

Algoritmo MIG-ALBHW para resolver o *Assembly Line Balancing with Hierarchical Worker Assignment*

Alexandre dos Santos Mello

Universidade Federal Fluminense
Instituto de Computação, Niterói - RJ, CEP: 24210-346
alexandre_mello@id.uff.br

Matheus Nohra Haddad

Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas, ICEA, João Monlevade - MG, CEP: 35931-088
matheus.haddad@ufop.edu.br

Luiz Satoru Ochi

Universidade Federal Fluminense
Instituto de Computação, Niterói - RJ, CEP: 24210-346
satoru@ic.uff.br

Pablo Luiz Araújo Munhoz

Universidade Federal de Ouro Preto
Departamento de Computação, ICEB, Ouro Preto - MG, CEP: 35400-000
pablo.munhoz@ufop.edu.br

Igor Machado Coelho

Universidade Federal Fluminense
Instituto de Computação, Niterói - RJ, CEP: 24210-346
imcoelho@ic.uff.br

RESUMO

A necessidade de estudos de problemas de alocação com recursos limitados é essencial para o desenvolvimento de Cidades e Regiões Inteligentes e Sustentáveis. Por tal fato, este artigo aborda o *Assembly Line Balancing with Hierarchical Worker Assignment* (ALBHW), visando encontrar a melhor combinação entre tarefas e trabalhadores em suas estações de trabalho, adicionando uma nova característica de hierarquia aos trabalhadores. Desta forma, trabalhadores qualificados para executar determinadas tarefas possuem um custo proporcional à sua qualificação. Para resolver o ALBHW propomos uma abordagem heurística, ainda não explorada na literatura. Nosso algoritmo híbrido, nomeado de MIG-ALBHW, une as meta-heurísticas *Iterated Local Search* e *Multi-Start*, além de conseguir gerar múltiplas soluções através da adaptação da fase construtiva do *Greedy Randomized Adaptive Search Procedure*. Experimentos computacionais realizados em um conjunto de 54 instâncias de *benchmark* demonstram a eficiência do MIG-ALBHW, pois 41 soluções ótimas foram encontradas e sendo em média 37,59% mais rápido.

PALAVRAS CHAVE. Cidades Inteligentes. *Assembly Line Balancing with Hierarchical Worker Assignment*. Heurísticas híbridas.

Cidades e Regiões Inteligentes e Sustentáveis, meta-heurística

ABSTRACT

The need to study allocation problems with limited resources is essential for the development of Smart and Sustainable Cities and Regions. For this reason, this article addresses the *Assembly Line Balancing with Hierarchical Worker Assignment* (ALBHW), aiming to find the best combination between tasks and workers at their workstations, adding a new hierarchy feature to workers. In this way, workers qualified to perform certain tasks have a cost proportional to their qualification. To solve ALBHW we propose a heuristic approach, not yet explored in the literature. Our hybrid algorithm, named *MIG-ALBHW*, combines the metaheuristics *Iterated Local Search* and *Multi-Start*, in addition to be able to generate multiple solutions by adapting the construction phase of the *MIG-ALBHW Greedy Randomized Adaptive Search Procedure*. Computational experiments carried out on a set of 54 instances of *benchmark* demonstrate the efficiency of *MIG-ALBHW*, as 41 optimal solutions were found and on average 37.59% faster.

KEYWORDS. Smart Cities. Assembly Line Balancing with Hierarchical Worker Assignment. Hybrid heuristics.

Smart Cities, metaheuristic

1. Introdução

Para o desenvolvimento de uma cidade inteligente é necessário que haja uma boa infraestrutura de indústrias na região, sendo de grande importância o fornecimento de máquinas, empregos, crescimento econômico, novas tecnologias, rotatividade de pessoas e atenção das demais cidades, além de que as cidades e a indústria dividem os mesmos recursos, como espaço físico, água, luz e serviços públicos, em geral. Deste modo, é nítida a relevância de estudar essa união. Para as Cidades e Regiões Inteligentes e Sustentáveis (CRIS) não seria diferente [Horejsi et al., 2020].

A utilização de métodos eficientes para resolver problemas de alocação com recursos limitados em CRIS permite que indústrias otimizem sua produção, além de oferecer flexibilidade para adaptações em linhas de produção personalizadas. Em um contexto de sustentabilidade, uma maior eficiência na linha de produção pode reduzir o consumo de energia e recursos. Podemos exemplificar isso em uma linha de produção de bicicletas e carros elétricos, bem como dispositivos inteligentes em uma cidade inteligente. Dessa forma, as cidades inteligentes podem otimizar sua produção e distribuição, promovendo eficiência e sustentabilidade.

Neste contexto, destaca-se o *Simple Assembly Line Balancing* (SALBP) que consiste em um sistema de linha de montagem, com tarefas indivisíveis e com ordem de execução. Cada estação da linha de montagem é responsável por executar uma parte do conjunto de tarefas, alocando um trabalhador e cumprindo um tempo de ciclo. Dessa forma, o SALBP tem como objetivo principal encontrar a melhor alocação de tarefas nas estações visando minimizar o número de estações e/ou o tempo total de um ciclo [Sungur e Yavuz, 2015].

Conhecido na literatura como problema de balanceamento de linhas de produção e designação hierárquica de trabalhadores, o *Assembly Line Balancing with Hierarchical Worker Assignment* (ALBHW), é uma derivação do SALBP que adiciona a característica de hierarquia aos trabalhadores. Assim, o conjunto de trabalhadores é classificado por um nível de qualificação para a execução das tarefas. O objetivo do ALBHW é encontrar a melhor alocação de tarefas aos trabalhadores em suas estações de trabalho, respeitando as restrições de precedência de tarefas, tempo total de um ciclo e qualificação dos trabalhadores. O balanceamento é necessário visto que a hierarquia irá definir o custo total, quanto mais qualificado o trabalhador para executar a tarefa, maior será o seu custo de alocação.

Por conta da principal característica do problema ser a hierarquia dos trabalhadores, algo presente em vários problemas reais e práticos, como saúde, indústrias, manutenção, escalonamento entre outros Sungur e Yavuz [2015], possui uma grande relevância em seu estudo, principalmente nos casos de indústrias para CRIS.

Este artigo propõe um método eficiente para resolver o ALBHW baseado nas meta-heurísticas *Iterated Local Search* (ILS) [Lourenço et al., 2010] e *Multi-Start* [Martí et al., 2013]. A busca local do método é realizada pelo *Randomized Variable Neighborhood Descent* (RVND) [Souza et al., 2010] a partir de um conjunto de seis estruturas de vizinhanças (EV). Para a construção de cada solução inicial, foi utilizado a fase construtiva parcialmente gulosa do *Greedy Randomized Adaptive Search Procedure* (GRASP) [Feo e Resende, 1995].

Este trabalho introduz pela primeira vez uma abordagem heurística eficiente para resolver o ALBHW. Após uma revisão literária notou-se a presença de uma série de modelagens exatas para o problema e somente uma abordagem heurística Topaloglu et al. [2020], que embora eficiente em muitos casos possuem limitações de tempo computacional, memória e entre outros. A Heurística proposta oferece uma alternativa para casos em que a modelagem já não consegue resolver com tanta eficiência.

O artigo está dividido da seguinte forma: a Seção 2 apresenta a descrição formal da modelagem matemática; a Seção 3 faz uma revisão literária de métodos mais comuns usados para

resolver o problema; na Seção 4 há a descrição da abordagem apresentada; na Seção 5 estão os experimentos computacionais realizados; a Seção 6 apresenta a conclusão e trabalhos futuros.

2. Descrição do Problema

Solucionar o ALBHW é encontrar a melhor alocação de trabalhadores e tarefas em suas estações de trabalho. Tal alocação acarreta a minimização dos custos, onde um trabalhador mais caro pode ser substituído por trabalhadores mais baratos, sempre respeitando as restrições de precedência, tempo máximo de uma estação e a capacidade de cada trabalhador executando uma tarefa. O ALBHW pode ser definido da seguinte forma: dada uma linha de montagem com um número finito de estações $S = \{1, \dots, m\}$, com n tarefas indivisíveis $T = \{1, \dots, n\}$ e com uma ordem de precedência. As tarefas predecessoras imediatas da tarefa i fazem parte do conjunto P_i de arcos $A = \{(j, i) | i \in T, j \in P_i\}$, então dizemos que se existe arco (j, i) , significa que j é um predecessor imediato a tarefa i , logo precisa ser executada antes de i . Toda estação s possui um tempo total de ciclo Z que não pode ser ultrapassado pelo conjunto T_s de tarefas atribuídas.

Todas as tarefas e trabalhadores possuem um nível de qualificação dado pelos conjuntos $K = \{1, \dots, \iota\}$ e $H = \{1, \dots, \nu\}$, logo quanto menor o número, mais qualificado é, e consequentemente maior será o seu custo c . O tipo de uma tarefa i é representado por k_i e só pode ser executada pelo trabalhador $h \forall h \leq k_i$. Temos que o trabalhador só poderá ser alocado a uma estação s se $h \leq \min \{k_i | i \in T_s\}$ for verdadeiro. Uma característica única deste problema é a qualificação mínima necessária para a execução de uma tarefa. Além disso, cada tipo de trabalhador i possui um tempo diferente definido por t_{ih} ao executar a tarefa. Este tempo de processamento da tarefa i é proporcional ao tipo de trabalhador, quanto menor o tipo do trabalhador, menos tempo será necessário para realizá-la. A função objetivo (Equação (1)) do ALBHW então busca minimizar o custo total C do balanceamento, onde c_h é o custo do trabalhador h e y_h é o número total de trabalhadores alocados nas estações.

Para melhor entendimento do ALBHW, temos uma definição formal dada pela modelagem matemática de Sungur e Yavuz [2015]. Sejam: (i, j) uma tarefa $(i, j \in T)$; h um tipo do trabalhador $h \in H$; s uma estação $(s \in S)$; m número máximo de estações; n número de tarefas; ι número de tarefas por tipo de trabalhador; T conjunto de tarefas; S conjunto de estações; K conjunto de tipos de tarefas; H conjunto de tipos de trabalhadores; P_i conjunto de predecessores imediatos a tarefa i ; A conjunto de precedências entre as tarefas; T_s conjunto de tarefas alocadas na estação s ; h_s tipo do trabalhador alocado na estação s ; k_i tipo da tarefa i ; c_h custo do trabalho h ; t_{ij} tempo da tarefa i ao ser executada pelo trabalhador h ; Z tempo total do ciclo; v_s variável de decisão se a estação s foi escalonada; x_{ihs} variável de decisão se a tarefa i foi escalonada na estação s do tipo de trabalhador h ; y_{hs} variável de decisão se o trabalhador do tipo h foi escalonado na estação s ; y_h quantidade total de trabalhadores h alocados; C custo total da solução.

$$\min \sum_h c_h y_h \quad (1)$$

Sujeito a:

$$\sum_{h \leq k_i} \sum_s x_{ihs} = 1, \quad \forall i \in T \quad (2)$$

$$\sum_h y_{hs} = v_s, \quad \forall s \in S \quad (3)$$

$$\sum_i x_{ihs} \leq n y_{hs}, \quad \forall h \in H, \forall s \in S \quad (4)$$

$$\sum_s y_{hs} = y_h, \quad \forall h \in H \quad (5)$$

$$\sum_{h \leq k_j} \sum_s s x_{jhs} \leq \sum_{h \leq k_i} \sum_s s x_{ih s}, \quad \forall (j, i) \in A \quad (6)$$

$$\sum_i \sum_{h \leq k_j} t_i n x_{ih s} \leq Z, \quad \forall s \in S \quad (7)$$

$$v_s \geq v_{s+1}, \quad \forall s \in S \mid s < m \quad (8)$$

$$v_s \in \{0, 1\}, \quad \forall s \in S \quad (9)$$

$$x_{ih s} \in \{0, 1\}, \quad \forall i \in T, \forall h \in H \mid h \leq k_i, \forall s \in S \quad (10)$$

$$y_{hs} \in \{0, 1\}, \quad \forall h \in H, \forall s \in S \quad (11)$$

O modelo é definido da seguinte forma: (1) A função objetivo; (2) certificam que a tarefa i está atribuída a uma única estação; (3) garantem que cada estação está atribuída a um trabalhador; (4) asseguram que o trabalhador h foi alocado em uma estação s ; (5) determinam o número de trabalhadores; (6) garantem que todas as precedências estão sendo respeitadas; (7) certificam que a soma dos tempos de execuções das tarefas i executadas pelo trabalhador h não ultrapassa o tempo total do ciclo; (8) fazem com que as estações sejam executadas de maneira sequencial; (9), (10) e (11) garantem a integridade das variáveis binárias de decisão.

Uma ilustração do problema pode ser visualizada na Figura 1. Cada coluna da imagem representa uma estação, logo a linha de produção possui seis estações. Cada nó representa uma tarefa e abaixo a qualificação mínima de um trabalhador para que ela possa ser executada. O rótulo acima do nó indica o tempo necessário para execução de acordo com a qualificação dos trabalhadores (tipo 1, tipo 2, tipo 3). Caso esse número não exista, significa que determinada tarefa não pode ser executada por aquele tipo. Então temos que a tarefa 11, por exemplo, pode ser executada pelos três tipos de trabalhadores, cada um com seu respectivo tempo de execução. A tarefa 11 possui como predecessoras as tarefas 9, 7, 3 e 1, ou seja, a tarefa 11 só pode ser executada quando todas essas tarefas forem finalizadas.

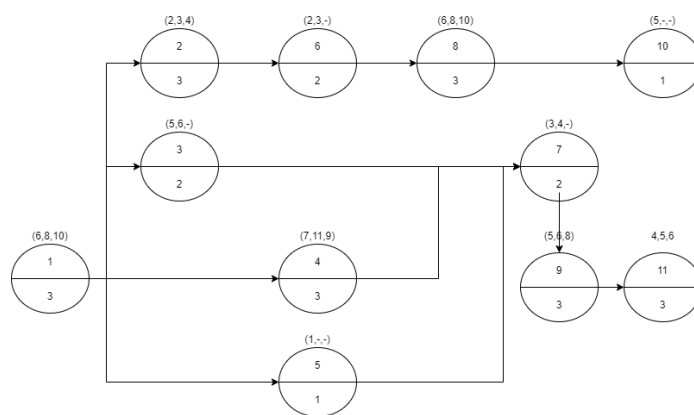


Figura 1: Exemplo do ALBHW (Adaptado de Sungur e Yavuz [2015])

A utilização de ferramentas eficientes para realizar o escalonamento de trabalhadores em relação a suas tarefas e estações de trabalho podem conduzir uma gestão mais eficiente, uma vez que bem escalonado o tempo de execução do ciclo pode ser reduzido, assim como o gasto com trabalhadores.

3. Referencial Teórico

Sungur e Yavuz [2015] apresentaram uma formulação de um modelo de programação inteira descrito na Seção 2 para a variação do problema tratado neste artigo. Os autores fizeram uma adaptação do conjunto de instâncias para o SALB proposto por [Otto et al., 2013]. Para o primeiro conjunto de instâncias pequenas, gerou soluções ótimas para todas as 225 instâncias. Para o conjunto de instâncias médias gerou soluções ótimas para 116 de 225.

Polat et al. [2016] utilizaram de uma abordagem de duas fases com o VNS para o *assembly line worker assignment and balancing problem type-II (ALWABP-2)*. Inicialmente o método faz a alocação das tarefas em estações de trabalho, minimizando o tempo de ciclo e respeitando a ordem de prioridade, em um segundo momento, o VNS aloca os trabalhadores nas estações criadas. Dessa forma, o algoritmo conseguiu atingir o melhor resultado da literatura em 16 e melhorou o resultado em 16 de 32 instâncias.

Campana e de O. Moreira [2018] propuseram uma formulação matemática juntamente com métodos heurísticos para solucionar o ALBHW. A heurística gulosa usada foi baseada em prioridades para a execução das tarefas, ou seja, quanto mais sucessores uma tarefa tem, maior prioridade ela terá de ser incluída na solução inicial. Sua proposta apresentou um percentual de 2% a 5% das soluções ótimas conhecidas na literatura.

Topaloglu et al. [2020] propuseram modelos para resolver o problema de balanceamento de linhas de produção e designação hierárquica de trabalhadores com restrições de posicionamento e paralelismos das estações de trabalho. Para este problema, existem tarefas isoladas que exigem serem alocadas sozinhas em uma estação de trabalho, um conjunto de tarefas que exigem estarem alocadas exatamente na mesma estação e permitir que estações sejam alocadas em paralelo caso o tempo de uma tarefa excedam o tempo total do ciclo. Inicialmente foi desenvolvido um modelo de programação inteira nomeado IP que se mostrou insuficiente para instâncias médias e grandes. Em seguida propuseram um algoritmo baseado no *Simulated Annealing* (OSAA) e para refinar as soluções, unificaram o OSAA com um procedimento de busca local (HSAA), ao total, três métodos foram apresentados. Os parâmetro do HSAA foram definidos pela utilização de um modelo exato. O IP conseguiu atingir o ótimo em 22 das 54 das instâncias testadas, o OSAA 34 de 54, enquanto o HSAA 48 das 54 testadas.

Após uma revisão da literatura, verificou-se a importância do estudo de métodos heurístico que possuem uma metodologia com uma complexidade computacional simplificada quando comparado a métodos exatos para o ALBHW, visto que o único encontrado baseado em *Simulated Annealing* adicionou novas restrições ao problema.

4. Metodologia

A metodologia utilizada para resolver o ALBHW está descrita a seguir da seguinte forma. Primeiro é mostrado como cada uma das soluções será representada (Seção 4.1) e em seguida como o algoritmo as avalia (Seção 4.2). Logo após, o algoritmo proposto *MIG-ALBHW* está descrito (Seção 4.3), sua heurística construtiva (Seção 4.4), sua busca local (Seção 4.5) e por fim suas estruturas de vizinhanças e perturbações (Seção 4.6).

4.1. Representação da Solução

O ALBHW é composto por um conjunto de tarefas a serem alocadas em determinadas estações, dessa forma, para representar uma solução do ALBHW são utilizados dois vetores $S = (E, T)$. O Vetor E , contém todas as tarefas separadas por zeros, ou seja, cada conjunto de tarefas entre zeros contém as tarefas de uma estação. Por sua vez, o Vetor T é composto pelos tipos de trabalhadores responsáveis por executar as tarefas de cada estação.

De acordo com a imagem, as tarefas $e_1 = \{1, 2, 3, 4, 7, 10, 12, 15\}$, pertencem a primeira estação com tipo de trabalhador 1, as tarefas $e_2 = \{5, 11, 14, 17, 18\}$ pertencem a segunda estação

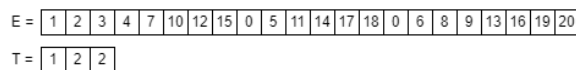


Figura 2: Representação da Solução

com qualificação de trabalhador tipo 2 e por fim as tarefas $e_3 = \{6, 8, 9, 13, 16, 19, 20\}$ fazem parte da terceira estação também do tipo de trabalhador 2. Essa solução possui valor de $fo = 240$ referente a primeira instância. Os trabalhadores do tipo 1 possuem 100 de custo enquanto os trabalhadores do tipo 2 possuem 70 de custo.

4.2. Função de Avaliação

A função $f(s)$ representada pela Equação (12) é responsável por avaliar o escalonamento dos trabalhadores a partir de uma solução do ALBHW. O cálculo dos custos dos trabalhadores é feito pela função $c(s)$. As constantes M_1 e M_2 são responsáveis por penalizar as soluções que violarem as restrições, tempo total de ciclo e a precedência de cada tarefa respectivamente. O valor que M_1 e M_2 serão multiplicados é calculado pelas funções $\mu(s)$ e $\nu(s)$. Essas funções retornam quantas violações de cada restrição a solução teve. Em uma solução viável, os valores retornados de $\mu(s)$ e $\nu(s)$ são zero, restando apenas o somatório dos custos dos trabalhadores alocados.

$$f(s) = \min \sum_{T=0}^n c(s) + M_1\mu(s) + M_2\nu(s) \quad (12)$$

4.3. Algoritmo MIG-ALBHW

Para resolver o ALBHW a abordagem proposta se baseia na utilização das meta-heurísticas *Iterated Local Search* (ILS) Lourenço et al. [2010] e *Multi-Start* [Martí et al., 2013] que utiliza como busca local o *Randomized Variable Neighborhood Descent* RVND [Souza et al., 2010]. O método de construção inicial da solução utiliza a fase construtiva do GRASP [Feo e Resende, 1995] (Seção 4.4). Nomeado de MIG-ALBHW, o método está representado pelo Algoritmo 1.

O Algoritmo é inicializado pelos valores dos parâmetros $ILSmax$ e $MSmax$. O laço iniciado na linha 6, é referente ao *Multi-Start* e irá acontecer $MSmax$ vezes, dessa forma gerando uma nova solução a cada iteração. O Laço iniciado na linha 10 irá ser executado $ILSmax$ vezes, conforme a melhora na solução corrente. Dessa forma, o MIG-ALBHW pode gerar múltiplas soluções iniciais que irão sofrer perturbações seguidas de buscas locais enquanto tiver uma melhora.

4.4. Heurística Construtiva

Para a construção inicial das soluções foi utilizado um algoritmo parcialmente guloso baseado na fase construtiva do GRASP, um procedimento que construirá a solução elemento a elemento, adaptado do método totalmente guloso apresentado por Scholl e Vob [2012], Moreira et al. [2012] e por Campana e de O. Moreira [2018]. O método irá criar uma lista chamada de *MaxIF* ordenada de maneira crescente para cada uma das tarefas com o maior número de precedentes. Diferentemente dos trabalhos citados, a adição de cada elemento será conforme o sorteio de uma lista restrita de candidatos extraída de *MaxIF* de acordo com o valor α . Dessa forma, a seleção de uma tarefa a ser adicionada solução é aleatória, porém dentro de um vetor restrito de tarefas candidatas.

A construção da solução inicial utilizando a lista *MaxIF* está representada pelo Algoritmo 2. O método é iniciado por uma lista de tarefas candidatas (LC) ordenada pelo critério do número de precedentes. O laço da linha 4 é iniciado com a criação da lista de candidatos restritos para a primeira estação. Dessa forma, o valor probabilístico *alpha* garante que a primeira tarefa será selecionada de maneira aleatória a partir do valor *alpha*. O tipo de cada estação será definido pelo

Algoritmo 1: MIG-ALBHW(ILS_{max}, MS_{max})

```

1  início
2      iter ← 0;
3      Seja LP a lista de perturbações;
4      iter ← melhorIter ← 0;
5      nivel ← 1;
6      enquanto (iter - melhorIter < MSmax) faça
7          iter + +;
8          vezes ← 0;
9          s ← MaxIF(α); {Gera uma solução Inicial}
10         enquanto (iter - melhorIter < ILSmax) faça
11             ntroca ← nivel + 1;
12             Seja p um número aleatório de LP;
13             s' ← Pertubacao(p, ntroca); {Aplicar ntroca perturbações}
14             s ← RVND(); {Aplica a Busca Local}
15             se (f(s) < f(s*)) então
16                 s* ← s;
17                 f* ← f(s);
18                 vezes ← 0;
19                 nivel ← 1;
20                 melhorIter ← iter;
21             fim
22             vezes + +;
23         fim
24         nivel + +;
25     fim
26     Retorne s;
27 fim
    
```

menor tipo de trabalhador dessa tarefa adicionada. As tarefas restantes são copiadas para a LC2 auxiliar que irá entrar no laço da linha 9 adicionando novas tarefas até que o critério de parada seja atingido. Neste caso, o critério de parada pode ser: tempo total de ciclo atingiu o máximo; ou não há mais tarefas (LC2 ficou vazio); ou nenhuma tarefa pode ser adicionada na estação.

Algoritmo 2: $MaxIF(LC)$

```

1  início
2      Seja LC2 a lista auxiliar;
3      repita
4          LCR ← α * LC;
5          Selecionar aleatoriamente uma tarefa i de LCR;
6          Remover i de LC;
7          LC2 ← LC;
8          repita
9              s' ← Adicionar(i);
10             se (tarefa i foi adicionada) então
11                 remover i de LC;
12                 remover i de LC2;
13             fim
14             senão
15                 remover i de LC2;
16             fim
17         até (Critério de Parada);
18     até (LC == 0);
19     Retorne s';
20 fim
    
```

Esse método semi-guloso permite a geração de diferentes soluções a cada iteração do método construtivo do GRASP. Uma vez que o valor de α se aproxima de 1, a solução gerada se torna mais determinística, quanto mais próximo de 0, a solução se torna mais aleatória.

4.5. Busca Local

A busca local utilizada pelo método é realizada através do *Randomized Variable Neighborhood Descent* [Souza et al., 2010] (Algoritmo 3) realizando a descida completa das vizinhanças de cada uma das estruturas de vizinhanças (EV) descritas na Seção 4.6. O método irá realizar o refinamento da solução corrente que acontece pelos movimentos responsáveis por explorar todas as estruturas de vizinhança. Tais movimentos são baseados em trocas, substituição e realocação de tarefas e trabalhadores nas estações, aceitando somente soluções melhores do que a corrente.

Algoritmo 3: $RVND(LV)$

```

1  início
2      Seja  $LV'$ ;
3      enquanto  $LV \neq 0$  faça
4          Selecione uma EV  $N \in LV'$  aleatoriamente;
5          Encontre o melhor vizinho  $s' \in N(s)$ ;
6          se  $(f(s) < f(s'))$  então
7               $s \leftarrow s'$ ;
8               $LV' \leftarrow LV$ ;
9          fim
10         senão
11             Remova  $N$  de  $LV'$ ;
12         fim
13     fim
14     Retorne  $s$ ;
15 fim
    
```

4.6. Estruturas de Vizinhança e Perturbações

Seis tipos de estruturas de vizinhanças (N1 a N6) são exploradas através de movimentos específicos, descritos a seguir (Conjunto):

- N1 — Uma tarefa i_1 da estação e_1 é trocada com a tarefa i_2 da estação e_2 ;
- N2 — Duas tarefas i_1 e i_2 são trocadas da estação e_1 com as tarefas j_1 e j_2 da estação e_2 ;
- N3 — Uma tarefa i_1 da estação e_1 é removida de e_1 e inserida em e_2 ;
- N4 — Uma nova estação e_1 é adicionada na solução;
- N5 — Uma estação e_1 é removida da solução, unidos as tarefas de duas estações;
- N6 — O tipo de uma estação e_1 é trocado;

Para a realização das perturbações na solução corrente durante o método ILS, uma sequência de aplicações de cada uma das EV foram usadas, dessa forma se torna possível fugir de ótimos locais. A sequência definida foi: $\{N1, N3, N4, N5\}$.

5. Experimentos Computacionais

Os experimentos computacionais para o MIG-ALBHW foram realizados no conjunto de instâncias proposto por Otto et al. [2013] e adaptado por [Sungur e Yavuz, 2015]. O Algoritmo foi projetado em c++ e compilado (*GNU GCC Compiler*) em computador *Asus* com 8GB de memória RAM com um processador *Intel(R) Core(TM) i7-7500U CPU @ 2.70ghZ 2.90 GHz*. O método foi testado em duas partes, a primeira para instâncias nas quais o modelo matemático finalizou e gerou o resultado ótimo e a segunda em que o processo foi encerrado por tempo limite de execução, gerando soluções de boa qualidade, porém não ótimas. Os resultados estão expressos na Tabela 1.

Para as tabelas de resultados entende-se: *Inst.* como o número da instância; *BKS* a melhor solução conhecida da literatura; *Melhor Sol.* como a melhor solução encontrada pelo método; *tempo(s)* o tempo de execução em segundos; *Sol.* o valor médio de soluções; *gap(%)* representa o desvio em relação ao valor de função objetivo da melhor solução da literatura; *ganho de tempo* representa o desvio em relação o tempo da melhor solução da literatura, com o MIG-ALBHW. O *gap* pode ser calculado pela Equação a seguir:

$$gap = \frac{x_{met} - x^*}{x^*} \times 100.$$

A variável x_{met} representa o melhor resultado do método proposto e x^* o melhor resultado presente na literatura. Os parâmetros do MIG-ALBHW foram definidos de forma empírica, sendo o parâmetro do ILS, *ILSmax* definido em 10, enquanto o *MSmax* que irá gerar múltiplas soluções durante a execução foi definido em 30. O valor de α no método construtivo do GRASP, foi definido

Tabela 1: Resultados do MIG-ALBHW para o ALBHW

Inst.	BKS	Sungur e Yavuz		MIG-ALBHW		Sol ²	gap(%)	ganho de tempo
		Melhor Sol	tempo(s)	Melhor Sol	tempo(s)			
291_0.7	240¹	240	0.80	240	0.59	248.20	0.00	26,3
292_0.7	240¹	240	1.00	240	0.62	256.70	0.00	38,0
297_0.7	268¹	268	1.10	268	0.55	268.00	0.00	50,0
303_0.7	268¹	268	0.80	268	1.10	277.30	0.00	-37,5
313_0.7	268¹	268	0.70	268	0.73	279.10	0.00	-4,3
318_0.7	823¹	823	87.0	823	4.50	903.50	0.00	94,8
319_0.7	972¹	972	16.0	972	4.00	989.40	0.00	75,0
320_0.7	865¹	865	19.0	865	6.81	922.00	0.00	64,2
324_0.7	746¹	746	153.0	753	7.63	775.60	0.94	95,0
329_0.7	804¹	804	739.0	804	4.80	863.30	0.00	99,4
348_0.7	438¹	438	6.60	438	1.90	482.70	0.00	71,2
349_0.7	387¹	387	16.0	387	2.60	404.10	0.00	83,8
350_0.7	387¹	387	3.80	387	2.36	418.00	0.00	37,9
352_0.7	317¹	317	2.00	317	1.24	322.10	0.00	38,0
353_0.7	466¹	466	14.0	466	1.88	496.10	0.00	86,6
367_0.7	268¹	268	1.20	268	1.01	277.40	0.00	15,8
368_0.7	268¹	268	1.10	268	1.30	268.00	0.00	-18,2
370_0.7	268¹	268	0.70	268	0.97	276.10	0.00	-38,6
373_0.7	268¹	268	0.80	268	1.34	274.00	0.00	-67,5
385_0.7	268¹	268	0.60	268	0.36	271.30	0.00	40,0
391_0.7	902¹	902	33.0	902	15.16	939.00	0.00	54,1
393_0.7	844¹	844	46.0	844	6.51	910.10	0.00	85,8
398_0.7	855¹	855	37.0	864	8.45	944.30	1.05	77,2
409_0.7	895¹	895	35.0	916	7.33	965.60	2.35	79,1
415_0.7	846¹	846	179.0	846	9.80	874.60	0.00	94,5
424_0.7	438¹	438	3.80	438	3.15	454.20	0.00	17,1
427_0.7	478¹	478	82.0	478	7.11	502.90	0.00	91,3
432_0.7	387¹	387	1.50	387	2.30	408.80	0.00	-53,3
437_0.7	387¹	387	3.70	387	3.72	396.30	0.00	-0,5
438_0.7	466¹	466	7.00	466	6.90	491.50	0.00	1,4
441_0.7	268¹	268	1.20	268	1.54	271.00	0.00	-28,3
442_0.7	268¹	268	1.10	268	0.77	275.20	0.00	30,0
444_0.7	268¹	268	0.80	268	0.46	272.20	0.00	42,5
448_0.7	268¹	268	1.00	268	1.87	270.10	0.00	-87,0
453_0.7	268¹	268	0.80	268	4.85	217.00	0.00	-506,3
466_0.7	995¹	995	6.40	995	24.66	1158.60	0.00	-285,3
468_0.7	974¹	974	12.0	974	10.41	1163.40	0.00	13,3
476_0.7	867¹	867	13.0	874	14.87	981.28	0.81	-14,4
477_0.7	795¹	795	12.0	795	18.95	878.00	0.00	-57,9
479_0.7	1023¹	1023	8.90	1023	13.99	1236.90	0.00	-57,2
491_0.7	487¹	487	4.30	487	5.20	511.30	0.00	-20,9
494_0.7	438¹	438	1.10	438	1.12	469.20	0.00	-1,8
497_0.7	487¹	487	2.00	487	4.77	562.25	0.00	-138,5
499_0.7	438¹	438	3.40	438	7.61	438.00	0.00	-123,8
501_0.7	387¹	387	1.50	387	12.69	395.30	0.00	-746,0
Média							0,11	-17,44

¹Otimidade provada por Sungur e Yavuz [2015]; ²Solução Média para 30 execuções; ³Solução Média para 5 execuções;

em 0.7, desta forma as soluções geradas possuem um certo nível de aleatoriedade garantindo uma maior variabilidade de soluções iniciais.

Na primeira bateria de testes (Tabela 1), o MIG-ALBHW foi executado 30 vezes para cada instância e atingiu o ótimo em 41 das 45 instâncias e teve um gap médio de 0.11%, dessas 45 conseguiu resolver a instância em 26 gastando menos tempo com gap médio de -17,44%. Já para a segunda bateria de testes (Tabela 2), o MIG-ALBHW foi executado 5 vezes utilizando as instâncias cujo modelo matemático de Sungur e Yavuz [2015] encerrou a execução por tempo de processamento máximo. Nesta bateria de testes, o MIG-ALBHW conseguiu atingir 4 de 9 dos melhores resultados encontrados na literatura com gap médio de 1,63% e em um tempo computacional bastante inferior com gap médio de 92,63%.

Tabela 2: Resultados do MIG-ALBHW para o ALBHW

Inst.	Sungur e Yavuz			MIG-ALBHW			Sol ³	gap(%)	ganho de tempo
	BKS	Melhor Sol	tempo(s)	Melhor Sol	tempo(s)				
327_0.7	2131	2131	7200	2210	402.21	2270.00	3.71		3,4
331_0.7	2368	2368	7200	2466	354.86	2491.00	4.14		4,4
335_0.7	2173	2173	7200	2240	378.85	2301.00	3.08		3,4
337_0.7	2207	2207	7200	2265	382.73	2387.30	2.63		3,4
341_0.7	2133	2133	7200	2158	643.32	2294.20	1.17		31,3
352_0.7	774	774	7200	774	808.26	930.00	0.00		28,3
354_0.7	972	972	7200	972	979.22	1054.40	0.00		26,3
363_0.7	853	853	7200	853	379.03	887.80	0.00		3,4
371_0.7	853	853	7200	853	447.25	871.60	0.00		2,4
Média							1,63		92,63

³Solução Média para 5 execuções;

As perturbações na solução quebrando e unindo duas estações tiveram um grande peso para atingir as soluções em que a instância possui uma característica única onde as tarefas possuem um tempo de ciclo muito alto, pois dificilmente duas tarefas poderiam ficar em uma mesma estação. Dessa forma, percebeu-se que unir e dividir estações são movimentos importantes na busca por melhores soluções.

6. Conclusão

A utilização de métodos eficientes para resolver o ALBHW em cidades inteligentes podem proporcionar inúmeras vantagens, como redução de custos, tempo, consumo de energia entre outros, além de maximizar a produção. Este artigo apresentou uma abordagem híbrida para solucionar o *Assembly Line Balancing with Hierarchical Worker Assignment*. O método foi nomeado de MIG-ALBHW e aborda a união do ILS, *Multi-Start* e GRASP. Múltiplas soluções iniciais são geradas dentro do ILS utilizando uma abordagem de geração de solução inicial adaptativa gulosa e randômica, garantindo a variabilidade das soluções iniciais, que são em seguida refinadas pelo RVND.

O MIG-ALBHW foi testado em 54 instâncias no total, na primeira bateria de testes atingindo o ótimo em 41 de 45 com gap médio de 0,11% e com gap médio de ganho de tempo de -17,44%. Na segunda bateria de testes igualando o melhor resultado da literatura em outras 4 de 9 com gap médio de 0,11% e gap médio de ganho de tempo de 92,63%.

Em algumas soluções, especificamente as que possuem um número de estações grande, as perturbações na solução quebrando e unindo duas estações tiveram um grande peso para atingir as soluções em que a instância possui uma característica única onde as tarefas possuem um tempo de ciclo muito alto, pois raramente duas tarefas poderiam ficar em uma mesma estação. Dessa forma, explorar novos movimentos de união e divisão de estações pode ser um importante trabalho futuro

na busca por melhores soluções. Além disso, também se considera importante utilizar métodos eficientes para uma melhor definição dos parâmetros como o *Irace* [López-Ibáñez et al., 2011], que poderá melhorar ainda mais os resultados obtidos pelo *MIG-ALBHW*.

Referências

- Campana, N. P. e de O. Moreira, M. C. (2018). Modelo matemático e heurísticas para o problema de balanceamento de linhas de produção e designação hierárquica de trabalhadores. *L Simpósio Brasileiro de Pesquisa Operacional*.
- Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *J. Glob. Optim.*, 6(2):109–133.
- Horejsi, P., Novikov, K., e Simon, M. (2020). A smart factory in a smart city: Virtual and augmented reality in a smart assembly line. *IEEE Access*, 8:94330–94340.
- Lourenço, H. R., Martin, O. C., e Stützle, T. (2010). Iterated local search: Framework and applications. In *International Series in Operations Research & Management Science*, p. 363–397. Springer US, Boston, MA.
- López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., e Birattari, M. (2011). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3.
- Martí, R., Resende, M. G. C., e Ribeiro, C. C. (2013). Multi-start methods for combinatorial optimization. *Eur. J. Oper. Res.*, 226(1):1–8.
- Moreira, M. C. O., Ritt, M., Costa, A. M., e Chaves, A. A. (2012). Simple heuristics for the assembly line worker assignment and balancing problem. *Journal of Heuristics*, 18(3):505–524.
- Otto, A., Otto, C., e Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *Eur. J. Oper. Res.*, 228(1):33–45.
- Polat, O., Kalayci, C. B., Mutlu, Ö., e Gupta, S. M. (2016). A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-II: an industrial case study. *Int. J. Prod. Res.*, 54(3):722–741.
- Scholl e Vob (2012). Simple assembly line balancing — heuristic approaches. *Journal of Heuristics*, 2(3):217–244.
- Souza, M. J. F., I. M. Coelho, S. R., Santos, H., e Merschmann, L. H. C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal Of Operational Research*, 207:1041–1051.
- Sungur, B. e Yavuz, Y. (2015). Assembly line balancing with hierarchical worker assignment. *Journal of Manufacturing Systems*, 37:290 – 298. ISSN 0278-6125. URL <http://www.sciencedirect.com/science/article/pii/S0278612514000983>.
- Topaloglu, S., Yildiz, G., e Okyay, R. (2020). Assembly line balancing problem with hierarchical worker assignment, positional constraints, task assignment restrictions and parallel workstations. *The International Journal of Industrial Engineering: Theory, Applications and Practice*, 27:345–377.