Mário Ribeiro Júnior

Algoritmo ILS para o Problema de Balanceamento de Linhas de Montagem

ALEGRE - ES

2025

Mário Ribeiro Júnior

Algoritmo ILS para o Problema de Balanceamento de Linhas de Montagem

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

Universidade Federal do Espírito Santo

Centro de Ciências Exatas, Naturais e da Saúde

Departamento de Computação

Orientador: Prof. Dr. Edmar Hell Kampke

ALEGRE - ES

2025

Mário Ribeiro Júnior

Algoritmo ILS para o Problema de Balanceamento de Linhas de Montagem

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

Aprovado em 20 de Março de 2025.

COMISSÃO EXAMINADORA

Prof. Dr. Edmar Hell Kampke
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Dayan de Castro Bissoli
Universidade Federal do Espírito Santo

Prof. Dr. Geraldo Regis Mauri
Universidade Federal do Espírito Santo

Digitalizado com CamScanner

Agradecimentos

Primeiramente, agradeço a Deus, fonte de toda sabedoria e força, por ter me sustentado em cada passo desta caminhada acadêmica.

Aos meus pais Mário e Joana, que sempre estiveram ao meu lado, me incentivando, acreditando em mim e me dando todo o apoio necessário para que eu pudesse chegar até aqui. Seus ensinamentos e amor incondicional foram fundamentais nessa jornada. Amo vocês!

À minha namorada, Adriele, pelo carinho, paciência e apoio inabalável em toda minha vida. Sua presença foi um grande alicerce para mim, trazendo motivação e conforto nos momentos de desafio. Sou grato por ter alguém tão especial ao meu lado. Amo você!

Ao meu orientador Edmar, pelo ensinamento, dedicação e incentivo ao longo desta caminhada. Sua orientação foi fundamental para o meu crescimento acadêmico e pessoal, e sou imensamente grato pelo conhecimento compartilhado e pelo apoio nesta trajetória.

Aos meus professores, que ao longo dos anos compartilharam seus conhecimentos e experiências, contribuindo para minha formação acadêmica e profissional.

Aos meus amigos pelo companheirismo. Em especial Álvaro, Cleiton, Daniel e Luann pelo apoio e por tornarem essa trajetória mais leve e significativa.

À Universidade Federal do Espírito Santo, pela oportunidade.

Resumo

Este trabalho aborda a aplicação do algoritmo *Iterated Local Search* (ILS) para resolver o Problema de Balanceamento de Linhas de Montagem Simples do tipo 2 (PBLMS-2). O PBLMS-2 é um problema de otimização com implicações significativas na eficiência e produtividade de linhas de montagem industriais. O objetivo principal é distribuir tarefas entre as estações de trabalho de maneira equilibrada e respeitando as restrições de precedência entre as tarefas, minimizando o tempo total de produção. O algoritmo ILS, uma meta-heurística iterativa baseada em busca local, é proposto como uma abordagem eficaz para encontrar soluções para o PBLMS-2. Ao aplicar o ILS ao PBLMS-2, foram encontrados resultados próximos aos melhores resultados conhecidos da literatura. As soluções obtidas com o ILS foram comparadas com os melhores resultados da literatura, como também comparadas com resultados da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP). Isso proporciona uma base sólida para avaliar a eficácia do ILS em comparação com outras abordagens e para compreender melhor o cenário atual da pesquisa neste domínio crítico da otimização industrial. Os resultados do ILS ficaram, respectivamente, 0,72% e 0,47% de distância entre os melhores resultados conhecidos e os resultados do algoritmo GRASP proposto na literatura.

Palavras-chave: Problema de Balanceamento de Linhas de Montagem Simples, *Iterated Local Search*, Otimização Combinatória, meta-heurísticas.

Abstract

This work addresses the application of the Iterated Local Search (ILS) algorithm to solve the Simple Assembly Line Balancing Problem Type 2 (SALBP-2). The SALBP-2 is an optimization problem with significant implications for the efficiency and productivity of industrial assembly lines. The main objective is to distribute tasks among workstations in a balanced manner while respecting precedence constraints between tasks, minimizing the total production time. The ILS algorithm, an iterative metaheuristic based on local search, is proposed as an effective approach to finding solutions for the SALBP-2. By applying ILS to SALBP-2, results close to the best-known results in the literature were obtained. The solutions achieved with ILS were compared with the best results from the literature, as well as with results from the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic. This provides a solid foundation for evaluating the effectiveness of ILS compared to other approaches and for better understanding the current research landscape in this critical domain of industrial optimization. The results of ILS were, respectively, 0.72% and 0.47% away from the best-known results and the results of the GRASP algorithm proposed in the literature.

Keywords: Simple Assembly Line Balancing Problem, Iterated Local Search, Combinatorial Optimization, Metaheuristics.

Lista de Figuras

Figura 1 – Classificações para PBLMs adaptado de Uddin e Lastra (2011) 14

Figura 2 – Solução para um grafo de precedência. 19
Figura 3 – Distribuição inicial de tarefas nas estações de trabalho. 22
Figura 4 – Transferência da última tarefa da primeira estação de trabalho para a primeira posição da segunda estação de trabalho. 22
Figura 5 – Resultado após realizar todas as possibilidades do primeiro tipo de movimento. 23
Figura 6 – Distribuição após primeira iteração realizada do segundo tipo de movimento. 23
Figura 7 – Vetor V preenchido com a última tarefa. 24
Figura 8 – Lista de candidatos a serem adicionados no vetor V. 24
Figura 9 – Vetor V preenchido com a nova ordenação. 24

Figura 10 – Exemplo de instância com nove tarefas. 26

Lista de Algoritmos

1 Algoritmo *ILS* adaptado de Lourenço, Martin e Stützle (2003) 20

Lista de Tabelas

Tabela 1 – Ambiente Computacional. 27
Tabela 2 – Resultados obtidos pelo algoritmo ILS. 28
Tabela 3 – Comparação dos resultados do ILS com os obtidos por Bissoli e Amaral (2016). 29

Lista de Abreviaturas e Siglas

CCENS	Centro de Ciências Exatas, Naturais e da Saúde
DCOMP	Departamento de Computação
FO	Função Objetivo
GRASP	*Greedy Randomized Adaptive Search Procedure*
ILS	*Iterated Local Search*
PBLM	Problema de Balanceamento de Linhas de Montagem
UFES	Universidade Federal do Espírito Santo

Sumário

1 Introdução **11**
1.1 O Problema e sua Importância 11
1.2 Objetivos 12
1.2.1 Objetivo Geral 12
1.2.2 Objetivos Específicos 12

2 Revisão de Literatura **14**
2.1 PBLM 14
2.2 ILS 16

3 Metodologia **17**
3.1 PBLMS-2 17
3.2 *ILS* 19
3.2.1 Solução Inicial 19
3.2.2 Busca Local 21
3.2.3 Perturbação 23
3.2.4 Aceitação 24

4 Resultados Obtidos **25**
4.1 Experimentos computacionais 25
4.1.1 Instâncias 25
4.1.2 Escolha de parâmetros 26
4.1.3 Detalhes da implementação e execução de testes 26
4.2 Resultados Computacionais 27
4.3 Análise e Discussão 28

5 Conclusão **30**

Referências **31**

1 Introdução

Na história como um todo, mas principalmente no mundo empresarial, há a busca ávida e incessante de excelência operacional e do uso eficiente de recursos. Entretanto, nos anos recentes, esses fatores tornaram-se mais proeminentes e elementos-chave neste cenário. Ambos os fatores são imprescindíveis a serem cumpridos para se garantir a competitividade das organizações empresariais (SLACK et al., 2009). Essa busca por excelência operacional é a chave para o sucesso industrial, e se torna mais visível em indústrias manufatureiras, já que essa área é determinante, devido a sua influência no crescimento e sustentabilidade empresarial (SARKER; SHANTHIKUMARI, 1983).

Nas indústrias manufatureiras os lucros tem um alto valor de importância e acaba sendo um objetivo em comum entre todas elas, independente de sua área de atuação (HASAN; AHMAD, 2018). Para ter uma boa otimização de recursos, se faz necessário realizar um planejamento adequado das suas linhas de montagem, já que as mesmas influenciam diretamente na otimização de distribuição de recursos, redução nos tempos de fabricação, eliminação de atrasos e estoques. Dessa forma, é importante garantir que os produtos entregues estejam com a qualidade desejada e dentro dos prazos exigidos (BAYBARS, 1986). Nesse contexto, o Problema de Balanceamento de Linha de Montagem (PBLM) se torna essencial e, uma vez equilibrado de forma satisfatória, o ciclo de produção acaba sendo mais eficiente num contexto operacional, sustentando a alta qualidade do produto final (REGO et al., 2011).

1.1 O Problema e sua Importância

O PBLM apesar de sua complexidade é extremamente fundamental, porque é amplamente utilizável em muitos setores industriais, como automotivo, mecânico e eletrônico (EREL; SARIN, 1998). Além disso, o desenvolvimento tecnológico contínuo, juntamente com a crescente solicitação por produtos altamente personalizados, e ao mesmo tempo com o estímulo de prazos de entrega mais curtos, podem causar gargalos na produção, tornando esse problema cada vez mais relevante (FARNES; PEREIRA, 2007).

Uma linha de montagem possui várias estações de trabalho, cada uma com capacidades e habilidades distintas. Além disso, também existem tarefas únicas, que possuem um determinado tempo de processamento associado a elas, que se refere ao tempo que uma estação de trabalho demora para concluir essa determinada tarefa. Podem existir também restrições de precedência de tarefas, que determina a sequência na qual as tarefas devem ser executadas, levando em consideração as dependências do processo (SALVESON, 1955).

Portanto, de acordo com Scholl (1999) e Kriengkorakot e Pianthong (2007) o PBLM consiste no desafio de alocar tarefas a estações de trabalho em uma linha de montagem, reduzir o tempo de ciclo, e assim, aprimorar a produção, respeitando as restrições de precedência e as habilidades da estação de trabalho. Este problema é altamente complexo, determinado pela consideração de múltiplos critérios de decisão, constituindo, portanto, um desafio tanto na teoria quanto na prática (REKIEK et al., 2002). Esse tempo é definido como o tempo gasto para produzir um produto do início até a conclusão do processamento, e a busca da redução desse tempo é essencial para a eficiência da produção em grande escala (SCHOLL, 1999).

O PBLM também tem como objetivo conseguir equilibrar razoavelmente a distribuição de tarefas em estações de trabalho. Isso ocasiona na alocação de um número equilibrado de tarefas às estações de trabalho, impedindo o trabalho exacerbado, o que significa uma estação de trabalho estar muito mais sobrecarregada do que as outras, e também a subutilização, que é uma estação ter muito menos tarefas do que as demais, o que por sua vez leva a gargalos na linha de montagem, ao desperdício de recursos investidos e aumento do custos de produção (SCHOLL, 1999).

Esse é um problema NP-difícil descrito por Klein e Scholl (1996). Isso significa que, não se conhece um algoritmo que resolva todos os casos em tempo polinomial. Para resolver esse problema, várias métodos de otimização podem ser utilizados. Esses métodos incluem não somente a programação linear, mas também vários tipos de heurísticas e meta-heurísticas, que buscam encontrar em um tempo curto a melhor distribuição das tarefas nas estações de trabalho, respeitando sempre todas as restrições e especificações do problema (ROCHA et al., 2019).

1.2 Objetivos

Nesta seção são apresentados o objetivo geral, bem como os específicos deste trabalho.

1.2.1 Objetivo Geral

O objetivo geral deste trabalho consiste em aplicar a meta-heurística *Iterated Local Search* (ILS) para resolução do PBLM.

1.2.2 Objetivos Específicos

a) Realizar um estudo de caso com os dados necessários para resolver o PBLM.

b) Implementar a meta-heurística ILS para resolver o PBLM.

c) Realizar testes de validação e calibragem de parâmetros no algoritmo desenvolvido utilizando instâncias específicas do problema básico.

d) Efetuar eventuais correções no algoritmo.

e) Executar a aplicação desenvolvida utilizando as instâcias propostas na literatura.

f) Avaliar o desempenho obtido e compará-lo com os apresentados na literatura.

2 Revisão de Literatura

Nesta seção serão apresentados trabalhos relacionados ao PBLM (Seção 2.1) e ao algoritmo ILS (Seção 2.2).

2.1 PBLM

O PBLM é regularmente classificado por restrições e objetivos, conforme mostrado na Figura 1.

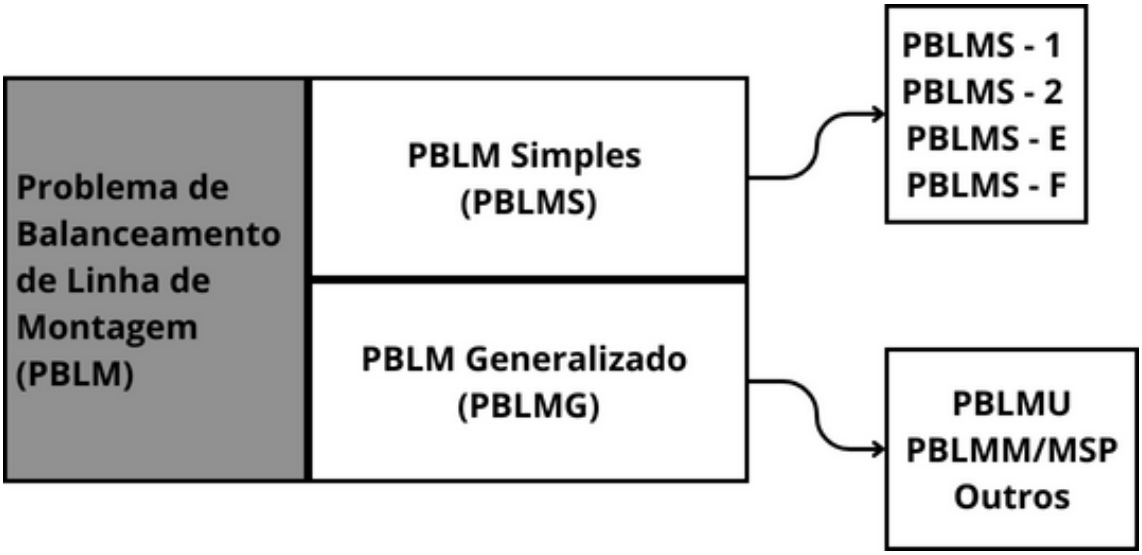

Figura 1 – Classificações para PBLMs adaptado de Uddin e Lastra (2011)

De acordo com a Figura 1, o PBLM é subdividido em duas classes: PBLM Simples (PBLMS) e PBLM Generalizado (PBLMG). O PBLMG é o problema de balanceamento de linha que visa resolver algumas características adicionais, como exemplo: linhas de montagem com *layout* em formato U (PBLMU); Agregação de estações de trabalho em paralelo (PBLMM/MSP); entre outros.

Por outro lado, o PBLMS é extremamente útil para montagem simples de produtos, em uma linha reta, para fornecer uma restrição exclusiva de precedência entre as tarefas. De acordo com Becker e Scholl (2006), o PBLMS possui quatro classificações:

- PBLMS-1: se baseia na distribuição de tarefas nas estações de trabalho de forma que o número de estações (m) é minimizado para uma determinada taxa de produção (tempo de ciclo fixo c), o que é análogo a minimizar o tempo ocioso total;

- PBLMS-2: busca minimizar o tempo de ciclo (maximizar a taxa de produção), submetendo-se a restrições de precedência para um determinado número (m) de estações disponíveis;
- PBLMS-E: é a versão mais geral do problema, que tem como objetivo maximizar a eficiência da linha (E), e paralelamente minimizar c e m, considerando suas correlações;
- PBLMS-F é um problema de possibilidade que se respalda em estabelecer se existe ou não um balanceamento de linha viável para uma dada combinação entre c e m.

O presente trabalho aborda uma resolução para o PBLMS-2. Existem várias abordagens na literatura baseadas em heurísticas e métodos exatos que focam em modelar e resolver o PBLMS-2.

De acordo com Klein e Scholl (1996), o PBLMS-2 é um problema NP-difícil que consiste em atribuir tarefas a um determinado número de estações de trabalho de uma linha de montagem, de forma que a taxa de produção seja maximizada. Além disso, possiveis restrições de precedência entre as tarefas devem ser consideradas. Os métodos existentes para o PBLMS-2 baseiam-se principalmente na resolução repetida de instâncias do PBLMS-1, que visa minimizar o número de estações para uma determinada taxa de produção. O procedimento *Branch and Bound* utlizado por Klein e Scholl (1996) resolveu o PBLMS-2 usando uma nova técnica de enumeração, o método de limite inferior local (*Local Lower Bound Method*), que é complementado por uma série de regras de delimitação e dominância. Os resultados apresentados se mostraram muito eficientes, sendo muito próximos aos melhores resultados conhecidos até aquele momento.

Scholl e Becker (2006) realizaram um levantamento abrangente de pesquisas envolvendo o PBLMS, com ênfase nas contribuições marcantes. Segundo os autores, o PBLM surge e necessita de resolução quando uma linha de montagem precisa ser configurada ou remodelada. Ela consiste na distribuição da carga de trabalho para fabricação de qualquer produto a ser montado entre os postos de trabalho ao longo da linha. Boysen e Fliedner (2008) propuseram uma heurística de dois estágios baseada em um algoritmo de grafos e otimização de colônias de formigas para resolver todas as versões do PBLM. Ao contrário de outros procedimentos, a abordagem apresentada pode ser facilmente modificada para incorporar todas as extensões do PBLM, entretanto, o método genérico não apresentou resultados satisfatórios quando comparados a algoritmos especializados.

Bissoli e Amaral (2016) apresentaram uma abordagem de implementação do algoritmo *Greedy Randomized Adaptive Search Procedure* (GRASP) para resolução do PBLMS-2. No Algoritmo GRASP foi utilizada a estratégia de *First Improvement Method* para um melhoramento iterativo na busca local. Esse melhoramento visa alterar somente a estação

de trabalho que está definindo o tempo de ciclo do problema. Utilizando também alguns conceitos de grafos, por exemplo a ordenação topológica, conseguiu encontrar soluções muito próximas das melhores encontradas até aquele momento em algumas instâncias disponibilizadas na literatura.

2.2 ILS

Também são apresentados na literatura artigos relacionados a utilização da meta-heurística ILS pra resolução de diversos problemas.

Chiarandini, Stützle et al. (2002) utilizou uma abordagem do ILS para o problema de coloração de grafos. O problema em questão tem como definição: para um determinado número de cores, é preciso encontrar uma coloração viável, uma vez encontrada essa coloração, o número de cores diminui e a busca local começa novamente. No artigo em questão foi aplicado o ILS e o mesmo mostrou bons resultados, comparado com os melhores conhecidos, para uma variedade de esquemas de perturbação. Além disso, apresentou resultados computacionais próximos aos melhores conhecidos em algumas instâncias difíceis do conjunto de *benchmark* conhecido como DIMACS.

Stützle (2006) utilizou uma abordagem ILS por meio de uma análise do espaço de busca para o Problema de Alocação Quadrática (PAQ). A primeira versão implementada teve um problema de estagnação que comprometeu o desempenho. Uma segunda implementação corrigiu o problema usando critérios de aceitação, o que permitiu movimentos para ótimos locais piores. Também propuseram extensões de ILS baseadas em população. A análise do ILS aprimorado mostrou desempenho superior quando comparado com outros algoritmos para o PAQ.

Também foi utilizado o ILS para o Problema de Roteamento de Veículos com *Backhauls* (PRVB) por Cuervo et al. (2014). O artigo propõem um algoritmo muito efetivo do ILS para o PRVB. Após a implementação do algoritmo foi utilizada análises estatísticas completas dos componentes da implementação. Também foi provado para o experimento em questão, que ao expandir a área de exploração e melhorar a eficiência da heurística de busca local, é possível desenvolver algoritmos meta-heurísticos mais simples e rápidos, sem comprometer a qualidade das soluções obtidas.

Diante do que foi exposto, não foi encontrado na literatura uma aplicação do ILS para o PBLMS-2. Baseado nas contribuições recentes em problemas correlatos, torna-se interessante a avaliação da implementação do ILS para resolver esse problema, que é o objetivo deste trabalho.

3 Metodologia

Neste trabalho foi implementada a meta-heurística ILS para o PBLMS-2. Na Seção 3.1 e apresentado o modelo matemático do PBLMS-2, e na Seção 3.2 a meta-heurística ILS que foi implementada é descrita.

3.1 PBLMS-2

O PBLMS-2 é responsável por distribuir tarefas sequenciais entre as estações de trabalho. O objetivo é conseguir a máxima utilização do trabalho e equipamento, ao mesmo tempo que procura reduzir o tempo de ciclo da estação de trabalho mais sobrecarregada.

Uma linha de montagem é composta por várias estações de trabalho dispostas sequencialmente em uma estrutura de manuseio de materiais, como uma esteira rolante. Nesse sentido, elas constituem estações de trabalho que propõem mover o material de uma estação de trabalho para a próxima e concluir o trabalho ao fim da linha. A linha de montagem consiste em uma série de tarefas específicas e relacionadas que devem ser concluídas para criar um produto final. Cada estação de trabalho realiza uma tarefa em uma série específica ou relacionada de tarefas. No entanto, em razão das restrições tecnológicas de fabricação e organizacionais, é necessário realizar tarefas simultaneamente e então criar produtos no fim do ciclo de execução. Um aspecto importante de uma linha de montagem é que cada tarefa é associada a uma estação de trabalho específica e cada tarefa, que deve ser concluída, ocorre em um determinado período de tempo. Dado que várias tarefas são realizadas simultaneamente, cada tarefa deve ser atribuída a uma estação de trabalho, assegurando que todas as estações de trabalho tenham um período de tempo suficiente para concluir a tarefa atribuída (BECKER; SCHOLL, 2006).

De acordo com Bissoli (2018), o PBLMS-2 pode ser formulado matematicamente da seguinte forma: Uma instância (T, G, m) consiste em três componentes. $T = 1, ..., n$ é um conjunto de n tarefas. Cada tarefa $i \in T$ tem o tempo de processamento determinado como $t_i > 0$. Sem perda de totalidade, os tempos de processamento são considerados valores inteiros. Além disso, é dado um grafo de precedência que é acíclico e orientado $G = (T, A)$, sendo A seu conjunto de arcos. Portanto, um arco $a_{i,j} \in A$ indica que a tarefa i deve ser processada antes da tarefa j. Dada uma tarefa $j \in T$, $P_j \subset T$ denota o conjunto de tarefas que devem ser processadas antes de j. Por fim, m representa um número determinado de estações de trabalho que estão ordenadas de 1 a m. Uma solução possível é alcançada por intermédio da atribuição de cada tarefa a exatamente uma estação de trabalho, de modo que as restrições de precedência entre as tarefas, representadas em G, sejam satisfeitas.

O modelo matemático para o PBLMS-2 foi proposto por Baybars (1986), sendo descrito da seguinte forma:

$$\text{Minimizar} \quad z \tag{3.1}$$

Sujeito a:

$$\sum_{s=1}^{m} x_{is} = 1, \quad \forall i \in T \tag{3.2}$$

$$x_{is} \leq \sum_{s'=1}^{s-1} x_{js'}, \quad \forall i \in T, \forall j \in P_i, s = 1, ..., m \tag{3.3}$$

$$\sum_{i \in T} t_i x_{is} \leq z, \quad s = 1, ..., m \tag{3.4}$$

$$x_{is} \in \{0, 1\} \quad \forall i \in T, s = 1, ..., m \tag{3.5}$$

$$z > 0 \tag{3.6}$$

Seja $x_{i,s}$ uma variável binária definida como 1 se, e somente se, a tarefa $i \in T$ é atribuída à estação de trabalho s ($1 \leq s \leq m$), e 0 caso contrário (3.5). A função objetivo (3.1) minimiza o tempo de ciclo z. A restrição (3.2) garante que cada tarefa $i \in T$ seja atribuída a uma única estação de trabalho. A restrição (3.3) reflete os relacionamentos de precedência entre as tarefas. Mais especificamente, se a tarefa $i \in T$ é atribuída à estação de trabalho s, todas as tarefas $j \in P_i$ devem ser atribuídas às estações de trabalho s', sendo $s' \leq s$. A restrição (3.4) garante que a soma dos tempos de processamento das tarefas atribuídas às estações de trabalho não excedam o tempo de ciclo z, definido com um valor positivo na restrição (3.6).

Bissoli (2018) apresenta um exemplo de solução para o PBLM-2 conforme mostra a Figura 2. Como é exemplificado, são nove tarefas e quatro estações de trabalho. O grafo da Figura 2 (a) demonstra as precedências entre as tarefas e também o custo de cada tarefa. Já a Figura 2 (b) demonstra as tarefas distribuídas nas estações de trabalho, como também apresenta uma solução para o problema, obtendo a solução $z = 13$.

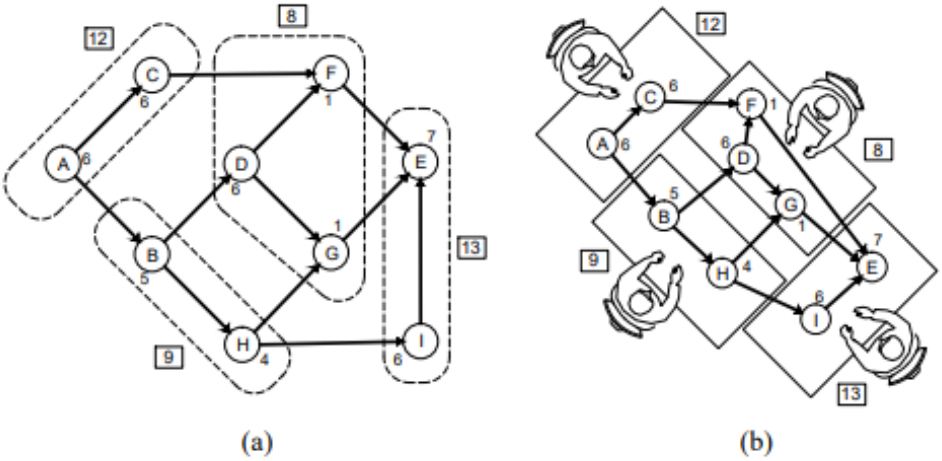

Figura 2 – Solução para um grafo de precedência.

Fonte: Bissoli (2018).

3.2 *ILS*

A meta-heurística *Iterated Local Search* (ILS) foi proposta por Lourenço, Martin e Stützle (2003). O ILS busca gerar novas soluções através de perturbações aplicadas na solução ótima local, possibilitando uma melhor exploração da vizinhança pela busca local (BISSOLI, 2018).

O Algoritmo 1 exemplifica um pseudocódigo genérico do ILS, que possui quatro módulos: o procedimento *SoluçaoInicial* (linha 2), que tem como objetivo gerar uma solução inicial qualquer; na *BuscaLocal* (linha 3), a solução é refinada; Em seguida, iterativamente é realizada uma Perturbação (linha 5) que consiste em executar movimentos para diversificar a solução, que novamente é refinada pela busca local (linha 6). O critério de aceitação (linha 7), decide em qual solução a próxima perturbação será aplicada.

Como parâmetro a ser ajustado, o ILS apresenta o número máximo de iterações *ILS-max*.

3.2.1 Solução Inicial

O algoritmo inicia com a *SoluçãoInicial* que é estruturada com uma ordenação e posteriormente, distribuição nas estações de trabalho. Como dito anteriormente, as tarefas tem restrições de precedências, portanto a ordenação é realizada respeitando essa regra. Utilizando esse conceito, foi utilizado um método de ordenação por topologia, ou como chamado na área de grafos: ordenação topológica (GOODRICH; TAMASSIA; GOLDWASSER, 2014). Esse tipo de ordenação visa ordenar vértices de um grafo orientado acíclico afim de que a ordenação respeite a precedência entre vértices. Na ordenação

topológica utilizada, foi adicionado um critério de desempate para a escolha de qual tarefa deverá ser adicionada na lista ordenada, visto que pode ter mais de uma tarefa candidata a entrar na lista. Como as tarefas possuem um tempo de processamento (t_i), o critério de desempate foi escolher a tarefa com o menor t_i para ser adicionada primeiro na lista.

Algoritmo 1: Algoritmo *ILS* adaptado de Lourenço, Martin e Stützle (2003)

Entrada: ILS_{max}
Saída: Solução S

1 **início**
2 $S_{inicial} \leftarrow SolucaoInicial();$
3 $S \leftarrow BuscaLocal(S_{inicial});$
4 **para** $i \leftarrow 1$ **até** ILS_{max} **faça**
5 $S_i \leftarrow Perturbacao(S);$
6 $S_i \leftarrow BuscaLocal(S_i);$
7 $S \leftarrow Aceitacao(S, S_i);$
8 **fim para**
9 **fim**

Após realizar a ordenação topológica, as tarefas são atribuídas nas estações de trabalho. Foi utilizada uma distribuição de tarefas por média de custo. Nesta abordagem é calculada a média de custo das tarefas, que é a soma de todos os tempos de processamento dividido pelo número de estações de trabalho. Essa média é utilizada pra realizar a distribuição das tarefas de forma a ter um limite de custo por estação de trabalho. O custo por estação de trabalho é a soma dos custos das tarefas atribuídas aquela estação de trabalho em questão. Numa primeira iteração, são atribuídas tarefas, de acordo com a ordenação topológica, na primeira estação de trabalho enquanto a soma dos custos das mesmas não ultrapassarem a média. Na segunda iteração serão adicionadas tarefas na segunda estação de trabalho até que a soma de custos ultrapasse a média. Portanto, estações de trabalho ímpares sempre terão a soma de custos menor do que a média e estações de trabalho pares tem a soma de custos maior que a média.

Realizada a ordenação e a distribuição, é definida a solução $S_{inicial}$. Conforme descrito anteriormente no modelo matemático, o PBLM-2 é um problema de minimização, portanto quando menor for o tempo de execução da estação de trabalho mais sobrecarregada, melhor será a solução.

Em seguida é realizada a busca local apresentada a seguir, que visa melhorar a solução inicial encontrada, explorando a vizinhança.

3.2.2 Busca Local

Essa parte da busca local é essencial para evitar que a heurística fique presa em ótimos locais.

A estrutura de vizinhança se baseia em trocar tarefas entre as estações de trabalho disponíveis sem ferir a precedência entre as tarefas. Portanto, para gerá-la, recebe-se as estações de trabalho preenchidas com as tarefas.

A busca local implementada neste trabalho possui dois tipos de movimento de trocas de tarefas entre estações de trabalho: transferir a última tarefa de uma estação de trabalho para a próxima estação de trabalho e transferir a primeira tarefa de uma estação de trabalho para a estação de trabalho anterior.

Para o primeiro movimento é retirada da estação de trabalho a última tarefa associada a ela, e esta tarefa é colocada na primeira posição da próxima estação de trabalho. Em seguida é calculada a Função Objetivo (FO). Se a solução encontrada for melhor do que a solução atual, essa nova solução será a atual, a busca local se reiniciará, e fará novos movimentos de transferência de tarefas. Esse movimento de transferência de última tarefa para a primeira posição da próxima estação de trabalho só termina quando não houver mais melhoras, ou seja, após realizar todos os movimentos possíveis sem melhorar a solução atual. Esse primeiro tipo de movimento é realizado da estação de trabalho 1 até a estação de trabalho m-1.

Realizado o primeiro tipo de movimento sem encontrar soluções melhores, a busca local começará a realizar o movimento de transferir a primeira tarefa de uma estação de trabalho para a anterior. Esse movimento é realizado em sentido contrário ao do primeiro movimento, ou seja, a partir da última estação de trabalho (m) até a segunda estação de trabalho. O movimento se baseia em pegar a primeira tarefa associada a estação de trabalho e transferi-la para a última posição da estação de trabalho anterior. As demais etapas são iguais ao do primeiro movimento. Portanto se a solução encontrada é melhor, todo procedimento de busca local é reiniciado a partir do primeiro movimento. A busca local termina quando o primeiro e o segundo movimento foram executados sem melhorar a solução vigente.

Abaixo é apresentado um exemplo que esclarece com mais nitidez o processo de geração de soluções vizinhas. Nas Figuras 3 a 6 é usado o termo máquina para se referir às estações de trabalho.

Utilizando como exemplo a linha de montagem da Figura 2 (a) e realizando a distribuição das tarefas nas estações de trabalho, na Figura 3 é mostrada a lista com as nove tarefas distribuídas em quatro estações de trabalho.

```
Distribuição de Máquinas:
Máquina 1: 1 2
Máquina 2: 8 3
Máquina 3: 9 4
Máquina 4: 7 6 5
```

Figura 3 – Distribuição inicial de tarefas nas estações de trabalho.

Dada a distribuição realizada, são realizados os movimentos de transferência da última tarefa da estação de trabalho atual, para a primeira posição da estação de trabalho seguinte. No exemplo da Figura 3, é realizada a movimentação da tarefa 2, para a estação de trabalho 2, como é mostrado na Figura 4.

```
Distribuição de Máquinas:
Máquina 1: 1
Máquina 2: 2 8 3
Máquina 3: 9 4
Máquina 4: 7 6 5
```

Figura 4 – Transferência da última tarefa da primeira estação de trabalho para a primeira posição da segunda estação de trabalho.

Após a transferência realizada, é calculada a FO com base nessa nova solução. Se a solução encontrada após esse movimento for melhor do que a atual, essa se torna a solução principal. Quando isso acontece a busca local é reiniciada, e as transferências começam novamente, mas agora, para a nova distribuição encontrada. Caso a solução não seja melhor que a atual, a alteração é desfeita, a iteração vai para a próxima estação de trabalho.

As iterações terminam quando forem feitas todas as transferências possíveis e as soluções não forem melhores que a solução vigente. Finalizada a primeira parte das buscas por vizinhos usando o primeiro movimento, é realizado o segundo tipo de movimento, que consiste em transferir a primeira tarefa de uma estação de trabalho para a anterior. Digamos que a solução encontrada após o primeiro movimento seja igual a que é apresentada na Figura 5. O segundo movimento partirá da última estação de trabalho

(máquina 4), retirando a primeira tarefa, que no caso é a 9, e colocará na última posição da estação de trabalho anterior (máquina 3).

```
Distribuição de Máquinas:
Máquina 1: 1 2
Máquina 2: 8 3
Máquina 3: 4 7 6
Máquina 4: 9 5
```

Figura 5 – Resultado após realizar todas as possibilidades do primeiro tipo de movimento.

Como pode-se ver na Figura 6, a tarefa 9 se deslocou para a estação de trabalho anterior. Realizado este movimento, é calculada a FO e comparada com a solução atual da mesma forma como no primeiro tipo de movimento. Então, caso a nova solução seja melhor que a anterior, essa se torna a nova melhor solução e a busca local será reiniciada. Caso a solução encontrada seja pior, o procedimento de busca local continua a partir da estação de trabalho anterior. Por fim, a iteração termina quando todas as transferências possíveis tenham sido realizadas e não se tenha encontrado uma solução vizinha melhor.

```
Distribuição de Máquinas:
Máquina 1: 1 2
Máquina 2: 8 3
Máquina 3: 4 7 6 9
Máquina 4: 5
```

Figura 6 – Distribuição após primeira iteração realizada do segundo tipo de movimento.

3.2.3 Perturbação

A perturbação é realizada para que se evite que o algoritmo fique preso em ótimos locais, e assim perca a eficiência. A perturbação implementada neste trabalho realiza uma nova ordenação baseada na última tarefa da última estação de trabalho. Essa tarefa é armazenada na última posição de um novo vetor V que será utilizado para representar a nova ordem perturbada. Em seguida, é criada uma lista L com todas as tarefas que não estão em V, mas que já possuem todos os seus sucessores em V. Iterativamente é selecionado aleatoriamente um elemento de L e adicionado na primeira posição disponível à direita em V . A lista L é então atualizada. Quando todas as tarefas estiverem em V, as mesmas são distribuídas nas estações de trabalho por custo, seguindo o esquema de média de

custos descrito anteriormente. Após a perturbação é realizada uma nova busca local para possivelmente encontrar soluções vizinhas melhores. Para exemplificar o procedimento de pertubação implementado, digamos que a melhor solução encontrada até o momento seja a que é apresentada na Figura 5 e as restrições de precedência as que são apresentadas na Figura 2. Inicialmente, a perturbação coloca a última tarefa da última estação de trabalho na última posição do vetor V, conforme exemplificado na Figura 7. Em seguida verificará quais são as tarefas candidatas a serem adicionadas em V, armazenando-las na Lista L.

Figura 7 – Vetor V preenchido com a última tarefa.

Com podemos ver na Figura 8 existem três tarefas predecessoras da última tarefa e que são candidatas a serem adicionadas em V. O critério de escolha nesse caso é a aleatoriedade. Realizada a atribuição da tarefa escolhida aleatoriamente em V, a lista L é atualizada. O processo é repetido até que todas as tarefas sejam atribuídas no vetor V como é demonstrado na Figura 9.

Figura 8 – Lista de candidatos a serem adicionados no vetor V.

Figura 9 – Vetor V preenchido com a nova ordenação.

3.2.4 Aceitação

Por fim, o módulo de aceitação é responsável por decidir se a solução resultante após a perturbação e busca local será aceita como a solução atual para a próxima iteração. Esse critério envolve comparar o resultado do cálculo da FO da solução obtida pela busca local e a solução atual (linha 7 do Algoritmo 1). Se o tempo de ciclo z da solução obtida após a busca local for melhor do que a solução encontrada até o momento, essa nova solução será a nova solução atual. A decisão influencia diretamente o direcionamento do algoritmo e seu progresso ao longo das iterações.

4 Resultados Obtidos

Neste capítulo são detalhados os experimentos computacionais para validar o método implementado, os resultados obtidos, e sua análise e discussão.

4.1 Experimentos computacionais

Para validar a qualidade das soluções obtidas pelo método proposto, experimentos computacionais foram realizados. Nas seções seguintes são detalhados alguns aspectos importantes desta etapa, como as instâncias utilizadas e a definição dos parâmetros do ILS.

4.1.1 Instâncias

As instâncias utilizadas são muito conhecidas e amplamente utilizadas na literatura. Essas instâncias estão disponíveis em: https://assembly-line-balancing.de.

As instâncias escolhidas para a realização dos experimentos computacionais foram as mesmas usadas no trabalho de Bissoli e Amaral (2016). Essa escolha foi feita para a realização de comparação dos resultados obtidos pelo algoritmo ILS proposto neste trabalho com os resultados do algoritmo GRASP de Bissoli e Amaral (2016).

As instâncias disponíveis no *website* citado acima possuem grau de complexidade que variam de acordo com o número de tarefas, sendo a menor com 30 tarefas e a maior com 297 tarefas.

As três instâncias escolhidas são denominadas: *Gunther*, *Hahn* e *Barthold*, que possuem 35, 53 e 148 tarefas, respectivamente.

Na Figura 10 é possível verificar o exemplo de uma instância para o PBLM-2.

Os números assinalados e os retângulos marcados representam respectivamente:

1. Número de tarefas;
2. Tempo de processamento de cada tarefa;
3. Relações de precedência entre as tarefas.

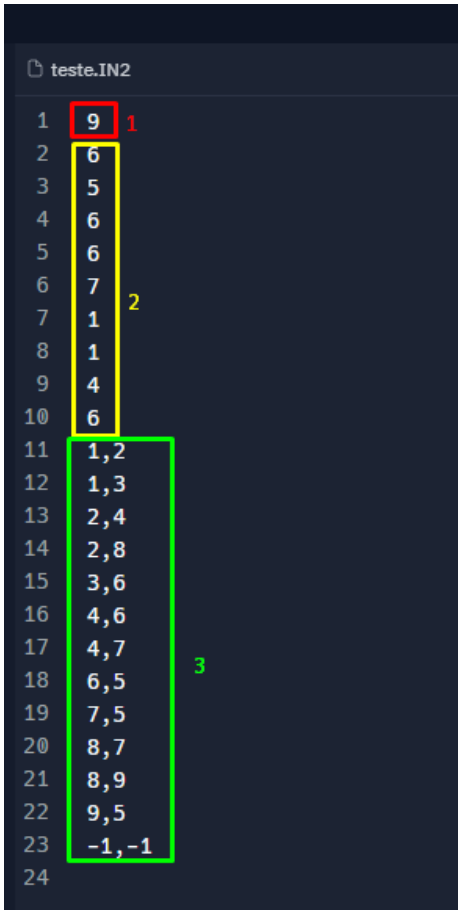

Figura 10 – Exemplo de instância com nove tarefas.

4.1.2 Escolha de parâmetros

Por padrão o algoritmo ILS recebe o parâmetro ILS_{max} que define o número de iterações. Com o intuito de comparar os resultados obtidos pelo ILS com os resultados de Bissoli e Amaral (2016), foi utilizado o mesmo número de iterações. Desta forma o ILS_{max} foi definido com um valor de 200000.

Existe também o parâmetro que define o número de estações de trabalho. Esse também seguirá a mesma parametrização de Bissoli e Amaral (2016), variando o número de estações de trabalho entre 3 e 15.

4.1.3 Detalhes da implementação e execução de testes

O algoritmo foi desenvolvido na linguagem *C*, compilado com o compilador *GCC Compiler* na versão 11.4.0. O computador utilizado tem as configurações descritas na Tabela 1.

Para cada uma das instâncias, e para cada valor de m, foram realizadas 20 execuções.

Tabela 1 – Ambiente Computacional.

Item	Valor
Processador	Ryzen 5 3.8GHz
Memória RAM	16GB
Sistema Operacional	Ubuntu 20.04.3

4.2 Resultados Computacionais

Nesta seção são apresentados os resultados computacionais obtidos a partir dos experimentos descritos acima.

Na Tabela 2 pode-se observar os resultados obtidos para cada instância. Para cada uma delas se tem o valor da melhor solução conhecida (BKS), a melhor solução obtida dentre as 20 execuções (z_{best}), o valor médio de z (z_{avg}) e a diferença percentual (Δ) entre BKS e z_{best}. Além disso, também é exibido na coluna ω o tempo gasto (em segundos) para que a solução com valor z_{best} fosse gerada.

É importante destacar que o valor BKS foi obtido do *website* citado anteriormente. Ele representa o melhor resultado conhecido através dos resultados obtidos de várias implementações de métodos de solução para o PBLMS-2. Um desses métodos de solução é a meta-heurística GRASP proposta por Bissoli e Amaral (2016), que foi escolhida representativamente para comparação dos resultados com o ILS proposto neste trabalho.

Os resultados apresentados na Tabela 2 demonstram que o método proposto neste trabalho foi eficaz em diferentes instâncias, sendo na maioria dos casos próximo da melhor solução conhecida (BKS). Isso pode ser observado através dos valores de Δ. Embora não tenha encontrado as melhores soluções conhecidas em todos os casos, na maioria deles o algoritmo mostrou-se consistente com os resultados, produzindo soluções repetíveis e confiáveis para essas instâncias, como pode ser observado na coluna z_{avg}.

Na Tabela 2 observa-se também que em 16 dos 31 casos de teste os valores de z_{best} foram iguais ao da coluna BKS, o que indica que para diferentes valores de m o algoritmo ILS proposto apresenta resultados satisfatórios. Além disso, nos demais casos o valor da diferença percentual (Δ) foi menor que 5%, apresentando uma média de apenas 0,72%.

Ainda na Tabela 2, observa-se que as melhores soluções foram encontradas com menos de 18 minutos, no pior caso, enquanto que no melhor caso foi encontrada com pouco mais de 10 segundos de execução.

Tabela 2 – Resultados obtidos pelo algoritmo ILS.

Instância	m	BKS	z_{best}	z_{avg}	$\Delta\%$	$\omega(seg.)$
Gunther (35)	6	84	84	84,00	0,00%	10,428
	7	72	73	73,05	1,39%	10,410
	8	63	63	63,35	0,00%	12,308
	9	54	55	56,10	1,85%	15,835
	10	50	50	50,75	0,00%	16,393
	11	48	48	48,00	0,00%	19,717
	12	44	44	44,35	0,00%	20,023
	13	42	44	44,00	4,76%	20,010
	14	40	41	41,90	2,50%	22,455
	15	40	40	40,00	0,00%	21,610
Hahn (53)	3	4787	4787	4787,00	0,00 %	14,883
	4	3677	3677	3685,70	0,00 %	18,372
	5	2823	2823	2828,25	0,00 %	21,693
	6	2400	2400	2400,00	0,00%	23,915
	7	2336	2388	2388,00	2,23 %	75,224
	8	1907	1965	1965,00	3,04 %	59,972
	9	1827	1827	1827,00	0,00 %	27,349
	10	1775	1775	1775,00	0,00%	64,421
Barthold (148)	3	1878	1878	1878,00	0,00 %	215,036
	4	1409	1409	1409,00	0,00 %	233,293
	5	1127	1127	1127,85	0,00 %	279,433
	6	939	940	940,80	0,11 %	305,653
	7	805	807	807,95	0,25 %	542,518
	8	705	707	708,20	0,28 %	611,562
	9	626	628	630,70	0,32 %	718,283
	10	564	566	568,50	0,35 %	779,736
	11	513	516	517,95	0,58 %	734,856
	12	470	474	476,30	0,85%	931,332
	13	434	438	441,80	0,92%	921,362
	14	403	410	412,30	1,74%	1019,182
	15	383	388	392,89	1,31%	1024,917
Média					0,72%	

4.3 Análise e Discussão

Na Tabela 3 é apresentada uma comparação dos resultados obtidos pelo ILS com os resultados do método GRASP de Bissoli e Amaral (2016).

Tabela 3 – Comparação dos resultados do ILS com os obtidos por Bissoli e Amaral (2016).

Instância	m	z_{bestG}	z_{bestI}	$\Delta\%$
Gunther (35)	6	**84**	**84**	0,00 %
	7	**72**	73	1,38 %
	8	**63**	**63**	0,00 %
	9	**54**	55	1,85 %
	10	**50**	**50**	0,00 %
	11	**48**	**48**	0,00 %
	12	**44**	**44**	0,00 %
	13	**42**	44	4,76 %
	14	**40**	41	2,50 %
	15	**40**	**40**	0,00 %
Hahn (53)	3	**4787**	**4787**	0,00 %
	4	**3677**	**3677**	0,00 %
	5	**2823**	**2823**	0,00 %
	6	**2400**	**2400**	0,00 %
	7	**2336**	2388	2,22 %
	8	**1910**	1965	2,87 %
	9	1866	**1827**	-2,09 %
	10	1866	**1775**	-4,87 %
Barthold (148)	3	**1878**	**1878**	0,00 %
	4	**1409**	**1409**	0,00 %
	5	**1127**	**1127**	0,00 %
	6	**939**	940	0,10 %
	7	**805**	807	0,24 %
	8	**705**	707	0,28 %
	9	**626**	628	0,15 %
	10	**564**	566	0,35 %
	11	**513**	516	0,58 %
	12	**470**	474	0,85 %
	13	**435**	438	0,68 %
	14	**404**	410	1,48 %
	15	**383**	388	1,30 %
Média				0,47%

Para cada instância e cada valor de m é apresentado na coluna z_{bestG} o valor da melhor solução encontrada por Bissoli e Amaral (2016) com o algoritmo GRASP. Na coluna seguinte (z_{bestI}) é apresentado o menor valor de z obtido pelo método proposto neste trabalho. Por fim, a coluna $\Delta\%$ apresenta a diferença percentual ente z_{bestG} e z_{bestI}. Os resultados do algoritmo ILS se aproximam bastante dos resultados obtidos pelo GRASP de Bissoli e Amaral (2016), sendo em média, apenas 0,47% pior que o GRASP. A diferença percentual não ultrapassou 5%. Em dois casos de teste o ILS atingiu a melhor solução conhecida (BKS), enquanto que o algoritmo GRASP de Bissoli e Amaral (2016) não atingiu os mesmos resultados. Além disso, em 13 instâncias os resultados do GRASP de Bissoli e Amaral (2016) foram iguais aos do ILS proposto nesse trabalho.

5 Conclusão

Neste trabalho foi abordado o PBLMS-2, sendo proposto um método que usa a meta-heurística ILS para resolvê-lo. O método proposto gerou resultados consistentes, em tempos consideravelmente baixos, levando em conta a implementação simples, sem haver uma estratégia sofisticada para busca local.

Os resultados obtidos mostraram também a eficácia dessa abordagem para diferentes instâncias do problema. Os resultados mostram que o método tem eficácia para diversos casos de testes, apresentando uma variação média de apenas 0,72% para as melhores soluções conhecidas na literatura.

Em comparação com Bissoli e Amaral (2016), o algoritmo ILS apresentou resultados melhores em dois casos de teste, apresentado uma diferença média de apenas 0,47% em relação aos melhores valores obtidos. Além disso, nesses dois casos de teste o algoritmo ILS encontrou o melhor valor conhecido até então na literatura.

Para trabalhos futuros, recomenda-se explorar outros movimentos de busca local, com o intuito de explorar ainda mais o espaço de soluções.

Referências

BAYBARS, I. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, v. 32, n. 8, 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, n. 3, p. 694–715, 2006.

BISSOLI, D. C. *Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção*. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Espírito Santo, 2018.

BISSOLI, D. C.; AMARAL, A. R. S. Um algoritmo grasp para o salbp-2. In: *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*. Rio de Janeiro, RJ, Brasil: SOBRAPO, 2016. v. 48, p. 2081–2091.

BOYSEN, N.; FLIEDNER, M. A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, v. 184, n. 1, p. 39–56, 2008.

CHIARANDINI, M.; STÜTZLE, T. et al. An application of iterated local search to graph coloring problem. In: *Proceedings of the computational symposium on graph coloring and its generalizations*. New York, NY, USA: Ithaca, 2002. p. 112–125.

CUERVO, D. P. et al. An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, v. 237, n. 2, p. 454–464, 2014.

EREL, E.; SARIN, S. C. A survey of the assembly line balancing procedures. *Production Planning & Control*, v. 9, n. 5, p. 414–434, 1998.

FARNES, V. C. F.; PEREIRA, N. A. Balanceamento de linha de montagem com o uso de heurística e simulação: estudo de caso na linha branca. *Revista Gestão da Produção Operações e Sistemas*, n. 1, p. 125–125, 2007.

GOODRICH, M. T.; TAMASSIA, R.; GOLDWASSER, M. H. *Data Structures and Algorithms in Java*. 6. ed. New York, NY, USA: John Wiley & Sons Inc, 2014.

HASAN, M. H.; AHMAD, R. A review of assembly line balancing problems. *Journal of Advanced Manufacturing Systems*, v. 17, n. 3, p. 369–386, 2018.

KLEIN, R.; SCHOLL, A. Maximizing the production rate in simple assembly line balancing — a branch and bound procedure. *European Journal of Operational Research*, v. 91, n. 2, p. 367–385, 1996.

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

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of metaheuristics*. Boston, MA, USA: Springer, 2003. p. 320–353.

REGO, C. et al. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, v. 211, n. 3, p. 427–441, 2011.

REKIEK, B. et al. State of art of optimization methods for assembly line design. *Annual Reviews in control*, v. 26, n. 2, p. 163–174, 2002.

ROCHA, C. et al. A survey on solution methods for multi-objective assembly line balancing problems. *Computers & Industrial Engineering*, v. 127, p. 832–848, 2019.

SALVESON, M. E. The assembly line balancing problem. *Journal of Industrial Engineering*, v. 6, n. 4, p. 18–25, 1955.

SARKER, B. R.; SHANTHIKUMARI, J. G. A generalized approach for serial or parallel line balancing. *International Journal of Production Research*, v. 21, n. 1, p. 109–133, 1983.

SCHOLL, A. *Balancing and sequencing of assembly lines*. Berlin, Heidelberg, Germany: Springer Berlin Heidelberg, 1999.

SCHOLL, A.; BECKER, C. 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.

SLACK, N. et al. *Administração da produção*. São Paulo, SP, Brasil: Atlas, 2009.

STÜTZLE, T. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, v. 174, n. 3, p. 1519–1539, 2006.

UDDIN, M. K.; LASTRA, J. L. M. Assembly line balancing and sequencing. In: GRZECHCA, W. (Ed.). *Assembly Line–Theory and Practice*. Rijeka, Croatia: Intechopen, 2011. p. 13–36.