho dispostas em uma linha, e por um sistema de transporte que move o produto para ser fabricado ao longo da linha. O produto é fabricado executando um determinado conjunto de tarefas. Cada uma dessas tarefas possui um tempo de processamento pré-definido. Para obter uma solução para o SALBP, todas as tarefas devem ser atribuídas a estações de trabalho sujeitas a restrições de precedência entre as tarefas. No contexto do SALBP-2, todas as estações de trabalho são consideradas com a mesma capacidade de processamento. Além disso, assume-se que a linha de montagem se move em velocidade constante, implicando que as estações de trabalho podem possuir

diferentes tempos de produção, considerando que cada tarefa possui um tempo distinto. (SCHOLL; BECKER, 2006).

O SALBP-2 é reconhecido como um problema de otimização combinatória do tipo *NP-hard* (SCHOLL; BECKER, 2006). Com isso, as utilizações de métodos exatos para a solução de grandes problemas não são indicadas por demandar um elevado tempo computacional.

1.1 O PROBLEMA E SUA IMPORTÂNCIA

A grande pressão do mercado junto à crescente competitividade entre as empresas faz com que organizações que ainda trabalham com atividades manuais, necessitem aperfeiçoar seus métodos de produção, assim gerando seus produtos em menor tempo possível (ZACHARIA; NEARCHOU, 2010). Com a necessidade de um aperfeiçoamento nesses métodos de produção, Tasan e Tunali (2008) dizem que esses procedimentos de aperfeiçoamento devem ser aptos a produzir em grande escala novos modelos de produtos, além de se ajustar rapidamente às mudanças do gosto do consumidor, deve integrar a tecnologia nos seus processos e além disso, produzir uma grande diversidade de produtos.

Segundo Baudin (2002), a complexidade das prováveis soluções torna o estudo da linha de montagem um tema relevante de pesquisa. Já que as empresas são obrigadas a expandir os limites de seus produtos para suprir as expectativas dos seus consumidores por um alto grau de customização, com isso mantendo em alto nível de qualidade e um baixo custo o produto (SIMARIA; VILARINHO, 2004). Com a finalidade de aumentar a produtividade e diminuir os custos dos processos da linha, é realizado um balanceamento das tarefas nas estações de trabalho, onde pode obter esse balanceamento através modelos matemáticos, heurísticas e meta-heurísticas, ou seja, métodos que buscam uma solução viável para uma determinada classe de problemas (HILLIER; LIEBERMAN, 2013).

Portanto, o estudo de métodos eficientes para resolução do problema de balanceamento de linha de montagem simples torna-se importante, para auxiliar a tomada de decisão em ambientes de sequenciamento de produção.

1.2 OBJETIVOS

A seguir são descritos os objetivos gerais e específicos relacionados a este trabalho.

1.2.1 OBJETIVO GERAL

O objetivo geral deste trabalho é implementar uma meta-heurística *Simulated Annealing* para solucionar o SALBP-2.

1.2.2 OBJETIVOS ESPECÍFICOS

Com o intuito de atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

a) Realizar revisão bibliográfica do SALBP-2;
b) Implementar uma meta-heurística *Simulated Annealing* para solucionar o problema SALBP-2;
c) Realizar experimentos computacionais e comparar com a literatura;
d) Discutir os experimentos computacionais.

1.3 APRESENTAÇÃO DO TRABALHO

O trabalho está dividido da seguinte forma. No Capítulo 2 é apresentada uma descrição do problema SALBP-2 e uma revisão bibliográfica dos principais e mais recentes trabalhos da literatura. O Capítulo 3 detalha como a meta-heurística *Simulated Annealing* foi aplicada ao SALBP-2. Os resultados são descritos no Capítulo 4 e, no Capítulo 5 são apresentadas as conclusões, no Capítulo 6 são enumeradas as referências utilizadas nesse trabalho.

2. O PROBLEMA DE BALANCEAMENTO DE LINHAS DE MONTAGEM

Uma linha de montagem é caracterizada como sistemas onde a matéria-prima entra gradualmente e se move pelo meio de uma sequência de estações de trabalhos enquanto as tarefas estão sendo executadas e transformando a matéria-prima no produto desejável (SOUZA et al., 2003). O esforço total em um procedimento de montagem é particionado em tarefas, onde essas mesmas precisam ter um tempo para serem realizadas. Essas tarefas possuem uma relação de precedência, isto é, para poder realizar uma tarefa na estação de trabalho, todas suas tarefas anteriores devem ter sido realizadas (BECKER e SCHOLL, 2006).

Porém, a grande dificuldade de potencializar a divisão de tarefas entre as estações de trabalho é conhecida como Problema de Balanceamento de Linha de Montagem (*Assembly Line Balancing Problem* - ALBP) (SCHOLL; BOYSEN e FLIEDNER, 2009). Kriengkorakot e Pianthong (2012) define o ALBP tecnicamente como uma linha de montagem consiste em um conjunto de N estações de trabalho, onde geralmente estão dispostas ao longo de uma esteira ou correia transportadora ou algum equipamento similar de transporte de matérias. As tarefas são lançadas consecutivamente na linha e são movidos de estação para estação. Portanto, o problema de balanceamento de linha de montagem consiste em equilibrar a carga de trabalho na linha de montagem de forma a otimizar algum objetivo. Contudo, a quantidade total de trabalho necessária para produzir um produto é dividida em um conjunto de $V = \{1, \ldots, n\}$ operações elementares denominadas tarefas. A execução de uma tarefa k possui um tempo específico $t(k)$. A carga total de uma estação de trabalho é medida pela soma dos tempos das tarefas $\sum t$ atribuídas a ela. Essas tarefas podem ser representadas por um grafo orientado de precedência, onde cada nó corresponde a uma tarefa, os pesos dos nós equivalem os tempos das tarefas e os arcos indicam as restrições de precedência. A Figura 1 mostra um diagrama de precedência com $n = 9$ tarefas, com respectivos tempos entre 2 e 9 unidades de tempo.

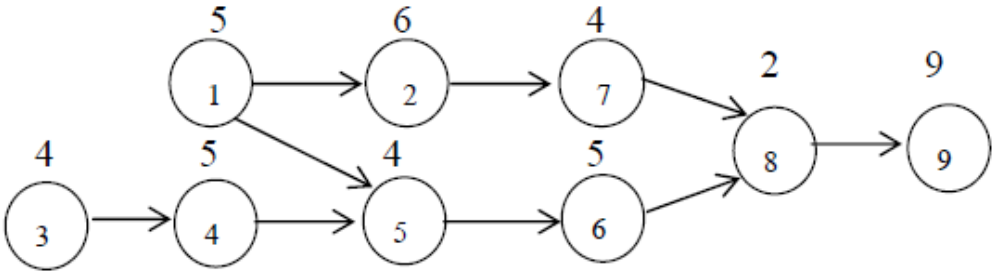


Figura 1 - Diagrama de precedência.

Fonte: Kriengkorakot e Pianthong (2012).

De acordo com Becker e Scholl (2006), o ALBP possui duas classificações, Problema de Balanceamento de Linha de Montagem Simples (*Simple Assembly Line Balancing Problem* - SALBP) e Problema de Balanceamento de Linhas de Montagem Generalizado (*Generalized Assembly Line Balancing Problem* - GALBP), onde elas se diferem nas restrições e funções objetivo. Para ser classificado como SALBP, deve-se levar em conta algumas ponderações (BOYSEN; FLIEDNER e SCHOLL, 2008; SCHOLL, BOYSEN e FLIEDNER, 2009):

1. O resultado final da produção deve gerar só um produto;
2. Todas as tarefas devem ser processadas de um determinado modo;
3. As linhas de montagem devem ter um tempo fixo;
4. A linha é considerada linear, sem linhas de abastecimento ou paralelas;
5. As sequências de processamento das tarefas devem seguir as restrições de precedência de cada tarefa;
6. Os tempos das tarefas são determinísticos, ou seja, dada uma certa entrada, ela produzirá sempre a mesma saída;
7. As únicas restrições para atribuir às tarefas devem ser de precedência;
8. Uma tarefa não pode ser processada em duas ou mais estações de trabalho;
9. Todas as estações são igualmente equipadas com máquinas e operadores.

Os problemas que não satisfazerem as nove condições acima são classificados como GALBP. Nessa classe está incluso o problema de equilíbrio *U-line* (UALBP) e o problema de balanceamento da linha de montagem do modelo misto (MALBP). Esses problemas são muito grandes e contém todas as extensões problemáticas que podem ser relevantes na prática como a seleção de equipamento, opções de

processamento, entre outros. (BECKER e SCHOLL, 2006). A Figura 2 mostra uma visão geral da classificação do ALBP.

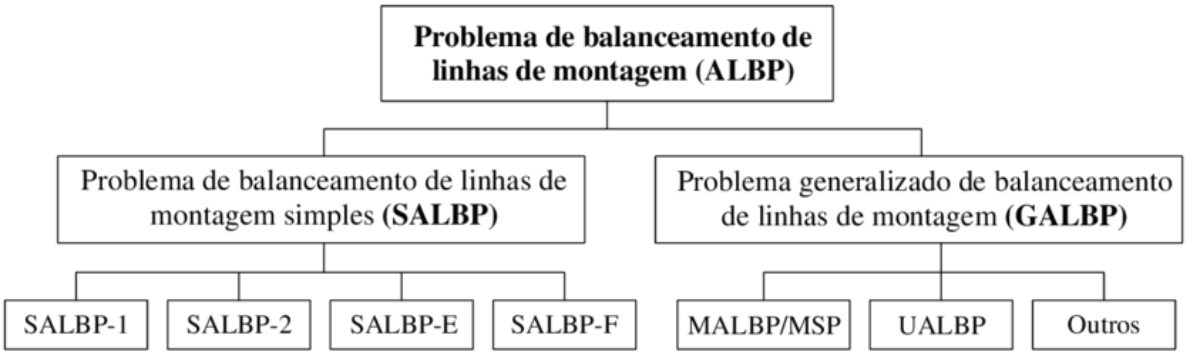


Figura 2 - Classificação dos problemas de balanceamento de linha de montagem.

Fonte: Bissoli e Amaral (2016).

O SALBP pode ser divididos em quatro tipos: O primeiro, chamado de SALBP-F, é um problema de viabilidade que consiste em estabelecer se uma linha é viável ou não para um certo tempo de ciclo e número de estações. O SALBP-1 minimiza o número de estações para um dado tempo de ciclo, ou seja, consiste em delegar tarefas a estações de trabalho, de modo que o número das estações seja minimizado para uma dada taxa de produção. O SALBP-2 minimiza o tempo de ciclo para um determinado número de estações, ou seja, visa aumentar a taxa de produção para um específico número de estações. O SALBP-E é a versão mais geral do problema, que objetiva maximizar a eficiência da linha minimizando simultaneamente o tempo de ciclo e o número de estações de trabalho (BECKER e SCHOLL, 2006; BOYSEN; FLIEDNER e SCHOLL, 2008).

2.1 REVISÃO DE LITERATURA

Uma característica principal do problema de balanceamento de linha de montagem é a existência de restrições de precedência. As restrições de precedência limitam que uma tarefa seja atribuída a uma estação, enquanto as suas tarefas anteriores não estejam concluídas. (TALBOT E PATTERSON,1984).

A seguir aborda-se alguns trabalhos recentes que solucionaram o SALBP-2 por meio de métodos heurísticos.

Nearchou (2007) aplica uma nova heurística baseada no método de evolução diferencial para poder equilibrar linhas de montagem de grande porte. Considerando

instâncias públicas de *benchmark* de ALBPs para quais existem limites superiores no valor da função objetivo ideal, o algoritmo considera o ALBP simples do tipo 2 (SALBP-2). A heurística foi testada em várias instâncias de grafos de precedência, onde representava as restrições de montagem com até 297 tarefas. Comparações estendidas com outros métodos de computação evolutiva previamente publicados mostraram um desempenho superior para a abordagem proposta. Além disso, o processo experimental mostrou que a aplicação do algoritmo de evolução diferencial (DEA) pode ser uma ferramenta de otimização promissora para a solução de muitos outros problemas de otimização de fabricação de natureza combinatória.

Kilincci (2010) propôs a utilização de uma heurística baseada em rede de Petri, para resolver o SALBP-2, onde a rede Petri é uma ferramenta matemática e gráfica usada para modelar e analisar sistemas inteiros. O algoritmo foi testado em 302 instâncias e foi comparada com heurísticas que possuem sua base em algoritmo de evolução diferencial de Nearchou (2007). Os resultados dos testes de comparação mostraram que a heurística baseada em rede de Petri obteve bons resultados para SALBP-2.

Blum (2011) aborda o SALBP-2 aplicando um método denominado *Iterative Beam Search* (IBS). O procedimento proposto é capaz de obter as melhores soluções conhecidas em 283 dos 302 casos de teste. Mais recentemente, Sikora et al., (2015) propuseram um algoritmo genético híbrido (AG) para o SALBP-2. O AG obteve respostas de alta qualidade, sendo que o ótimo foi encontrado em entorno de 42% das 302 instâncias testadas. No entanto, o AG apresentou-se com desempenho inferior quando comparado com os resultados gerados pelo método SALOME (KLEIN; SCHOLL, 1996) para resolver o SALBP.

Zhang et al., (2016) apresentaram um novo algoritmo de evolução diferencial para a solucionar o SALBP-2. O algoritmo apresentado pode lidar diretamente com variáveis inteiras de SALBP-2 em um espaço discreto. O processo experimental global mostrou que o IDEA pode ser uma ferramenta de otimização promissora para a solução de muitos outros problemas de valor inteiro.

Bissoli e Amaral (2016) propõem o algoritmo GRASP para resolver o SALBP-2, no qual é usada uma Lista restrita de candidatos na etapa de construção da solução, para realizar as atribuições das tarefas nas máquinas. O algoritmo proposto foi testado em um conjunto de instâncias *benchmark* da literatura e os resultados

obtidos mostram que atinge os melhores valores para algumas soluções conhecidas na literatura e em outras instâncias se aproxima das ótimas soluções.

2.2 MODELO MATEMÁTICO PARA O SALBP-2

Blum (2011) apresenta um modelo matemático para o SALBP-2, sendo que à função objetivo consiste em minimizar o chamado o tempo de ciclo, usando de forma expressa como um problema de programação inteira (PI).

Minimizar

$$z \tag{1}$$

Sujeito a:

$$\sum_{s=1}^{m} x_{is} = 1 \ \forall i \in T \tag{2}$$

$$x_{is} \leq \sum_{s\prime=1}^{s} x_{js\prime} \ \forall i \in T, s = 1, \ldots, m, \ j \in P_i \tag{3}$$

$$\sum_{i \in T} t_i x_{is} \leq z \ s = 1, \ldots, \quad m \tag{4}$$

$$x_{is} \in \{0,1\} \ \forall i \in T, \ s = 1, \ldots, \quad m \tag{5}$$

$$z > 0 \tag{6}$$

Este modelo PI faz uso das seguintes variáveis e constantes: x_{is} é um binário, variável que é definida como 1, se, e somente se tarefa $i \in T$ é atribuída à estação de trabalho s , x recebe o valor 1. A função objetivo (1) minimiza o tempo de ciclo de $z > 0$. A Restrição (2) garante que cada tarefa $i \in T$ é atribuída a uma única estação de trabalho $1 \leq s \leq m$. A restrição (3) reflete as relações de precedência entre as tarefas. Mais especificamente, se a tarefa $i \in T$ é atribuída a uma estação de trabalho $1 \leq s \leq m$, todas as tarefas $j \in P_i$ devem ser atribuídas às estações de trabalho $1 \leq s' \leq m$ com $s' \leq s$. A restrição (4) garante que a soma dos tempos de processamento das tarefas atribuídas a uma estação de trabalho s não exceda o tempo de ciclo z.

2.3 CONSIDERAÇÕES FINAIS

Considerando os estudos encontrados na literatura que resolveram o SALBP-2 (NEARCHOU, 2007; KILINCCI, 2010; BLUM, 2011; BISSOLI, AMARAL ,2016; ZHANG ET AL, 2016) constatou-se que ainda não houve a aplicação do SA para o SALBP-2. Dessa forma, nesse estudo é proposto solucionar o SALBP-2 pelo SA, que é descrito na seção seguinte.

3. METODOLOGIA

A metodologia abordada neste trabalho consiste na aplicação da meta-heurística *Simulated Annealing* (SA) para resolução do de problema de balanceamento de linha de montagem simples do tipo dois (SALBP-2), que é descrito formalmente a seguir.

3.1 REPRESENTAÇÃO DE UMA SOLUÇÃO

A representação da solução adotada neste trabalho considera o número de estações de trabalho e de tarefas. A solução é representada por uma matriz de n linhas por m colunas, sendo n o número de estações de trabalho que podem variar de acordo com dado de números de estações de trabalhos para cada instância, sendo assim i representa o índice cada estação de trabalho e k representa o índice de cada tarefa na estação de trabalho .

Considerando uma programação elaborada com 35 tarefas a serem distribuídas em 8 estações de trabalho, onde cada tarefa deve respeitar seus predecessores, isto é, uma tarefa não pode estar em um posto que esteja anterior ao seu predecessor. Uma possível representação da solução apresentada na Figura 3. Nesse exemplo, as linhas representam as estações de trabalho e as colunas representam a quantidade de tarefas em cada estação de trabalho.

Posto de Trabalho 1	1	5	6	8	7	2	17
Posto de Trabalho 2	12	18	10				
Posto de Trabalho 3	9	14	15	19			
Posto de Trabalho 4	20	16	3				
Posto de Trabalho 5	13	4	21	25			
Posto de Trabalho 6	22	23	24	26	27		
Posto de Trabalho 7	11	28					
Posto de Trabalho 8	29	30	31	32	33	35	34

Figura 3 - Representação de uma de solução com 8 estações de trabalhos e 35 tarefas.

O tempo de ciclo de cada estação é representado por um vetor de p_i onde o valor máximo do índice i é o valor máximo de n, cada tempo de ciclo é calculado com a soma dos custos das tarefas em cada estação de trabalho. A Tabela 1 apresenta os custos de cada tarefa separadamente, na Tabela 2 é mostrado cada tarefa e seus predecessores, na coluna de predecessores é apresentado as tarefas que devem ser executadas antes da chamada da tarefa que se encontra na coluna tarefa, o vazio nesta coluna indica que a tarefa não possui predecessores, por fim a Tabela 3 apresenta os valores presentes em p_i para a solução ilustrada na Figura 3.

Tabela 1 - Representação dos custos de cada tarefa.

Tarefa	Custo
1	29
2	3
3	5
4	22
5	6
6	14
7	2
8	5
9	22
10	30
11	23
12	30
13	23
14	2
15	19
16	29
17	2
18	2
19	19
20	29
21	6
22	10
23	16
24	23
25	5
26	5
27	5
28	40
29	2
30	5
31	5
32	1
33	40
34	2
35	2

Tabela 2 - Representação dos predecessores das tarefas.

Tarefa	Predecessores
1	
2	1
3	2
4	3
5	1
6	5
7	1,6
8	6
9	8
10	1
11	4
12	1
13	9
14	7,10
15	14
16	15
17	
18	7,12
19	18
20	17,19
21	16,20
22	21
23	22
24	23
25	21
26	
27	24,26
28	11,13,27
29	28
30	21
31	30
32	21,31
33	11,13,27,32
34	27
35	33

Tabela 3 - Representação do tempo de ciclo de cada estação de trabalho.

Estação de Trabalho	1	2	3	4	5	6	7	8
Tempo de Ciclo	61	62	62	63	56	59	63	57

3.2 ESTRUTURAS DE VIZINHANÇA

Uma estrutura de vizinhança pode ser definida pelos movimentos entre as tarefas da solução para outras estações de trabalho, ou seja, as diferentes estruturas de

vizinhança são obtidas a partir de uma operação chamada movimento. Dada uma solução s conhecida, uma solução s' pode ser obtida por meio de dois movimentos de troca realizados na matriz de solução.

Para este trabalho, foram desenvolvidos 2 tipos de movimentos, que são descritos a seguir:

3.2.1 MOVIMENTO TROCA DE TAREFA ENTRE ESTAÇÃO DE TRABALHO A ESQUERDA OU DIREITA

O movimento esquerda ou direita (MED) realiza a troca de tarefas entre estações de trabalhos perto uma das outras. Essa troca altera a ordem que as tarefas são executadas, assim mudando o tempo de ciclo da estação de trabalho. O MED só gera movimentos viáveis, gerando soluções que respeitam a precedências das tarefas. Pode-se definir o MED da seguinte forma:

1. Escolhe-se uma estação de trabalho aleatoriamente.
2. Aleatoriamente obtêm-se uma tarefa da estação de trabalho definida.
3. Aleatoriamente é definido se a tarefa irá de um movimento para esquerda ou direita.
4. Valida-se se o movimento irá gerar uma solução viável.
5. Enquanto a validação retornar verdadeiro para solução inviável, realiza-se os passos 1, 2 e 3, e em seguida realiza-se o passo 4, e é validado se a solução é viável para aquele movimento.
6. O movimento realiza a troca de uma tarefa entre duas estações de trabalho.

A Figura 4 apresenta um exemplo onde a estação 5 é escolhida para ter uma tarefa removida. A tarefa 25 é escolhida aleatoriamente. Em seguida, é escolhido um movimento aleatoriamente para esquerda, movendo a tarefa para a estação 4. A matriz solução após o movimento de troca é exibida na Figura 5.

Posto de Trabalho 1	1	5	6	8	7	2	17
Posto de Trabalho 2	12	18	10				
Posto de Trabalho 3	9	14	15	19			
Posto de Trabalho 4	20	16	3				
Posto de Trabalho 5	13	4	21	25			
Posto de Trabalho 6	22	23	24	26	27		
Posto de Trabalho 7	11	28					
Posto de Trabalho 8	29	30	31	32	33	35	34

Figura 4 – Matriz Solução antes do movimento esquerda ou direita.

Posto de Trabalho 1	1	5	6	8	7	2	17
Posto de Trabalho 2	12	18	10				
Posto de Trabalho 3	9	14	15	19			
Posto de Trabalho 4	20	16	3	25			
Posto de Trabalho 5	13	4	21				
Posto de Trabalho 6	22	23	24	26	27		
Posto de Trabalho 7	11	28					
Posto de Trabalho 8	29	30	31	32	33	35	34

Figura 5 – Matriz Solução depois do movimento esquerda ou direita.

3.2.2 MOVIMENTO TROCA DE TAREFA ALEATORIAMENTE ENTRE AS ESTAÇÕES DE TRABALHO

O movimento aleatório (MA) realiza a troca de tarefas aleatoriamente entre estações de trabalhos. Essa troca altera a ordem que as tarefas são executadas, portanto modificando o tempo de ciclo da estação trabalho que é retirada a tarefa, e da

estação de trabalho que recebe a tarefa. O MA é dividido em dois métodos, um que só gera movimentos aleatórios viáveis (MAV) e outro método que gera movimentos aleatórios viáveis e inviáveis (MAVI), toda solução inviável é penalizada somando seu tempo de ciclo por um valor empírico. Pode-se definir o MA da seguinte forma:

3.2.2.1 MOVIMENTO VIÁVEL

1. É definida uma estação de trabalho aleatoriamente para ser movida uma tarefa.
2. A tarefa é definida aleatoriamente da estação de trabalho de origem.
3. Uma nova estação de trabalho de destino é selecionada aleatoriamente para receber a tarefa.
4. Uma validação é realizada para verificar se movimento gera uma solução viável.
5. Enquanto a validação retornar false para uma solução viável, realiza-se os passos 1, 2 e 3, e em seguida realiza-se o passo 4, e é validado se a solução é viável para aquele movimento.
6. O movimento realiza a troca de uma tarefa entre duas estações de trabalho.

3.2.2.2 MOVIMENTO VIÁVEL OU INVIÁVEL

1. É definida uma estação de trabalho origem e uma estação de trabalho destino aleatoriamente.
2. Sorteia-se uma tarefa para ser retirada da estação de trabalho origem.
3. A tarefa sorteada é adicionada a estação de trabalho de destino.
4. O movimento gera uma solução viável ou inviável, caso uma solução não respeite as restrições de precedências, a mesma será penalizada no cálculo da função objetivo.
5. O movimento realiza a troca de uma tarefa entre duas estações de trabalho.

A Figura 6 apresenta um exemplo onde a estação de trabalho 8 é escolhida como origem e a estação 2 é escolhida como destino, é sorteada na estação de trabalho a tarefa 33, para se mover para a estação de trabalho de destino. A matriz solução após o movimento de troca é exibida na Figura 7.

Posto de Trabalho 1	1	5	6	8	7	2	17
Posto de Trabalho 2	12	18	10		⇦		
Posto de Trabalho 3	9	14	15	19			
Posto de Trabalho 4	20	16	3	25			
Posto de Trabalho 5	13	4	21				
Posto de Trabalho 6	22	23	24	26	27		
Posto de Trabalho 7	11	28					
Posto de Trabalho 8	29	30	31	32	33 ⇩	35	34

Figura 6 - Matriz Solução antes do movimento aleatório.

Posto de Trabalho 1	1	5	6	8	7	2	17
Posto de Trabalho 2	12	18	10	33			
Posto de Trabalho 3	9	14	15	19			
Posto de Trabalho 4	20	16	3	25			
Posto de Trabalho 5	13	4	21				
Posto de Trabalho 6	22	23	24	26	27		
Posto de Trabalho 7	11	28					
Posto de Trabalho 8	29	30	31	32	35	34	

Figura 7 - Matriz Solução depois do movimento aleatório.

3.3 META-HEURÍSTICA SIMULATED ANNEALING

A meta-heurística escolhida para resolver o problema SALBP-2 é o *Simulated Annealing* (SA) (KIRKPATRICK et al., 1983). O SA vem apresentando bons resultados em vários problemas de otimização combinatória (RIBEIRO, MAURI, LORENA, 2011; AHONEN, ALVARENGA, AMARAL, 2013; CELLIN, AMARAL, 2016; BISSOLI et al., 2018).

O SA baseia-se em uma analogia da termodinâmica, no processo de recozimento de metais. Se um metal líquido é resfriado lentamente, seus átomos formam um cristal puro que corresponde ao estado da energia mínima para o metal. O metal atinge um estado com maior energia se for resfriado rapidamente (KIRKPATRICK et al.,1983).

Segundo Kirkpatrick et al. (1983), o SA é um método de busca local que admite passos de piora como estratégia, para fugir de possíveis ótimos locais, uma solução s gerada é dada como entrada ao SA que gera através de uma perturbação uma solução s' vizinha de s. A cada geração de s' é realizada a verificação da variação da função objetivo Δ, onde $\Delta = f(s') - f(s)$. Caso $\Delta \leq 0$, a solução s' dispõe de uma função objetivo melhor que s, logo s' torna-se a solução s. Se $\Delta \geq 0$ a probabilidade de aceitar a solução é calculada da seguinte forma: $e^{(-\Delta/T)}$, onde o $T0$ é a temperatura com um valor preliminarmente definido.A possibilidade de aceitação, faz com que ajuste a aceitação ou não de soluções piores que a atual. À medida que a temperatura se aproxima de zero, a probabilidade de se aceitar soluções piores é diminuída. Ao chegar na temperatura de congelamento, o algoritmo retorna um ótimo local e assim o método é encerrado.

É proposto um aprimoramento no SA, sendo incorporado um método de busca local (MBL), para melhorar a melhor solução encontrada, onde irá ser realizado movimentos de troca gerando novos vizinhos para a melhor solução obtida até o momento, a melhor solução encontrada assumi a posição de melhor solução no *Simulated Annealing* (JÚNIOR; SOUZA; MARTINS, 2005).

3.4 PSEUDOCÓDIGO DO SIMULATED ANNEALING PARA O SALBP-2

Algoritmo 1: PSEUDOCÓDIGO DO SIMULATED ANNEALING

```
Entrada: Solução S, IterT, TC,T0, β, SAmax
Saída: Solucão S*
1  início
2      T ← T0;
3      S* ← S;
4      IterT ← 0;
5      enquanto T > TC faça
6          enquanto IterT < SAmax faça
7              IterT ← IterT+1;
8              Gere um número aleatório k ∈ (0, 1);
9              se k = 0 então
10                 S' ← GerarVizinhoMED(S);
11             fim
12             senão
13                 S' ← GerarVizinhoMAV(S);
14             fim
15             Δfo ← fo(S') − fo(S);
16             se Δfo <= 0 então
17                 S ←S';
18                 se fo(S') <= fo(S*) então
19                     Melhore s' usando MBL;
20                     S* ←S';
21                 fim
22             fim
23             senão
24                 Gere um número aleatório p ∈ (0, 1);
25                 se p < e^(−Δfo/T) então
26                     S ←S';
27                 fim
28             fim
29         fim
30         T ← T * β;
31         IterT ← 0;
32     fim
33 fim
```

Figura 8 - Pseudocódigo para o Simulated Annealing.

3.5 SOLUÇÃO INICIAL

Inicialmente uma solução s é obtida através de uma heurística construtiva aleatória para o SALBP-2. É recebida como entrada uma instância, com um conjunto de n tarefas, com seus custos e seus predecessores e um conjunto de m máquinas.

O processo de construção baseia-se em receber uma tarefa$n$ e realizar uma validação para saber a viabilidade da solução. Se retornar -1, onde esse valor é definido empiricamente para definir um status de uma tarefa que pode ser inserida em qualquer estação de trabalho, caso contrário irá retornar a última estação de trabalho que essa tarefa pode ser inserida. Sabendo quais estações de trabalho são viáveis, aleatoriamente escolhe-se uma estação de trabalho e insere-se a tarefa na matriz solução. Tal procedimento é repetido para todas tarefas. Por fim é gerada uma solução s viável. A Figura 9 apresenta o pseudocódigo para a construtiva aleatória.

Algoritmo 2: ALGORITMO DE CONSTRUÇÃO ALEATÓRIA

```
   Entrada: Conjunto de n tarefas
   Saída: Solução S
 1 início
 2 |   para i ← 1 até MaxIter faça
 3 |   |   k ← verificarViabilidadeEstacoes(i,S);
 4 |   |   se k = -1 então
 5 |   |   |   idEstacao ← sortearEstacao();
 6 |   |   fim
 7 |   |   se k != -1 então
 8 |   |   |   idEstacao ← sortearAleatorioEstacaoViavel();
 9 |   |   fim
10 |   |   S[idEstacao][controlador] ←idtarefa;
11 |   fim
12 fim
```

Figura 9 - Pseudocódigo para a construtiva aleatória.

3.6 BUSCA LOCAL

Como mecanismo de busca local foi definida uma heurística que tem como objetivo gerar n soluções usando um método que gera um vizinho s' por x vezes tem de se obter a melhor solução gerada entre os vizinhos.

A geração de vizinhança é usada o método MAVI, onde irá gerar soluções viáveis e inviáveis. Se a busca resultar em uma solução melhor que a atual, esta passará a ocupar o posto de melhor solução e a busca local realizará novas buscas a partir da melhor solução. Caso a busca apresente piora em relação a melhor solução, esta será desfeita e a busca local continua considerando a melhor solução anterior.

3.7 FUNÇÃO OBJETIVO

A função objetivo visa minimizar o tempo de ciclo entre as estações de trabalho. Com isso nesse trabalho é adotada técnicas de penalização da função objetivo. Quando uma solução se mostrar inviável, por alguma tarefa não respeitar sua restrição de precedência, seja $f(s)$ a função objetivo da solução s, um valor de penalização p definido por um valor empírico que seja maior que soma dos custos das tarefas.

Cada valor empírico de penalização foi calculado usando testes exaustivos de geração de vizinhança, onde o valor que houvesse maior geração de soluções viáveis era definido como valor de penalização, com isso o valor definido é acrescentado a $f(s)$ sempre que este violar uma restrição de precedência entre as tarefas. A função objetivo é calculada a partir do maior valor de tempo de ciclo entre as estações de trabalho.

4. RESULTADOS

Os resultados dividem-se em duas seções, a configuração dos experimentos que será abordado na seção 4.1 e análise dos resultados abordado na seção 4.2.

4.1 CONFIGURAÇÃO DOS EXPERIMENTOS

A meta-heurística *Simulated Annealing* (SA) foi modelada e implementada na linguagem C++ e compilada com o G++. Os testes foram realizados em um computador com processador Intel Core I5 2.71GHz, 8GB de memória RAM e em um Sistema Operacional *Windows 10 Home Single Language*. Para os mesmos, foram utilizadas instâncias testadas por Bissoli e Amaral (2016) para solução do problema SALBP-2 aplicando a meta-heurística GRASP, nas quais a instâncias da literatura usadas foram, *Gunther, Hahn, Barthold*.

Cada instância possui um número máximo de tarefas, a Tabela 4 mostra a relação do número de tarefas por instância.

Tabela 4 - Representação das quantidades de tarefas por instância.

Grafo	Quantidade de Tarefas
Gunther	35
Hahn	53
Barthold	148

Cada instância apresenta um valor *p* de penalização definido, a instância *Gunther* possui *p* igual a 1000, a instância *Hahn* possui *p* igual a 15000, a instância *Barthold* possui *p* igual a 10000.

Para cada instância testada foi realizada uma calibragem para o SA, foi definido por experiência prática que o parâmetro α tem o valor fixado como 0,975, onde possui um resfriamento lento. Para precisão computacional a temperatura de congelamento (TC) é fixada em 0.001.

A temperatura inicial (T0) e os número de iterações para atingir o equilíbrio térmico (SAmax) foram calibrados usando valores empíricos junto a testes computacionais exaustivos, no final são definidos os parâmetros para a determinada instância. As Tabelas 5, 6 e 7 apresentam a calibragem do Simulated Annealing para as instâncias *Gunther*, *Hahn, Barthold*.

Tabela 5 - Representação dos parâmetros do SA para a instância Gunther.

Grafo	*m*	TC	α	SAmax	T0
	6	*0.001*	*0.975*	20000	1000
	7	*0.001*	*0.975*	5000	40000
	8	*0.001*	*0.975*	5000	60000
	9	*0.001*	*0.975*	5000	60000
Gunther	10	*0.001*	*0.975*	5000	30000
	11	*0.001*	*0.975*	1000	40000
	12	*0.001*	*0.975*	1000	30000
	13	*0.001*	*0.975*	1000	30000
	14	*0.001*	*0.975*	1000	30000
	15	*0.001*	*0.975*	1000	20000

Tabela 6 - Representação dos parâmetros do SA para a instância Hahn.

Grafo	*m*	TC	α	SAmax	T0
	3	*0.001*	*0.975*	5000	100000
	4	*0.001*	*0.975*	5000	100000
	5	*0.001*	*0.975*	5000	100000
	6	*0.001*	*0.975*	1000	30000
Hahn	7	*0.001*	*0.975*	1000	50000
	8	*0.001*	*0.975*	5000	100000
	9	*0.001*	*0.975*	10000	100000
	10	*0.001*	*0.975*	10000	300000

Tabela 7 - Representação dos parâmetros do SA para a instância Barthold.

Grafo	*m*	TC	α	SAmax	T0
	3	*0.001*	*0.975*	1000	10000
	4	*0.001*	*0.975*	1000	10000
	5	*0.001*	*0.975*	1000	50000
	6	*0.001*	*0.975*	1000	50000
	7	*0.001*	*0.975*	1000	50000
Barthold	8	*0.001*	*0.975*	5000	100000
	9	*0.001*	*0.975*	1000	50000
	10	*0.001*	*0.975*	10000	100000
	11	*0.001*	*0.975*	10000	100000
	12	*0.001*	*0.975*	1000	50000
	13	*0.001*	*0.975*	5000	100000
	14	*0.001*	*0.975*	5000	100000
	15	*0.001*	*0.975*	5000	100000

4.2 ANÁLISES DOS RESULTADOS

Para obter os resultados dos experimentos computacionais, a meta-heurística *Simulated Annealing (SA)* foi executada 10 vezes para cada número m de estações de trabalho de cada instância.

A Tabela 8, a Tabela 9 e a Tabela 10 apresentam os resultados da seguinte maneira: As duas primeiras colunas representam as instâncias usadas nos experimentos computacionais, onde a primeira representa o grafo de precedência e na segunda o número de estações de trabalho (*m*). A terceira coluna representa *o Best Known Solution* (BKS), onde é a melhor solução conhecida pela literatura (BLUM, 2011). A quarta coluna apresenta a melhor solução obtida pelo *Simulated Annealing* dentre as dez execuções. A quinta coluna representa a diferença percentual entre a melhor solução alcançada e o BKS. A sexta e sétima coluna apresentam a média das soluções geradas e o tempo médio das dez execuções realizada pelo SA. A oitava coluna apresenta o desvio padrão entre as 10 soluções encontrada pelo SA.

Tabela 8 - Representação dos resultados para a instância Gunther.

Grafo	*m*	BKS	Melhor	Diferença Percentual	Média	T(s)	Desvio Padrão
	6	84	**84**	0,00%	84,00	72,20	0,00%
	7	72	**72**	0,00%	72,70	85,30	1,72%
	8	63	**63**	0,00%	63,60	28,60	1,10%
	9	54	**54**	0,00%	58,40	33,50	3,72%
Gunther	10	50	**50**	0,00%	50,70	40,30	1,62%
	11	48	**48**	0,00%	48,20	8,30	0,87%
	12	44	**44**	0,00%	44,60	85,30	1,16%
	13	42	**42**	0,00%	43,10	25,00	2,31%
	14	40	**40**	0,00%	41,80	69,60	2,47%
	15	40	**40**	0,00%	41,10	30,30	2,42%
Média		53,70	53,70	0,00%	54,82	47,84	1,74%

Tabela 9 - Representação dos resultados para a instância Hahn.

Grafo	*m*	BKS	Melhor	Diferença Percentual	Média	T(s)	Desvio Padrão
	3	4787	**4787**	0,00%	44788,00	147,60	1,42%
	4	3677	**3677**	0,00%	3788,50	148,70	9,26 %
	5	2823	**2823**	0,00%	2850,60	159,90	1,89%
	6	2400	2419	0,56%	2445,80	241,00	1,76%
Hahn	7	2336	**2336**	0,00%	2400,50	87,00	1,44%
	8	1907	1917	0,37%	2233,00	261,80	10,83%
	9	1827	1827	0,00%	2123,00	597,30	14,69%
	10	1775	1792	0,67%	2236,00	598,00	12,99%
Média		2691,50	2697,25	0,20%	7858,18	280,16	6,79%

Tabela 10 - Representação dos resultados para a instância Barthold.

Grafo	*m*	BKS	Melhor	Diferença Percentual	Média	T(s)	Desvio Padrão
	3	1878	**1878**	0,00%	1878,88	135,10	0,03%
	4	1409	**1409**	0,00%	1497,22	135,50	8,76%
	5	1127	**1127**	0,00%	1132,44	285,80	0,41%
	6	939	940	0,08%	969,33	1087,40	4,33%
	7	805	807	0,18%	856,66	1120,40	10,85%
Barthold	8	705	817	10,41%	1675,77	214,00	26,92%
	9	626	1375	52,94%	1947,11	982,90	14,52%
	10	564	1448	62,14%	2015,11	1049,90	15,98%
	11	513	708	22,59%	1586,88	1170,10	25,65%
	12	470	2271	92,92%	2514,11	183,90	5,42%
	13	434	920	50,76%	2406,77	563,20	41,92%
	14	403	2621	103,73%	3169,22	545,50	12,01%
	15	383	2969	109,10%	3225,55	825,80	7,70%
Média		788,92	1483,85	38,83%	1913,47	638,42	13,42

Os experimentos computacionais mostram que algoritmo SA obteve êxito em se aproximar ou encontrar os valores das melhores soluções conhecidas (BKS) para as instâncias *Gunther* e *Hahn* com um desvio padrão médio de 1,74% e 6,79%, já para a instância *Barthold* obteve um desvio médio padrão de 13,42%

O resultado para instância *Barthold* mostram que o SA não obteve êxito em encontrar todos os BKS ou se aproximar das melhores soluções conhecidas, só se obteve as melhores soluções conhecidas para *m* entre 3 a 5, para *m* entre 6 e 7 obteve uma diferença de 0,08% e 0,18% em relação aos BKS, para *m* de 8 a 15 obteve resultados distantes tendo uma diferença entre 10,41% a 109,10% em relação dos valores das melhores soluções conhecidas.

5. CONCLUSÕES

O presente trabalho abordou o problema de balanceamento de linha de montagem tipo 2 (SALBP-2), que é uma variação do SALBP. Para resolução do problema proposto, foi desenvolvida uma meta-heurística *Simulated Annealing.*

Os resultados obtidos pelo Simulated *Annealing* para instâncias com tamanho entre 35 a 53 tarefas foram satisfatórias, obtendo uma média da diferença percentual de 0,09% em relação às melhores soluções conhecidas. Para os resultados com a instância de tamanho de 148 tarefas o SA não obteve resultados satisfatórios, obtendo uma média da diferença percentual de 38,83% em relação às melhores soluções conhecidas. Para todos os casos testados a meta-heurística *Simulated Annealing* obteve soluções viáveis e foi capaz de alcançar algumas das melhores soluções conhecidas.

Em trabalhos futuros, para explorar instâncias mais complexas, sugere-se realizar estudos sobre técnicas de calibragem para o algoritmo SA, e por fim propor uma combinação do SA com outra meta-heurística para obtenção de melhores resultados.

6. REFERÊNCIAS

AHONEN, Hannu; DE ALVARENGA, Arlindo Gomes; AMARAL, André RS. Simulated annealing and tabu search approaches for the corridor allocation problem. **European Journal of Operational Research**, v. 232, n. 1, p. 221-233, 2014.

BAUDIN, Michel. **Lean assembly: the nuts and bolts of making assembly operations flow**. CRC Press, 2002.

BECKER, Christian; SCHOLL, Armin. A survey on problems and methods in generalized assembly line balancing. **European journal of operational research**, v. 168, n. 3, p. 694-715, 2006.

DE CASTRO BISSOLI, Dayan; AMARAL, André Renato Sales. UM ALGORITMO GRASP PARA O SALBP-2. In: Simpósio Brasileiro de Pesquisa Operacional, 2016, Vitória - ES. Anais do XLVIII SBPO, 2016. p. 2081-2091.

BISSOLI, DAYAN C.; S. ALTOE, WAGNER A.; MAURI, GERALDO R.; S. AMARAL, ANDRE R. A simulated annealing metaheuristic for the bi-objective flexible job shop scheduling problem. In: 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE), **2018, San Salvador. 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)**, 2018. p. 1

BLUM, C. Iterative beam search for simple assembly line balancing with a fixed number of work stations. **Statistics and Operations Research Transactions**, 35(2):145–164, 2011.

CARNAHAN, Brian J.; NORMAN, Bryan A.; REDFERN, Mark S. Incorporating physical demand criteria into assembly line balancing. **Iie Transactions**, v. 33, n. 10, p. 875-887, 2001.

CELLIN, B. P.; AMARAL, A. R. S. Um algoritmo Simulated Annealing e um híbrido VNS/ILS para resolver o problema de layout em múltiplas linhas. In: Simpósio Brasileiro de Pesquisa Operacional, 2016, Vitória - ES. Anais do XLVIII SBPO, 2016. p. 2092-2103.

GAITHER, Norman; FRAZIER, Greg. **Production and operations management**. Thomson South-Western, 1999.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introdução à pesquisa operacional**. McGraw Hill Brasil, 2013.

JÚNIOR, Aloísio de Castro Gomes; SOUZA, Marcone Jamilson Freitas; MARTINS, Alexandre Xavier. Algoritmos Simulated Annealing eficientes para resolução do problema de roteamento de veículos com janela de tempo. In: Simpósio Brasileiro de Pesquisa Operacional, 2005, Gramado - RS. Anais do XXXVII SBPO, 2005. p. 1271-1281.

KILINCCI, Ozcan. A Petri net-based heuristic for simple assembly line balancing problem of type 2. **The International Journal of Advanced Manufacturing Technology**, v. 46, n. 1-4, p. 329-338, 2010.

KIRKPATRICK, Scott; GELATT, C. Daniel; VECCHI, Mario P. Optimization by simulated annealing. **science**, v. 220, n. 4598, p. 671-680, 1983.

KLEIN, Robert; SCHOLL, Armin. Maximizing the production rate in simple assembly line balancinga branch and bound procedure. **European Journal of Operational Research**, v. 91, n. 2, p. 367-385, 1996.

KRIENGKORAKOT, Nuchsara; PIANTHONG, Nalin. The assembly line balancing problem: review articles. **Engineering and Applied Science Research**, v. 34, n. 2, p. 133-140, 2007.

NEARCHOU, Andreas C. Balancing large assembly lines by a new heuristic based on differential evolution method. **The International Journal of Advanced Manufacturing Technology**, v. 34, n. 9-10, p. 1016-1029, 2007.

PAIVA, E.; CARVALHO, Jr. L.; FENSTERSEIFER, J. **Estratégia de produção e de operações. Porto Alegre: Bookman**, 2004.

RIBEIRO, Glaydston Mattos; MAURI, Geraldo Regis; LORENA, Luiz Antonio Nogueira. A simple and robust Simulated Annealing algorithm for scheduling workover rigs on onshore oil fields. **Computers & Industrial Engineering**, v. 60, n. 4, p. 519-526, 2011.

SALVESON, Melvin E. The assembly line balancing problem. **The Journal of Industrial Engineering**, p. 18-25, 1955.

SCHOLL, Armin; BECKER, Christian. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. **European Journal of Operational Research**, v. 168, n. 3, p. 666-693, 2006.

SCHOLL, Armin; BOYSEN, Nils; FLIEDNER, Malte. Optimally solving the alternative subgraphs assembly line balancing problem. **Annals of Operations Research**, v. 172, n. 1, p. 243, 2009.

SCHOLL, Armin; BOYSEN, Nils; FLIEDNER, Malte. The sequence-dependent assembly line balancing problem. **OR Spectrum**, v. 30, n. 3, p. 579-609, 2008.

SIKORA, Celso Gustavo Stall et al. Genetic algorithm for type-2 assembly line balancing. In: **2015 Latin America Congress on Computational Intelligence (LA-CCI)**. IEEE, 2015. p. 1-6.

SIMARIA, Ana Sofia; VILARINHO, Pedro M. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. **Computers & Industrial Engineering**, v. 47, n. 4, p. 391-407, 2004.

SOUZA, Mariella Consoni Florenzano et al. Análise da alocação de mão-de-obra em linhas de multimodelos de produtos com demanda variável através do uso da simulação: um estudo de caso. **Revista Produção**, v. 13, n. 3, p. 63, 2003.

TALBOT, F. Brian; PATTERSON, James H. An integer programming algorithm with network cuts for solving the assembly line balancing problem. **Management Science**, v. 30, n. 1, p. 85-99, 1984.

TASAN, Seren Ozmehmet; TUNALI, Semra. A review of the current applications of genetic algorithms in assembly line balancing. **Journal of intelligent manufacturing**, v. 19, n. 1, p. 49-69, 2008.

ZACHARIA, P. Th; NEARCHOU, Andreas C. Multi-objective fuzzy assembly line balancing using genetic algorithms. **Journal of Intelligent Manufacturing**, v. 23, n. 3, p. 615-627, 2012.

ZHANG, Haijun et al. An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2. **Assembly Automation**, v. 36, n. 3, p. 246-261, 2016.