Douglas Geovanini de Paiva Mosca Leite,
Luiz Otávio...
& Marcos...

Trabalho Prático Final – GCC118 Programação Matemática – PCC540 Linear and Integer Programming

Brasil

2025, Lavras

Douglas Geovanini de Paiva Mosca Leite,
Luiz Otávio...
& Marcos...

Trabalho Prático Final – GCC118 Programação Matemática – PCC540 Linear and Integer Programming

Trabalho Prático Final – GCC118 Programação Matemática – PCC540 Linear and Integer ProgrammingLaTeX.

Universidade Federal de Lavras – UFLA

Brasil
2025, Lavras

Resumo

O Problema de Balanceamento de Linhas de Produção e Designação de Trabalhadores (ALWABP) consiste em atribuir tarefas a estações de trabalho e designar trabalhadores a essas estações, considerando relações de precedência entre tarefas, tempos de execução variáveis por trabalhador e restrições de incapacidade específicas, com o objetivo de minimizar o tempo de ciclo da linha de produção. Este trabalho implementa e calibra uma metaheurística *Iterated Local Search* (ILS) via Optuna, comparando seu desempenho com um modelo exato resolvido pelo Gurobi sob limite de 700 segundos. Os parâmetros otimizados (*cooling_rate* = 0,9265, *initial_temp_factor* = 0,0543, perturbações entre 1 e 5) foram aplicados a 48 instâncias de quatro famílias. O Gurobi resolveu 31 instâncias; o ILS encontrou soluções viáveis para todas, igualando o valor do Gurobi nas famílias Heskiaoff e Roszieg e apresentando gaps de 2–18% em Tonge e Wee-Mag, onde o solver exato frequentemente expirou. Os resultados indicam que o ILS oferece boa robustez e qualidade competitiva, sobretudo quando o tempo limita a convergência do modelo exato.

Lista de ilustrações

Sumário

1 Introdução

A Pesquisa Operacional (PO) consolidou-se como uma área científica interdisciplinar dedicada ao apoio à tomada de decisões em sistemas complexos, empregando métodos matemáticos, estatísticos e computacionais para modelar, analisar e otimizar processos. Seu surgimento remonta à Segunda Guerra Mundial, quando estudos aplicados à logística, ao planejamento de recursos e ao uso estratégico de tecnologias militares impulsionaram o desenvolvimento de ferramentas analíticas avançadas, posteriormente estendidas a diversos setores civis (ARENALES et al., 2007). Nesse contexto, os modelos de otimização desempenham papel central ao fornecer abstrações matemáticas capazes de representar, de forma estruturada, aspectos fundamentais de um sistema real, permitindo identificar soluções viáveis e eficientes para problemas de alocação, sequenciamento e dimensionamento (GOLDBARG; LUNA, 2005).

Entre os problemas estudados no âmbito da otimização combinatória destaca-se o Assembly Line Worker Assignment and Balancing Problem (ALWABP), ou Problema de Alocação e Balanceamento de Trabalhadores em Linhas de Montagem. Essa variante avançada do balanceamento clássico de linhas incorpora, além da distribuição de tarefas, a alocação de trabalhadores com diferentes níveis de habilidade, tornando-se particularmente relevante em ambientes como centros de trabalho protegidos, que empregam pessoas com deficiência. Nesse tipo de cenário, as capacidades individuais influenciam significativamente os tempos de processamento, exigindo modelos capazes de integrar restrições de precedência, heterogeneidade humana e limites de tempo de ciclo. O ALWABP, portanto, além de seu valor operacional, representa um desafio associado à inclusão produtiva, ao reconhecer a diversidade de perfis e promover práticas de organização do trabalho socialmente responsáveis (YILMAZ, 2021).

Dada a complexidade estrutural do ALWABP, métodos exatos podem tornar-se inviáveis para instâncias de tamanho real. Assim, o emprego de metaheurísticas torna-se uma alternativa promissora, uma vez que essas estratégias se baseiam em mecanismos estocásticos capazes de explorar o espaço de soluções de maneira eficiente, evitando a explosão combinatória típica de algoritmos exatos (LUKE, 2013; GLOVER; SÖRENSEN, 2015). Entre esses métodos, o sorteado foi Iterated Local Search (ILS), uma abordagem que combina simplicidade arquitetural e alta capacidade exploratória. No ILS, uma solução inicial é refinada por sucessivas buscas locais, intercaladas com perturbações controladas que promovem diversificação, enquanto um critério de aceitação orienta a evolução da trajetória de busca. Essa arquitetura, composta pelos módulos de geração inicial, busca local, perturbação e aceitação, permite construir uma sequência de soluções encadeadas que exploram o espaço de busca de maneira sistemática e eficiente (LOURENÇO; MARTIN;

STÜTZLE, 2001).

Diante desse contexto, este trabalho tem como objetivo resolver o Problema de Balanceamento de Linhas de Produção e Designação de Trabalhadores (ALWABP) por meio da implementação de uma metaheurística Iterated Local Search (ILS) e avaliando seu desempenho para a solução do problema.

2 O problema

O **ALWABP** (Assembly Line Worker Assignment and Balancing Problem) é um problema de otimização que combina balanceamento de linha de produção com designação de trabalhadores. Suas características principais são:

2.1 Características Fundamentais

- Linha de produção com m estações ordenadas linearmente
- Conjunto de k trabalhadores, onde $|S| = |W|$ (número de estações igual ao número de trabalhadores)
- Conjunto de n tarefas a serem distribuídas pelas estações
- Relações de precedência entre tarefas definidas por um grafo direcionado $G = (V, E)$
- Tempos de execução variáveis: t_{wi} representa o tempo da tarefa i pelo trabalhador w
- Restrições de incapacidade: I_w define tarefas que o trabalhador w não pode executar

2.2 Restrições do Problema

- Cada tarefa é designada a exatamente uma estação
- Cada trabalhador é alocado a exatamente uma estação
- Cada estação possui exatamente um trabalhador
- Precedências devem ser respeitadas: se $i \preceq j$, então i em estação anterior ou igual a j
- Tarefas em I_w não podem ser executadas pelo trabalhador w

2.3 Função Objetivo

Minimizar o **tempo de ciclo** da linha, definido como o maior tempo de execução entre todas as estações.

Informações Disponíveis

Para a implementação, dispõe-se de:

- Número de estações (m), trabalhadores (k) e tarefas (n)
- Matriz de tempos t_{wi} para todas combinações trabalhador-tarefa
- Conjuntos I_w de tarefas impossíveis para cada trabalhador
- Grafo de precedências entre as tarefas

3 Formulação Matemática do Problema

A seguir apresenta-se a formulação do problema de Balanceamento de Linhas de Produção com Designação de Trabalhadores (ALWABP) como um modelo de Programação Linear Inteira. O objetivo é minimizar o tempo de ciclo da linha, considerando a heterogeneidade dos trabalhadores, restrições operacionais e relações de precedência entre tarefas.

3.0.1 Variáveis de Decisão

- $v_{sw} \in \{0, 1\}$: variável binária que indica se o trabalhador w é alocado à estação s.
- $z_{siw} \in \{0, 1\}$: variável binária que indica se a tarefa i é executada na estação s pelo trabalhador w.
- $C \in R_+$: variável contínua que representa o tempo de ciclo da linha, definido como o maior tempo de processamento entre as estações.

3.0.2 Função Objetivo

O objetivo consiste em minimizar o tempo de ciclo da linha de produção. O tempo de ciclo é linearizado impondo que, para cada estação, o tempo total de execução das tarefas alocadas ao trabalhador desta estação não exceda C.

$$\min C$$

3.0.3 Restrições

(1) Atribuição única de cada tarefa

Cada tarefa deve ser executada exatamente uma vez, em uma única estação e por um único trabalhador:

$$\sum_{s \in S} \sum_{w \in W} z_{siw} = 1 \quad \forall i \in N$$

(2) Um trabalhador por estação

Cada estação deve possuir exatamente um trabalhador alocado:

$$\sum_{w \in W} v_{sw} = 1 \quad \forall s \in S$$

(3) Trabalhador em apenas uma estação

Cada trabalhador pode ocupar apenas uma estação da linha:

$$\sum_{s \in S} v_{sw} = 1 \quad \forall w \in W$$

(4) Vinculação tarefa–estação–trabalhador

Uma tarefa só pode ser atribuída a um trabalhador se este estiver de fato na estação selecionada:

$$z_{siw} \leq v_{sw} \quad \forall s \in S,\ \forall i \in N,\ \forall w \in W$$

(5) Incapacidades de execução

Caso o trabalhador w seja incapaz de executar a tarefa i (ou seja, $i \in I_w$), tal combinação é proibida:

$$z_{siw} = 0 \quad \forall s \in S,\ \forall w \in W,\ \forall i \in I_w$$

(6) Definição linear do tempo de ciclo

O tempo total de processamento das tarefas atribuídas à estação s não pode exceder o tempo de ciclo C:

$$\sum_{i \in N} \sum_{w \in W} t_{wi}\, z_{siw} \leq C \quad \forall s \in S$$

Essa restrição define o valor de C como o maior carregamento entre as estações.

(7) Restrições de precedência

Se a tarefa i deve preceder a tarefa j no processo produtivo, então i deve ser alocada a uma estação numericamente menor ou igual àquela que executa j:

$$\sum_{s \in S} \sum_{w \in W} s\, z_{siw} \ \leq \ \sum_{s \in S} \sum_{w \in W} s\, z_{sjw} \quad \forall (i,j) \in E$$

(8) Domínio das variáveis

$$v_{sw}, z_{siw} \in \{0, 1\}, \qquad C \geq 0$$

4 Iterated Local Search

??

O Iterated Local Search (ILS) é uma meta-heurística estocástica de otimização combinatorial pertencente à classe de algoritmos de busca baseada em trajetórias. O ILS tem como objetivo focar a busca em um subespaço reduzido, composto pelas soluções que são ótimos locais S^* para um determinado motor de otimização (normalmente uma busca local). A ideia central do ILS não é explorar o espaço de busca de forma completamente nova a cada iteração, mas sim realizar uma amostragem enviesada (ou dirigida) do conjunto de ótimos locais (LOURENÇO; MARTIN; STÜTZLE, 2001).

4.1 O Algoritmo Iterated Local Search

O ILS implementa uma busca heurística em S^* através de um passeio estocástico. Dado um ótimo local corrente s^*, o algoritmo aplica uma perturbação para gerar uma solução intermediária $s' \in S$. Em seguida, aplica-se a Busca Local a s' para se obter um novo ótimo local s^{**}. Um critério de aceitação decide se s^{**} substitui s^* como a solução corrente para a próxima iteração (LOURENÇO; MARTIN; STÜTZLE, 2001). A Figura 1 ilustra graficamente este processo. O algoritmo então, é executado sequencialmente de forma cíclica até que um critério de parada, como um número máximo de iterações ou um tempo de execução, seja satisfeito.

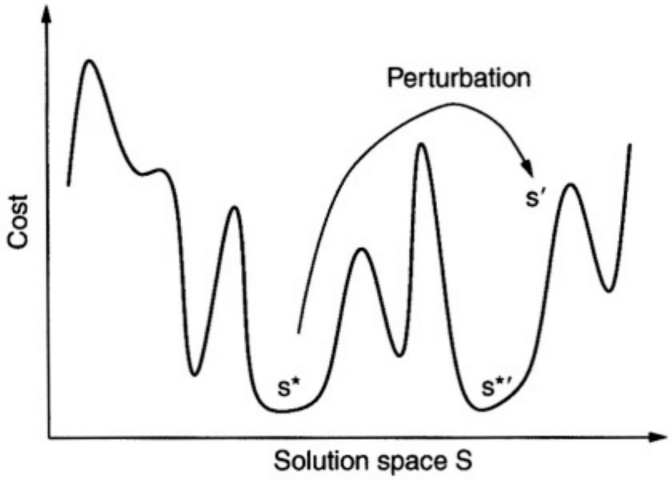

Figura 1 – Representação do ILS.

Fonte: (LOURENÇO; MARTIN; STÜTZLE, 2001)

A eficácia do ILS decorre do equilíbrio entre dois comportamentos complementares: a *intensificação*, realizada pela busca local, que aprofunda a exploração em torno de uma

solução; e a *diversificação*, introduzida pela perturbação e pelo critério de aceitação, que permite ao algoritmo explorar novas regiões do espaço de busca. Este equilíbrio é crucial, pois perturbações muito pequenas podem causar um retorno frequente ao mesmo s^*, enquanto perturbações muito grandes degradam a busca a uma reinicialização aleatória.

A estrutura geral do ILS é sintetizada no pseudocódigo 1:

Algorithm 1 Iterated Local Search (ILS)

```
1: s_0 = GeraçãoDaSoluçãoInicial()
2: s* = BuscaLocal(s_0)
3: while (Critério de parada não satisfeito) do
4:     s' = Perturbação(s*)
5:     s'' = BuscaLocal(s')
6:     s* = CritérioDeAceitação(s*, s'')
7: end while
8: return s*
```

4.2 Implementação do ILS ao problema ALWABP

A implementação do ILS para o problema ALWABP segue a arquitetura geral apresentada na Seção 4.1, sendo adaptada para incorporar as especificidades do domínio. A escolha das quatro componentes principais foi orientada pelas características do problema, em particular pela presença de relações de precedência rígidas, pela heterogeneidade dos tempos de execução e pelas restrições de incapacidade dos trabalhadores.

Cada uma dessas componentes e estratégias é detalhada nas subseções a seguir, com foco nas decisões de implementação tomadas para que o ILS atue de forma eficaz sobre a instância específica do ALWABP.

4.2.1 Geração de Solução Inicial

A qualidade da solução inicial pode influenciar a trajetória de busca do algoritmo ILS, embora seu efeito seja atenuado pelos mecanismos subsequentes de perturbação e aceitação. Para o ALWABP, adotou-se a heurística *Ranked Positional Weight* (RPW), proposta inicialmente por (HELGESON; BIRNIE, 1961) para problemas de balanceamento de linha. Esta heurística foi adaptada para incorporar as características específicas do ALWABP, particularmente a heterogeneidade dos trabalhadores e as restrições de incapacidade. O procedimento consiste em duas fases sequenciais:

Fase 1 - Cálculo dos Pesos Posicionais: Para cada tarefa i, determina-se um peso posicional ($RPW(i)$) que reflete sua carga de trabalho e a carga de todas as tarefas

que dependem dela na cadeia de precedências. O cálculo é realizado de forma recursiva a partir das tarefas sem sucessores (folhas do grafo) em direção às raízes:

$$RPW(i) = \bar{t}_i + \sum_{j \in S(i)} RPW(j) \tag{4.1}$$

onde $\bar{t}_i$ representa o tempo médio de execução da tarefa i considerando todos os trabalhadores, e $S(i)$ denota o conjunto de todos os sucessores diretos e indiretos de i no grafo de precedências. Esta métrica prioriza tarefas com maior duração própria e maior impacto subsequente no fluxo de produção.

Fase 2 - Atribuição Sequencial com Viabilidade: As tarefas são ordenadas em lista decrescente de $RPW(i)$. Em seguida, percorre-se esta lista sequencialmente, tentando alocar cada tarefa à primeira estação de trabalho que satisfaça um conjunto de condições de viabilidade: Todos as tarefas predecessores da tarefa em questão já devem ter sido alocados a estações com índice igual ou anterior ao da estação candidata; O trabalhador designado para a estação candidata deve ser capaz de executar a tarefa ($i \notin I_w$).

A designação inicial dos trabalhadores às estações é feita de forma aleatória, garantindo uma bijeção entre os conjuntos de trabalhadores e estações. Durante o processo de atribuição, caso uma tarefa não encontre uma estação viável com a configuração corrente de trabalhadores, o algoritmo inicia um subprocedimento de reatribuição de trabalhadores. Este subprocedimento permuta as alocações dos trabalhadores entre estações até que seja encontrada uma configuração que permita a alocação da tarefa bloqueada, mantendo a viabilidade das alocações já realizadas.

O resultado deste processo na maioria das vezes é uma solução viável para o ALWABP, que serve como ponto de partida para a primeira aplicação da busca local dentro do ciclo do ILS.

4.2.2 Busca Local

A busca local tem como função aprimorar uma solução corrente através da exploração sistemática de seu espaço de vizinhança. Para o ALWABP, foi implementado o *Variable Neighborhood Descent* (VND), um método que explora sequencialmente múltiplas estruturas de vizinhança predefinidas (HANSEN; MLADENOVIć, 2001).

O VND aplica duas vizinhanças distintas, descritas a seguir, em uma ordem fixa. Sempre que uma melhoria é encontrada em uma vizinhança, o algoritmo retorna à primeira para reiniciar a busca. O procedimento termina quando nenhuma das vizinhanças consegue produzir uma melhoria na solução corrente, indicando que um ótimo local em relação ao conjunto de vizinhanças foi alcançado. Esta abordagem de busca sequencial em múltiplas

vizinhanças tem sido aplicada com sucesso em problemas de balanceamento de linha com trabalhadores heterogêneos, como demonstrado por (POLAT et al., 2016).

Movimentação de Tarefas: A primeira vizinhança é gerada por movimentos que realocam uma única tarefa entre estações diferentes. Formalmente, dado um par de estações distintas (s_1, s_2) e uma tarefa i atualmente alocada à estação s_1, um vizinho é criado movendo-se i para s_2. Para que o movimento seja considerado viável, ele deve satisfazer três condições: A realocação não pode violar as relações de precedência do grafo, todos os predecessores de i devem estar em estações com índice menor ou igual a s_2, e todos os seus sucessores devem estar em estações com índice maior ou igual a s_2; O trabalhador w_2, alocado à estação destino s_2, deve ser capaz de executar a tarefa i ($i \notin I_{w_2}$); O movimento só é aceito se resultar em uma redução no tempo de ciclo global C.

O cálculo do novo tempo de ciclo C' após o movimento é eficiente, pois apenas as cargas das duas estações envolvidas são afetadas. Sejam C_1 e C_2 os tempos atuais das estações s_1 e s_2, respectivamente, e $t_{i,w}$ o tempo da tarefa i quando executada pelo trabalhador w. O novo tempo de ciclo é dado por:

$$C' = \max \left\{ C_1 - t_{i,w_1}, \quad C_2 + t_{i,w_2}, \quad \max_{k \notin \{s_1, s_2\}} C_k \right\} \tag{4.2}$$

onde w_1 e w_2 são os trabalhadores das estações origem e destino. A busca nesta vizinhança avalia exaustivamente todos os movimentos viáveis e aplica aquele que proporciona a maior redução em C.

Troca de Tarefas: A segunda vizinhança é gerada pela operação de troca de tarefas entre duas estações diferentes. Dadas duas estações s_1 e s_2, com tarefas i e j alocadas a elas, respectivamente, um vizinho é criado trocando-se as posições de i e j. A viabilidade da troca é verificada com base em critérios similares aos do movimento: Ambos os trabalhadores, w_1 (de s_1) e w_2 (de s_2), devem ser capazes de executar a nova tarefa que lhes será atribuída; As restrições de precedência devem ser preservadas para ambas as tarefas após a troca. Isso implica em verificar as relações de i e de j com suas novas posições nas estações s_2 e s_1, respectivamente; A troca deve resultar em um tempo de ciclo C' estritamente menor que o atual.

4.2.3 Perturbação

A perturbação constitui o mecanismo central de diversificação no ILS, sendo responsável por conduzir a busca para regiões distintas do espaço de soluções quando a busca local atinge um ótimo local. A implementação proposta para o ALWABP emprega dois operadores estocásticos de perturbação: **Realocação Aleatória de Tarefas** e **Permutação Aleatória de Trabalhadores**. Para controlar a intensidade e a direção da diversificação, cada operador é acionado segundo uma probabilidade pré-definida: **70%**

e **30%** respectivamente. A escolha destes operadores e suas probabilidades foi orientada pela estrutura específica do problema, que envolve dois conjuntos de decisões acopladas.

O operador de **Realocação Aleatória de Tarefas** seleciona aleatoriamente uma tarefa i e uma estação de destino s_{dest} diferente de sua estação atual s_{orig}. O movimento é executado apenas se for considerado viável, isto é, se satisfizer todas as restrições do problema: O trabalhador w_{dest} alocado à estação s_{dest} deve ser capaz de executar a tarefa i (i.e., $i \notin I_{w_{\text{dest}}}$); Todos os predecessores de i devem estar em estações com índice menor ou igual a s_{dest}, e todos os seus sucessores devem estar em estações com índice maior ou igual a s_{dest}.

Já o operador de **Permutação Aleatória de Trabalhadores** seleciona aleatoriamente um par de estações distintas, s_a e s_b, e troca os trabalhadores w_a e w_b alocados a elas. A troca é condicionada a uma verificação de viabilidade completa: ambos os trabalhadores devem ser capazes de executar *todas* as tarefas atualmente alocadas à sua nova estação. Formalmente, para cada tarefa i na estação s_a, deve valer $i \notin I_{w_b}$, e analogamente para as tarefas em s_b em relação a w_a. Este operador altera o perfil de competências associado a cada estação, podendo viabilizar novos arranjos de tarefas que eram previamente inviáveis devido a restrições de incapacidade.

A intensidade da perturbação é modulada pelo parâmetro *strength*, que determina o número total de operações de perturbação aplicadas consecutivamente para gerar s' a partir de s^*. O número de movimentos é dado por movimentos $= strength \times 10$.

4.2.4 Critério de Aceitação

O critério de aceitação do algoritmo segue a estrutura clássica do *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), permitindo tanto a aceitação de soluções melhores quanto a aceitação probabilística de soluções piores, com o objetivo de escapar de ótimos locais e explorar regiões mais amplas do espaço de busca.

Uma solução candidata s' é aceita em substituição à solução corrente s de acordo com a seguinte regra:

$$\text{aceitar}(s, s') = \begin{cases} \text{verdadeiro}, & \text{se } f(s') < f(s) \\ \text{verdadeiro}, & \text{se rand}() < \exp\left(-\dfrac{f(s') - f(s)}{T}\right) \\ \text{falso}, & \text{caso contrário} \end{cases}$$

em que:

- $f(s)$ representa o tempo de ciclo da solução s; - T é a temperatura atual do sistema, que controla a probabilidade de aceitação de soluções piores; - rand() retorna um número aleatório uniformemente distribuído no intervalo $[0, 1]$.

A temperatura inicial T_0 é definida como uma fração do custo inicial, conforme prática comum em algoritmos de *Simulated Annealing* (LAARHOVEN; AARTS, 1987):

$$T_0 = 0.1 \times f(s_{\text{inicial}})$$

Ao longo das iterações, a temperatura é progressivamente reduzida por meio de um fator de resfriamento $\alpha = 0.95$, seguindo a atualização:

$$T_{k+1} = \alpha \times T_k$$

Essa estratégia de resfriamento controlado permite um equilíbrio entre a fase exploratória inicial e a fase de exploração intensificada nas iterações finais.

4.2.5 Estratégias Adicionais

4.2.5.1 Adaptação Dinâmica da Perturbação

Foi implementada uma estratégia de adaptação dinâmica da perturbação, na qual o parâmetro *strength* que define a intensidade da perturbação, é ajustado dinamicamente durante a execução do algoritmo. Inicialmente, *strength* é definido com o valor padrão (normalmente 2). Sempre que o algoritmo ultrapassa um limiar pré-definido de iterações consecutivas (padrão 50) sem melhorias na solução, o valor de *strength* é incrementado em uma unidade, até atingir um limite máximo pré-definido (5 padrão).

4.2.5.2 Reinicialização

Foi implementado uma estratégia de reinicialização para evitar cenários de exaustão da região de busca local. Dessa forma, quando o algoritmo executa um número pré-definido de iterações (1000 padrão) iterações consecutivas sem que o *Simulated Anealing* aceite nenhuma nova solução o procedimento é acionado:

1. Gera uma nova solução inicial através da heurística RPW
2. Aplica VND nesta nova solução
3. Reinicia a temperatura para o valor inicial
4. Define $strength = valor_inicial + 1$ (ligeiramente maior que o inicial)
5. Mantém a melhor solução encontrada globalmente

A reinicialização permite que o algoritmo explore regiões distantes do espaço de busca quando a região atual está esgotada.

4.2.5.3 Timeout Adaptativo

Foi implementada uma estratégia de timeout adaptativo para equilibrar o tempo de execução do algoritmo frente a diferentes níveis de complexidade das instâncias do ALWABP. Quando a opção `adaptive-timeout` está ativada, o limite máximo de execução é ajustado dinamicamente com base no progresso recente da busca. A cada 50 iterações (após as primeiras 100), o algoritmo computa uma taxa de melhorias das soluções:

$$\text{taxa} = \frac{\text{número de melhorias nas últimas 50 iterações}}{50} \tag{4.3}$$

Se taxa > 0.1, o tempo máximo é estendido para $1.2 \times T_{\max}$, onde $T_{\max}$ é o tempo originalmente especificado. Este mecanismo concede tempo adicional quando o algoritmo está produzindo melhorias consistentes.

4.2.5.4 Critério de Parada

A implementação emprega um critério de parada multicondicional, garantindo que a execução seja interrompida assim que um dos limites for alcançado: (i) o número máximo de iterações pré-definido é atingido; (ii) o tempo máximo de execução é excedido; ou (iii) a solução encontrada apresenta uma qualidade considerada suficientemente próxima do ótimo, especificamente quando seu valor se situa dentro de uma tolerância de 0,01% em relação ao valor ótimo conhecido (UB) quando está disponível.

5 Metodologia

Este capítulo descreve a metodologia empregada para a avaliação do Solver Gurobi e a implementação algoritmo ILS proposto para o ALWABP descrito no capítulo **??**. São detalhados os procedimentos de otimização de parâmetros, a seleção das instâncias de teste, o protocolo experimental adotado e as métricas utilizadas para análise de desempenho.

5.1 Otimização de Parâmetros

A calibração dos parâmetros do algoritmo ILS foi realizada por meio de otimização automática utilizando o framework Optuna (AKIBA et al., 2019), que implementa o algoritmo Tree-structured Parzen Estimator (TPE) para busca eficiente no espaço de hiperparâmetros.

Os parâmetros selecionados para otimização e seus respectivos intervalos de busca são apresentados na Tabela 1. A definição dos intervalos dos parâmetros *taxa de resfriamento* e *fator de temperatura inicial* baseou-se em valores reportados na literatura para problemas similares (KIRKPATRICK; GELATT; VECCHI, 1983) e o restante foi definido com base em testes exploratórios preliminares. Durante a otimização, foi imposta a restrição lógica de que o parâmetro de perturbação máxima deve ser estritamente maior que o de perturbação inicial, garantindo a possibilidade de intensificação progressiva da diversificação.

Tabela 1 – Espaço de busca dos hiperparâmetros

Parâmetro	**Intervalo**	**Tipo**
Taxa de resfriamento (α)	[0.85, 0.99]	Contínuo
Fator de temperatura inicial	[0.05, 0.30]	Contínuo
Perturbação inicial	[1, 4]	Inteiro
Perturbação máxima	[3, 8]	Inteiro
Limiar de melhoria	[20, 200]	Inteiro
Limiar de estagnação	[500, 2000]	Inteiro

Para cada configuração de parâmetros avaliada (*trial*), o algoritmo ILS foi executado sobre uma instância de calibração com limite de tempo fixo. A função objetivo minimizada foi o tempo de ciclo da melhor solução encontrada. Formalmente:

$$f(\theta) = C^*(\theta)$$

onde θ representa um vetor de parâmetros e $C^*(\theta)$ denota o tempo de ciclo da solução retornada pelo ILS quando executado com a parametrização θ.

O processo de otimização seguiu o seguinte protocolo:

1. Seleção da *72 Tonge* como instância representativa do conjunto de teste para calibração
2. Execução de 20 trials do algoritmo TPE
3. Limite de tempo de 300 segundos por trial
4. Registro do tempo de ciclo final e do gap em relação ao valor ótimo conhecido
5. Armazenamento da melhor configuração encontrada em arquivo YAML para uso nos experimentos posteriores

Ao término do processo de otimização, a melhor configuração de parâmetros identificada pelo algoritmo TPE é apresentada na Tabela 2.

Tabela 2 – Parâmetros otimizados do algoritmo ILS

Parâmetro	**Valor**	**Descrição**
Tempo máximo (s)	700	Limite de tempo por execução
Taxa de resfriamento (α)	0,9265	Fator multiplicativo da temperatura
Fator de temperatura inicial	0,0543	Proporção do custo inicial usada em T_0
Perturbação inicial	1	Intensidade mínima de diversificação
Perturbação máxima	5	Intensidade máxima de diversificação
Limiar de melhoria	133	Iterações sem ganho para elevar perturbação
Limiar de estagnação	938	Iterações sem aceitação para reinicializar

A configuração da Tabela 2 corresponde diretamente ao arquivo `best_params.yaml` gerado ao final do processo de calibração com Optuna. O melhor vetor encontrado apresenta *cooling_rate* = 0,9265, *initial_temp_factor* = 0,0543, *perturbation_initial* = 1, *perturbation_max* = 5, *improvement_threshold* = 133 e *stagnation_threshold* = 938. Esses valores foram reutilizados integralmente em todas as execuções subsequentes do ILS.

5.2 Instâncias de Teste

5.2.1 Conjunto de Dados

Os experimentos foram conduzidos sobre instâncias do benchmark ALWABP disponibilizadas pelo professor no campus virtual da disciplina, que representam diferentes configurações de linhas de produção originadas de quatro problemas base da literatura: Heskiaoff, Roszieg, Tonge e Wee-Mag. O conjunto final consistiu em 48 instâncias distribuídas entre os quatro problemas base, com 12 instâncias de cada família. A relação completa das instâncias avaliadas é apresentada na Tabela 3.

Tabela 3 – Instâncias do benchmark ALWABP utilizadas nos experimentos

Heskiaoff		**Roszieg**		**Tonge**		**Wee-Mag**	
Nome	**ID**	**Nome**	**ID**	**Nome**	**ID**	**Nome**	**ID**
heskia	1	roszieg	1	tonge	1	wee-mag	1
heskia	2	roszieg	2	tonge	2	wee-mag	2
heskia	11	roszieg	11	tonge	11	wee-mag	11
heskia	12	roszieg	12	tonge	12	wee-mag	12
heskia	41	roszieg	41	tonge	41	wee-mag	41
heskia	42	roszieg	42	tonge	42	wee-mag	42
heskia	51	roszieg	51	tonge	51	wee-mag	51
heskia	52	roszieg	52	tonge	52	wee-mag	52
heskia	61	roszieg	61	tonge	61	wee-mag	61
heskia	62	roszieg	62	tonge	62	wee-mag	62
heskia	71	roszieg	71	tonge	71	wee-mag	71
heskia	72	roszieg	72	tonge	72	wee-mag	72

5.3 Protocolo Experimental

5.3.1 Implementação

Ambos os algoritmos, ILS e modelo exato via Gurobi, foram implementados em Python 3.8. O modelo de programação linear inteira foi resolvido utilizando o solver Gurobi 9.5, enquanto o ILS foi implementado utilizando estruturas de dados nativas do Python para maximizar a portabilidade.

5.3.2 Execução dos Experimentos

Para cada instância do conjunto de teste, foram realizadas as seguintes etapas:

Resolução via Gurobi:

- Limite de tempo de 700 segundos
- Registro do tempo de ciclo da melhor solução encontrada
- Armazenamento da solução em arquivo texto

Resolução via ILS:

- Cinco replicações independentes com sementes distintas (10, 11, 12, 13, 14)
- Limite de tempo base de 700 segundos, ajustável dinamicamente quando a opção de timeout adaptativo está ativada
- Critério de parada antecipada quando a solução atinge tolerância de 0,01% em relação ao valor ótimo conhecido

- Utilização do valor obtido pelo Gurobi como referência para parada antecipada quando disponível
- Armazenamento individual de cada solução

O limite de 700 segundos foi aplicado rigidamente a ambos os métodos. Quando o Gurobi não encontrava uma solução dentro do tempo, o resultado era marcado como "ERRO ou TIMEOUT" e o tempo registrado como 700 segundos; o ILS era executado mesmo assim para fornecer uma solução viável. Cada execução do ILS utilizou as sementes 10, 11, 12, 13 e 14 para permitir análise estatística.

A execução de múltiplas replicações do ILS visa capturar a variabilidade inerente ao comportamento estocástico da metaheurística, permitindo análise estatística dos resultados.

6 Resultados e Discussão

Esta seção apresenta os resultados consolidados das 48 instâncias executadas. Os arquivos `results.csv` e `results_ils_single_results.csv` registram os tempos de ciclo e tempos de execução para cada replicação, enquanto o `experiment.out` resume o andamento experimento a experimento. O limite de 700 segundos foi respeitado em todas as execuções. Todas as métricas apresentadas referem-se às cinco replicações do ILS (sementes 10–14), permitindo análise da variabilidade estocástica.

6.1 Visão Geral

O Gurobi obteve solução em 31 das 48 instâncias; nas demais, atingiu o timeout de 700 segundos sem devolver um tempo de ciclo válido. O ILS retornou soluções viáveis em 100% das instâncias, mesmo quando o Gurobi não conseguiu produzir resposta. A Tabela 4 resume a taxa de sucesso do Gurobi por família de instâncias.

Tabela 4 – Resumo de resolução por família

Família	**Instâncias**	**Gurobi resolveu**	**Gurobi timeout**
Heskiaoff	12	12	0
Roszieg	12	12	0
Tonge	12	5	7
Wee-Mag	12	2	10
Total	48	31	17

6.2 Resultados Detalhados por Instância

As Tabelas 5–8 apresentam os resultados completos para cada instância. A coluna *UB* indica o upper bound conhecido da literatura; *Gurobi CT* e *Gurobi (s)* mostram o tempo de ciclo e o tempo de execução do solver exato ("–" indica timeout); *ILS Melhor*, *ILS Média* e *ILS (s)* referem-se ao melhor ciclo entre as cinco replicações, à média dos ciclos finais e ao tempo médio de execução. A coluna *Melhoria (%)* indica a redução percentual média entre a solução inicial construtiva (C_{inicial}) e a solução final refinada pelo ILS (C_{final}), calculada como:

$$\text{Melhoria}(\%) = \frac{C_{\text{inicial}} - C_{\text{final}}}{C_{\text{inicial}}} \times 100$$

Tabela 5 – Resultados – Família Heskiaoff

Inst.	UB	Gurobi CT	Gurobi (s)	ILS Melhor	ILS Média	ILS (s)	Melhoria (%)
1_hes	94	94	0,11	94	94,00	5,48	53,92
2_hes	95	95	0,16	95	95,00	6,15	23,26
11_hes	169	169	0,15	172	172,00	21,76	66,02
12_hes	107	107	0,11	107	108,20	10,88	44,11
41_hes	35	35	0,75	35	35,00	13,72	42,62
42_hes	40	40	1,39	40	40,00	9,88	69,92
51_hes	51	51	1,43	51	51,00	9,67	69,64
52_hes	50	50	1,10	50	51,40	30,17	56,51
61_hes	66	66	1,01	66	66,00	7,33	60,76
62_hes	56	56	0,89	56	56,00	11,80	66,31
71_hes	91	91	0,40	91	91,00	0,17	32,09
72_hes	65	65	1,18	65	69,60	43,55	52,78

Tabela 6 – Resultados – Família Roszieg

Inst.	UB	Gurobi CT	Gurobi (s)	ILS Melhor	ILS Média	ILS (s)	Melhoria (%)
1_ros	20	20	0,11	20	20,40	2,87	75,42
2_ros	22	22	0,08	22	22,80	7,64	65,45
11_ros	30	30	0,07	26	27,60	3,77	65,50
12_ros	27	27	0,08	25	25,40	3,73	62,20
41_ros	10	10	0,58	10	10,00	1,61	57,63
42_ros	10	10	0,75	10	10,00	1,67	60,94
51_ros	11	11	0,39	11	11,00	1,37	82,65
52_ros	10	10	0,49	10	10,00	2,73	79,76
61_ros	16	16	0,47	16	16,00	2,09	55,80
62_ros	13	13	0,37	13	13,00	0,80	52,21
71_ros	15	15	0,64	15	15,00	1,76	41,86
72_ros	16	16	0,69	16	16,00	0,62	45,95

Tabela 7 – Resultados – Família Tonge

Inst.	UB	Gurobi CT	Gurobi (s)	ILS Melhor	ILS Média	ILS (s)	Melhoria (%)
1_ton	87	87	128,91	94	99,20	194,36	92,95
2_ton	87	87	86,42	89	96,80	181,70	93,39
11_ton	110	110	55,91	112	118,40	111,94	93,54
12_ton	108	108	158,63	117	121,40	113,60	91,92
41_ton	28	–	700,00	33	36,20	263,81	97,70
42_ton	32	–	700,00	38	41,40	250,29	96,31
51_ton	35	–	700,00	41	43,40	191,23	97,56
52_ton	43	43	207,97	46	49,60	198,30	96,86
61_ton	61	–	700,00	75	76,20	258,03	93,40
62_ton	66	–	700,00	77	80,80	261,89	93,09
71_ton	54	–	700,00	63	68,40	252,63	93,93
72_ton	57	–	700,00	71	78,00	249,86	91,93

6.3 Análise de Desempenho por Família

6.3.1 Heskiaoff

As 12 instâncias da família Heskiaoff (28 tarefas, 4 ou 7 trabalhadores) foram resolvidas pelo Gurobi em frações de segundo (máximo 1,43 s em 51_hes). O ILS igualou o valor ótimo em 10 das 12 instâncias; nas duas exceções (11_hes e 72_hes), a melhor

Tabela 8 – Resultados – Família Wee-Mag

Inst.	UB	Gurobi CT	Gurobi (s)	ILS Melhor	ILS Média	ILS (s)	Melhoria (%)
1_wee	25	–	700,00	26	29,00	505,99	70,35
2_wee	26	–	700,00	26	28,00	461,70	72,76
11_wee	29	–	700,00	32	33,00	362,03	83,33
12_wee	30	30	573,12	34	36,00	320,84	79,26
41_wee	10	–	700,00	11	11,40	585,88	62,99
42_wee	9	–	700,00	12	12,60	569,64	68,50
51_wee	12	–	700,00	14	14,60	410,38	86,89
52_wee	9	–	700,00	12	12,40	434,73	89,14
61_wee	15	–	700,00	19	19,20	582,00	64,84
62_wee	18	18	190,84	19	19,80	571,91	64,77
71_wee	18	–	700,00	21	22,40	574,57	66,27
72_wee	16	–	700,00	21	21,00	586,14	58,17

solução ficou $\approx 2\%$ acima do ótimo. A melhoria média sobre a solução construtiva variou de 23% (2_hes) a 70% (42_hes), demonstrando a eficácia da busca local mesmo quando a solução inicial estava distante do ótimo.

6.3.2 Roszieg

A família Roszieg (25 tarefas, 4 ou 6 trabalhadores) apresentou comportamento similar: todas as 12 instâncias foram resolvidas pelo Gurobi em menos de 1 segundo. O ILS reproduziu o ótimo em 10 casos e superou o valor do Gurobi em 2 (11_ros e 12_ros obtiveram ciclos 26 e 25, respectivamente, enquanto o Gurobi reportou 30 e 27). A redução média de ciclo em relação à solução inicial foi de 60%, com tempos abaixo de 8 segundos por replicação.

6.3.3 Tonge

A família Tonge (70 tarefas, 10 ou 17 trabalhadores) mostrou o maior contraste entre os métodos. O Gurobi resolveu apenas 5 das 12 instâncias (1_ton, 2_ton, 11_ton, 12_ton e 52_ton), com tempos de 55–208 segundos; as demais atingiram o timeout de 700 segundos. O ILS produziu soluções viáveis para todas, porém com gaps em relação ao bound de referência que variaram de 2% (2_ton) a 25% (72_ton). As melhorias médias sobre a solução inicial superaram 91%, refletindo a grande distância entre as soluções construtivas e os ótimos locais alcançados. Os tempos médios do ILS ficaram entre 111 e 264 segundos.

6.3.4 Wee-Mag

A família Wee-Mag (75 tarefas, 11 ou 19 trabalhadores) representou o maior desafio. O Gurobi resolveu apenas 2 das 12 instâncias (12_wee em 573 s e 62_wee em 191 s). O ILS entregou soluções viáveis para todas, com gaps de 4–31% em relação aos bounds

conhecidos. A melhoria média sobre a solução inicial ficou entre 58% (72_wee) e 89% (52_wee). Os tempos de execução do ILS variaram de 320 a 586 segundos, permanecendo abaixo do limite de 700 s em todos os casos.

6.4 Comparação entre ILS e Gurobi

Qualidade da solução. Nas 24 instâncias das famílias Heskiaoff e Roszieg, o ILS igualou ou superou o valor do Gurobi em 22 casos; nas 2 exceções (11_hes e 72_hes), a diferença foi de 2–7%. Nas famílias Tonge e Wee-Mag, onde o Gurobi frequentemente expirou, o ILS apresentou gaps de 2–31% em relação aos bounds conhecidos, porém entregou soluções factíveis em 100% das instâncias.

Tempo de execução. O Gurobi resolveu as 24 instâncias menores em menos de 2 segundos; nas 5 instâncias Tonge resolvidas, o tempo variou de 56 a 209 segundos. O ILS consumiu em média 10 s nas famílias pequenas e 200–580 s nas famílias maiores, permanecendo sempre abaixo do limite de 700 s.

Robustez. O ILS entregou solução viável em todas as 48 instâncias, cobrindo os 17 casos em que o Gurobi excedeu o tempo. As melhorias médias sobre a solução construtiva inicial superaram 90% nas famílias Tonge e Wee-Mag, demonstrando a capacidade do algoritmo de escapar de ótimos locais ruins através da perturbação adaptativa e do critério de aceitação tipo *Simulated Annealing*.

Limitações observadas. O ILS não alcançou o ótimo em instâncias Tonge e Wee-Mag com maior número de trabalhadores (IDs 41–72), sugerindo que a combinação de muitas tarefas e muitos trabalhadores dificulta a convergência. Em alguns casos, a variabilidade entre replicações foi elevada (por exemplo, 72_hes variou de 65 a 77), indicando sensibilidade à semente.

7 Conclusão

Este trabalho abordou o problema ALWABP por meio de duas abordagens complementares: um modelo de programação linear inteira resolvido pelo solver Gurobi e uma metaheurística *Iterated Local Search* (ILS) calibrada automaticamente via Optuna. Os experimentos envolveram 48 instâncias de quatro famílias clássicas, cada uma executada 5 vezes com sementes distintas, sob limite de 700 segundos.

Principais achados. O Gurobi resolveu 31 das 48 instâncias (65%), todas das famílias Heskiaoff e Roszieg e 7 de Tonge/Wee-Mag. O ILS entregou soluções viáveis em 100% dos casos, igualando o ótimo em 22 das 24 instâncias menores e apresentando gaps de 2–31% nas famílias mais complexas. A melhoria média sobre a solução construtiva inicial foi de 55% em Heskiaoff, 60% em Roszieg, 94% em Tonge e 73% em Wee-Mag, evidenciando a eficácia da busca local combinada à perturbação adaptativa.

Vantagens do ILS. A implementação mostrou-se robusta e portátil, independente de licenças comerciais. A calibração automática de parâmetros via Optuna reduziu a necessidade de ajustes manuais e permitiu identificar configurações não intuitivas (por exemplo, perturbação inicial igual a 1 e limiar de melhoria em 133 iterações). O critério de aceitação tipo *Simulated Annealing* garantiu escapar de ótimos locais sem perder qualidade nas regiões promissoras.

Limitações. O gap persistente em instâncias Tonge e Wee-Mag com muitos trabalhadores (IDs 41–72) indica que a combinação de 70–75 tarefas com 17–19 trabalhadores impõe desafios de escalabilidade ao ILS. A variabilidade entre replicações (por exemplo, 65 a 77 em 72_hes) sugere sensibilidade à semente e possível benefício de mais replicações ou mecanismos de intensificação.

Referências

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019. (KDD '19), p. 2623–2631. ISBN 9781450362016. Disponível em: <https://doi.org/10.1145/3292500.3330701>. Citado na página 21.

ARENALES, M. et al. *Pesquisa Operacional*. 1. ed. Rio de Janeiro, RJ: Elsevier : ABEPRO, 2007. (COLEÇÃO CAMPUS - ABEPRO Engenharia de Produção). ISBN 9788535251937. Disponível em: <www.campus.com.br>. Citado na página 5.

GLOVER, F.; SÖRENSEN, K. Metaheuristics. *Scholarpedia*, v. 10, n. 4, p. 6532, 2015. Disponível em: <http://dx.doi.org/10.4249/scholarpedia.6532>. Citado na página 5.

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização combinatória e programação linear: modelos e algoritmos*. 2. ed. Rio de Janeiro: Elsevier, 2005. ISBN 978-85-352-1520-5. Citado na página 5.

HANSEN, P.; MLADENOVIć, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467, 2001. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221700001004>. Citado na página 15.

HELGESON, W.; BIRNIE, D. P. Assembly line balancing using the ranked positional weight technique. *Journal of industrial engineering*, v. 12, n. 6, p. 394–398, 1961. Citado na página 14.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. Disponível em: <https://www.science.org/doi/abs/10.1126/science.220.4598.671>. Citado 2 vezes nas páginas 17 e 21.

LAARHOVEN, P. J. M. van; AARTS, E. H. L. Simulated annealing. In: _____. *Simulated Annealing: Theory and Applications*. Dordrecht: Springer Netherlands, 1987. p. 7–15. ISBN 978-94-015-7744-1. Disponível em: <https://doi.org/10.1007/978-94-015-7744-1_2>. Citado na página 18.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. A gentle introduction to iterated local search. *Proceedings of MIC 2001 - 4th Metaheuristics International Conference*, MIC, v. 1, p. 1–6, 2001. Disponível em: <https://www.metaheuristics.org/downloads/mic2001-ils.pdf>. Citado 2 vezes nas páginas 6 e 13.

LUKE, S. *Essentials of Metaheuristics*. second. [S.l.]: Lulu, 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/. Citado na página 5.

POLAT, O. et al. A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-ii: an industrial case study. *International Journal of Production Research*, Taylor & Francis, v. 54, n. 3, p. 722–741, 2016. Disponível em: <https://doi.org/10.1080/00207543.2015.1055344>. Citado na página 16.

YILMAZ, H. Modeling and solving assembly line worker assignment and balancing problem with sequence-dependent setup times. *Soft Computing*, v. 25, p. 12899–12914, 2021. ISSN 1432-7643. Disponível em: <https://doi.org/10.1007/s00500-021-06107-3>. Citado na página 5.