Curso Superior em Engenharia de Computação
Departamento de Computação

Otimização Topológica da Malha Ferroviária Brasileira: Uma Abordagem via Meta-heurísticas Híbridas

José Marconi de Almeida Júnior

Divinópolis
2025

1 Introdução

1.1 Introdução Textual do Problema

O sistema de transporte de cargas no Brasil enfrenta um desequilíbrio estrutural histórico, caracterizado por uma dependência excessiva do modal rodoviário, que concentra cerca de 65% da movimentação nacional, conforme dados da Confederação Nacional do Transporte [1]. Essa matriz logística resulta em custos elevados de frete, maior impacto ambiental e gargalos no escoamento de produção, especialmente considerando as dimensões continentais do país. A expansão e a otimização da malha ferroviária apresentam-se como soluções estratégicas para aumentar a competitividade nacional e promover a integração regional.

O problema central abordado neste trabalho é o *Network Design Problem* (Problema de Projeto de Redes) aplicado à conexão das 27 capitais brasileiras. O objetivo é projetar uma topologia de rede que minimize os custos de implantação (construção de trilhos) sem comprometer a eficiência logística (tempo médio de viagem entre quaisquer duas capitais).

1.2 Formulação Matemática

O problema pode ser modelado como um grafo não direcionado $G = (V, E)$, onde $V = \{1, ..., 27\}$ representa o conjunto de capitais e E o conjunto de possíveis conexões ferroviárias. Seja x_{ij} uma variável binária que assume valor 1 se a aresta (i, j) é construída e 0 caso contrário. O objetivo é minimizar uma função bicritério $F(x)$, sujeita à restrição de que o grafo permaneça conexo:

$$MinF(x) = \alpha \cdot \sum_{(i,j)\in E} d_{ij}x_{ij} + \beta \cdot \mu(G_x) \tag{1}$$

Onde d_{ij} é a distância física (custo de construção) entre as cidades i e j, e $\mu(G_x)$ é a média das distâncias mínimas (caminho mais curto) entre todos os pares de vértices no grafo construído, calculada via algoritmo de Floyd-Warshall. Os parâmetros α e β ponderam a importância relativa entre economizar na obra (α) e agilizar o transporte (β).

1.3 Revisão Bibliográfica

O problema de projeto de redes (*Network Design*) é classificado como NP-Difícil na literatura clássica, conforme estabelecido por Magnanti e Wong [2], o que justifica o uso de métodos heurísticos para instâncias complexas. Trabalhos recentes, como o de Almeida Júnior [3], propuseram o uso de Árvores Geradoras Mínimas (MST) baseadas no algoritmo de Kruskal [4]. Embora a MST garanta o menor custo de construção possível, ela frequentemente resulta em topologias lineares ineficientes do ponto de vista logístico.

1.4 Justificativa

A justificativa deste estudo reside na necessidade de ferramentas computacionais que apoiem a tomada de decisão em políticas públicas de infraestrutura. A aplicação de técnicas de Pesquisa Operacional em dados georreferenciados reais, obtidos do IBGE [5], permite simular cenários de investimento onde o equilíbrio entre custo inicial e eficiência operacional é crítico.

1.5 Contribuição

A principal contribuição deste trabalho é a aplicação e comparação de meta-heurísticas avançadas — *Iterated Local Search* (ILS), *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Variable Neighborhood Search* (VNS) — para refinar a solução inicial baseada em MST, propondo uma malha que equilibra custo e eficiência através da criação estratégica de ciclos.

1.6 Organização do Trabalho

O trabalho está organizado da seguinte forma: a Seção 2 apresenta uma revisão bibliográfica mais aprofundada; a Seção 3 detalha a metodologia computacional e os algoritmos; a Seção 4 apresenta os experimentos computacionais e a análise visual dos resultados; e a Seção 5 conclui o estudo.

2 Revisão Bibliográfica

A otimização de redes de transporte é um campo vasto. Magnanti e Wong [2] fornecem a base teórica para a modelagem de *Network Design*, destacando a complexidade combinatória de selecionar arestas em grafos densos.

No contexto de heurísticas construtivas, o algoritmo de Kruskal [4] é seminal para a obtenção de Árvores Geradoras Mínimas (MST). Entretanto, para problemas que exigem redundância e eficiência de fluxo, a MST é insuficiente. Nesse cenário, meta-heurísticas de busca local tornam-se essenciais. Lourenço et al. [6] formalizaram o *Iterated Local Search* (ILS) como um método robusto que alterna entre busca local e perturbação. Paralelamente, Hansen e Mladenović [7] desenvolveram o *Variable Neighborhood Search* (VNS), que explora sistematicamente mudanças de vizinhança para escapar de ótimos locais. Este trabalho integra essas abordagens clássicas ao problema específico da malha brasileira.

3 Metodologia

3.1 Explicação do Código Principal

A metodologia computacional foi desenvolvida em linguagem C++, estruturada sob o paradigma de Orientação a Objetos. A classe `RedeLogistica` encapsula a lógica do grafo, mantendo métodos distintos para construção, perturbação e busca local, garantindo modulari-

dade e fácil manutenção.

3.2 Representação da Solução

A solução é representada por uma matriz de adjacência binária 27×27. Esta estrutura permite acesso constante $O(1)$ para verificação de arestas e facilita a implementação de operações de inversão de bits (*bit-flip*).

3.3 Função de Avaliação

A função de avaliação (fitness) verifica primeiramente a conectividade do grafo via Busca em Largura (BFS). Se o grafo for desconexo, aplica-se uma penalidade infinita. Para soluções conexas, calcula-se o custo bicritério conforme a Equação 1. A eficiência logística $\mu(G_x)$ é obtida através do algoritmo de Floyd-Warshall ($O(N^3)$), recalculado a cada nova solução proposta para garantir precisão nas distâncias reais de viagem.

3.4 Construção

Foram implementados dois métodos para gerar soluções iniciais:

- **MST Determinística (Baseline):** Utiliza o algoritmo de Prim para gerar a árvore de custo mínimo absoluto, servindo como ponto de partida padrão.

- **Construção GRASP:** Utiliza uma Lista de Candidatos Restrita (RCL) baseada na cardinalidade ($\alpha = 0.2$). Isso permite a geração de diversas árvores iniciais de boa qualidade, introduzindo diversidade estocástica no processo de busca.

3.5 Movimentos e Busca Local

O operador de movimento (vizinhança) definido foi o *Bit Flip* de aresta: a cada passo, o algoritmo seleciona aleatoriamente um par de cidades e inverte o estado da conexão (constrói se não existir, remove se existir). O motor de refinamento é o *Hill Climbing* (Subida de Encosta, algoritmo 1) estocástico, que aceita apenas movimentos que melhorem estritamente a função objetivo.

Algoritmo 1 Busca Local: Hill Climbing Estocástico

Entrada: Solução Inicial S, Iterações $MaxIter$
Saída: Solução Otimizada S

```
1: CustoAtual ← CalcularCusto(S)
2: for i ← 0 to MaxIter do
3:     u, v ← EscolherParAleatorio()
4:     S' ← InverterAresta(S, u, v)                ▷ Movimento Bit-Flip
5:     NovoCusto ← CalcularCusto(S')
6:     if NovoCusto < CustoAtual then
7:         S ← S'
8:         CustoAtual ← NovoCusto
9:     else
10:        DesfazerMovimento(S, u, v)
11:    end if
12: end for
```

3.6 Meta-heurísticas

- **Iterated Local Search (ILS) - Algoritmo 2:** Aplica perturbações aleatórias (modificação forçada de $k = 3$ arestas) na melhor solução corrente e reinicia a busca local, iterando por um número fixo de ciclos.

Algoritmo 2 Meta-heurística ILS (Iterated Local Search)

Entrada: Instância do Problema
Saída: Melhor Solução Global S^*

```
1: S ← GerarSolucaoMST()
2: S* ← S
3: for ciclo ← 0 to MaxCiclos do
4:     S ← HillClimbing(S)                         ▷ Busca Local
5:     if Custo(S) < Custo(S*) then
6:         S* ← S
7:     end if
8:     S ← Perturbar(S*, força=3)                  ▷ Modifica 3 arestas aleatórias
9: end for
```

- **Variable Neighborhood Search (VNS) - Algoritmo 3:** Alterna sistematicamente entre vizinhanças de magnitudes crescentes ($k = 1, 2, 3$) para ”agitar”a solução apenas quando a busca local estagna.

Algoritmo 3 Meta-heurística VNS (Variable Neighborhood Search)

Entrada: Solução Inicial S, Vizinhaças k_{max}, Iterações $MaxIter$
Saída: Melhor Solução Global S

$S \leftarrow$ GerarSolucaoMST()
for $iter \leftarrow 0$ **to** $MaxIter$ **do**
    $k \leftarrow 1$
    while $k \leq k_{max}$ **do**
        ▷ **1. Agitação (Shake)**
        $S' \leftarrow$ Perturbar(S, força=k) ▷ Modifica k arestas
        ▷ **2. Busca Local**
        $S'' \leftarrow$ HillClimbing(S')
        ▷ **3. Mudança de Vizinhança**
        if $Custo(S'') < Custo(S)$ **then**
            $S \leftarrow S''$ ▷ Melhorou: Centro da busca muda
            $k \leftarrow 1$ ▷ Reinicia vizinhança
        else
            $k \leftarrow k + 1$ ▷ Não melhorou: Aumenta a perturbação
        end if
    end while
end for

- **GRASP(Algoritmo 4):** Executa múltiplas iterações independentes de construção aleatorizada seguidas de busca local, explorando diferentes bacias de atração.

Algoritmo 4 Meta-heurística GRASP (Greedy Randomized Adaptive Search Procedure)

Entrada: Parâmetro α (fator de aleatoriedade), Iterações $MaxIter$
Saída: Melhor Solução Global S^*

$S^* \leftarrow \emptyset$
$Custo(S^*) \leftarrow \infty$
for $i \leftarrow 0$ **to** $MaxIter$ **do**
    ▷ **Fase 1: Construção Gulosa Aleatorizada**
    $S \leftarrow \emptyset$
    while S não é uma solução completa **do**
        $C \leftarrow$ Lista de todas as arestas candidatas viáveis
        $C_{restrita} \leftarrow$ Selecionar as melhores $\alpha\%$ de C (RCL)
        $e \leftarrow$ Escolher aleatoriamente uma aresta de $C_{restrita}$
        $S \leftarrow S \cup \{e\}$
    end while
    ▷ **Fase 2: Busca Local**
    $S \leftarrow$ HillClimbing(S)
    if $Custo(S) < Custo(S^*)$ **then**
        $S^* \leftarrow S$
    end if
end for

4 Experimentos Computacionais e Resultados

4.1 Introdução

Os algoritmos foram implementados em C++ e compilados via GCC utilizando o prompt de comando do WSL dentro do VSCode. Os testes foram realizados em um computador equipado com processador Intel Core i7 de oitava geração com CPU de 1.80GHz, 16GB de RAM, em ambiente Windows, mas utilizando WSL. O código fonte e os dados gerados foram disponibilizados para reprodutibilidade.

4.2 Instâncias

A instância utilizada compreende as coordenadas geográficas (latitude/longitude) das 27 capitais brasileiras. .Os dados de posição e extensão territorial foram extraídos do Anuário Estatístico do Brasil [5]. A Tabela 1 detalha as coordenadas exatas (latitude e longitude) de cada capital utilizadas para o cálculo da matriz de distâncias euclidianas."

Tabela 1 – Coordenadas das Capitais Brasileiras utilizadas nos Experimentos (Fonte: Dados do experimento baseados no IBGE)

UF	Lat	Lon	UF	Lat	Lon
AC	-9.97	-67.81	PB	-7.12	-34.86
AL	-9.64	-35.73	PR	-25.44	-49.27
AM	-3.11	-60.01	PE	-8.06	-34.88
AP	0.02	-51.06	PI	-5.06	-42.80
BA	-12.96	-38.50	RJ	-22.90	-43.22
CE	-3.75	-38.53	RN	-5.81	-35.20
DF	-15.79	-47.88	RO	-8.76	-63.89
ES	-20.29	-40.30	RS	-30.05	-51.18
GO	-16.68	-49.24	RR	2.82	-60.67
MA	-2.55	-44.25	SC	-27.59	-48.57
MT	-15.60	-56.09	SE	-10.93	-37.07
MS	-20.46	-54.62	SP	-23.56	-46.64
MG	-19.91	-43.93	TO	-10.24	-48.32
PA	-1.46	-48.47			

4.3 Calibração do Algoritmo

Para a função objetivo, definiram-se os pesos $\alpha = 1.0$ e $\beta = 500.0$. O peso elevado para a eficiência (β) força o algoritmo a criar "atalhos"caros apenas se eles trouxerem um ganho logístico global expressivo. As meta-heurísticas foram configuradas com 30 ciclos externos e 500 iterações de busca local interna.

4.4 Resultados Quantitativos

A Tabela 2 apresenta o desempenho comparativo. O "Gap"indica a melhoria percentual em relação à solução inicial (MST).

Tabela 2 – Comparativo Final de Desempenho (Custo vs. Tempo)

Método	Custo Final	Tempo (s)	Gap (%)
MST (Inicial)	14.394,06	0,000	0,00%
HC Puro	8.983,03	3,109	37,59%
GRASP	9.268,80	4,765	35,61%
VNS	8.957,30	16,614	37,77%
ILS	**8.947,07**	**4,635**	**37,84%**

4.5 Análise Visual das Topologias

A seguir, são apresentadas as topologias geradas por cada método.

A Figura 1 exibe a solução inicial (MST). Nota-se uma estrutura esparsa e linear, obrigando grandes desvios para conectar regiões distantes (ex: Sul ao Nordeste).

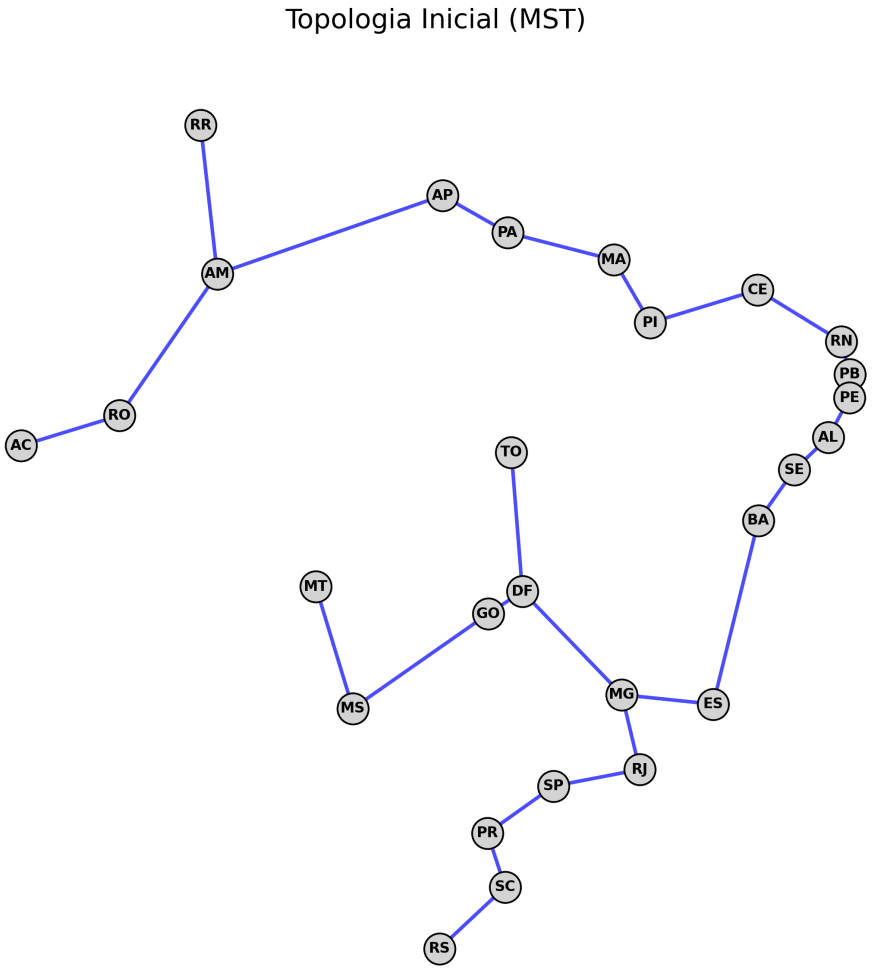

Figura 1 – Topologia Inicial (MST) - Custo: 14.394,06

A Figura 2 mostra o resultado do Hill Climbing Puro. O método simples já foi capaz de criar ciclos importantes, reduzindo o custo em 37,59%.

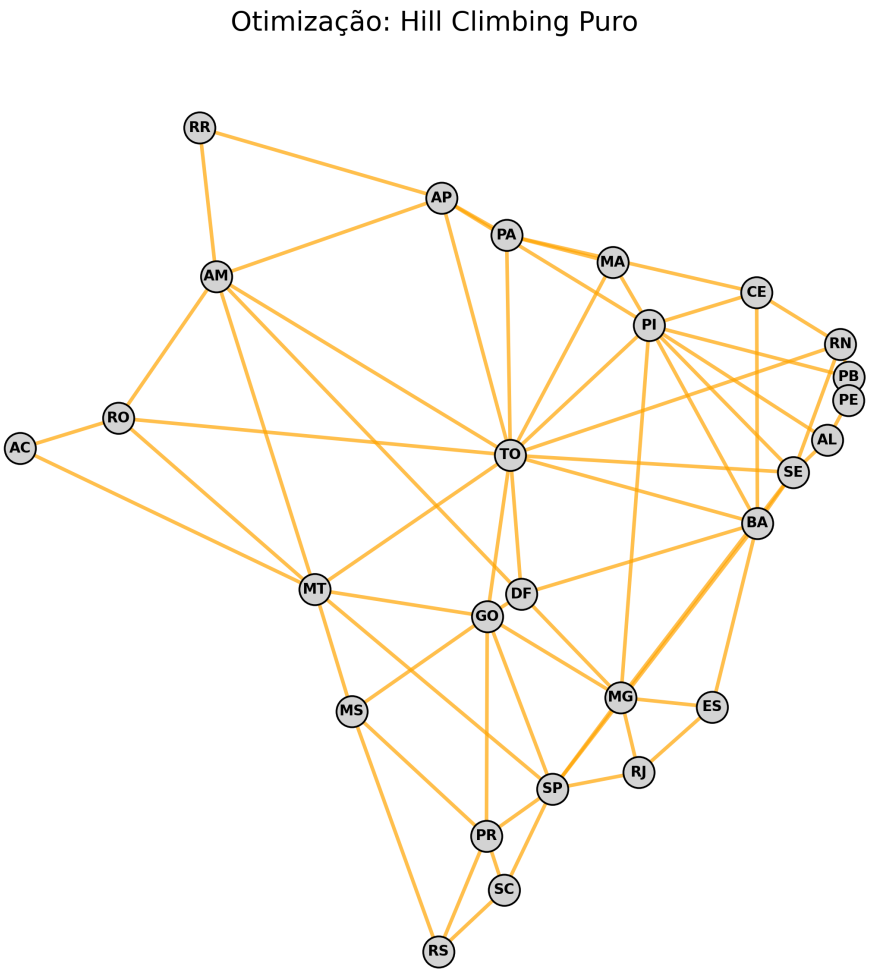

Figura 2 – Otimização via Hill Climbing Puro

A Figura 3 apresenta o resultado do GRASP. Apesar da diversificação na construção, o método convergiu para uma solução ligeiramente inferior (Gap de 35,61%) nesta instância específica.

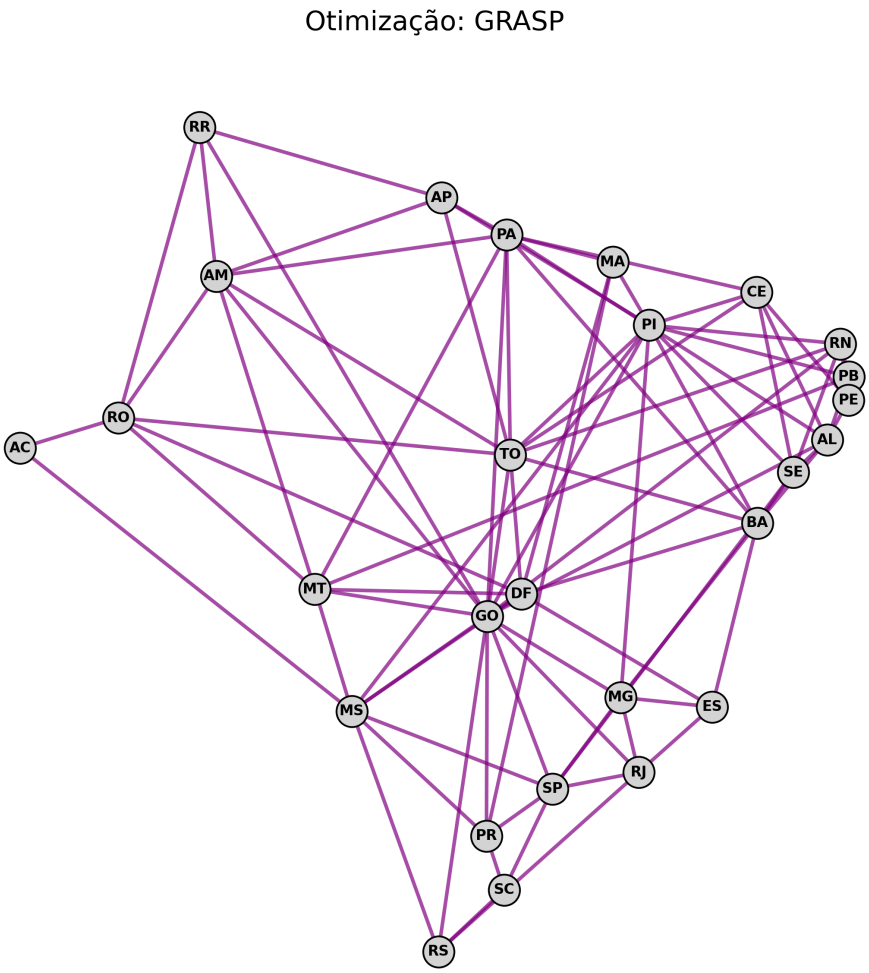

Figura 3 – Otimização via GRASP

A Figura 4 exibe a solução do VNS. O método obteve um resultado excelente (Gap de 37,77%), visualmente muito denso e eficiente, porém com o maior tempo computacional (16,6s).

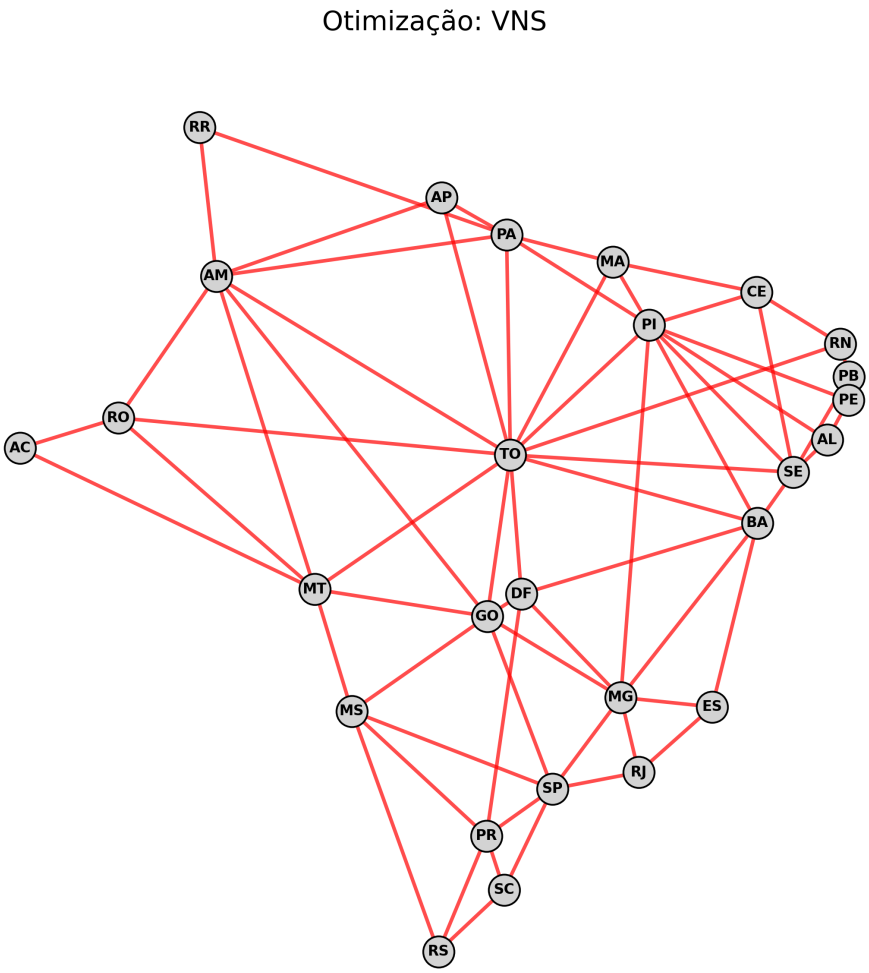

Figura 4 – Otimização via VNS

Por fim, a Figura 5 apresenta a solução do ILS, que obteve o melhor desempenho global (Gap de 37,84%). A topologia equilibra perfeitamente a densidade de conexões com o custo de implantação, criando atalhos estratégicos entre o Centro-Oeste e as demais regiões.

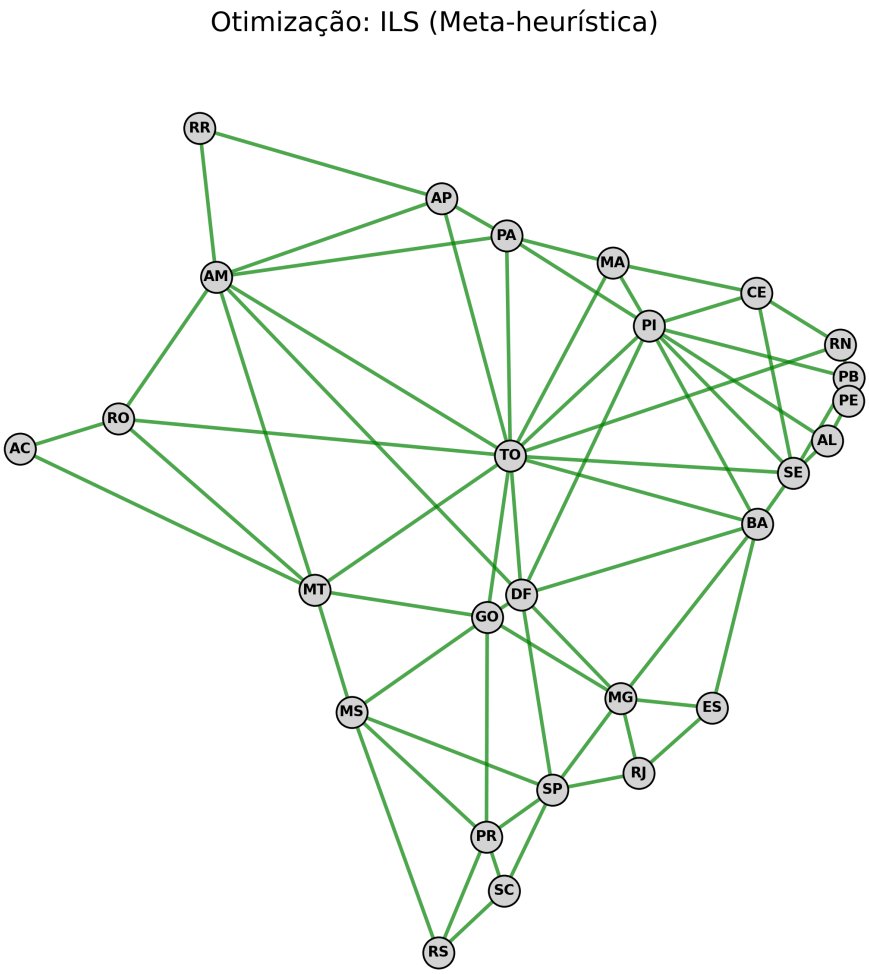

Figura 5 – Solução Otimizada via ILS (Melhor Resultado)

5 Conclusão

Este trabalho retomou o problema de otimização da malha ferroviária nacional através de uma abordagem bicritério, visando corrigir as ineficiências logísticas apontadas em soluções baseadas puramente em Árvores Geradoras Mínimas (MST).

A metodologia empregada comparou métodos construtivos e meta-heurísticas de refinamento. Os resultados demonstraram que o método ILS foi o mais eficiente, proporcionando uma redução de 37,84% na função de custo em relação à baseline, com um tempo computacional competitivo de 4,6 segundos. O VNS obteve resultados muito próximos, mas com um custo computacional significativamente maior.

Como trabalhos futuros, sugere-se a inclusão de restrições geográficas reais (rios, relevo) na matriz de custos e a aplicação de algoritmos populacionais para comparar com as abordagens de trajetória única aqui apresentadas.

Referências

1 Confederação Nacional do Transporte. *Plano CNT de Transporte e Logística*. 2023. Acesso em: 25 nov. 2025. Disponível em: ⟨https://www.cnt.org.br⟩.

2 MAGNANTI, T. L.; WONG, R. T. Network design and transportation planning: Models and algorithms. *Transportation Science*, v. 18, n. 1, p. 1–55, 1984.

3 JúNIOR, J. M. A. Integrando mst no planejamento de redes ferroviárias/metroviárias. *Anais de Engenharia de Computação - CEFET-MG*, p. 1–6, 2023.

4 KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, v. 7, n. 1, p. 48–50, 1956.

5 Instituto Brasileiro de Geografia e Estatística. *Anuário Estatístico do Brasil: Posição e Extensão*. Rio de Janeiro: IBGE, 2023. v. 83.

6 LOURENçO, H. R.; MARTIN, O. C.; STüTZLE, T. Iterated local search. In: *Handbook of Metaheuristics*. [S.l.]: Springer, 2003. p. 320–353.

7 HANSEN, P.; MLADENOVIć, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467, 2001.