

ALGORITMO ILS PARA O PROBLEMA DE BALANCEAMENTO DE LINHAS DE MONTAGEM

Mário Ribeiro Júnior, Edmar Hell Kampke, Dayan de Castro Bissoli, Antonio Almeida de Barros Junior

Universidade Federal do Espírito Santo
Alto Universitário, s/n, Campus Universitário, Alegre – ES, 29500-000
mario.ribeiro.88@edu.ufes.br, edmar.kampke@ufes.br,
dayan.bissoli@ufes.br, antonio.barros@ufes.br

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 objetivo principal do PBLMS-2 é distribuir tarefas entre estações de trabalho de maneira equilibrada, respeitando as restrições de precedência entre as tarefas e minimizando o tempo de ciclo. Ao aplicar o ILS ao PBLMS-2 foram encontrados resultados próximos aos melhores conhecidos da literatura. Os valores obtidos são também comparados com os resultados de outro trabalho da literatura. Ao final dos experimentos computacionais foi constatado que os valores obtidos pelo ILS ficaram, respectivamente, 3,10% e 1,83% de distância entre os melhores resultados conhecidos e em relação a um algoritmo GRASP proposto na literatura para o PBLMS-2.

PALAVRAS CHAVE. Problema de Balanceamento de Linhas de Montagem Simples, *Iterated Local Search*, Otimização Combinatória, Meta-heurísticas.

MH – Meta-heurísticas; OD – Otimização Discreta

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 main objective of SALBP-2 is to distribute tasks among workstations in a balanced manner while respecting precedence constraints between tasks and minimizing cycle time. By applying ILS to SALBP-2 results close to the best-known results in the literature were obtained. The values obtained are also compared with the results of another work in the literature. At the end of the computational experiments, it was found that the values obtained by ILS were, respectively, 3.10% and 1.83% away from the best known results and in relation to a GRASP algorithm proposed in the literature for SALBP-2.

KEYWORDS. Simple Assembly Line Balancing Problem. *Iterated Local Search*. Combinatorial Optimization. Metaheuristics.

MH - Metaheuristics; OD - Discrete Optimization

1. Introdução

Nas indústrias manufatureiras a lucratividade depende diretamente da otimização dos recursos [Hasan e Ahmad, 2018]. Assim, para atingir esse objetivo, se faz necessário realizar um planejamento adequado das suas linhas de montagem, já que as mesmas influenciam diretamente na distribuição de recursos, redução nos tempos de fabricação e eliminação de atrasos e estoques. Nesse contexto, o Problema de Balanceamento de Linhas de Montagem (PBLM) se torna essencial e, uma vez equilibrado de forma satisfatória, o ciclo de produção acaba sendo mais eficiente num contexto operacional [Rego et al., 2011].

Apesar da sua complexidade o PBLM é fundamental, porque é amplamente utilizável em muitos setores industriais, como automotivo, mecânico e eletrônico [Erel e 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 e Pereira, 2007].

Uma linha de montagem possui várias estações de trabalho, cada uma com capacidades e habilidades distintas. As tarefas possuem tempos de processamento associados, definidos como o tempo necessário para que uma estação de trabalho conclua determinada tarefa. Podem existir 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], o PBLM consiste no desafio de alocar tarefas a estações de trabalho em uma linha de montagem, de forma a reduzir o tempo de ciclo, e assim, aprimorar a produção, respeitando as restrições de precedência e as habilidades da estação de trabalho. O PBLM também tem como objetivo conseguir equilibrar satisfatoriamente a distribuição de tarefas em estações de trabalho, impedindo o trabalho excessivo, ou seja, evitando que uma estação de trabalho esteja muito mais sobrecarregada do que as outras, e também a subutilização, que é uma estação de trabalho 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 dos custos de produção [Scholl, 1999].

Klein e Scholl [1996] afirmam que o PBLM é NP-difícil. Dessa forma, uma alternativa viável para resolução do PBLM é o uso de métodos heurísticos ou meta-heurísticos. Neste trabalho a meta-heurística *Iterated Local Search* (ILS) foi escolhida para ser aplicada na resolução de uma variante do PBLM. A justificativa pela escolha da ILS é por ela ter sido usada satisfatoriamente em diferentes problemas [Chiarandini et al., 2002; Stützle, 2006; Cuervo et al., 2014] e por não ter sido encontrada na literatura uma aplicação da ILS para a variante do PBLM abordado neste trabalho.

Diante disso, na Seção 2 é apresentada uma breve revisão da literatura sobre o PBLM. Na Seção 3 a metodologia desenvolvida neste trabalho é descrita. Já a Seção 4 é dedicada à apresentação e discussão dos resultados obtidos nos experimentos computacionais. Por fim, a Seção 5 apresenta as conclusões deste trabalho.

2. Revisão de Literatura

De acordo com Uddin e Lastra [2011], o PBLM é subdividido em duas classes: PBLM Simples (PBLMS) e PBLM Generalizado (PBLMG). O PBLMS é extremamente útil para montagens simples de produtos, em uma linha reta, para fornecer uma restrição exclusiva de precedência entre as tarefas. De acordo com Becker e Scholl [2006], e conforme apresentado na Figura 1, o PBLMS possui quatro tipos (PBLMS-1, PBLMS-2, PBLMS-E e PBLMS-F).

Por outro lado, 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; Agregação de estações de trabalho em paralelo; entre outros.

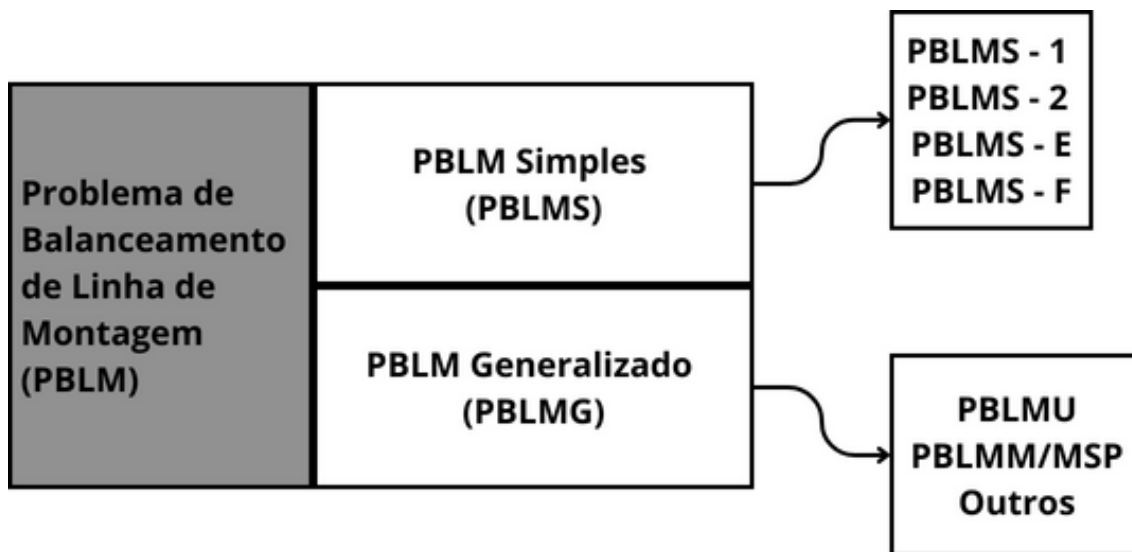


Figura 1: Classificações para PBLMs adaptado de Uddin e Lastra [2011].

Neste trabalho é abordado o PBLMS do Tipo 2 (PBLMS-2) que busca minimizar o tempo de ciclo (maximizar a taxa de produção), respeitando as restrições de precedência para um determinado número (m) de estações de trabalho disponíveis. Na literatura existem várias abordagens, baseadas em heurísticas e métodos exatos, que focam em modelar e resolver o PBLMS-2.

O procedimento *Branch and Bound* utilizado 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 PBLMS 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 colônia 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. Utilizando também alguns conceitos de grafos, como exemplo a ordenação topológica, o método conseguiu encontrar soluções muito próximas das melhores encontradas até aquele momento para algumas instâncias disponibilizadas na literatura.

No trabalho de Pitakaso et al. [2023] foi aplicada a meta-heurística *Variable Neighbourhood Strategy Adaptive Search* (VaNSAS) para o PBLMS-2. Os resultados obtidos, após a realização dos experimentos computacionais em um conjunto com 128 instâncias, mostraram que a solução ótima foi atingida em 98,4% dos casos.

3. Metodologia

Na Subseção 3.1 é apresentado o modelo matemático do PBLMS-2, e na Subseção 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 do equipamento, ao mesmo tempo que procura-se reduzir o tempo de ciclo da estação de trabalho mais sobrecarregada.

De acordo com Bissoli [2018], o PBLMS-2 pode ser formulado matematicamente da seguinte forma: Uma instância (T, G, m) consiste em três componentes. $T = \{1, \dots, n\}$ 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 $G = (T, A)$ que é acíclico e orientado, 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 da tarefa 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:

$$\text{Minimizar } z \quad (1)$$

Sujeito a:

$$\sum_{s=1}^m x_{is} = 1, \quad \forall i \in T \quad (2)$$

$$x_{is} \leq \sum_{s'=1}^s x_{js'}, \quad \forall i \in T, \forall j \in P_i, s = 1, \dots, m \quad (3)$$

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

$$x_{is} \in \{0, 1\} \quad \forall i \in T, s = 1, \dots, m \quad (5)$$

$$z > 0 \quad (6)$$

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 (5). A função objetivo (1) minimiza o tempo de ciclo z . A restrição (2) garante que cada tarefa $i \in T$ seja atribuída a uma única estação de trabalho. A restrição (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 (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 (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, com $z = 13$.

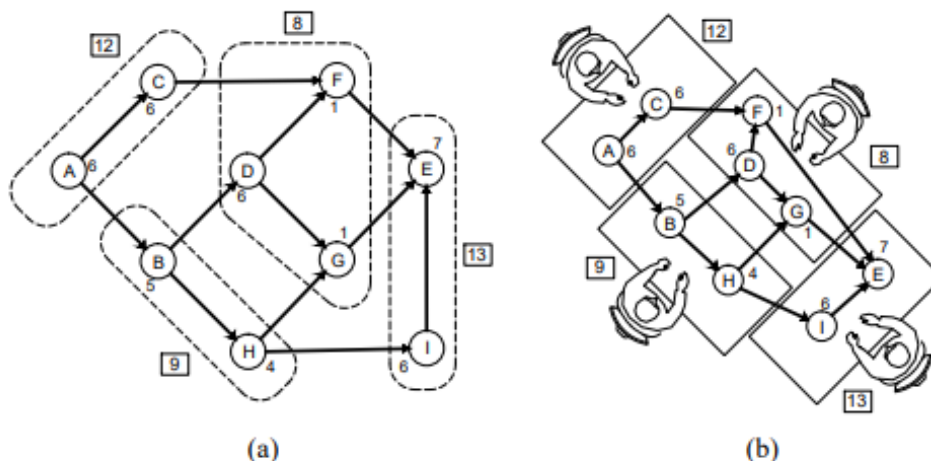


Figura 2: Solução para um grafo de precedência.

Fonte: Bissoli [2018].

3.2. ILS

A meta-heurística ILS foi proposta por Lourenço et al. [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 primeiro deles (linha 2) tem como objetivo gerar uma solução inicial qualquer; na Busca Local (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 apenas o número máximo de iterações ILS_{max} .

3.2.1. Solução Inicial

O algoritmo inicia com a obtenção de uma solução inicial ($S_{inicial}$) que é estruturada com uma ordenação das tarefas e, posteriormente, distribuição nas estações de trabalho. Como dito anteriormente, as tarefas tem restrições de precedência, portanto a ordenação é realizada respeitando essa regra. A partir disso, foi utilizado um método de ordenação topológica [Goodrich et al., 2014], que visa ordenar vértices de um grafo orientado e 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 et al. [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 o custo total da estação de trabalho não ultrapassar a média. Na segunda iteração serão adicionadas tarefas na segunda estação de trabalho até que o custo total da estação de trabalho ultrapasse a média. A distribuição continua iterativamente para as demais estações de trabalho, de tal forma que estações de trabalho ímpares sempre terão o seu custo total menor do que a média e estações de trabalho pares tem o seu custo total maior que a média.

3.2.2. Busca Local

Essa etapa é essencial para evitar que a heurística fique presa em ótimos locais. Para isso, utiliza-se uma estrutura de vizinhança que 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 calculado o valor de z . 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.

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.

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 de z 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

Nesta seção 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 subseções seguintes são detalhados alguns aspectos importantes desta etapa, como as instâncias utilizadas e a definição do parâmetro 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/salbp>.

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 quatro instâncias escolhidas são denominadas: *Gunther*, *Hahn*, *Barthold* e *Scholl*, que possuem 35, 53, 148 e 297 tarefas, respectivamente.

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.

4.1.3. Detalhes da implementação e execução de testes

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

Tabela 1: Ambiente Computacional.

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

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

4.2. Resultados Computacionais

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

Nas Tabelas 2 e 3 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 pelo algoritmo GRASP de Bissoli e Amaral [2016] (z_{bestG}), a melhor solução obtida dentre as 20 execuções do algoritmo ILS proposto neste trabalho (z_{bestI}), o valor médio de z (z_{avgI}) e as diferenças percentuais (Δ_{BKS} e Δ_G) entre BKS e z_{bestI} , e entre z_{bestG} e z_{bestI} , respectivamente. Além disso, também é exibido na coluna ω o tempo gasto (em segundos) para que a solução com valor z_{bestI} fosse gerada.

É importante destacar que o valor BKS foi obtido do *website* citado na Subseção 4.1.1. Ele representa o estado da arte, ou seja, 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.

4.3. Análise e Discussão

Os resultados apresentados na Tabela 2 demonstram que o método proposto neste trabalho foi eficaz nas três menores instâncias usadas nos experimentos computacionais, sendo na maioria dos casos próximo da melhor solução conhecida (BKS). Isso pode ser observado através dos valores de Δ_{BKS} . 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_{avgI} .

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_{BKS}) 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. Os resultados do algoritmo ILS se aproximam bastante dos resultados obtidos pelo GRASP de Bissoli e Amaral [2016], sendo em média, apenas 0,48% 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.

Já na Tabela 3 são apresentados os resultados obtidos nos experimentos computacionais realizados com a instância denominada *Scholl*. É importante destacar que essa instância é a maior

Tabela 2: Resultados obtidos pelo algoritmo ILS - Parte 1.

Instância	m	BKS	z_{bestG}	z_{bestI}	z_{avgI}	$\Delta_{BKS}\%$	$\Delta_G\%$	$\omega(seg.)$
Gunther (35)	6	84	84	84	84,00	0,00%	0,00%	10,428
	7	72	72	73	73,05	1,39%	1,39%	10,410
	8	63	63	63	63,35	0,00%	0,00%	12,308
	9	54	54	55	56,10	1,85%	1,85%	15,835
	10	50	50	50	50,75	0,00%	0,00%	16,393
	11	48	48	48	48,00	0,00%	0,00%	19,717
	12	44	44	44	44,35	0,00%	0,00%	20,023
	13	42	42	44	44,00	4,76%	4,76%	20,010
	14	40	40	41	41,90	2,50%	2,50%	22,455
	15	40	40	40	40,00	0,00%	0,00%	21,610
Hahn (53)	3	4787	4787	4787	4787,00	0,00%	0,00%	14,883
	4	3677	3677	3677	3685,70	0,00%	0,00%	18,372
	5	2823	2823	2823	2828,25	0,00%	0,00%	21,693
	6	2400	2400	2400	2400,00	0,00%	0,00%	23,915
	7	2336	2336	2388	2388,00	2,22%	2,22%	75,224
	8	1907	1910	1965	1965,00	3,04%	2,87%	59,972
	9	1827	1866	1827	1827,00	0,00%	-2,09%	27,349
	10	1775	1866	1775	1775,00	0,00%	-4,87%	64,421
Barthold (148)	3	1878	1878	1878	1878,00	0,00%	0,00%	215,036
	4	1409	1409	1409	1409,00	0,00%	0,00%	233,293
	5	1127	1127	1127	1127,85	0,00%	0,00%	279,433
	6	939	939	940	940,80	0,11%	0,11%	305,653
	7	805	805	807	807,95	0,24%	0,24%	542,518
	8	705	705	707	708,20	0,28%	0,28%	611,562
	9	626	626	628	630,70	0,32%	0,32%	718,283
	10	564	564	566	568,50	0,35%	0,35%	779,736
	11	513	513	516	517,95	0,58%	0,58%	734,856
	12	470	470	474	476,30	0,85%	0,85%	931,332
	13	434	435	438	441,80	0,92%	0,68%	921,362
	14	403	404	410	412,30	1,74%	1,48%	1019,182
	15	383	383	388	392,89	1,31%	1,31%	1024,917
Média						0,72%	0,48%	

disponível no *website* citado anteriormente e possui 297 tarefas. Apesar do algoritmo ILS não ter obtido a melhor solução conhecida em nenhum dos casos de teste, mais uma vez mostrou-se consistente com os resultados, produzindo soluções repetíveis e com baixo desvio (Δ_{BKS}).

Os valores apresentados na Tabela 3 mostram que, em média, o algoritmo ILS obteve resultados que se distanciam apenas 5,73% dos melhores valores conhecidos na literatura, sendo que no pior caso a diferença foi de 9,30%.

Em comparação com o algoritmo GRASP de Bissoli e Amaral [2016], para a instância *Scholl*, nota-se que o algoritmo GRASP também não atingiu, em nenhum dos casos de teste, o melhor valor conhecido *BKS*. No entanto, o algoritmo GRASP apresentou resultados de melhor qualidade quando comparados com os resultados do algoritmo ILS, sendo que a diferença percentual média foi de 3,33%, variando entre 1,28% e 5,87%.

Tabela 3: Resultados obtidos pelo algoritmo ILS - Parte 2.

Instância	m	BKS	z_{bestG}	z_{bestI}	z_{avgI}	$\Delta_{BKS}\%$	$\Delta_G\%$	$\omega(seg.)$
Scholl (297)	25	2787	2819	2855	2867,35	2,44%	1,28%	4647,559
	26	2680	2710	2765	2777,25	3,17%	2,03%	5683,178
	27	2580	2608	2660	2673,00	3,10%	1,99%	8950,329
	28	2488	2515	2574	2591,95	3,46%	2,35%	9656,719
	29	2402	2427	2477	2499,70	3,12%	2,06%	10338,462
	30	2322	2350	2417	2428,85	4,09%	2,85%	9782,217
	31	2247	2281	2339	2353,15	4,09%	2,54%	10984,553
	32	2177	2210	2280	2297,95	4,73%	3,17%	11118,577
	33	2111	2141	2208	2223,20	4,59%	3,13%	11442,618
	34	2049	2088	2146	2157,45	4,73%	2,78%	8910,325
	35	1991	2042	2083	2104,15	4,62%	2,01%	6644,100
	36	1935	1990	2049	2061,80	5,89%	2,96%	7175,518
	37	1883	1932	1981	1994,70	5,20%	2,54%	6140,903
	38	1834	1877	1936	1952,10	5,56%	3,14%	7062,052
	39	1787	1826	1882	1912,25	5,32%	3,07%	6957,146
	40	1742	1784	1840	1870,80	5,63%	3,14%	7055,304
	41	1700	1740	1812	1827,20	6,59%	4,14%	7735,884
	42	1659	1702	1762	1783,30	6,21%	3,53%	7339,372
	43	1621	1686	1746	1756,80	7,71%	3,56%	7963,099
	44	1584	1637	1706	1729,20	7,70%	4,22%	7316,634
	45	1549	1601	1677	1692,50	8,26%	4,75%	7817,669
	46	1515	1558	1631	1654,55	7,66%	4,69%	7310,963
	47	1483	1532	1593	1621,40	7,42%	3,98%	6521,290
	48	1452	1518	1587	1600,50	9,30%	4,55%	6298,194
	49	1427	1485	1546	1560,30	8,34%	4,11%	10787,286
	50	1394	1451	1506	1524,65	8,03%	3,79%	8815,405
	51	1386	1413	1496	1509,15	7,94%	5,87%	9000,272
	52	1386	1390	1461	1484,30	5,41%	5,11%	10244,318
Média						5,73%	3,33%	

Por fim, a coluna ω da Tabela 3 destaca a complexidade da instância *Scholl*, uma vez que para alguns casos de teste a melhor solução (z_{bestI}) foi obtida após um pouco mais de três horas de execução.

5. Conclusão

Neste trabalho foi abordado o PBLMS-2, sendo proposto um método que usa a meta-heurística ILS para resolvê-lo. A ordenação topológica é usada para construir uma solução viável. Nessa etapa o procedimento tenta equilibrar a distribuição das tarefas nas estações de trabalho através de uma média de custos. Além disso, os procedimentos de busca local e perturbação executados iterativamente também garantem a viabilidade da solução.

Para um conjunto menor de três instâncias usadas nos testes, o ILS proposto gerou resultados consistentes, em tempos consideravelmente baixos, levando em conta a implementação simples, sem haver uma estratégia complexa para busca local. Já para a maior instância disponível para o PBLMS-2, o algoritmo ILS apresentou uma variação média de 5,73% em relação as melhores soluções conhecidas na literatura. Esse valor é muito superior à variação média de 0,72% obtido

para o primeiro conjunto de instâncias.

Em comparação com o algoritmo GRASP proposto por Bissoli e Amaral [2016], o algoritmo ILS apresentou resultados melhores em dois casos de teste. Além disso, nesses dois casos de teste o algoritmo ILS encontrou o melhor valor conhecido até então na literatura. Em média, o algoritmo ILS apresentou resultados que se distanciaram apenas 1,83% em relação aos melhores valores obtidos pelo algoritmo GRASP de Bissoli e Amaral [2016].

Para trabalhos futuros, recomenda-se explorar outros mecanismos de perturbação e outros movimentos de busca local, com o intuito de aumentar a diversificação e explorar ainda mais o espaço de soluções.

Referências

- Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32(8):909–932.
- Becker, C. e Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694–715.
- Bissoli, D. C. (2018). *Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção*. Doutorado em ciência da computação, Universidade Federal do Espírito Santo.
- Bissoli, D. C. e Amaral, A. R. S. (2016). Um algoritmo grasp para o salbp-2. In *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, volume 48, p. 2081–2091, Rio de Janeiro, RJ, Brasil. SOBRAPO.
- Boysen, N. e Fliedner, M. (2008). A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, 184(1):39–56.
- Chiarandini, M., Stützle, T., et al. (2002). An application of iterated local search to graph coloring problem. In *Proceedings of the computational symposium on graph coloring and its generalizations*, p. 112–125, New York, NY, USA. Ithaca.
- Cuervo, D. P., Goos, P., Sörensen, K., e Arráiz, E. (2014). An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 237(2): 454–464.
- Erel, E. e Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, 9(5):414–434.
- Farnes, V. C. F. e Pereira, N. A. (2007). 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*, (1):125–125.
- Goodrich, M. T., Tamassia, R., e Goldwasser, M. H. (2014). *Data Structures and Algorithms in Java*. John Wiley & Sons Inc, New York, NY, USA, 6 edition.
- Hasan, M. H. e Ahmad, R. (2018). A review of assembly line balancing problems. *Journal of Advanced Manufacturing Systems*, 17(3):369–386.
- Klein, R. e Scholl, A. (1996). Maximizing the production rate in simple assembly line balancing — a branch and bound procedure. *European Journal of Operational Research*, 91(2):367–385.

- Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In Glover, F. e Kochenberger, G., editors, *Handbook of metaheuristics*, p. 320–353. Springer, Boston, MA, USA.
- Pitakaso, R., Sethanan, K., Jirasirilerd, G., e Golinska-Dawson, P. (2023). A novel variable neighborhood strategy adaptive search for salbp-2 problem with a limit on the number of machine's types. *Annals of Operations Research*, 324(1):1501–1525.
- Rego, C., Gamboa, D., Glover, F., e Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–441.
- Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering*, 6(4):18–25.
- Scholl, A. (1999). *Balancing and sequencing of assembly lines*. Springer Berlin Heidelberg, Berlin, Heidelberg, Germany.
- Scholl, A. e Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666–693.
- Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539.
- Uddin, M. K. e Lastra, J. L. M. (2011). Assembly line balancing and sequencing. In Grzechca, W., editor, *Assembly Line—Theory and Practice*, p. 13–36. Intechopen, Rijeka, Croatia.