Relatório Técnico: Otimização de Balanceamento de Linha (ALWABP) via Colônia de Formigas Híbrida

Guilherme Freire, Mateus Poddis, Arthur Catarino

Dezembro, 2025

Resumo

Este relatório documenta a implementação de uma meta-heurística baseada em Otimização por Colônia de Formigas (ACO) para a resolução do Problema de Balanceamento de Linha de Montagem com Atribuição de Trabalhadores (ALWABP). O trabalho foca na minimização do tempo de ciclo (E_{max}) em um cenário onde a eficiência dos trabalhadores é heterogênea. A solução proposta introduz uma estratégia de feromônio em dois níveis, heurísticas baseadas em *Order Strength*, pré-processamento topológico e uma busca local dedicada à redução de gargalos, além de tratar restrições de inviabilidade através de penalização dinâmica.

1 Introdução

O alocamento eficiente de recursos é um desafio central na Pesquisa Operacional. O problema abordado, ALWABP (*Assembly Line Worker Assignment and Balancing Problem*), estende o problema clássico de balanceamento de linhas ao considerar que o tempo de execução de uma tarefa varia conforme o trabalhador designado para a estação.

O objetivo é minimizar o *makespan* ou tempo de ciclo, definido pela estação com a maior carga de trabalho. Devido à natureza NP-difícil do problema, métodos exatos tornam-se inviáveis para grandes instâncias, motivando o uso de meta-heurísticas. Este projeto implementa um algoritmo ACO híbrido, robusto a dados de entrada que representam falhas ou indisponibilidade (divisão por zero), utilizando técnicas de penalização "Grande-M".

2 Formulação Matemática

A modelagem matemática segue a estrutura formalizada para o problema. Seja N o conjunto de tarefas e K o conjunto de trabalhadores/estações. Definimos as variáveis de decisão:

- $X_{nek} \in \{0, 1\}$: Variável binária que indica se a tarefa n é realizada na estação e pelo trabalhador k.
- $Y_{ek} \in \{0, 1\}$: Variável binária que indica se o trabalhador k está alocado à estação e.

- E_{max}: O tempo de ciclo (estação com a carga mais pesada).

A Função Objetivo é minimizar o tempo de ciclo:

$$\text{Min } E_{max} \tag{1}$$

Sujeito às seguintes restrições principais:

1. **Definição do Tempo de Ciclo:** A carga de qualquer estação não pode exceder E_{max}:

$$\forall e \in K, \quad \sum_{n \in N} \sum_{k \in K} X_{nek} \cdot T_{nk} \leq E_{max} \tag{2}$$

2. **Unicidade da Tarefa:** Cada tarefa deve ser executada exatamente uma vez:

$$\forall n \in N, \quad \sum_{e \in K} \sum_{k \in K} X_{nek} = 1 \tag{3}$$

3. **Alocação Única:** Cada estação possui apenas um trabalhador e cada trabalhador assume apenas uma estação:

$$\forall e \in K, \sum_{k \in K} Y_{ek} = 1 \quad \text{e} \quad \forall k \in K, \sum_{e \in K} Y_{ek} = 1 \tag{4}$$

4. **Vínculo Lógico:** Uma tarefa só ocorre se o trabalhador estiver na estação:

$$X_{nek} \leq Y_{ek} \tag{5}$$

5. **Precedência:** Respeito ao grafo $G_{ij} = 1$, onde a tarefa i deve ocorrer em uma estação anterior ou igual à tarefa j.

3 Algoritmo Proposto e Implementação

A solução foi implementada em Python utilizando uma variação do *Max-Min Ant System*. O algoritmo se destaca por utilizar uma estrutura de decisão hierárquica, pré-processamento topológico e tratamento numérico de exceções.

3.1 Pré-processamento e Análise da Instância

A eficiência do algoritmo depende fundamentalmente da preparação dos dados. A função de leitura realiza um processamento intensivo:

- **Grafos:** Constrói-se a matriz de tempos T_{nk} e o grafo de precedências G. Gera-se também o grafo inverso G^{-1}, crucial para a validação eficiente na busca local.
- **Topologia:** As tarefas são ordenadas topologicamente e segmentadas em regiões correspondentes ao número de estações ($|K|$). Isso cria um viés positivo para alocar especialistas em tarefas iniciais nas primeiras estações.
- **Métricas:** Calculam-se o *Lower Bound* (LB) teórico, a Carga Alvo (C_{alvo}) média ideal e o *Order Strength* (OS), que mede a densidade do grafo e pondera as heurísticas.

3.2 Representação da Solução

A solução é encapsulada na classe `Formiga`, composta por uma lista de objetos `Estação`. Cada estação armazena o trabalhador alocado, o conjunto de tarefas e a carga atual. A formiga possui métodos intrínsecos para cálculo do C_{max}.

3.3 Estrutura de Feromônios em Dois Níveis

O algoritmo utiliza duas matrizes distintas, inicializadas com $\tau_0 = 100/C_{alvo}$:

- `feromoniosTE`: Guia a alocação de Trabalhadores às Estações.
- `feromoniosTarefas`: Guia a sequência de tarefas dentro de cada estação.

3.4 Construção da Solução

3.4.1 Fase 1: Alocação de Trabalhadores

A formiga define o trabalhador k para a estação e utilizando a regra de transição:

$$P_{ek} = \frac{[\tau_{ek}]^\alpha \cdot [\eta_{ek}]^\beta}{\sum_{l \in \mathcal{U}} [\tau_{el}]^\alpha \cdot [\eta_{el}]^\beta} \tag{6}$$

A heurística η_{ek} é ponderada pelo *Order Strength* (OS):

$$\eta_{ek} = OS \cdot \left(\frac{1}{\bar{T}_{local}} \right) + (1 - OS) \cdot \left(\frac{1}{\bar{T}_{global}} \right) \tag{7}$$

Isso equilibra a eficiência local (necessária em grafos densos) com a eficiência global (útil em grafos desconexos).

3.4.2 Fase 2: Alocação de Tarefas

Para cada estação, tarefas candidatas são selecionadas com probabilidade:

$$P_{ij} = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{u \in \Omega} [\tau_{iu}]^\alpha \cdot [\eta_{iu}]^\beta} \tag{8}$$

A heurística é gulosa ($1/T_{n,k}$). O preenchimento respeita um C_{alvo} dinâmico, recalculado a cada estação para evitar desbalanceamento de carga no final da linha.

3.5 Tratamento de Inviabilidade (Penalização)

Tarefas alocadas a trabalhadores incapazes ($T_{nk} = \infty$ ou $D_i \leq 0$) são tratadas via penalização ("Grande-M") na função de custo:

$$Tempo_{calc} = \begin{cases} \frac{T_i}{D_i} & \text{se } D_i > 0 \\ 999999 & \text{se } D_i \leq 0 \end{cases} \tag{9}$$

Essa lógica força o algoritmo a evitar configurações inválidas naturalmente.

3.6 Busca Local (Shift)

Após a construção, aplica-se um refinamento determinístico:

1. Identifica-se a estação gargalo (maior carga).
2. Tenta-se mover tarefas para outras estações, validando se:
   - O trabalhador destino é apto;
   - As precedências são respeitadas (via G e G^{-1});
   - A nova carga destino não excede o gargalo original.

3.7 Atualização de Feromônios

A evaporação ocorre com taxa $\rho = 0.1$:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} \tag{10}$$

O reforço é elitista (apenas as 10% melhores formigas), proporcional à qualidade da solução:

$$\Delta\tau = \frac{100}{C_{max}} \tag{11}$$

4 Resultados e Discussões

Esta seção apresenta a análise comparativa entre o método exato (Solver Gurobi) e a meta-heurística proposta baseada em Otimização por Colônia de Formigas (ACO Proposto). O algoritmo proposto integra uma heurística construtiva com estratégias de busca local para o problema de balanceamento de linha.

Os testes foram realizados em quatro classes clássicas de instâncias: *Hes*, *Ros*, *Ton* e *Wee*. O método exato e o ACO foram limitados a 300 segundos (5 minutos) de execução por instância, enquanto o ACO Proposto utilizou um critério de parada baseado em iterações sem melhoria.

4.1 Análise de Desempenho Geral

A Tabela 1 resume as médias obtidas para cada classe. Observa-se o comportamento do algoritmo desde a solução inicial (primeira iteração válida) até a solução final refinada.

Tabela 1: Comparativo de Desempenho: ACO Proposto vs. Solver Gurobi

Classe	Sol. Inicial	Sol. Final	Melhoria (%)	Gap Ótimo (%)	Tempo ACO (s)	Tempo Solve
Ros	22.37	21.05	5.89	26.30	1.77	
Hes	117.13	101.93	12.98	33.10	2.23	
Ton	170.53	153.30	10.11	109.28	9.38	30
Wee	48.05	40.95	14.78	107.34	15.84	28

4.2 Eficiência Computacional

Os resultados demonstram uma clara distinção entre instâncias de baixa e alta complexidade:

- **Instâncias Leves (Ros, Hes):** O Solver Gurobi foi capaz de provar a otimalidade em menos de 1 segundo, superando o tempo de convergência do ACO (2s). Nestes casos, o método exato é preferível.
- **Instâncias Complexas (Ton, Wee):** O Gurobi atingiu sistematicamente o tempo limite de 300 segundos (5 minutos) sem garantir a otimalidade global em todos os casos. Em contraste, o **ACO Proposto** encontrou suas soluções finais em uma média de 9 a 16 segundos. Isso representa uma redução de tempo na ordem de 95%, tornando a meta-heurística uma alternativa viável para cenários onde a resposta rápida é crítica.

4.3 Evolução da Solução e Qualidade

A coluna "Melhoria (%)" na Tabela 1 destaca a eficácia das estratégias de busca local e atualização de feromônio implementadas no algoritmo modificado.

- **Refinamento da Solução:** Em todas as classes, a solução final foi consistentemente melhor que a inicial. O ganho variou entre 5,89% (Ros) e 14,78% (Wee). Isso confirma que o algoritmo não está estagnado na solução construtiva e consegue explorar o espaço de busca efetivamente.
- **Gap em relação ao Ótimo:** Apesar da melhoria, o Gap para o ótimo (ou melhor limitante encontrado pelo Gurobi) permanece alto nas instâncias difíceis (acima de 100% para Ton e Wee). Isso indica que, embora o ACO seja rápido, ele tende a convergir para ótimos locais distantes do ótimo global nessas topologias complexas.

5 Conclusão

A abordagem de Colônia de Formigas Híbrida para o ALWABP mostrou-se uma estratégia robusta e viável. A decomposição do problema em duas matrizes de feromônio permitiu que o algoritmo aprendesse simultaneamente a configuração ótima da equipe e o balanceamento da linha. A técnica de penalização assegurou a estabilidade matemática necessária, enquanto a busca local garantiu um refinamento médio de 10% sobre as soluções iniciais. Embora não supere o método exato em instâncias pequenas, o algoritmo oferece um ganho de tempo crítico em instâncias complexas.

Referências

[1] Dorigo, M.; Stützle, T. *Ant Colony Optimization*. MIT Press, 2004.

[2] Stützle, T.; Hoos, H. H. *MAX-MIN Ant System*. Future Generation Computer Systems, v. 16, n. 8, p. 889-914, 2000.

[3] Pinto, M. P. A. *Algoritmo Colônia de Formigas para o Problema de Pouso de Aeronaves em Múltiplas Pistas*. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Universidade de Pernambuco (UPE), Escola Politécnica de Pernambuco, Recife, 2022.