UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE COMPUTAÇÃO

Estudo do TATSP

MC798 - Programação Linear Inteira

Eliel Lucas de Oliveira Carvalho
ra: 295745

Campinas
2025

1 Introdução

O *Trigger Arc Traveling Salesman Problem* (TATSP) é uma variação recente do clássico *Traveling Salesman Problem* (TSP). Nesse modelo, certos arcos, denominados *trigger arcs*, podem alterar o custo de outros arcos quando percorridos antes deles. Tal dinâmica reflete aplicações práticas, como a movimentação de estantes em armazéns compactáveis: abrir um corredor desloca prateleiras vizinhas e muda o tempo (ou energia) necessário para transitar por elas.

Este estudo tem três objetivos principais:

1. compreender formalmente o TATSP e reproduzir a formulação inteira proposta por (CERRONE et al., 2025);
2. avaliar relaxações (linear e Lagrangeana) e estratégias de geração de colunas para obter limites inferiores e superiores em instâncias de diferentes escalas;
3. experimentar heurísticas de construção e refinamento, guiadas por soluções fracionárias, capazes de produzir tours viáveis em tempo razoável.

Durante a execução do projeto, explorei técnicas exatas (programação linear inteira) e métodos aproximados (subgradiente, busca local 3-Opt). Também implementei as rotinas em Julia+JuMP, utilizando o solver Gurobi.

Os próximos capítulos apresentam, respectivamente: a formulação matemática do TATSP, as relaxações e heurísticas desenvolvidas, a metodologia computacional e a análise crítica dos resultados obtidos.

2 Formulação Matemática do TATSP (PLI)

2.1 Definição e Notação

- $G = (N, A)$: grafo dirigido com conjunto de nós N e arcos A.
- $c(a)$: custo base do arco $a \in A$.
- R: conjunto de relações gatilho. Cada relação $r = (t, a)$ indica que o arco t é gatilho para o arco a.
- $c(r)$: custo alternativo de a se a relação $r = (t, a)$ estiver ativa.
- Apenas a última relação válida para um arco-alvo é aplicada.

A solução procurada é um ciclo hamiltoniano no grafo, e o custo total depende da ordem dos arcos no tour, já que isso determina quais relações são ativadas.

2.2 Variáveis de Decisão

- $x_a \in \{0, 1\}$: indica se o arco a pertence ao tour.
- $y_a \in \{0, 1\}$: indica se o arco a paga seu custo base.
- $y_r \in \{0, 1\}$: indica se a relação $r = (t, a)$ está ativa.
- $\hat{y}_r \in \{0, 1\}$: variável auxiliar para validar se o gatilho t precede o arco-alvo a.
- $u_i \in \mathbb{Z}$: ordem de visita do nó i (para evitar subtours via restrições MTZ).

2.3 Função Objetivo

$$\min \sum_{a \in A} c(a) \cdot y_a + \sum_{r \in R} c(r) \cdot y_r \tag{1}$$

Ou seja, cada arco presente no tour paga:

- seu custo base $c(a)$, se nenhuma relação for ativada;
- ou o custo modificado $c(r)$, se uma relação válida estiver ativa.

2.4 Restrições

1. **Tamanho do tour:**

$$\sum_{(i,j) \in A} x_{ij} = |N| \tag{2}$$

2. **Eliminação de subtours (MTZ):**

$$u_i + 1 \leq u_j + M(1 - x_{ij}) \quad \forall (i,j) \in A,\ j \neq 0 \tag{3}$$

3. **Entrada e saída únicas por nó:**

$$\sum_{(i,j) \in A} x_{ij} = 1 \quad \forall j \in N \tag{4}$$

$$\sum_{(i,j) \in A} x_{ij} = 1 \quad \forall i \in N \tag{5}$$

4. **Cada arco ativo paga um custo:**

$$y_{ij} + \sum_{r \in R_{(i,j)}} y_r = x_{ij}, \quad \forall (i,j) \in A \tag{6}$$

5. **Ativação depende do gatilho estar no tour:**

$$y_r \leq x_{ij}, \quad \text{para } r = ((i,j),(h,k)) \tag{7}$$

6. **Gatilho deve preceder o arco-alvo:**

$$u_i + 1 \leq u_h + M(1 - y_r) \tag{8}$$

7. **Relação desativada se gatilho vier depois:**

$$u_h + 1 \leq u_i + M(1 - \hat{y}_r) \tag{9}$$

8. **Coerência entre ordem e uso das variáveis:**

$$x_{ij} \leq (1 - x_{hk}) + (1 - y_{hk}) + \hat{y}_r \tag{10}$$

9. **Último gatilho prevalece:**

$$u_{\hat{i}} - M \cdot \hat{y}_{r_2} \leq u_i + M(2 - y_{r_1} - x_{\hat{i}\hat{j}}) - 1 \tag{11}$$

2.5 Observação sobre Complexidade

Quando $R = \varnothing$ o modelo reduz-se ao TSP clássico, problema NP-difícil. A presença de gatilhos adiciona $O(|R|)$ variáveis e restrições, o que faz com que instâncias de porte médio já se tornem difíceis para métodos exatos. Relaxações e heurísticas tornam-se, portanto, componentes essenciais.

3 Relaxações e Heurísticas

Este capítulo resume as abordagens aproximadas avaliadas.

3.1 Relaxação Linear (LP)

A relaxação linear remove o domínio binário de $x_a, y_a, y_r, \hat{y}_r$, permitindo valores contínuos em $[0, 1]$. Mantém-se u_i inteiro para preservar o efeito antissubtour das restrições MTZ. A solução fracionária produz um limite inferior rápido e serve de referência para heurísticas de arredondamento.

3.2 Relaxação Lagrangeana

A relaxação Lagrangeana é uma técnica de decomposição útil para obter limitantes inferiores mais fortes ao custo de resolver um problema dual. Neste trabalho, optei por relaxar as restrições de grau, que garantem que cada nó tenha exatamente um arco de entrada e um de saída no tour.

Para isso, associei multiplicadores de Lagrange às restrições violadas:

- λ_j^{in}: multiplicador da restrição de entrada do nó j;
- λ_i^{out}: multiplicador da restrição de saída do nó i.

As duas restrições abaixo, originalmente impostas com igualdade no modelo inteiro, foram transferidas para a função objetivo com penalidades:

$$\sum_{\substack{a \in A \\ \text{dest}(a)=j}} x_a = 1 \quad \forall j \in N \tag{12}$$

$$\sum_{\substack{a \in A \\ \text{orig}(a)=i}} x_a = 1 \quad \forall i \in N \tag{13}$$

Função Lagrangeana

A função Lagrangeana resulta da penalização das restrições acima com os respectivos multiplicadores:

$$\mathcal{L}(x, y, \lambda) = \sum_{a \in A} c_a \cdot y_a + \sum_{r \in R} c_r \cdot y_r \tag{14}$$

$$+ \sum_{j \in N} \lambda_j^{\text{in}} \left(\sum_{\substack{a \in A \\ \text{dest}(a)=j}} x_a - 1 \right) \tag{15}$$

$$+ \sum_{i \in N} \lambda_i^{\text{out}} \left(\sum_{\substack{a \in A \\ \text{orig}(a)=i}} x_a - 1 \right) \tag{16}$$

Reorganizando os termos, chego à forma compacta:

$$\mathcal{L}(x, y, \lambda) = \sum_{a \in A} \left(c_a \cdot y_a + (\lambda_{\text{dest}(a)}^{\text{in}} + \lambda_{\text{orig}(a)}^{\text{out}}) \cdot x_a \right) + \sum_{r \in R} c_r \cdot y_r - \sum_{i \in N} \left(\lambda_i^{\text{in}} + \lambda_i^{\text{out}} \right) \tag{17}$$

Essa expressão define o problema Lagrangeano com custos reduzidos nos arcos, ajustados dinamicamente com os multiplicadores.

Problema Dual

O objetivo agora é encontrar os multiplicadores que maximizam o valor mínimo da função Lagrangeana. Isso define o problema dual:

$$\max_{\lambda \in \mathbb{R}^{2n}} \quad \min_{(x,y)\in\mathcal{F}} \mathcal{L}(x, y, \lambda) \tag{18}$$

O conjunto $\mathcal{F}$ contém as soluções que satisfazem todas as demais restrições do modelo original, como:

- restrições MTZ (eliminação de subtours),
- ativação válida das relações gatilho,
- exclusividade entre custo base e custo via gatilho,
- e coerência entre ordem de visita e ativação de relações.

As restrições de grau, no entanto, foram retiradas explicitamente da formulação primal e tratadas indiretamente via penalização.

Resolução

Para resolver o problema dual, utilizei o método do subgradiente, que atualiza iterativamente os multiplicadores com base na violação das restrições de grau.

Em cada iteração, resolvo o subproblema interno:

$$\min_{(x,y)\in\mathcal{F}} \mathcal{L}(x, y, \lambda)$$

utilizando um modelo de programação linear contínua, onde as variáveis x, y, $\hat{y}$ e u são relaxadas para valores reais.

Apesar de a solução obtida ser fracionária, ela fornece um limitante inferior válido para o problema original. Ao longo das iterações, ajusto os multiplicadores conforme a violação observada, buscando maximizar o valor da função Lagrangeana.

3.3 Subproblema Lagrangeano

A formulação Lagrangeana do Trigger Arc TSP é construída a partir do modelo inteiro original, relaxando-se as restrições de grau por meio de multiplicadores de Lagrange λ_i^{out} e λ_i^{in}, para cada nó $i \in \{1, \ldots, n\}$. O problema dual resultante é resolvido iterativamente com o método do subgradiente (ver Capítulo 4).

Em cada iteração, os multiplicadores λ são fixados, e o modelo que preciso resolver é chamado de subproblema primal Lagrangeano. Ele fornece o valor da função Lagrangeana para os multiplicadores atuais e serve de base para o cálculo do subgradiente.

Variáveis de decisão:

- $x_a \in [0, 1]$: variável que indica se o arco a está presente no tour;
- $y_a \in [0, 1]$: indica se o custo base é aplicado ao arco a;
- $y_r, \hat{y}_r \in [0, 1]$: variáveis relacionadas à ativação e à precedência das relações de gatilho;
- $u_i \in [1, n]$: ordem de visita dos nós, usada para eliminar subtours com as restrições MTZ.

Função objetivo:

$$\min_{x,y^a,y^r} \quad \sum_{a\in A} c_a \cdot y_a^a + \sum_{r\in R} c_r \cdot y_r^r + \sum_{a\in A} \lambda_{u(a)}^{\text{out}} \cdot x_a + \sum_{a\in A} \lambda_{v(a)}^{\text{in}} \cdot x_a \tag{19}$$

O objetivo é minimizar o custo total da rota, considerando os custos base e de gatilho, além dos termos adicionados pelos multiplicadores que penalizam a violação das restrições de grau.

Restrições do subproblema: Mesmo com a relaxação de grau, várias restrições do modelo original permanecem ativas. O subproblema inclui:

- **Seleção de n arcos:** garante que o tour tenha exatamente n arcos:

$$\sum_{a\in A} x_a = n$$

- **Eliminação de subtours:** uso das restrições MTZ com variáveis u_i contínuas;
- **Compatibilidade de custos:** para cada arco incluído no tour, deve ser pago exatamente um custo — seja o custo base ou o custo de uma relação ativada;
- **Validação das relações de gatilho:** um arco-alvo só pode ativar uma relação se o gatilho correspondente estiver presente e tiver sido percorrido antes;
- **Último gatilho prevalecente:** no caso de múltiplas relações para o mesmo arco-alvo, apenas a mais recente (na ordem do tour) deve ser considerada ativa.

Resolução

Resolvo o subproblema como um modelo de programação linear contínua (LP) utilizando o solver Gurobi. Isso permite obter uma solução fracionária de forma eficiente, que é utilizada para:

1. Calcular o valor da função Lagrangeana na iteração atual;
2. Obter o subgradiente, a partir da violação das restrições de grau;
3. Atualizar os multiplicadores de Lagrange para a próxima iteração;
4. Servir de base para heurísticas construtivas que buscam soluções inteiras a partir da solução fracionária.

A estrutura do subproblema é semelhante à do modelo inteiro original, porém com menor complexidade computacional devido à relaxação das variáveis e restrições. Na prática, isso permite avaliar rapidamente a qualidade de cada vetor de multiplicadores testado ao longo da otimização dual.

3.4 Geração de Colunas

3.4.1 LP

A Geração de Colunas (GC) é uma técnica clássica para decompor problemas de grande escala. Sua aplicação ao TSP foi estabelecida por (HELD; KARP, 1970), que a utilizaram em uma formulação onde as colunas correspondiam a 1-árvores. A abordagem aqui implementada, embora inspirada no mesmo princípio iterativo de mestre-subproblema, adota uma formulação mais direta por *set partitioning*, na qual cada coluna representa um tour hamiltoniano completo e viável do Trigger Arc TSP.

Para cada coluna $c \in \mathcal{C}$, seja:

- d_c: o custo total do tour T_c;
- $\theta_c \in [0, 1]$: variável de decisão associada à coluna c;
- $a_{jc} = 1$ se o nó j pertence ao tour T_c, e 0 caso contrário.

A formulação do problema mestre é:

$$\begin{aligned} \min_{\theta} \quad & \sum_{c \in \mathcal{C}} d_c \cdot \theta_c \\ \text{sujeito a} \quad & \sum_{c \in \mathcal{C}} a_{jc} \cdot \theta_c = 1 \quad \forall j \in \{1, \ldots, n\} \\ & \theta_c \geq 0 \quad \forall c \in \mathcal{C} \end{aligned}$$

A resolução do problema mestre fornece dois resultados importantes:

- Um limitante inferior (LB) dado pelo valor ótimo da solução fracionária;
- Um vetor de multiplicadores duais π_j para cada restrição de cobertura de nó.

Subproblema de Pricing

O subproblema de *pricing* utiliza os multiplicadores duais π_j do problema mestre para identificar novas colunas (tours) com custo reduzido negativo. Ele é formulado como um problema TSP com custos reduzidos:

$$\text{Para cada arco } a = (u, v), \text{ define-se o custo reduzido: } \tilde{c}_a = c_a + \pi_u + \pi_v$$

Com isso, o subproblema consiste em encontrar um tour hamiltoniano com o menor custo total reduzido $\sum_a \tilde{c}_a x_a$. Se o custo reduzido total da nova coluna for negativo, ela é adicionada ao problema mestre. Caso contrário, a otimização é considerada convergida.

Este subproblema é resolvido como um modelo de programação inteira, utilizando as mesmas restrições do Trigger Arc TSP, mas com custos ajustados.

3.4.2 UB

Para obter um limitante superior viável (solução inteira), o mesmo problema mestre anterior foi utilizado, mas com variáveis inteiras:

$$\begin{aligned} \min_{\theta} \quad & \sum_{c \in \mathcal{C}} d_c \cdot \theta_c \\ \text{sujeito a} \quad & \sum_{c \in \mathcal{C}} a_{jc} \cdot \theta_c = 1 \quad \forall j \in \{1, \ldots, n\} \\ & \theta_c \in \{0, 1\} \quad \forall c \in \mathcal{C} \end{aligned}$$

Essa formulação busca escolher exatamente uma coluna (tour) que cubra todos os nós, dentre as disponíveis. Como cada nó deve estar presente em exatamente um tour, e θ_c é binária, o modelo seleciona um tour completo viável dentre os gerados, retornando um limitante superior (UB) para o Trigger Arc TSP.

3.5 Heurísticas Baseadas em LP e Lagrange

- **Arredondamento da LP**: começa na solução fracionária, constrói um ciclo guloso guiado por x_a e refina por 3-Opt.
- **Heurística Lagrangeana**: utiliza o vetor x obtido na melhor iteração dual, aplica inserção com reparo, seguida de uma busca local do tipo Busca Local Iterada (ILS) com movimentos 3-Opt. Esta abordagem, que alterna entre uma busca local intensiva e uma perturbação para escapar de ótimos locais, é uma meta-heurística eficaz para o TSP e suas variantes, conforme demonstrado em (CASTELLUCCI, 2018).

Os pseudocódigos completos, que dão mais clareza sobre os métodos, estão na seção 4.3.

4 Metodologia

4.1 Ambiente Computacional

- **Linguagem:** Julia 1.10.4
- **Solver:** Gurobi
- **Biblioteca de otimização:** JuMP
- **Ambiente:** Sistema Operacional: Linux, CPU: i5, RAM: 16 GB

4.2 Instâncias Utilizadas

A Tabela 2 apresenta o conjunto de instâncias utilizado no estudo, detalhando suas respectivas dimensões (N, A e R). Parte das instâncias disponibilizadas pelo professor não foram executadas e outras não foram executadas os algoritmos que precisavam do PLI, tendo em vista a falta de poder computacional para instâncias maiores.

Tabela 1: Conjunto de instâncias utilizadas nos experimentos computacionais.

Nome Arquivo	$\|N\|$	$\|A\|$	$\|R\|$
grf1	20	300	5651
grf2	20	300	5708
grf5	20	200	1880
grf6	20	200	1732
grf19	20	300	5708
grf101	18	90	1144
grf102	22	132	2135
grf103	22	110	1824
grf104	27	162	3410
grf105	26	130	2664
grf106	32	192	4985
grf107	37	222	6860
grf108	37	222	6860
grf109	37	222	6860
grf110	37	222	6860
grf111	42	252	9035

(Continua na próxima página)

Tabela 1 (continuação)

Nome Arquivo	$\|N\|$	$\|A\|$	$\|R\|$
grf112	50	350	13168
grf113	58	464	23583
grf114	66	594	34640
grf117	52	312	14285
grf118	62	434	23988

4.3 Descrição e Pseudocódigos

4.3.1 Relaxação LP

- Resolve o modelo do TATSP com variáveis contínuas, retornando um limitante inferior (LB) e a solução fracionária x^*.

function LP_Relaxation(T)
    Extrair parâmetros: número de nós (n), arcos (m), relações (R)
    Criar modelo com variáveis contínuas:
    $x[a],\ ya[a],\ yr[r],\ \hat{y}[r] \in [0, 1]$
    $u[i] \in [1, n]$ (inteiras para MTZ)
    Adicionar restrições:
    Seleção de n arcos
    Grau de entrada e saída = 1
    Subtour elimination (MTZ)
    Custo base ou ativação de relação
    Ordem entre gatilho e alvo (último gatilho prevalece)
    Resolver modelo com Gurobi
    Obter solução fracionária x_{frac} e valor do LB
    Medir tempo de resolução
    return (x_{frac}, $tempo$, LB)
end function

4.3.2 Heurística de Arredondamento da LP

- Executa múltiplas iterações para construir e melhorar soluções viáveis com base na solução fracionária x_{frac}.

function Arredondamento_LP(x_{frac}, T)
    for cada iteração até o limite pré-definido **do**
        Construir tour com heurística gulosa guiada por x_{frac}
        if tour inválido **then**
            Tentar reparar a solução
        end if
        Aplicar busca local com 3-Opt
        Atualizar melhor solução (se houver melhora)
        Registrar tempo e custo
    end for
    return melhor solução e custo encontrado
end function

4.3.3 Construção Gulosa Guiada

- Constrói um tour hamiltoniano de forma gulosa, orientada pela solução fracionária x^*.

- Em cada passo, escolhe o arco de menor custo penalizado com base em x^*.
- Em casos de beco sem saída, utiliza uma busca em largura (BFS) para reconectar com um nó não visitado.
- Essa abordagem é inspirada no uso de soluções fracionárias de relaxações lineares como guia para heurísticas construtivas, tal como discutido em (BERTSIMAS; VOHRA, 1998).

function CONSTRUCAO_GULOSA_GUIADA(T, x^*, α)
    Inicializar: $n \leftarrow$ número de nós em T
    nos_visitados $\leftarrow \{1\}$, tour_final $\leftarrow [\,]$, no_atual $\leftarrow 1$
    while |nos_visitados| $< n$ **do**
        Obter arcos candidatos saindo de no_atual para nós não visitados
        if há candidatos diretos **then**
            Para cada arco a, computar $score(a) = custo(a) - \alpha \cdot x^*_a$
            Selecionar arco com menor *score*
            Atualizar tour e nó atual; marcar destino como visitado
        else
            Usar BFS para alcançar nó não visitado mais próximo
            Adicionar arcos intermediários ao tour e atualizar nós visitados
        end if
    end while
    Tentar fechar o ciclo retornando ao nó 1
    if existe arco direto **then**
        Adicionar ao tour
    else
        Usar BFS para retornar e adicionar arcos ao tour
    end if
    Calcular custo final do tour
    return (tour_final, custo)
end function

4.3.4 Relaxação Lagrangeana

- Aplica método do subgradiente para resolver a relaxação dual do TATSP, retornando o melhor limitante inferior (LB) e os multiplicadores ótimos.

function RELAXACAO_LAGRANGEANA(T)
    Inicializar multiplicadores $\lambda_{in}, \lambda_{out}$
    while critério de parada não atingido **do**
        Resolver subproblema com λ atuais
        Atualizar λ usando subgradiente
        Atualizar melhor LB e solução x_{lag}
    end while
    return melhor LB, multiplicadores e solução x_{lag}
end function

4.3.5 Método do Subgradiente

- O método resolve iterativamente o problema dual da relaxação Lagrangeana, utilizando atualizações de multiplicadores com base no subgradiente das restrições de grau, conforme detalhado em (MARGARIDA, 2009).

- A cada iteração, o subproblema primal é resolvido com os multiplicadores correntes, e as violações das restrições de grau definem a direção de atualização (MARGARIDA, 2009).

- O passo é adaptativo e diminui com o tempo. O processo retorna o melhor limitante inferior (LB) encontrado.

function METODO_SUBGRADIENTE($T, \lambda_{in}, \lambda_{out}, UB, \alpha$)
  Inicializar: $LB \leftarrow -\infty$, $x_{lag} \leftarrow$ null
  while tempo não excedido e iteração $< K$ **do**
    Resolver subproblema lagrangeano com os λ atuais
    Obter valor da função lagrangeana Z e solução x
    if $Z > LB$ **then**
      Atualizar $LB \leftarrow Z$, $x_{lag} \leftarrow x$
    end if
    Calcular subgradientes das restrições de grau:
    $g_i^{out} \leftarrow \sum_{a:\ u(a)=i} x_a - 1$
    $g_i^{in} \leftarrow \sum_{a:\ v(a)=i} x_a - 1$
    Calcular norma: $\|g\|^2 \leftarrow \sum_i (g_i^{out})^2 + (g_i^{in})^2$
    if $\|g\|^2 < \epsilon$ **then**
      break
    end if
    Calcular passo adaptativo: $t \leftarrow \alpha \cdot \dfrac{UB - Z}{\|g\|^2}$
    Atualizar multiplicadores:
    $\lambda_{out} \leftarrow \lambda_{out} + t \cdot g^{out}$
    $\lambda_{in} \leftarrow \lambda_{in} + t \cdot g^{in}$
    Reduzir fator de passo: $\alpha \leftarrow 0{,}95 \cdot \alpha$
  end while
  return ($LB,\ x_{lag},\ \lambda_{in},\ \lambda_{out}$)
end function

4.3.6 Column Generation (LB)

- Método iterativo para refinar o limitante inferior (LB), adicionando colunas (tours fracionários) ao problema mestre restrito.

function COLUMN_GENERATION_LB(T)
  Gerar colunas iniciais com heurística
  while houver coluna com custo reduzido negativo **do**
    Resolver problema mestre restrito (LP com colunas atuais)
    Obter multiplicadores duais do mestre
    Resolver subproblema de pricing com os multiplicadores
    if nova coluna encontrada **then**
      Adicionar nova coluna ao mestre
    else
      Encerrar iteração
    end if
  end while
  return Melhor LB obtido
end function

4.3.7 Column Generation (UB)

- Variante que gera tours viáveis (inteiros) como colunas, utilizando-os para encontrar um bom limitante superior (UB).

function COLUMN_GENERATION_UB(T)
    Gerar colunas iniciais viáveis com heurística
    while houver coluna promissora **do**
        Resolver problema inteiro com as colunas atuais
        Avaliar solução e armazenar como novo UB (se melhor)
        Gerar nova coluna viável
        if nenhuma nova coluna promissora **then**
            Encerrar
        else
            Adicionar coluna ao modelo
        end if
    end while
    return Melhor UB encontrado
end function

4.3.8 ILS com 3-Opt guiado pela Relaxação Lagrangeana

- Gera um *tour* inicial com a heurística gulosa guiada usando os valores fracionários x^{RLX}.
- Aplica busca local 3-Opt até ótimo local.
- Iterativamente perturba a sequência de nós e volta a aplicar 3-Opt, aceitando soluções piores até $\alpha_{\text{accept}} \cdot \text{custo}_{\text{best}}$ para diversificação.
- Reinicia aleatoriamente a cada `restart_every` iterações ou quando não há melhora por `max_no_improve` iterações.

function ILS_3OPT(T, x^{RLX}, α_{accept}, max_iter, max_no_improve, restart_every)
    $tour \leftarrow$ GREEDY_GUIDED_CONSTRUCTION(T, x^{RLX})
    while THREE_OPT(T, $tour$) **do**
    end while
    ($best\_tour$, $best\_cost$) $\leftarrow$ ($tour$, COST(T, tour))
    ($curr\_tour$, $curr\_cost$) $\leftarrow$ ($best\_tour$, $best\_cost$)
    $no\_improve \leftarrow 0$
    for $it = 1$ **to** max_iter **do**
        if itmod restart_every $= 0$ **then** ▷ reinício controlado
            $curr\_tour \leftarrow$ GREEDY_GUIDED_CONSTRUCTION(T, $\mathcal{U}(0,1)$)
            while THREE_OPT(T, $curr\_tour$) **do**
            end while
            $curr\_cost \leftarrow$ COST(T, $curr\_tour$)
        end if
        $seq \leftarrow$ PERTURBAR_SEQUÊNCIA_CONECTADA(T, $curr\_tour$)
        $tour \leftarrow$ NODE_SEQ_TO_ARCS(T, seq)
        while THREE_OPT(T, $tour$) **do**
        end while
        $cost \leftarrow$ COST(T, $tour$)
        if $cost < best\_cost$ **then** ▷ melhoria absoluta
            ($best\_tour$, $best\_cost$) $\leftarrow$ ($tour$, $cost$)
            ($curr\_tour$, $curr\_cost$) $\leftarrow$ ($tour$, $cost$)

```
            no_improve ← 0
        else if cost < α_accept · best_cost then                ▷ aceita levemente pior
            (curr_tour, curr_cost) ← (tour, cost)
            no_improve ← no_improve + 1
        else
            no_improve ← no_improve + 1
        end if
        if no_improve ≥ max_no_improve then
            break                                               ▷ sem progresso
        end if
    end for
    return (best_tour, best_cost)
end function
```

4.3.9 Busca Local 3-OPT

Decidi utilizar uma heurística de busca local para TSP, definido por (ROCKI; SUDA, 2012). A qual faz o seguinte:

- Aplica vizinhança 3-opt ao tour atual, tentando melhorar o custo total.
- Gera rearranjos possíveis de três segmentos consecutivos e verifica se formam um tour válido.
- Substitui o tour se encontrar uma melhoria de custo.

```
function THREE_OPT(T, tour)
    Calcular custo atual do tour c_atual
    Extrair sequência de nós order a partir dos arcos do tour
    for i = 2 até n − 3 do
        for j = i + 1 até n − 2 do
            for k = j + 1 até n − 1 do
                Gerar segmentos seg1, seg2, seg3, seg4 a partir de order
                Gerar todas as 7 permutações 3-opt de (seg2, seg3)
                for cada rearranjo new_order do
                    Construir novo tour new_tour a partir de new_order
                    if tour é válido e possui tamanho correto then
                        Calcular novo custo c_novo
                        if c_novo < c_atual then
                            Atualizar tour original com new_tour
                            return true
                        end if
                    end if
                end for
            end for
        end for
    end for
    return false
end function
```

4.3.10 Resolução Exata via ILP (Branch & Bound interno do Gurobi)

- Constrói um modelo **ILP** completo no `JuMP`, com variáveis $x, y_a, y_r, \hat{y}_r, u$ e todas as restrições do TATSP.

- Ajusta parâmetros do solver (`TimeLimit`, `MIPGap`, tolerâncias numéricas).
- Invoca `optimize!`; o Gurobi executa internamente seu Branch & Bound.
- Extrai a solução $x^\star$; reconstrói o tour, calcula o custo real e registra **UB** e **LB**.

function SOLVE_ILP(T)
  Construir modelo PLI $\mathcal{M}$ em `JuMP` ▷ variáveis $x, y_a, y_r, \hat{y}_r, u$
  Definir função-objetivo do TATSP relaxado
  Configurar parâmetros do Gurobi (*TimeLimit*, *MIPGap*)
  `optimize!`($\mathcal{M}$) ▷ Branch & Bound interno
  if status ∈ {OPTIMAL, TIME_LIMIT} **then**
    $x^\star \leftarrow$ valores de x em $\mathcal{M}$
    $tour \leftarrow$ BUILD_TOUR($x^\star$)
    $UB \leftarrow$ COST_TATSP($T, tour$)
    $LB \leftarrow$ objetivo de $\mathcal{M}$
    return ($tour$, UB, LB)
  else
    return null ▷ não convergiu / erro
  end if
end function

5 Análise dos Resultados Computacionais

Este capítulo detalha os resultados obtidos com as diferentes abordagens de otimização propostas. A análise inicia com uma avaliação global da consistência e da eficácia dos limites encontrados. Em seguida, vemos o desempenho de cada método para a obtenção de limites inferiores (LB) e superiores (UB). Por fim, o capítulo consolida as observações em conclusões estratégicas sobre a combinação mais eficiente de métodos e aponta direções para futuros trabalhos.

5.1 Organização dos Experimentos

Todos os algoritmos foram executados no ambiente computacional descrito na Seção 4.1. Para cada instância, foram calculados os seguintes limites, os quais os tempos de execução (em segundos) são apresentados nas colunas "t":

Para Limites Inferiores (LB):

- LB_{LP}: Valor da relaxação linear do modelo.
- LB_{Lag}: Valor obtido pela relaxação Lagrangeana via método do subgradiente.
- LB_{Col}: Valor dual do problema mestre ao término da geração de colunas.
- LB_{ILP}: Melhor limite dual (*best bound*) obtido pelo solver Gurobi ao resolver o modelo exato.
- GAP_{ILP}: Gap retornado ao fim da solução exata com Gurobi.
- $Nós\_B\&B_{ILP}$: Nós explorados no Branch and Bound interno do Gurobi.

Tabela 2: Limites inferiores por instância

Instância	LP LB	t	Lagrange LB	t	Colunas LB	t	ILP LB	t	ILP Gap(%)	ILP Nós B&B
grf1	91,41	8	91,38	806	114,34	28	107.48	1502	12,00	3895
grf2	34,55	5	34,50	1055	55,46	65	44.53	1166	0,00	9900
grf5	103,28	2	103,26	334	121,02	3	113.17	762	0,00	66465
grf6	47,42	1	47,30	264	54,24	3	49.93	11	0,00	547
grf19	34,54	8	34,49	1061	56,66	83	44.53	1306	0,00	9900
grf101	16,32	1	16,31	233	17,55	3	16.96	285	0,00	34611
grf102	19,9	2	19,89	536	21,83	5	20.65	1501	3,15	20591
grf103	19,9	2	19,9	629	21,81	8	20.56	1501	2,24	21250
grf104	19,9	2	19,9	629	21,81	8	25.73	1501	4,89	5519
grf105	23,52	3	23,52	1053	26,01	10	24.46	1501	3,35	12986
grf106	28,9	9	28,9	1055	32,18	44	29.94	1502	3,27	2161
grf107	33,43	14	33,42	1063	37,95	17	35.24	1502	5,05	184137
grf108	33,4	14	33,39	1061	38,24	35	38.28	1502	12,59	1609
grf109	33,4	13	33,39	1062	38,01	19	35.77	1022	6,45	1569
grf110	33,41	9	33,41	1055	37,16	122	38.95	1023	14,09	1659
grf111	37,9	23	37,8	813	43,21	205	45.33	1024	16,23	1147
grf112	45,11	45	45,00	810	51,24	535	55.41	1025	18,51	127
grf113	52,30	107	52,19	611	-	-	-	-	-	-
grf114	59,5	141	59,39	673	-	-	-	-	-	-
grf117	46,91	61	46,79	629	-	-	-	-	-	-
grf118	55,90	109	55,80	623	-	-	-	-	-	-

Para Limites Superiores (UB):

- UB_{LP}: Solução obtida por uma heurística de arredondamento da solução fracionária do LP.
- UB_{Lag}: Solução gerada por uma heurística guiada pelos multiplicadores Lagrangeanos.
- UB_{Col}: Melhor solução inteira (tour) encontrada durante o processo de geração de colunas.
- UB_{ILP}: Melhor solução inteira encontrada pelo solver Gurobi até o limite de tempo.
- GAP_{ILP}: Gap retornado ao fim da solução exata com Gurobi.
- $Nós\_B\&B_{ILP}$: Nós explorados no Branch and Bound interno do Gurobi.

Tabela 3: Limites superiores por instância

Instância	LP-Round UB	t	Heur. Lag. UB	t	Colunas UB	t	ILP UB	t	ILP Gap(%)	ILP Nós B&B
grf1	111,64	43	99,82	801	109,17	1526	114,34	1502	12,00	3895
grf2	48,98	45	46,14	1051	56,67	1574	45,41	1166	0,00	9900
grf5	125,79	23	118,39	466	121,02	340	117,39	762	0,00	66465
grf6	56,82	9	52,57	109	54,24	267	50,65	11	0,00	547
grf19	48,98	66	45,3	1051	59,12	1573	45,41	1306	0,00	9900
grf101	17,69	4	17,26	60	17,55	227	17,84	285	0,00	34611
grf102	21,45	2	20,81	536	21,84	537	21,06	1501	3,15	20591
grf103	21,81	1	21,01	993	21,82	441	21,96	1501	2,24	21250
grf104	27,43	7	25,6	531	27,25	1515	26,89	1501	4,89	5519
grf105	25,42	12	24,96	1051	25,82	1369	26,61	1501	3,35	12986
grf106	31,51	27	31,10	1051	31,92	1544	32,68	1502	3,27	2161
grf107	37,83	4	36,24	1051	38,17	1523	37,24	1502	5,05	184137
grf108	37,13	27	35,65	1051	38,16	1546	38,14	1502	12,59	1609
grf109	37,48	28	36,03	1051	38,35	1021	39,31	1022	6,45	1569
grf110	38,64	2	35,72	1051	38,41	1021	37,52	1023	14,09	1659
grf111	44,19	40	40,73	801	43,54	1021	43,10	1024	16,23	1147
grf112	51,19	74	48,02	803	51,24	1021	51,02	1025	18,51	127
grf113	59,08	14	55,45	610	-	-	-	-	-	-
grf114	67,44	138	63,55	652	-	-	-	-	-	-
grf117	53,23	16	49,90	602	-	-	-	-	-	-
grf118	63,88	9	60,22	632	-	-	-	-	-	-

5.2 Coerência entre limites inferiores e superiores

O primeiro passo na validação dos resultados é verificar a coerência entre os melhores limites encontrados. Para cada instância, definimos o melhor limite inferior, $LB^{\max}$, e o melhor limite superior, $UB^{\min}$, como:

$$LB^{\max} = \max\{LB_{LP}, LB_{Lag}, LB_{Col}, LB_{ILP}\}$$

$$UB^{\min} = \min\{UB_{LP}, UB_{Lag}, UB_{Col}, UB_{ILP}\}$$

Validação de Consistência A validação do método foi realizada com base na relação fundamental de que o limite inferior ($LB^{\max}$) não pode exceder o limite superior ($UB^{\min}$). Contudo, foram observadas inconsistências nessa relação para algumas instâncias.

Nesses casos, o limite inferior calculado superou o melhor limite superior conhecido. Um exemplo notório é a instância `grf2`, com o resultado $55.46 > 45.41$. A principal suspeita para essa anomalia é uma falha de implementação no método de geração de colunas, a qual não pôde ser investigada por limitações de tempo.

Como um fator adicional, notou-se que, para algumas dessas mesmas instâncias, o próprio limite superior de referência não era garantidamente ótimo, pois o modelo ILP que o produziu encerrou a execução com um *gap* de otimalidade. Isso significa que o valor de $UB^{\min}$ poderia ser ainda menor.

5.3 Desempenho dos Métodos para Limites Inferiores (LB)

- **Relaxação Linear (LP):** O LB_{LP} serve como um *baseline*, sendo calculado rapidamente. No entanto, sua qualidade é relativamente baixa, apresentando limites inferiores significativamente abaixo do melhor UB conhecido, o que evidencia a fragilidade da relaxação linear padrão para este problema.

- **Relaxação Lagrangeana:** O LB_{Lag} proporciona apenas uma leve melhoria sobre o LB_{LP}, a um custo computacional consideravelmente maior.

- **Geração de Colunas:** A abordagem LB_{Col} resultou em limites inferiores elevados, porém identificou-se que tais valores foram obtidos devido a uma falha na implementação. Isso ficou claro ao se observar que os limites inferiores estavam acima dos limites superiores encontrados pelo método exato com Gurobi.

- **Limite do Solver (ILP):** O LB_{ILP} representa o melhor limite dual fornecido pelo solver Gurobi dentro do tempo limite. Nos casos em que a otimalidade não foi comprovada, esse limite atuou como benchmark.

5.4 Desempenho dos Métodos para Limites Superiores (UB)

- **Arredondamento da LP (UB_{LP}):** Esta heurística apresentou o menor tempo de execução, porém entregou soluções com qualidade moderada, frequentemente distantes do melhor limite superior conhecido.

- **Heurística Lagrangeana (UB_{Lag}):** Esta foi a estratégia com o melhor equilíbrio entre tempo de execução e qualidade da solução. Guiada pelos multiplicadores Lagrangeanos, gerou consistentemente soluções próximas às obtidas pelo método exato.

- **Tour Inteiro da Geração de Colunas (UB_{Col}):** Este método não produziu o melhor limite superior em nenhuma das instâncias avaliadas, provavelmente devido a problemas na implementação.

- **Solver ILP (Gurobi):** O método exato (UB_{ILP}) constituiu a referência principal em termos de qualidade, provando otimalidade para instâncias menores. Mesmo nas situações em que a solução ótima não foi encontrada dentro do limite de tempo, ele frequentemente obteve o melhor limite superior entre os métodos avaliados, demonstrando robustez apesar do alto custo computacional.

5.5 Quantidade de nós na árvore de Branch&Bound

A última coluna das Tabelas 2 e 3, referente ao número de nós explorados no processo de Branch and Bound (B&B), mostra informações importantes sobre a complexidade de cada instância e a eficácia dos limites inferior e superior empregados na resolução do TATSP. O número de nós B&B representa o esforço necessário do solver para podar o espaço de busca e alcançar (ou se aproximar de) a otimalidade.

Nas instâncias com 20 vértices, como `grf1`, `grf2`, `grf5`, `grf6` e `grf19`, é observado um comportamento variado. Algumas, como `grf6`, foram resolvidas com apenas 547 nós — um indicativo de bons limites e estrutura favorável. Por outro lado, `grf5` exigiu mais de 66 mil nós, evidenciando que, mesmo com um tamanho moderado, a instância pode apresentar estrutura difícil de ser explorada eficientemente, talvez devido à densidade de relações ou à qualidade dos limites fornecidos.

Instâncias pequenas, como `grf101` (com apenas 18 nós), também mostram comportamento imprevisível: foram necessários mais de 34 mil nós para resolver essa instância, reforçando que o número de vértices não é o único fator determinante da dificuldade. O mesmo ocorre em `grf102`–`grf104`, que apresentam entre 20 e 27 nós e ainda assim demandam dezenas de milhares de nós na árvore.

Ao analisarmos as instâncias maiores (acima de 30 vértices), destaca-se `grf107`, que exigiu mais de 184 mil nós para ser resolvida, o maior valor registrado entre todas as instâncias. Curiosamente, outras instâncias com exatamente o mesmo número de nós e arestas, como `grf108`–`grf110`, foram resolvidas com cerca de 1.600 nós, o que sugere variações significativas causadas possivelmente pela distribuição dos custos, topologia do grafo ou pela aleatoriedade das relações gatilho. Isso evidencia que, além do tamanho, a estrutura combinatória da instância impacta diretamente na dificuldade da resolução.

Finalmente, nas maiores instâncias testadas, como `grf111` e `grf112`, o número de nós B&B não ultrapassou 1.200, apesar dos gaps finais ainda elevados (16–18%). Esse padrão sugere que, mesmo sem explorar extensivamente a árvore de busca, o solver foi capaz de encontrar boas soluções factíveis rapidamente, mas não pôde provar otimalidade devido a limites inferiores ainda distantes do valor ótimo.

Em resumo, os resultados confirmam que:

- O número de vértices não é um preditor confiável da dificuldade.
- A qualidade dos limites inferiores e superiores influencia diretamente a eficiência do Branch and Bound.
- A estrutura interna da instância (topologia e relações gatilho) é um fator crítico no tamanho da árvore de busca.
- Instâncias com gaps altos e poucos nós B&B, como `grf112`, indicam a necessidade de melhorias nos LBs, enquanto instâncias com muitos nós, como `grf107`, sugerem ineficiência na poda da árvore.

5.6 Trajetórias de convergência (caso-exemplo `grf104`)

Para ilustrar o comportamento dinâmico dos algoritmos, as Figuras 1–3 mostram a evolução dos valores de LB/UB ao longo das iterações na instância `grf104`.

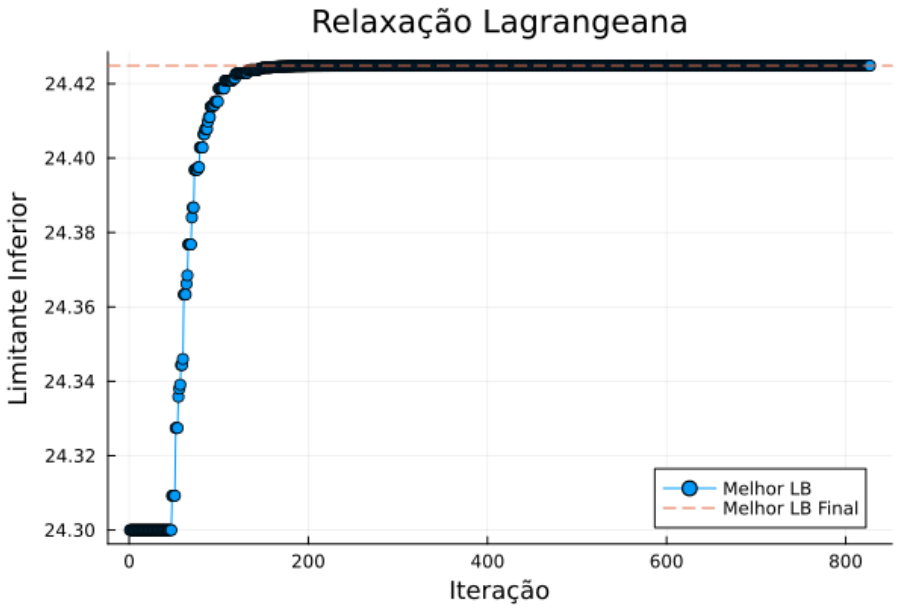

Figura 1: Relaxação Lagrangeana — evolução do limite inferior.

Observações A curva da relaxação Lagrangeana mostra uma ascensão rápida nas 50 primeiras iterações, subindo de aproximadamente 24,30 até estabilizar em 24,42. A partir desse ponto, as melhorias são marginais, com o algoritmo seguindo até o critério de parada sem grandes avanços. O comportamento reflete uma boa calibragem inicial dos multiplicadores de Lagrange.

Figura 2: Heurística de arredondamento da LP — melhor custo ao longo do tempo.

Observações A heurística de arredondamento encontra uma solução inicial com custo 27,43, que permanece inalterada durante toda a execução (cerca de 7 segundos). Isso evidencia a limitação do método: apesar de ser extremamente rápido, apresenta baixa capacidade de refino e adaptação.

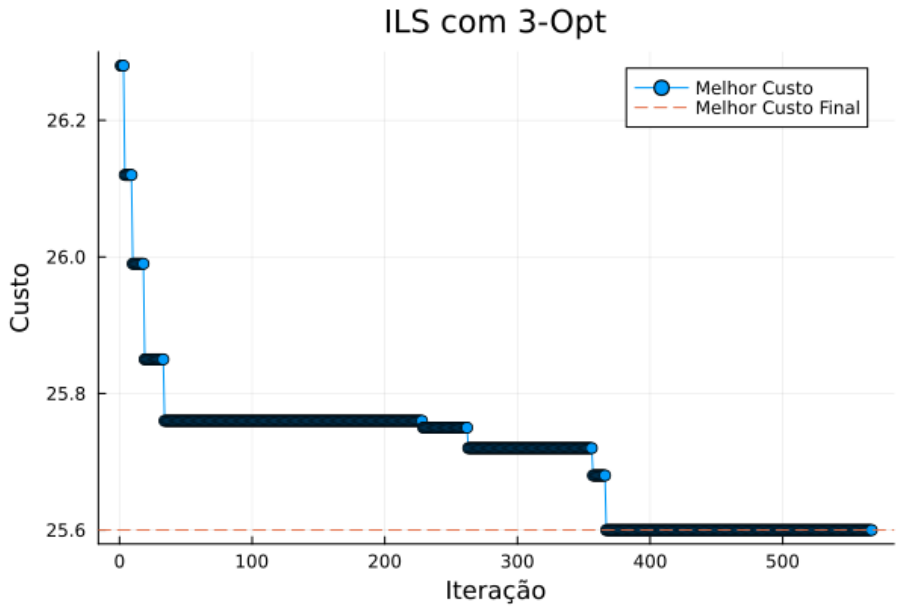

Figura 3: ILS+3-Opt — evolução do custo.

Observações O ILS parte de uma solução inicial de custo 26,24, gerada pela construção gulosa guiada. Logo nas primeiras iterações, melhora para 25,80 e segue com pequenas quedas graduais até atingir 25,60 na iteração 380. A partir daí, o custo permanece constante até o final. O gráfico mostra um comportamento típico de buscas locais intensas: grande melhora inicial, seguida de pequenos refinamentos e eventual estagnação em um ótimo local profundo.

Esse padrão de ganho inicial rápido seguido de platô também aparece em outras instâncias, reforçando as características gerais das abordagens.

6 Conclusão

Este trabalho investigou a variante *Trigger Arc Traveling Salesman Problem* (TATSP) por meio de diversas abordagens, incluindo modelagem exata, relaxações, geração de colunas e heurísticas. As análises empíricas realizadas em diferentes instâncias evidenciaram os seguintes aspectos principais:

- **Relaxação Linear (LP) como linha de base**: Embora rápida, essa abordagem forneceu limites inferiores relativamente fracos. Contudo, serviu como uma importante referência inicial e base para métodos heurísticos de arredondamento.
- **Relaxação Lagrangeana**: Essa técnica proporcionou apenas pequenas melhorias em relação à LP, demandando maior esforço computacional. Ainda assim, destacou-se pela capacidade de gerar multiplicadores que auxiliaram significativamente na obtenção de soluções inteiras de alta qualidade, posicionando-se como a melhor alternativa entre as técnicas heurísticas para limites superiores.
- **Geração de colunas**: Problemas de implementação impactaram negativamente o desempenho desse método, resultando em limites inferiores inconsistentes e superiores aos valores conhecidos como corretos.
- **Limites superiores**: A heurística guiada pela relaxação Lagrangeana destacou-se claramente como a melhor técnica prática, produzindo soluções consistentemente próximas aos valores obtidos pelo método exato. O solver ILP (Gurobi) confirmou sua posição como

método de referência, garantindo ótimos comprovados para instâncias menores e obtendo bons resultados mesmo em cenários mais complexos.

Contribuições

1. Implementação abrangente e detalhada do modelo inteiro do TATSP usando JuMP, incluindo variáveis específicas e restrições relacionadas a *trigger arcs*.

2. Desenvolvimento e avaliação de três diferentes relaxações (LP, Lagrangeana, geração de colunas dual) e duas heurísticas principais (arredondamento LP e heurística lagrangeana).

Limitações

- Problemas identificados na implementação da geração de colunas prejudicaram a validação plena dos resultados obtidos por esse método.

- As buscas locais realizadas (3-Opt e ILS) apresentaram limitações ao atingir rapidamente pontos de estagnação, indicando necessidade de mecanismos adicionais para exploração e diversificação das soluções.

- As restrições relacionadas ao tempo disponível de execução limitaram o desempenho do método exato do solver Gurobi, especialmente para instâncias maiores, deixando parte dos resultados dependentes dos limites estabelecidos pelo solver.

Em resumo, o estudo demonstrou que uma integração bem articulada entre técnicas exatas e heurísticas é eficaz para enfrentar o TATSP em instâncias de porte moderado.

Referências

Referências

BERTSIMAS, D.; VOHRA, R. V. Rounding algorithms for covering problems. *Mathematical Programming*, Springer, v. 80, n. 1, p. 63–89, 1998.

CASTELLUCCI, P. B. An iterated local search for the travelling salesman problem. In: *Anais do XIV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*. São Paulo, SP, Brasil: Sociedade Brasileira de Computação (SBC), 2018. p. 499–510.

CERRONE, C. et al. The trigger arc tsp: Optimise the picking process in warehouses with compactable storage systems. *Working Paper*, 2025. Preprint.

HELD, M.; KARP, R. M. The traveling-salesman problem and minimum spanning trees. *Operations Research*, INFORMS, v. 18, n. 6, p. 1138–1162, 1970.

MARGARIDA, P. M. *Relaxação Lagrangeana*. 2009. Material de aula online. Disponível em: <https://www.ime.unicamp.br/~chico/mt852/slidesrellag.pdf>. Acessado em: 16 de julho de 2025.

ROCKI, K.; SUDA, R. Accelerating 2-opt and 3-opt local search using gpu in the travelling salesman problem. In: . [S.l.: s.n.], 2012.