THAÍS GIOVANNA LOPES

LUÍS KENNEDY GERVÁSIO TUROLA

PROGRAMAÇÃO MATEMÁTICA

PROJETO PRÁTICO FINAL:

ASSEMBLY LINE WORKER ASSIGNMENT AND BALANCING PROBLEM (ALWABP)

LAVRAS – MG

2025

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Problema

O Balanceamento de Linhas de Montagem (SALBP) é um dos problemas clássicos da Engenharia de Produção e da Otimização. Basicamente, ele consiste em descobrir a melhor forma de distribuir tarefas em estações de trabalho para que a produção flua de maneira rápida e eficiente. O modelo clássico, contudo, tem uma grande falha: ele assume que todos os trabalhadores são idênticos, o que raramente acontece no mundo real.

O *Assembly Line Worker Assignment and Balancing Problem* (ALWABP), proposto por Miralles et al. (2007), trás um avanço, pois inclui as diferênças de capacidades entre os trabalhadores. No ALWABP, cada funcionário tem um tempo diferente (ou até mesmo é incapaz) para realizar certas tarefas. O problema ganha outra dimensão: é preciso designar o trabalhador certo para a estação certa e, ao mesmo tempo, distribuir as tarefas de forma otimizada. Reconhecido como um problema NP-difícil, o problema exige métodos computacionais avançados para ser resolvido de forma eficiente em escala industrial.

1.2 Objetivo do Relatório

Este trabalho tem como objetivo demonstrar as soluções desenvolvidas pela equipe sobre duas lentes diferentes. Primeiro, apresenta-se uma abordagem exata a partir de uma formulação matemática do ALWABP como um modelo de Programação Linear Inteira que foi resolvido utilizando o *solver* comercial *Gurobipy*. Em seguida, apresenta-se uma meta-heurística adaptativa, *Adaptative Neighborhood Local Search* (ANLS), definida por sorteio em aula para resolver as mesmas instâncias e analisar seu comportamento. Por fim, os resultados obtidos são comparados em termos de qualidade das soluções e tempo computacional, permitindo avaliar as vantagens e limitações de cada abordagem.

O restante deste relatório está organizado da seguinte forma: na Seção 2 apresenta-se a formulação matemática do ALWABP; na Seção 3 descreve-se a meta-heurística adotada e suas escolhas metodológicas; na Seção 4 discutem-se os resultados computacionais obtidos; e na Seção 5 apresentam-se as conclusões e considerações finais.

2 FORMULAÇÃO MATEMÁTICA

Agora apresenta-se a formulação de programação Inteira utilizada no solveer *Assembly Line Worker Assignment and Balancing Problem* (ALWABP). Seja um conjunto de tarefas $I = \{1, \ldots, n\}$, um conjunto de trabalhadores $W = \{1, \ldots, k\}$, e um conjunto de estações $S = \{1, \ldots, m\}$, com $m = k$. Cada trabalhador $w \in W$ possui um tempo de processamento t_{wi} para cada tarefa $i \in I$, podendo ocorrer incapacidade ($t_{wi} = \infty$). As precedências são dadas por um conjunto de pares (i, j) indicando que a tarefa i deve ser concluída antes da tarefa j.

2.1 Variáveis de decisão

$$x_{is} = \begin{cases} 1, & \text{se a tarefa } i \text{ é atribuída à estação } s; \\ 0, & \text{caso contrário;} \end{cases}$$

$$y_{ws} = \begin{cases} 1, & \text{se o trabalhador } w \text{ é designado à estação } s; \\ 0, & \text{caso contrário;} \end{cases}$$

$$z_{wsi} = \begin{cases} 1, & \text{se o trabalhador } w \text{ executa a tarefa } i \text{ na estação } s; \\ 0, & \text{caso contrário;} \end{cases}$$

$T_s \geq 0$ tempo total de processamento da estação s;

$C \geq 0$ tempo de ciclo da linha (variável objetivo).

2.2 Função Objetivo

Minimizar o tempo de ciclo da linha:

$$\min C \tag{2.1}$$

2.3 Restrições

2.3.1 Atribuição única de tarefas

$$\sum_{s \in S} x_{is} = 1, \quad \forall i \in I. \tag{2.2}$$

2.3.2 Cada trabalhador ocupa exatamente uma estação

$$\sum_{s \in S} y_{ws} = 1, \quad \forall w \in W. \tag{2.3}$$

2.3.3 Cada estação recebe exatamente um trabalhador

$$\sum_{w \in W} y_{ws} = 1, \quad \forall s \in S. \tag{2.4}$$

2.3.4 Coerência entre variáveis x, y e z

Uma tarefa só pode ser executada por um trabalhador se este estiver na estação em que a tarefa foi atribuída:

$$z_{wsi} \leq x_{is}, \quad \forall w, s, i, \tag{2.5}$$

$$z_{wsi} \leq y_{ws}, \quad \forall w, s, i. \tag{2.6}$$

2.3.5 Cada tarefa é executada por exatamente um trabalhador em sua estação

$$\sum_{w \in W} z_{wsi} = x_{is}, \quad \forall s \in S, \ \forall i \in I. \tag{2.7}$$

2.3.6 Incapacidade de trabalhadores

Se $t_{wi} = \infty$, o trabalhador w não pode executar a tarefa i:

$$z_{wsi} = 0, \quad \forall w \in W,\ \forall s \in S,\ \forall i \in I \text{ incapaz para } w. \tag{2.8}$$

2.3.7 Cálculo do tempo total da estação

$$T_s = \sum_{w \in W} \sum_{i \in I} t_{wi}\, z_{wsi}, \quad \forall s \in S. \tag{2.9}$$

2.3.8 Definição do tempo de ciclo

$$C \geq T_s, \quad \forall s \in S. \tag{2.10}$$

2.3.9 Restrições de precedência

Se i precede j, então a estação de i não pode ser posterior à de j:

$$\sum_{s \in S} s\, x_{is} \ \leq \ \sum_{s \in S} s\, x_{js}. \tag{2.11}$$

3 METODOLOGIA HEURÍSTICA: ADAPTIVE LARGE NEIGHBORHOOD SEARCH (ALNS)

A meta-heurística utilizada no projeto é baseada no *Adaptive Large Neighborhood Search* (ALNS), implementado integralmente em Python e estruturado para resolver o problema ALWABP considerando precedência, tempos dependentes do trabalhador e restrições de incapacidade. A seguir descrevemos a estrutura exata da implementação, alinhada ao código fornecido.

3.1 Estrutura Geral do ALNS

O algoritmo inicializa um objeto `ALNS` contendo:

- operadores de remoção: `random_remove`, `worst_remove`, `shaw_remove`;
- operadores de inserção: `greedy_insert` e `regret_insert`;
- operadores de trabalhador: `swap_workers` e `reassign_worst_station_worker`;
- pesos, pontuações e contadores individuais para cada operador;
- temperatura inicial estimada automaticamente;
- parâmetros de controle: `segment_length`, `cooling`, `max_iter`, `time_limit`.

A cada iteração, os operadores são selecionados por *roleta* proporcional aos seus pesos adaptativos, seguindo a lógica clássica do ALNS.

3.2 Construção da Solução Inicial

A solução inicial é construída por uma heurística gulosa implementada em `initial_solution_greedy`. Os passos reais do código são:

1. Os trabalhadores são embaralhados aleatoriamente e atribuídos a cada estação:

$$\text{worker\_by\_station}[s] = \text{trabalhadores embaralhados}.$$

2. Obtém-se uma ordenação topológica das tarefas via `networkx`:

$$\text{ordem} = \texttt{topological\_sort}(G).$$

3. Para cada tarefa i, o algoritmo avalia todas as estações s e identifica aquelas onde:
   - não há violação de precedências;
   - o trabalhador da estação não é incapaz ($t_{wi} \neq \infty$);
   - a inserção é viável segundo `feasible_insertion_station`.
4. Entre as estações viáveis, a heurística escolhe aquela que minimiza:

$$\widehat{C} = \max\{T_s + t_{wi},\ \max_{s'} T_{s'}\}.$$

5. Se nenhuma estação for viável, aplica-se o *fallback* exatamente como no código, tentando inserir a tarefa nas últimas estações permitidas.

Se a construção falhar completamente, o algoritmo retorna `None` e o ALNS tenta uma nova construção com semente distinta.

3.3 Estimativa Automática da Temperatura Inicial

Ao contrário de muitos ALNS tradicionais, a implementação do projeto calcula a temperatura inicial de forma automatizada. O procedimento é:

1. gera-se um conjunto de soluções perturbadas por remoção e inserção;

2. mede-se o desvio padrão dos custos C;

3. define-se:

$$T_0 = \begin{cases} \max(\sigma, 1.0), & \sigma > 0, \\ 10.0, & \text{se todas as soluções têm o mesmo custo.} \end{cases}$$

Assim, a temperatura inicial é sensível à instância e ao comportamento real dos operadores.

3.4 Ciclo Principal do ALNS

A cada iteração k, o algoritmo executa:

1. Seleção dos operadores por roleta:

$$r \sim \text{Remoção}, \quad i \sim \text{Inserção}, \quad w \sim \text{Operadores de trabalhador.}$$

2. Clonagem da solução atual:

$$S' \leftarrow S.$$

3. Com probabilidade 0,3, aplica-se um operador de trabalhador:

$$S' \leftarrow w(S').$$

4. Remove-se um conjunto de tarefas Q, cujo tamanho é sorteado entre $q_{\min}$ e $q_{\max}$:

$$R = r(S', q).$$

5. Insere-se novamente as tarefas removidas usando o operador de inserção selecionado. Se este falhar, tenta-se `greedy_insert` como *fallback*.

6. Avalia-se o custo da solução candidata:

$$C(S') = \max_{s} T_s.$$

7. A solução é aceita segundo o critério de *Simulated Annealing*:

$$\text{aceita}(S') = \begin{cases} 1, & C(S') < C(S), \\ \exp\left(\dfrac{C(S) - C(S')}{T}\right), & \text{caso contrário.} \end{cases}$$

8. Atualizam-se pesos e pontuações dos operadores de acordo com:

   - melhoria global: σ_1;
   - melhoria da solução corrente: σ_2;
   - aceitação sem melhoria: σ_3.

9. A cada `segment_length` iterações:

$$w \leftarrow (1-\rho)\, w + \rho\, \overline{\pi},$$

   onde $\overline{\pi}$ é a pontuação média do operador.

10. Registra-se a assinatura da solução para evitar repetição:

$$\text{signature}(S) = ((w_s,\ \text{tarefas ordenadas}),\ \forall s)\,.$$

3.5 Critério de Parada

O ALNS é encerrado quando:

- atinge `max_iter`;
- ou ultrapassa `time_limit`.

O algoritmo retorna:

$$(S_0,\ S^{\cdot\ t)},$$

onde S é a melhor solução encontrada e t é o tempo necessário para alcançá-la.

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos pela meta-heurística desenvolvida e pelo solver. Para cada instância, analisaram-se: (i) o valor da solução inicial (SI), (ii) o valor final obtido pela heurística (SF), (iii) o desvio relativo entre SI e SF, (iv) o desvio relativo entre SF e o ótimo fornecido pelo solver, e (v) os tempos computacionais de ambas as abordagens. A Tabela 4.1 consolida todos os valores obtidos.

4.1 Qualidade das soluções da heurística

Os resultados mostram que a meta-heurística foi capaz de produzir reduções substanciais em relação à solução inicial em praticamente todas as instâncias. Os desvios entre SI e SF frequentemente ultrapassaram 80%, chegando a valores superiores a 95% nas instâncias da família *ton* e alguns da família *wee*. Isso evidencia que o método de construção inicial gera soluções bastante distantes do ótimo, porém, a implementação da meta-heurística apresentou um refinamento satisfatório.

O desvio entre o valor final da heurística e o valor ótimo variou de acordo com a classe da instância. Em instâncias de menor complexidade estrutural (por exemplo, *hes* e *ros*), a heurística obteve soluções próximas ou iguais ao ótimo. Já em instâncias mais exigentes, especialmente das classes *ton*, *wee* e algumas *ton* mais profundas (como 41_ton, 52_wee, 61_ton, 72_ton), tiveram desvios consideráveis, em alguns casos superiores a 100%. Esse comportamento é esperado, uma vez que tais instâncias apresentam muitas tarefas, forte dispersão de tempos entre trabalhadores e precedências complexas.

4.1.1 Comparação com os valores ótimos do solver

O solver obteve o valor ótimo rapidamente nas instâncias pequenas e médias, porém atingiu o tempo limite em diversas instâncias grandes, particularmente

nas famílias *wee*, *ton* e algumas *hes*. Nessas situações, mesmo após longos períodos de execução (1800 segundos), o solver ainda não havia certificado a otimalidade da solução.

Isso indica que, apesar da formulação matemática ser adequada, sua escalabilidade é limitada para instâncias de alta complexidade, reforçando a relevância da heurística como alternativa prática.

4.1.2 Análise de tempo computacional

Os tempos de execução da heurística se mantiveram baixos em todas as instâncias, variando tipicamente entre 0.04 e 80 segundos. Mesmo nas instâncias mais complexas, o tempo da meta-heurística permaneceu inferior a 90 segundos.

Em contraste, o solver apresentou tempos muito superiores:

- para instâncias pequenas, foi eficiente, concluindo em menos de 1 segundo;
- para instâncias médias, o tempo cresceu para dezenas ou centenas de segundos;
- para instâncias grandes, atingiu o limite de 1800 segundos.

Dessa forma, a meta-heurística demonstra clara vantagem em escalabilidade, sendo capaz de produzir soluções de qualidade aceitável em tempo drasticamente menor.

4.2 Síntese dos resultados

De forma geral, observou-se:

- A meta-heurística reduz drasticamente a solução inicial, mostrando boa capacidade de refinamento.

- A qualidade final depende fortemente da classe da instância: ótima para instâncias simples, moderada para médias, e limitada para instâncias largas e profundas.

- O solver fornece garantias de otimalidade, mas não escala bem para instâncias grandes.

- Em cenários industriais reais, onde soluções rápidas são preferíveis à otimalidade exata, a heurística se mostra mais adequada.

Tabela 4.1 – Resultados obtidos pela heurística e pelo solver.

Instância	SI	SF	Desv. SI–SF (%)	Desv. SF–Opt (%)	T_{heur}	T_{solver}
11_hes	624	172	72.4359	1.7751	0.14	0.13
11_ros	91	33	63.7363	10	0.60	0.05
11_ton	2289	180	92.1363	63.6364	9.12	52.04
11_wee	905	36	96.0221	24.1379	11.03	1800.13
12_hes	438	107	75.5708	0	0.04	0.12
12_ros	104	27	74.0385	0	0.27	0.07
12_ton	2057	157	92.3675	45.3704	30.51	163.92
12_wee	547	39	92.8702	30	36.83	1264.43
1_hes	262	94	64.1221	0	1.83	0.09
1_ros	95	20	78.9474	0	0.04	0.07
1_ton	2264	108	95.2297	24.1379	31.62	85.93
1_wee	925	31	96.6486	24	10.33	1800.13
2_hes	278	95	65.8273	0	0.39	0.17
2_ros	118	22	81.3559	0	0.44	0.08
2_ton	2269	108	95.2402	24.1379	16.40	86.15
2_wee	286	32	88.8112	23.0769	30.84	1800.13
41_hes	131	35	73.2824	0	4.24	0.56
41_ros	83	10	87.9518	0	2.56	0.51
41_ton	2109	87	95.8748	210.7143	63.74	1800.29
41_wee	813	15	98.1550	50	46.46	1800.36
42_hes	153	41	73.2026	2.5	0.40	0.77
42_ros	39	11	71.7949	10	0.09	0.57
42_ton	2077	73	96.4853	128.1250	31.85	1800.27
42_wee	112	13	88.3929	44.4444	81.48	1800.38
51_hes	235	54	77.0213	5.8824	0.17	1.55
51_ros	109	11	89.9083	0	0.08	0.28
51_ton	2008	63	96.8625	80	58.26	1800.26
51_wee	877	16	98.1756	33.3333	67.13	1800.35
52_hes	147	50	65.9864	-90.0596	0.67	1.46
52_ros	83	11	86.7470	10	0.12	0.50
52_ton	2258	77	96.5899	79.0698	15.72	150.80
52_wee	880	17	98.0682	88.8889	35.24	1800.37
61_hes	410	66	83.9024	0	1.07	1.06
61_ros	95	16	83.1579	0	0.76	0.50
61_ton	2170	167	92.3041	173.7705	16.87	1800.26
61_wee	937	22	97.6521	46.6667	56.84	1800.34
62_hes	366	56	84.6995	0	1.62	0.81
62_ros	118	13	88.9831	0	0.17	0.40
62_ton	2590	168	93.5135	154.5455	42.97	1800.25
62_wee	1262	27	97.8605	50	36.34	173.77
71_hes	124	91	26.6129	0	0.22	0.49
71_ros	118	15	87.2881	0	0.92	0.52
71_ton	2398	236	90.1585	337.0370	58.32	1800.26
71_wee	961	23	97.6067	27.7778	58.72	1800.34
72_hes	363	86	76.3085	32.3077	0.30	1.12
72_ros	121	16	86.7769	0	0.99	0.41
72_ton	2875	128	95.5478	124.5614	37.54	1800.26
72_wee	209	24	88.5167	50	59.65	1800.34

5 CONCLUSÃO

Os resultados obtidos demonstram que a heurística desenvolvida é capaz de gerar soluções de boa qualidade em tempo computacional reduzido, especialmente quando comparada ao solver exato, cujo tempo de execução frequentemente atinge o limite máximo estabelecido. Em diversas instâncias, a heurística apresentou desvios moderados em relação ao valor ótimo conhecido, mas com ganhos significativos de eficiência, o que reforça sua aplicabilidade em cenários em que soluções rápidas são prioritárias.

Além disso, observou-se um comportamento consistente entre as famílias de instâncias: problemas maiores ou mais densos resultaram em desvios mais elevados, porém com tempos de processamento ainda aceitáveis. Esses resultados indicam que a abordagem proposta é adequada como método inicial ou como componente dentro de um processo híbrido de otimização.

Por fim, o desempenho observado sugere oportunidades de melhorias futuras, como ajustes nos mecanismos de busca local, integração de critérios adaptativos e emprego de estratégias multi-start.

REFERÊNCIAS

Google DeepMind. **Google Gemini**. <https://gemini.google.com/>. [Ferramenta online; acesso em 6 de dezembro de 2025].

MIRALLES, C. et al. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. **Discrete Applied Mathematics**, v. 156, n. 3, p. 352–367, fev. 2008. Disponível em: <https://doi.org/10.1016/j.dam.2005.12.012>.

OpenAI. **ChatGPT (Modelo GPT-5.1)**. <https://chat.openai.com/>. [Ferramenta online; acesso em 6 de dezembro de 2025].

ROPKE, S.; PISINGER, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. **Transportation Science**, nov. 2006. Disponível em: <https://www.researchgate.net/publication/220413334_An_Adaptive_Large_Neighborhood_Search_Heuristic_for_the_Pickup_and_Delivery_Problem_with_Time_Windows>.