# Trabalho Prático Final – GCC118/PCC540

## Resolução do Problema ALWABP via Modelo MILP e Metaheurística VNS

Caio Bueno Finocchio Martins  Tobias Maugus Bueno Cougo

Universidade Federal de Lavras (UFLA)
Departamento de Ciência da Computação

## 1 Introdução

O problema de *Assembly Line Worker Assignment and Balancing Problem* (ALWABP) consiste em designar tarefas a estações de trabalho e, simultaneamente, alocar trabalhadores a essas estações, levando em conta tempos de processamento heterogêneos e restrições de incapacidade. O objetivo é minimizar o tempo de ciclo da linha de montagem ($C_{\max}$), respeitando precedências tecnológicas e limitações de cada trabalhador.

O ALWABP generaliza o problema clássico de balanceamento de linhas, pois considera que trabalhadores distintos podem executar a mesma tarefa em tempos diferentes e, em alguns casos, podem ser completamente incapazes de realizá-la. Trata-se, portanto, de um problema de otimização combinatória de alta complexidade, para o qual modelos exatos podem se tornar inviáveis em instâncias maiores. Neste trabalho, analisam-se duas abordagens: (i) um modelo exato MILP resolvido pelo Gurobi; e (ii) uma metaheurística *Variable Neighborhood Search* (VNS).

O objetivo central é comparar o desempenho da VNS com o modelo exato, avaliando sua eficácia e seus limites enquanto heurística para o ALWABP.

## 2 Formulação do problema como MILP

Sejam: $N = \{1, \ldots, n\}$ o conjunto de tarefas, $W = \{1, \ldots, k\}$ o conjunto de trabalhadores e $S = \{1, \ldots, m\}$ o conjunto de estações, assumindo-se $m = k$, isto é, uma estação por trabalhador. O conjunto de precedências é denotado por $P \subseteq N \times N$, onde $(i, j) \in P$ indica que a tarefa $i$ deve ser concluída antes de $j$. O tempo de processamento da tarefa $i$ pelo trabalhador $w$ é dado por $t_{wi}$, podendo assumir um valor muito grande (ou infinito) para modelar incapacidade. Seja, ainda, $I_w \subseteq N$ o subconjunto de tarefas que o trabalhador $w$ não consegue executar.

As variáveis de decisão principais são:

- $C_{\max} \in \mathbb{R}_+$: tempo de ciclo da linha de produção (a ser minimizado);
- $y_{si} \in \{0, 1\}$: vale 1 se a tarefa $i$ é atribuída à estação $s$;
- $z_{ws} \in \{0, 1\}$: vale 1 se o trabalhador $w$ é alocado à estação $s$;

- $u_{wis} \in \{0,1\}$: variável auxiliar para linearizar $y_{si} \cdot z_{ws}$, indicando que a tarefa $i$ é executada na estação $s$ pelo trabalhador $w$.

A função objetivo busca minimizar o tempo de ciclo:

$$\min C_{\max}. \tag{1}$$

As principais restrições são:

**Atribuição de tarefas e trabalhadores** Cada tarefa é atribuída a exatamente uma estação:

$$\sum_{s \in S} y_{si} = 1, \quad \forall i \in N. \tag{2}$$

Cada trabalhador é alocado a exatamente uma estação:

$$\sum_{s \in S} z_{ws} = 1, \quad \forall w \in W. \tag{3}$$

Cada estação possui exatamente um trabalhador:

$$\sum_{w \in W} z_{ws} = 1, \quad \forall s \in S. \tag{4}$$

**Linearização e incapacidade** A variável auxiliar $u_{wis}$ lineariza o produto $y_{si} z_{ws}$:

$$u_{wis} \leq y_{si}, \qquad \forall i \in N,\ w \in W,\ s \in S, \tag{5}$$
$$u_{wis} \leq z_{ws}, \qquad \forall i \in N,\ w \in W,\ s \in S, \tag{6}$$
$$u_{wis} \geq y_{si} + z_{ws} - 1, \qquad \forall i \in N,\ w \in W,\ s \in S. \tag{7}$$

Para tarefas em que o trabalhador é incapaz ($i \in I_w$), impõe-se:

$$u_{wis} = 0, \quad \forall i \in I_w,\ w \in W,\ s \in S. \tag{8}$$

**Restrição de tempo de ciclo** Para cada estação, o tempo total de processamento das tarefas atribuídas não pode exceder $C_{\max}$:

$$\sum_{i \in N} \sum_{w \in W} t_{wi}\, u_{wis} \leq C_{\max}, \quad \forall s \in S. \tag{9}$$

**Precedências** Se $(i,j) \in P$, a estação que executa $i$ deve ser anterior ou igual à que executa $j$:

$$\sum_{s \in S} s\, y_{si} \ \leq \ \sum_{s \in S} s\, y_{sj}, \quad \forall (i,j) \in P. \tag{10}$$

O solver Gurobi é utilizado para resolver este modelo exato, com limite de tempo de 20 minutos por instância e coleta dos valores de solução, tempo computacional e *gap* final.

# 3 Algoritmo proposto: Variable Neighborhood Search (VNS)

Além do modelo MILP, foi implementada uma metaheurística *Variable Neighborhood Search* (VNS) para resolver o ALWABP. A VNS é baseada na exploração sistemática de diferentes estruturas de vizinhança, alternando fases de perturbação (*shaking*) e busca local.

## 3.1 Representação da solução

Uma solução $S$ é representada por:

- um vetor de estações das tarefas, $Y$, em que $Y[i]$ indica a estação à qual a tarefa $i$ está atribuída;
- um vetor de trabalhadores por estação, $Z$, em que $Z[s]$ indica qual trabalhador está alocado à estação $s$.

Dessa forma, cada par (estação, trabalhador) define a capacidade de execução das tarefas naquela posição da linha.

## 3.2 Função de avaliação

Dada uma solução $(Y, Z)$, o tempo de processamento de cada estação é calculado somando-se, para todas as tarefas atribuídas àquela estação, os tempos $t_{w,i}$ do trabalhador alocado $w = Z[s]$. O valor de $C_{\max}(S)$ é definido como o maior tempo entre todas as estações.

A solução é considerada factível se:

- todas as restrições de precedência são respeitadas, isto é, nenhuma tarefa é atribuída a uma estação posterior à estação de algum de seus predecessores;
- nenhuma tarefa é atribuída a uma estação cujo trabalhador alocado seja incapaz de executá-la.

Caso alguma dessas condições não seja satisfeita, a solução recebe uma penalização por infactibilidade (um valor de $C_{\max}$ muito elevado), garantindo que soluções factíveis sejam sempre preferidas no processo de busca.

## 3.3 Geração da solução inicial

A solução inicial é construída de forma gulosa, em duas etapas:

1. **Alocação de trabalhadores às estações.** Gera-se uma permutação aleatória dos trabalhadores e cada trabalhador é atribuído a exatamente uma estação. Assim, a numeração das estações é determinada pela ordem dessa permutação.
2. **Atribuição gulosa de tarefas.** Constrói-se uma ordenação topológica das tarefas a partir do grafo de precedências. Em seguida, percorrem-se as tarefas nessa ordem e, para cada tarefa $i$, busca-se a primeira estação $s$ (na ordem das estações) tal que: (i) o trabalhador $Z[s]$ seja capaz de executar a tarefa $i$, e (ii) todas as tarefas predecessoras de $i$ tenham sido atribuídas a estações com índice menor ou igual a $s$. A tarefa é então fixada nessa estação.

Esse procedimento produz uma solução inicial factível (quando possível), que tende a respeitar precedências e capacidades de maneira simples e serve de ponto de partida para a VNS.

## 3.4 Estruturas de vizinhança

Na fase de *shaking*, são utilizadas $K_{\max} = 3$ vizinhanças:

| $k$ | Vizinhança | Descrição |
|---|---|---|
| 1 | *Task Swap* | troca as estações de duas tarefas distintas; |
| 2 | *Task Reassignment* | move uma tarefa de sua estação atual para outra estação; |
| 3 | *Worker Swap* | troca os trabalhadores alocados a duas estações. |

A vizinhança é escolhida de acordo com o valor corrente de $k$, e a solução é aleatoriamente perturbada dentro dessa estrutura antes de aplicar a busca local.

## 3.5 Busca local (VND)

Após o *shaking*, aplica-se um *Variable Neighborhood Descent* (VND) com duas vizinhanças, exploradas em esquema de *first improvement*:

| $\ell$ | Vizinhança de busca local | Descrição |
|---|---|---|
| 1 | *Task Reassignment* | tenta mover tarefas individualmente entre estações; |
| 2 | *Worker Swap* | tenta trocar trabalhadores entre estações. |

Se um movimento que melhora $C_{\max}$ é encontrado na vizinhança atual, a solução é atualizada e o VND retorna à vizinhança $\ell = 1$. Caso contrário, passa-se para a próxima vizinhança. O VND termina quando nenhuma das vizinhanças produz melhoria.

## 3.6 Critérios de parada

O algoritmo VNS interno utiliza um limite fixo de iterações $VNS\_MAX\_ITER = 500$: o laço principal é encerrado assim que esse número de iterações é atingido.

Além disso, no contexto experimental do trabalho, foi imposto um **limite de tempo por instância**:

- Para cada instância, são realizadas sucessivas replicações da VNS, cada uma com uma *seed* diferente, até um máximo de $NUM\_REPLICATIONS = 13$ replicações.
- O tempo é cronometrado a partir do início da primeira replicação.
- Se o tempo acumulado das replicações para uma dada instância alcançar $TIME\_LIMIT = 600$ segundos (10 minutos), novas replicações não são iniciadas para essa instância.

Dessa forma, o experimento combina um critério de parada por número de iterações dentro de cada execução da VNS e um critério adicional de tempo de CPU por instância quando se considera o conjunto de replicações.

# 4 Resultados computacionais e análise

## 4.1 Configuração experimental

Os experimentos foram conduzidos em um notebook com as seguintes características de hardware:

- Processador: Intel® Core™ Ultra 7 155H (3,80 GHz);
- Memória RAM instalada: 16 GB;
- Sistema operacional: 64 bits, arquitetura x64.

O modelo exato foi resolvido com o solver Gurobi, com limite de 20 minutos por instância. A metaheurística VNS foi implementada em Python e executada com os parâmetros resumidos a seguir:

- $VNS\_MAX\_ITER = 1000$;
- $VNS\_K\_MAX = 3$;
- Conjunto de sementes para instâncias pequenas: {0, 1, 7, 13, 17, 23, 31, 42, 47, 73, 101, 202, 606};
- Conjunto de sementes para instâncias grandes: {97, 131, 197, 223, 281, 313, 487, 557, 613, 733, 1707, 1905, 1936};
- Limite de tempo por instância para a heurística: $TIME\_LIMIT = 500$ segundos.

As instâncias do VNS são executadas em paralelo (*ProcessPoolExecutor*), mas, para cada instância, as replicações são executadas em série dentro de um único processo, respeitando o limite de tempo. O tempo de cada instância é registrado como o intervalo entre o início da primeira replicação e o término da última replicação de fato executada (ou o momento em que o limite de tempo foi atingido).

## 4.2 Métricas comparadas

Para cada instância, são reportados:

- $SI$: valor da solução inicial produzida pela heurística;
- $SF$: melhor valor de solução final obtido pela VNS, considerando todas as replicações executadas;
- $SO$: valor ótimo ou melhor *upper bound* conhecido, extraído de um arquivo de referência;
- *Improvement%* (coluna `Improvement%` na Tabela 1): desvio percentual da solução final em relação à inicial, calculado como

$$100 \times \frac{SI - SF}{SI};$$

- *Gap_ to_ Optimal%* (coluna `GTO%` na Tabela 1): desvio percentual da solução final em relação ao valor ótimo, calculado como

$$100 \times \frac{SF - SO}{SO};$$

- *Total_ Time_ s* (coluna `VnsTime`): tempo total consumido pela heurística para aquela instância (soma das replicações executadas até o limite de tempo);
- *SOL_ GUROBI* (coluna `SG`): valor objetivo retornado pelo Gurobi;
- *TIME_ GUROBI* (coluna `GurobiTime`): tempo de execução do Gurobi;
- *GAP_ GUROBI_ OPT* (coluna `GGO%`): *gap* percentual final reportado pelo solver (MIPGap), em relação ao limite inferior interno.

Nas estatísticas agregadas (tempo médio, mínimo, máximo, total), o tempo total corresponde à soma dos tempos individuais das instâncias. No caso da VNS, como as instâncias são executadas em paralelo, esses tempos representam o esforço computacional somado, e não necessariamente o tempo de parede observado na execução completa.

## 4.3 Discussão qualitativa dos resultados

A Tabela 1 apresenta os resultados completos das 48 instâncias, já contendo os valores revisados de solução inicial ($SI$), solução final da VNS ($SF$), solução ótima conhecida ($SO$), solução retornada pelo Gurobi ($SG$), tempos de execução e *gaps*.

A seguir apresentamos o resumo estatístico global atualizado:

- Total de instâncias avaliadas: **48**
- Instâncias com solução Gurobi: **48**
- Instâncias onde o Gurobi atingiu o ótimo: **28**
- Instâncias onde a VNS atingiu o ótimo: **14**

**Estatísticas de tempo do Gurobi:**

- Tempo médio: **591.34 s**
- Tempo mínimo: **0.09 s**
- Tempo máximo: **1200.67 s**
- Tempo total: **28384.12 s**

**Estatísticas da heurística (VNS):**

- Tempo médio: **256.88 s**
- Tempo mínimo: **28.97 s**
- Tempo máximo: **538.19 s**
- Tempo total: **12330.34 s**

Tabela 1: Resultados completos por instância.

| Instance | Seed | SI | SF | SO | SG | GurobiTime | GGO% | VnsTime | Improvement% | GTO% |
|---|---|---|---|---|---|---|---|---|---|---|
| 11_hes | 202 | 423.0 | 173.0 | 169.0 | 169.0 | 0.28 | 0.0 | 71.00 | 59.10 | 2.37 |
| 11_ros | 1 | 50.0 | 31.0 | 30.0 | 30.0 | 0.09 | 0.0 | 32.04 | 38.00 | 3.33 |
| 11_ton | 281 | 363.0 | 178.0 | 110.0 | 110.0 | 958.7 | 0.0 | 241.07 | 50.96 | 61.82 |
| 11_wee | 197 | 233.0 | 51.0 | 29.0 | 35.0 | 1200.34 | 82.8571 | 441.26 | 78.11 | 75.86 |
| 12_hes | 101 | 497.0 | 107.0 | 107.0 | 107.0 | 0.47 | 0.0 | 61.03 | 78.47 | 0.00 |
| 12_ros | 1 | 64.0 | 27.0 | 27.0 | 27.0 | 0.25 | 0.0 | 28.97 | 57.81 | 0.00 |
| 12_ton | 733 | 1252.0 | 149.0 | 108.0 | 108.0 | 1200.34 | 50.0 | 290.55 | 88.10 | 37.96 |
| 12_wee | 223 | 243.0 | 52.0 | 30.0 | 33.0 | 1200.21 | 45.4545 | 398.90 | 78.60 | 73.33 |
| 1_hes | 606 | 328.0 | 94.0 | 94.0 | 94.0 | 0.55 | 0.0 | 121.16 | 71.34 | 0.00 |
| 1_ros | 31 | 25.0 | 20.0 | 20.0 | 20.0 | 0.17 | 0.0 | 32.14 | 20.00 | 0.00 |
| 1_ton | 97 | 486.0 | 165.0 | 87.0 | 87.0 | 964.16 | 0.0 | 326.92 | 66.05 | 89.66 |
| 1_wee | 223 | 274.0 | 54.0 | 25.0 | 29.0 | 1200.23 | 44.8276 | 497.83 | 80.29 | 116.00 |
| 2_hes | 13 | 457.0 | 95.0 | 95.0 | 95.0 | 1.76 | 0.0 | 142.09 | 79.21 | 0.00 |
| 2_ros | 0 | 121.0 | 22.0 | 22.0 | 22.0 | 0.85 | 0.0 | 35.63 | 81.82 | 0.00 |
| 2_ton | 1707 | 401.0 | 119.0 | 87.0 | 91.0 | 1200.35 | 36.2637 | 322.85 | 70.32 | 36.78 |
| 2_wee | 97 | 311.0 | 51.0 | 26.0 | 28.0 | 1200.56 | 35.7143 | 508.52 | 83.60 | 96.15 |
| 41_hes | 606 | 235.0 | 35.0 | 35.0 | 35.0 | 2.25 | 0.0 | 150.31 | 85.11 | 0.00 |
| 41_ros | 13 | 30.0 | 10.0 | 10.0 | 10.0 | 1.49 | 0.0 | 43.14 | 66.67 | 0.00 |
| 41_ton | 313 | 403.0 | 76.0 | 28.0 | 31.0 | 1200.34 | 70.9677 | 483.22 | 81.14 | 171.43 |
| 41_wee | 131 | 349.0 | 42.0 | 10.0 | 10.0 | 1200.39 | 20.0 | 538.19 | 87.97 | 320.00 |
| 42_hes | 42 | 302.0 | 40.0 | 40.0 | 40.0 | 4.96 | 0.0 | 142.06 | 86.75 | 0.00 |
| 42_ros | 13 | 57.0 | 11.0 | 10.0 | 10.0 | 2.28 | 0.0 | 50.60 | 80.70 | 10.00 |
| 42_ton | 1936 | 445.0 | 89.0 | 32.0 | 35.0 | 1200.60 | 74.2857 | 435.75 | 80.00 | 178.12 |
| 42_wee | 487 | 469.0 | 37.0 | 9.0 | 13.0 | 1200.67 | 92.3077 | 532.97 | 92.11 | 311.11 |
| 51_hes | 0 | 290.0 | 57.0 | 51.0 | 51.0 | 6.12 | 0.0 | 131.15 | 80.34 | 11.76 |
| 51_ros | 23 | 61.0 | 12.0 | 11.0 | 11.0 | 1.59 | 0.0 | 41.62 | 80.33 | 9.09 |
| 51_ton | 97 | 503.0 | 76.0 | 35.0 | 39.0 | 1200.30 | 51.2821 | 391.58 | 84.89 | 117.14 |
| 51_wee | 197 | 222.0 | 44.0 | 12.0 | 14.0 | 1200.61 | 42.8571 | 520.59 | 80.18 | 266.67 |
| 52_hes | 73 | 303.0 | 54.0 | 50.0 | 50.0 | 5.85 | 0.0 | 92.74 | 82.18 | 8.00 |
| 52_ros | 101 | 121.0 | 11.0 | 10.0 | 10.0 | 1.33 | 0.0 | 57.27 | 90.91 | 10.00 |
| 52_ton | 1707 | 390.0 | 99.0 | 43.0 | 45.0 | 1200.46 | 4.4444 | 357.54 | 74.62 | 130.23 |
| 52_wee | 97 | 152.0 | 31.0 | 9.0 | 13.0 | 1200.44 | 84.6154 | 513.31 | 79.61 | 244.44 |
| 61_hes | 42 | 918.0 | 66.0 | 66.0 | 66.0 | 3.32 | 0.0 | 228.44 | 92.81 | 0.00 |
| 61_ros | 0 | 72.0 | 17.0 | 16.0 | 16.0 | 1.64 | 0.0 | 82.76 | 76.39 | 6.25 |
| 61_ton | 313 | 1053.0 | 183.0 | 61.0 | 73.0 | 1200.41 | 38.3562 | 482.14 | 82.62 | 200.00 |
| 61_wee | 733 | 539.0 | 62.0 | 15.0 | 20.0 | 1200.44 | 45.0 | 503.89 | 88.50 | 313.33 |
| 62_hes | 23 | 549.0 | 56.0 | 56.0 | 56.0 | 4.05 | 0.0 | 221.65 | 89.80 | 0.00 |
| 62_ros | 202 | 61.0 | 13.0 | 13.0 | 13.0 | 2.95 | 0.0 | 123.92 | 78.69 | 0.00 |
| 62_ton | 131 | 693.0 | 138.0 | 66.0 | 71.0 | 1200.29 | 53.5211 | 411.77 | 80.09 | 109.09 |
| 62_wee | 223 | 406.0 | 56.0 | 18.0 | 19.0 | 1200.36 | 5.2632 | 456.67 | 86.21 | 211.11 |
| 71_hes | 1 | 554.0 | 91.0 | 91.0 | 91.0 | 1.39 | 0.0 | 174.16 | 83.57 | 0.00 |
| 71_ros | 47 | 65.0 | 16.0 | 15.0 | 15.0 | 1.53 | 0.0 | 86.30 | 75.38 | 6.67 |
| 71_ton | 281 | 1114.0 | 264.0 | 54.0 | 73.0 | 1200.60 | 67.1233 | 340.76 | 76.30 | 388.89 |
| 71_wee | 487 | 687.0 | 83.0 | 18.0 | 19.0 | 1200.65 | 52.6316 | 365.31 | 87.92 | 361.11 |
| 72_hes | 7 | 618.0 | 72.0 | 65.0 | 65.0 | 4.94 | 0.0 | 147.29 | 88.35 | 10.77 |
| 72_ros | 7 | 56.0 | 16.0 | 16.0 | 16.0 | 2.19 | 0.0 | 63.84 | 71.43 | 0.00 |
| 72_ton | 131 | 716.0 | 229.0 | 57.0 | 59.0 | 1200.19 | 45.7627 | 291.56 | 68.02 | 301.75 |
| 72_wee | 281 | 595.0 | 54.0 | 16.0 | 21.0 | 1200.18 | 42.8571 | 315.88 | 90.92 | 237.50 |

**Visão global revisada** Os novos resultados mostram:

- A VNS atingiu o ótimo em **14 instâncias**.
- A melhoria média sobre $SI$ continua elevada (**aprox. 75%**).
- Contudo, a distância ao ótimo ($Gap\_to\_Optimal$) permanece alta em instâncias grandes.
- O Gurobi segue sendo dominante: atinge o ótimo em **28 instâncias** e fornece limites inferiores fortes nas demais.

De maneira clara, a VNS melhora agressivamente a construção inicial, mas esbarra em ótimo local com facilidade. As famílias `ton` e `wee` permanecem extremamente difíceis, mantendo *gaps* altos tanto para o solver quanto para a heurística.

# 5 Conclusão

Os resultados atualizados reforçam a principal conclusão: **o modelo MILP resolvido pelo Gurobi é superior em qualidade**, enquanto a VNS é competitiva apenas como heurística rápida para gerar bons valores iniciais ou soluções viáveis em larga escala.

Embora a VNS tenha encontrado 14 ótimos, ela continua apresentando *gaps* grandes nas instâncias de maior porte, indicando que sua estrutura atual atinge um **limite natural de desempenho**.

Portanto, no ALWABP, a VNS cumpre bem o papel de refinadora de soluções iniciais, mas não substitui o modelo exato quando a qualidade da solução é crítica. Trabalhos futuros devem priorizar mecanismos mais fortes de diversificação e intensificação, como VNS reativo, busca tabu ou hibridização MILP+VNS.

# Referências


- Miralles, C., Garcia-Sabater, J. P., Andres, C., & Cardos, M. (2007). Advantages of assembly lines in sheltered work centres for disabled. *International Journal of Production Economics*, 110(1–2), 187–197.
- Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), 1097–1100.
- Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3), 449–467.
- Gendreau, M., & Potvin, J. Y. (Eds.). (2019). *Handbook of Metaheuristics* (3rd ed.). Springer.
- Torres, P., & Lima, C. (2018). *Metaheuristics for Logistics*. Wiley.
- Knuth, D. E. (1997). *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (3rd ed.). Addison-Wesley.