Trabalho Final CAL
Otimização de Rotas com Heurística

Vinícus Hansen, Monique H. Mendes, Marco R. Rigon

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

Abstract. *This paper presents a heuristic approach to the route optimization problem involving multiple mandatory stops between cities in southern Brazil, using the A* algorithm with practical distance constraints. To address the problem's combinatorial complexity, we implemented optimizations such as geographical filtering, route cost memorization, and a heuristic based on the Cheapest Insertion method. Experimental results show that these strategies enable efficient processing of significantly larger instances using real-world public road distance data. The findings highlight the effectiveness of heuristic techniques in solving complex optimization problems with real-world applicability.*

Resumo. *Este trabalho propõe uma solução heurística para o problema de roteamento com múltiplas paradas obrigatórias entre cidades da região Sul do Brasil, utilizando o algoritmo A* com restrições práticas de distância. Para contornar a complexidade combinatória do problema, foram implementadas otimizações como limitação geográfica, memorização de rotas e uma heurística baseada no método Cheapest Insertion. Os testes demonstraram que essas abordagens permitem lidar com instâncias maiores em tempos de execução viáveis, utilizando dados reais de distâncias rodoviárias públicas. Os resultados evidenciam o potencial de heurísticas computacionais em problemas de otimização com aplicabilidade prática.*

1. Introdução

Problemas de roteamento em transporte são desafiadores, especialmente com restrições como paradas obrigatórias. Neste trabalho, aplicamos o algoritmo A* para calcular os trajetos entre cidades da região Sul do Brasil, combinando-o com a heurística Cheapest Insertion para traçar a rota mais eficiente considerando várias paradas obrigatórias.

2. Definição do problema

O problema abordado neste trabalho consiste em encontrar a rota mais eficiente entre duas cidades distintas da região Sul do Brasil — abrangendo os estados do Paraná (PR), Santa Catarina (SC) e Rio Grande do Sul (RS) — considerando a necessidade de passagem por um conjunto fixo de cidades intermediárias (paradas obrigatórias). A definição de "mais eficiente" neste contexto refere-se à menor distância total percorrida entre o ponto de origem e o destino final, respeitando a ordem livre de visita às cidades obrigatórias.

A rede rodoviária é representada por um grafo completo e ponderado, onde:

- Os vértices representam cidades reais da região Sul.

- As arestas são ponderadas pela distância rodoviária entre os pares de cidades, obtida a partir de dados públicos.

A principal variável real considerada neste problema é a distância rodoviária entre municípios, que reflete a malha viária atual e suas limitações geográficas e logísticas. O dataset utilizado é o disponibilizado por [4], que contém a matriz de distâncias rodoviárias entre cidades brasileiras. Esse conjunto de dados permite construir um grafo completo realista, com base em informações oficiais, servindo de base para a execução dos algoritmos de busca. Em termos de complexidade, o problema se aproxima de uma generalização do Caixeiro Viajante com paradas obrigatórias, para o qual não se conhece solução em tempo polinomial. Assim, faz-se necessária a utilização de heurísticas, como o algoritmo A*, que pode explorar de forma eficiente o espaço de busca, guiado por uma função heurística admissível que estima a distância restante ao objetivo.

3. Preparando os Dados

Para possibilitar a construção do grafo completo e ponderado utilizado neste trabalho, foi necessário realizar um pré-processamento dos dados brutos, integrando diferentes fontes públicas. As etapas a seguir descrevem esse processo:

1. **Download do dataset de distâncias rodoviárias**: O ponto de partida foi o dataset disponibilizado por [4], que contém a distância rodoviária entre todos os municípios brasileiros.
2. **Filtragem por região geográfica**: Como o escopo do trabalho está restrito à região Sul do Brasil, foi necessário filtrar apenas os municípios dos estados do Paraná, Santa Catarina e Rio Grande do Sul. Para isso, utilizamos os dados do Instituto Brasileiro de Geografia e Estatística (IBGE) [1], que fornecem os códigos oficiais de cada município brasileiro, organizados por estado.
3. **Mapeamento dos códigos para nomes de cidades**: Após selecionar os municípios da região Sul com base nos seus respectivos códigos, realizamos a tradução desses códigos para os nomes correspondentes das cidades, também com base nos dados do IBGE [1].

   O Dataset resultante foi disponibilizado no Kaggle [3].

4. Solução Proposta

Utilizamos o algoritmo A* com heurística nula (equivalente ao Dijkstra), adaptado para limitar transições a distâncias máximas de 400 km, simulando a autonomia de veículos e reduzindo o espaço de busca. O algoritmo recebe como entrada um grafo ponderado da rede rodoviária, além dos nós origem e destino. A busca é conduzida por uma fila de prioridade que seleciona o próximo nó com menor custo acumulado, enquanto um conjunto de nós visitados evita ciclos e redundâncias. Essa fila gerencia os nós a serem explorados para cada par de cidades, enquanto o tratamento das paradas obrigatórias atua em um nível superior, organizando a sequência dos pares a serem percorridos e permitindo o roteamento eficiente em trajetos com múltiplas paradas. Inicialmente, todas as permutações das paradas eram testadas, o que causava crescimento fatorial do tempo. Para superar isso, implementamos a memorização de custos para evitar recomputações redundantes, seguida da heurística Cheapest Insertion, que insere paradas otimizando o custo incremental, permitindo lidar com mais paradas eficientemente. A implementação pode ser encontrada em [2]

5. Testes Realizados e Resultados

Para avaliar o desempenho das diferentes versões do algoritmo proposto, foram conduzidos experimentos com conjuntos de testes compostos por N paradas obrigatórias. A lista de paradas foi previamente definida, conforme N aumenta, são selecionados mais elementos da lista de forma que os elementos utilizados anteriormente são mantidos.

A Figura 1 ilustra uma execução do algoritmo com 7 paradas obrigatórias. As cidades destacadas em amarelo representam pontos que, embora não sejam obrigatórios, foram incluídos por comporem um caminho de menor custo total.

```
--- 7 parada(s) obrigatória(s) ---
Custo total: 849468
Caminho: Novo Hamburgo -> Dois Irmãos -> Morro Reuter -> Nova Petrópolis -> Igrejinha -> São F
rancisco de Paula -> Cambará do Sul -> Meleiro -> Criciúma -> Morro da Fumaça -> Joinville ->
Tijucas do Sul -> Curitiba
Tempo de execução: 22.70 segundos
```

Figure 1. Exemplo de Execução

Devido a limitações do Google Maps na visualização de trajetos longos com múltiplos pontos, o caminho completo precisou ser dividido em duas partes, apresentadas nas Figuras 2 e 3. A distância total estimada pelo Google Maps para essa rota foi de aproximadamente 900 km. Vale destacar que o Google pode priorizar tempo de viagem, enquanto nosso algoritmo considera exclusivamente a menor distância.

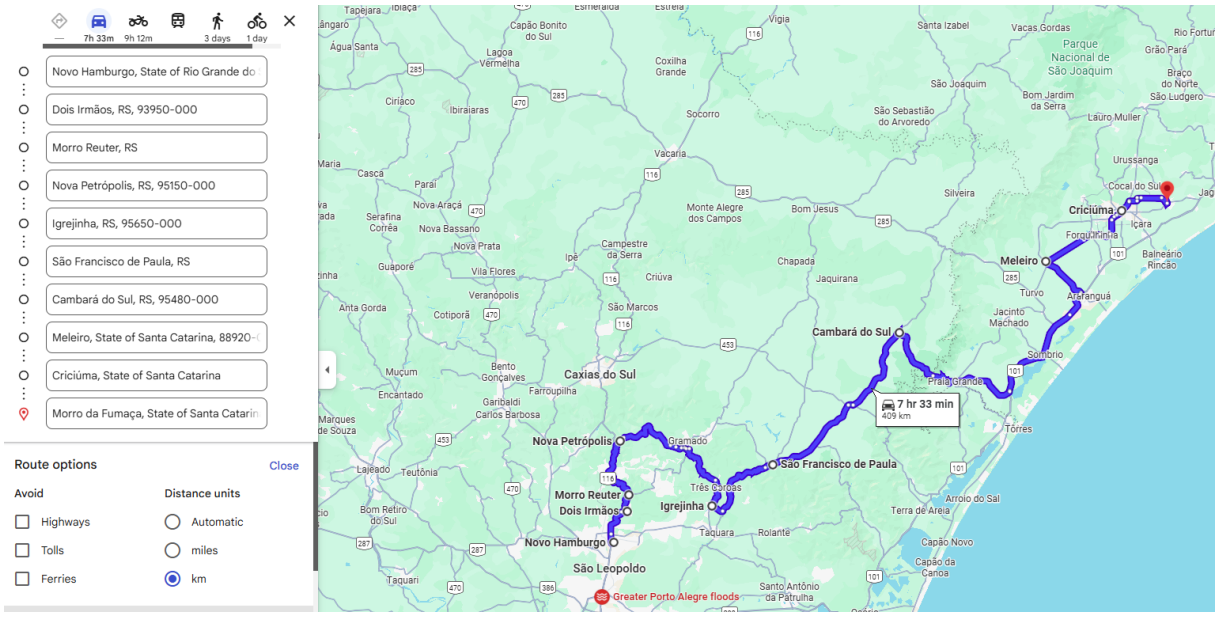

Figure 2. Caminho Equivalente (Parte 1)

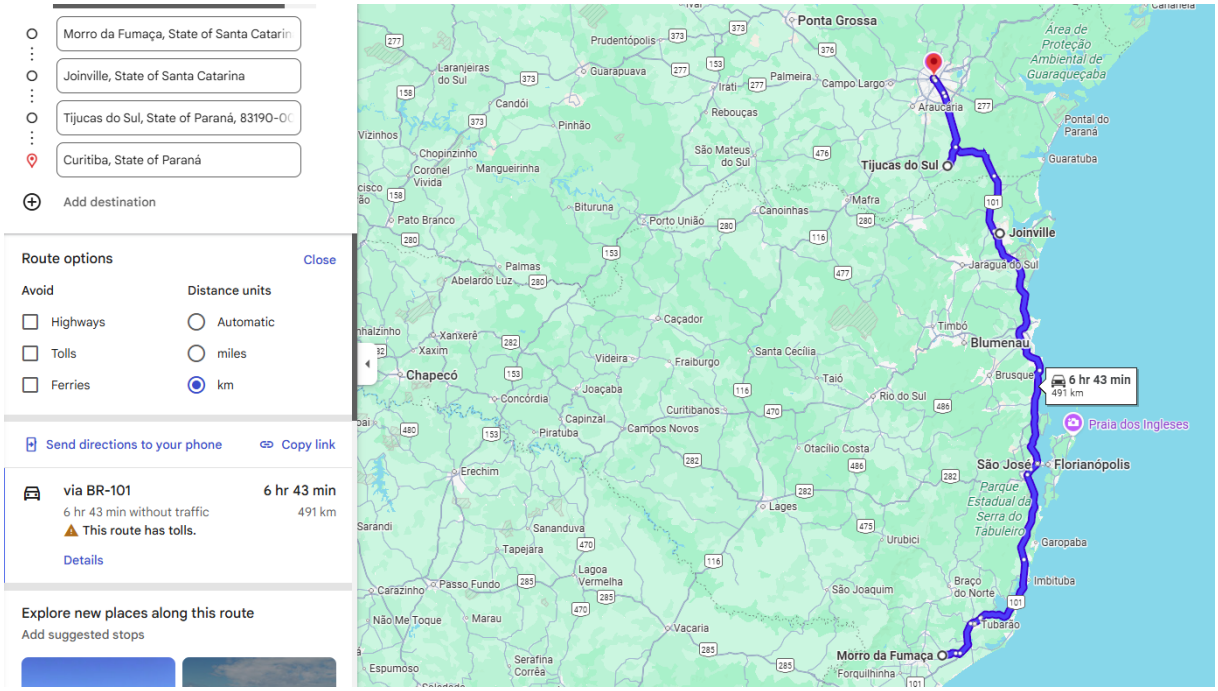

Figure 3. Caminho Equivalente (Parte 2)

A seguir, são apresentados os tempos de execução para cada versão do algoritmo implementado. Na tabela 1 podemos visualizar a evolução da viabilidade de execução do algoritmo em função do número de paradas.

Table 1. Comparação dos tempos de execução entre versões do algoritmo.

Versão	Paradas	Tempo (s)
V1 - Permutação total	5	˜600
V2 - Restrição geográfica (400 km)	5	˜120
V3 - Restrição + Memorização	5	8.19
V3 - Restrição + Memorização	10	57.97
V4 - Heurística Cheapest Insertion	10	34.80
V4 - Heurística Cheapest Insertion	20	153.88

Já na Figura 4 temos um gráfico de complexidade tradicional para uma análise mais completa. Nesse gráfico desconsideramos as execuções com tempo maior de 200 segundos, já que se tratam de tempos exponenciais. Além disso, não vimos necessidade de rodar a versão Heurística com mais de 20 paradas, devido à clara ”Derrota” dos algoritmos anteriores.

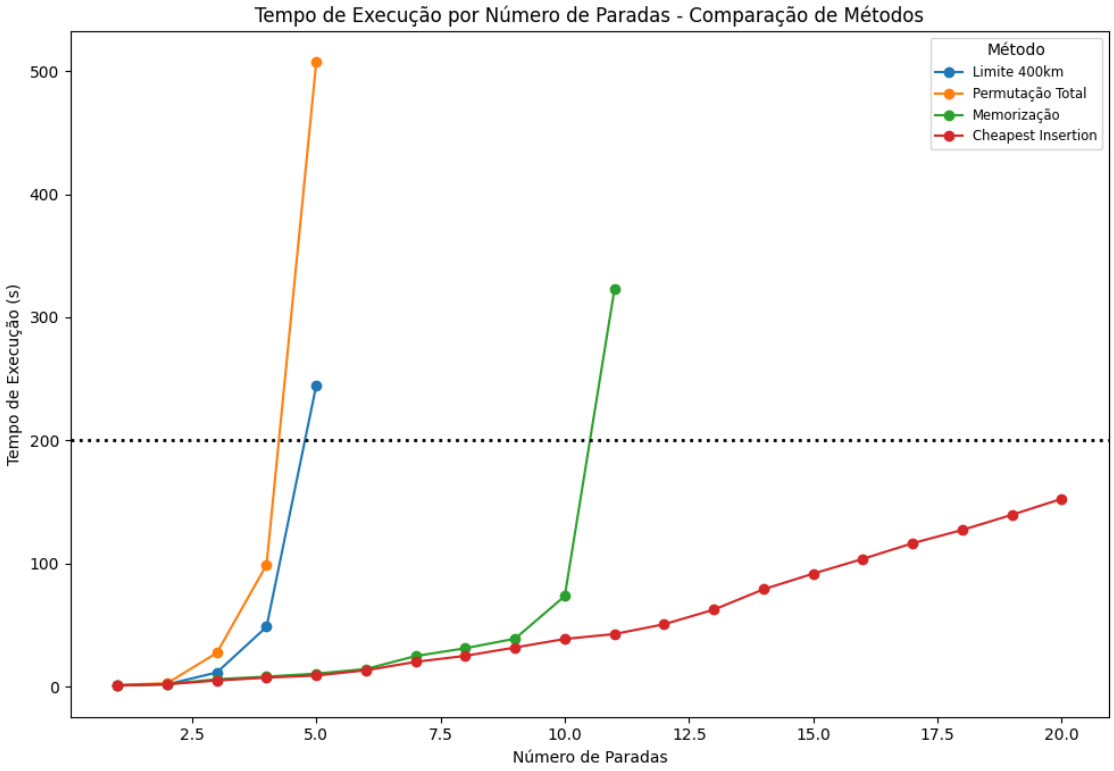

Figure 4. Gráfico de Complexidade

Os resultados evidenciam o impacto positivo das otimizações introduzidas. A introdução da restrição geográfica já representa uma melhora em relação à abordagem exaustiva (V1). A versão V3, que adiciona memorização de custos, reduz ainda mais o tempo de execução. Finalmente, a V4, com uso de heurística, apresenta o melhor desempenho relativo quando o número de paradas cresce, permitindo lidar com instâncias maiores de forma eficiente.

6. Conclusão

Os resultados obtidos neste trabalho demonstram que o uso de heurísticas computacionais foi fundamental para viabilizar a resolução eficiente do problema de roteamento com múltiplas paradas obrigatórias. As otimizações implementadas permitiram um aumento significativo na escalabilidade da solução, tornando possível lidar com instâncias contendo até quatro vezes mais paradas do que seria viável por meio de abordagens exaustivas. Além disso, foi possível aplicar essas técnicas em um cenário realista, utilizando dados públicos de distâncias rodoviárias entre cidades brasileiras. Essa abordagem reforça a aplicabilidade prática dos algoritmos heurísticos em problemas complexos do mundo real, onde soluções exatas se tornam inviáveis devido à complexidade computacional envolvida.

References

[1] *Códigos dos Municípios | IBGE*. https://shorturl.at/XL9al. (Visited on 06/12/2025).

[2] Hansen. *Trabalho final de CAL*. en. https://shorturl.at/QaHc9. (Visited on 06/23/2025).

[3] Vinícius Hansen. *Distância Rodoviária Cidades Sul Brasil*. en. https://shorturl.at/io2co, 2025. (Visited on 06/12/2025).
[4] Raphael Saldanha. *Road distances and trip duration matrix for Brazilian municipalities*. https://zenodo.org/records/11400243, 2024. (Visited on 06/12/2025).