

Otimização de gasto de combustível em entrega de pacotes

Bruno Rodrigues Faria¹, Lucas de Paula Vieira Santos², Maria Luísa Tomich Raso³

¹Instituto de Ciências Exatas e Informática - ICEI (PUC - MG)

Abstract. *This article describes in detail the solution implemented for the package delivery optimization problem, aiming at lower fuel consumption, also known as the 'Vehicle Routing Problem' (VRP), a type of problem similar to the Traveling Salesman Problem (TSP). For this, two different types of algorithms were used, one brute force and the other by branch-and-bound, in addition to a variation of branch-and-bound aiming for efficiency.*

Resumo. *Este artigo descreve com detalhes a solução implementada para o problema de otimização de entrega de pacotes visando menor gasto de combustível, também conhecido como 'Vehicle Routing Problem' (VRP), um tipo de problema similar ao Problema do Caixeiro Viajante (TSP). Para isso, foram utilizados dois tipos diferentes de algoritmos, um de força bruta e o outro por branch-and-bound, além de uma variação do branch-and-bound visando velocidade.*

1. Introdução

Problemas de otimização desempenham um papel crucial em diversos setores produtivos, permitindo encontrar soluções eficientes para desafios logísticos e de distribuição. Um exemplo comum é o planejamento do transporte de cargas entre diferentes pontos geográficos. O objetivo deste trabalho é abordar o problema da coleta e entrega de produtos em uma rede de lojas, utilizando três estratégias de otimização: força bruta, branch-and-bound e branch-and-bound alternativo.

No desenvolvimento deste trabalho, serão consideradas as coordenadas das lojas, a capacidade máxima de carga do caminhão, o rendimento do combustível e as informações sobre as origens e destinos dos produtos, todas contidas em um arquivo de texto.

Por meio de uma animação gráfica, pretende-se ilustrar a sequência das viagens do caminhão e a variação de sua carga entre as filiais, proporcionando uma visualização clara das rotas percorridas e do consumo de combustível associado a cada trajeto.

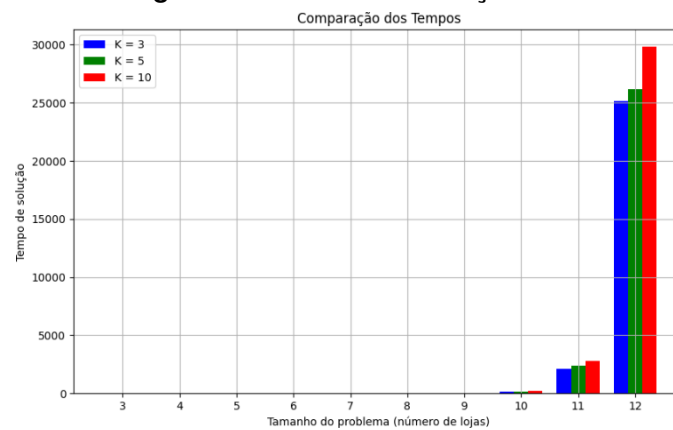
A partir da análise comparativa entre as estratégias de força bruta, branch-and-bound e branch-and-bound alternativo, este estudo busca fornecer uma contribuição para a otimização do transporte de cargas em redes de lojas, destacando a importância de escolher o método mais adequado para a resolução eficiente desse tipo de problema.

2. Solução proposta

2.1. Algoritmos utilizados

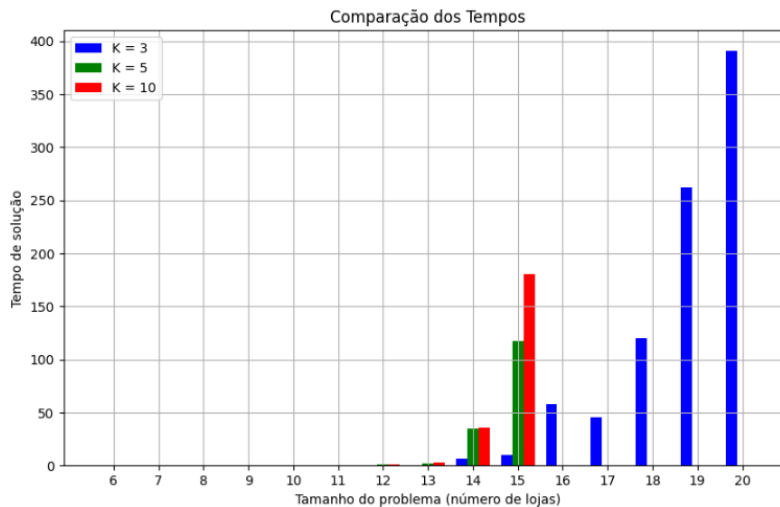
A abordagem inicial consiste na implementação de um algoritmo de força bruta, que busca gerar todas as soluções possíveis e selecionar aquela que resulta no menor gasto de combustível, respeitando as restrições de carga do caminhão. Essa abordagem permite examinar exaustivamente todas as combinações de trajetos, possibilitando a identificação da solução ótima, mas pode ser computacionalmente intensiva. Com isso chegamos a conclusão que o algoritmo de força bruta possui uma complexidade $\mathcal{O}(n!)$, onde n é o número de lojas diferente de 0 como mostrado no gráfico abaixo, observa-se que aumenta muito rápido e as outras execuções ficam quase impossíveis de ser vistas na plotagem, por conta da complexidade:

Figure 1. Gráfico barra - força bruta



Em seguida, propõe-se a aplicação de um algoritmo de branch-and-bound, que se baseia na solução encontrada pela força bruta para eliminar ramos da árvore de soluções que já se mostraram mais custosos. Essa estratégia reduz o espaço de busca e acelera o tempo de processamento, mantendo a busca pela solução ótima. Será comparada a eficiência e a qualidade das soluções obtidas por ambas as abordagens, assim como seus tempos de processamento. A partir disso, pode-se observar que o algoritmo de branch and bound possui uma solução com complexidade melhor que o força bruta, com tempo de execução igual a $\mathcal{O}(n^2 * 2^n)$, como demonstrado no gráfico abaixo. Observar-se um aumento exponencial nas barras em decorrência do número de lojas, mas também variações dependendo da quantidade de itens que o caminhão pode carregar:

Figure 2. Gráfico barra - branch and bound



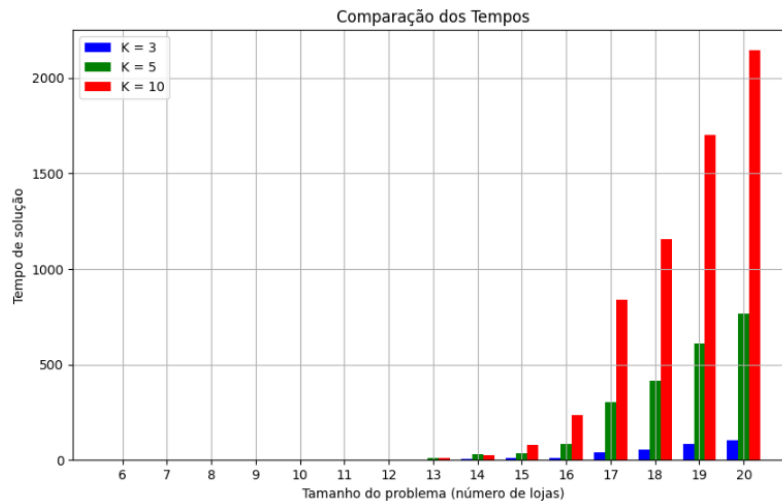
Como pode ser visto na Figura 2, a partir de 16 lojas o cálculo do caminho passa a considerar apenas como $K = 3$, pois ele limita fortemente o tempo de execução do programa, que começa a acelerar devido a sua complexidade exponencial.

Infelizmente, para o VRP e problemas do tipo NP, uma solução para achar o limite máximo ou mínimo de forma definitiva, rápida e consistente é desconhecida. No entanto, é importante ressaltar que, em situações práticas, fornecer limites inferiores ou superiores como uma medida da qualidade é de grande relevância para uma determinada solução proposta.

Embora o branch-and-bound já possua diversas condições de poda que resultam em abordagens bastante eficientes, os cálculos de upper bound ou lower bound podem ser melhorados por diferentes métodos, resultando em soluções mais velozes, embora apresentem menor precisão. Portanto, utilizando o branch-and-bound e alterando o cálculo de limite, assim como adição de mais restrições, foi desenvolvido o branch-and-bound alternativo, que tem como objetivo ter uma execução mais rápida, mesmo que isso signifique uma diminuição na qualidade da solução.

Foram observados diversos métodos para o cálculo do lower bound: "*Nearest Neighbor Algorithm*", que calcula o menor caminho teórico baseado nas menores distâncias para a próxima loja; "*Minimum Spanning Tree*", que utiliza algoritmos de Kruskal ou Prim para analisar a melhor alternativa; "*Held-Karp Bound*", que a cada iteração classifica um valor baseado em arestas e altera o limite; "*Cheapest Link*", que calcula o valor de um ciclo teórico utilizando os menores valores de cada vértice. [Cook], [Urrutia et al. 2007], [Sal], [Rathinam and Sengupta 2006]

Figure 3. Gráfico barra - branch and bound - alternativo



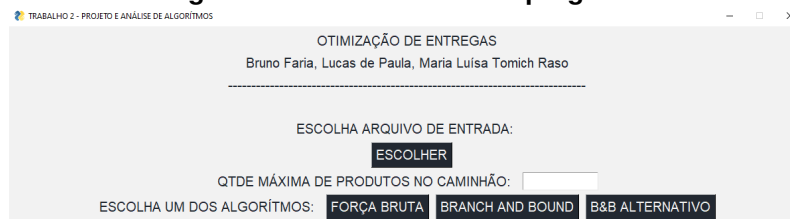
3. Implementação

Em primeiro lugar, foi criada uma interface gráfica que apresenta a possibilidade de escolha do arquivo de entrada e do tipo de solução a ser utilizada. Ambos estão melhor descritos nas subseções abaixo. Além disso, foi criada uma interface secundária, que aparece após a escolha do algoritmo de solução, e que apresenta o resultado gráfico das lojas pelo caminho otimizado, percorrido pelo caminhão, mostrando a mudança de carga e de combustível entre as suas filiais.

3.1. Interfaces e modelo de arquivo de entrada

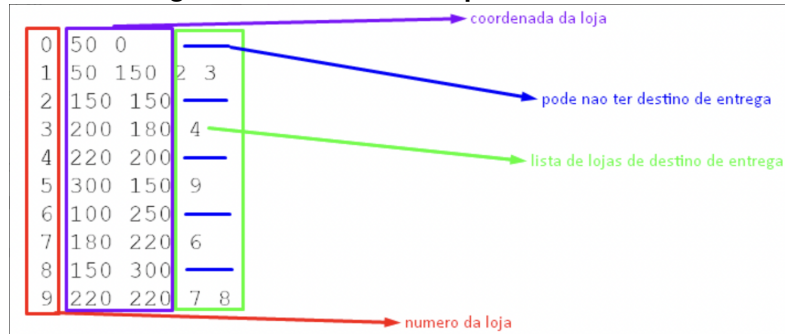
A seguinte interface é mostrada quando o programa roda:

Figure 4. Interface inicial do programa.



Nela, é possível escolher um arquivo de entrada, o número K de itens no caminhão, e qual a solução desejada. O arquivo de entrada é no formato texto, e possui a seguinte estrutura:

Figure 5. Estrutura do arquivo de entrada.

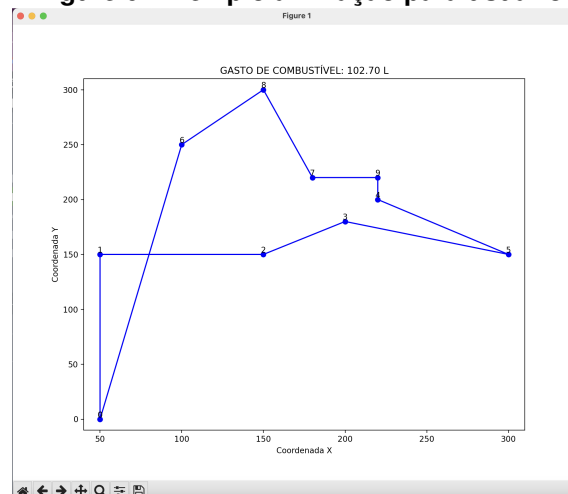


The diagram shows a table representing the input file structure. Annotations include: a purple arrow pointing to the first column labeled 'coordenada da loja', a blue arrow pointing to the second column labeled 'pode nao ter destino de entrega', a green arrow pointing to the third column labeled 'lista de lojas de destino de entrega', and a red arrow pointing to the last column labeled 'numero da loja'.

| | | | | |
|---|-----|-----|---|---|
| 0 | 50 | 0 | | |
| 1 | 50 | 150 | 2 | 3 |
| 2 | 150 | 150 | | |
| 3 | 200 | 180 | 4 | |
| 4 | 220 | 200 | | |
| 5 | 300 | 150 | 9 | |
| 6 | 100 | 250 | | |
| 7 | 180 | 220 | 6 | |
| 8 | 150 | 300 | | |
| 9 | 220 | 220 | 7 | 8 |

Sendo assim, a depender das escolhas feitas na interface inicial, uma segunda interface é mostrada, dinamicamente, em que um gráfico é plotado com as lojas (com formatos de pontos) em suas respectivas coordenadas no mapa (que foi lido no arquivo de entrada) e o caminho otimizado entre elas vai sendo traçado segundo a abordagem escolhida, mostrando também o gasto de combustível sendo atualizado e as cargas atualizadas do caminhão em cada ponto. Veja, a seguir, um exemplo, lembrando que a imagem apresenta apenas o formato final do gráfico, mas o usuário consegue visualizar ele sendo construído aos poucos:

Figure 6. Exemplo animação para usuário



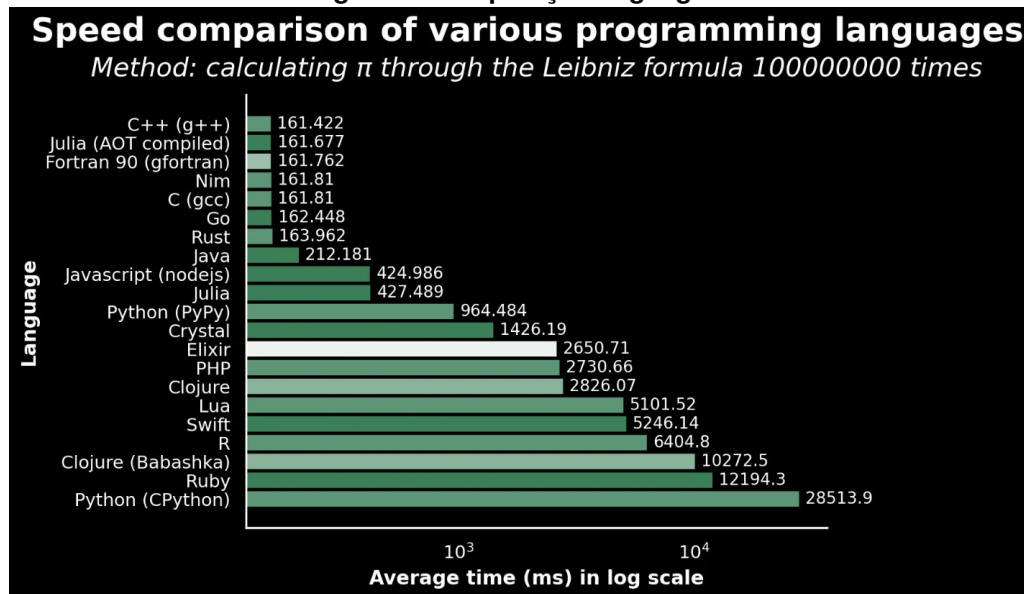
3.2. Escolha da linguagem de programação

A linguagem escolhida para a implementação da solução proposta foi a linguagem Python, pelos seguintes motivos:

1. Facilidade de construção de interfaces
2. Facilidade de plotagem de gráficos dinâmicos
3. Simplicidade e eficiência na codificação - variáveis, escopo, funções

No entanto, alguns desafios também foram encontrados em sua utilização como, por exemplo, o tempo de execução comparado com as demais linguagens. Exemplo, ao chegar em uma certa quantidade de lojas o programa começa a demorar muito, sendo inviável no força bruta e viável no branch and bound, como mostra o gráfico de comparação de linguagens:

Figure 7. Comparação linguagens



3.3. Força Bruta

O algoritmo força bruta apresentado aqui aborda o problema de encontrar a rota de entrega ideal em um cenário de logística com várias lojas e entregas. Este artigo fornece uma explicação de alto nível da lógica por trás do algoritmo sem se aprofundar em detalhes de variáveis específicas.

Dado um conjunto de lojas, entregas, uma matriz de distância e uma restrição no número de produtos que um caminhão pode transportar, o objetivo é determinar a rota mais eficiente que minimize o custo total da entrega. O algoritmo considera tanto a distância percorrida quanto o consumo de combustível como fatores na determinação da solução.

A partir do conjunto de lojas, o algoritmo de força bruta testa todas as possibilidades de solução até o final, ou seja, não aplica nenhum tipo de poda, motivo pelo qual sua complexidade é fatorial e, portanto, muito alta. Porém ele deve avaliar ainda assim se a loja é válida, vendo se todos produtos foram entregues e se ela chegou vazia ao destino, pois pode ter um caminho que seja válido com menor custo de combustível mas que não atenda as restrições padrões do problema.

3.3.1. Visão geral do algoritmo de força bruta

Inicialização:

- Inicialize uma variável para armazenar o número de permutações.
- Extraia os IDs da loja dos dados de entrada, excluindo a origem e o destino.
- Defina a melhor rota e custo inicial como indefinidos.
- Inicialize variáveis para armazenar o melhor custo, a lista de custos para plotagem, a lista de itens transportados e a contagem de permutações.

Gerar permutações:

- Gere recursivamente todas as permutações possíveis dos IDs da loja.
- Para cada permutação:
 - Acrescente a origem (0) e o destino (0) para formar uma rota completa.
 - Incrementar a contagem de permutações.
 - Chame a função auxiliar para calcular a viagem total para esta permutação.

Calcular Viagem Total para uma Permutação:

- Inicialize variáveis para armazenar o consumo de combustível, itens transportados e informações de rota.
- Iterar sobre as lojas na permutação:
 - Verifique se a loja pode ser visitada com base na rota atual.
 - Se a loja pode ser visitada:
 - * Verifique se a loja atual é válida com base nos itens transportados e na lista de entregas.
 - * Se a ramificação for válida:
 - Calcule o consumo de combustível atual.
 - Verifique se o custo atual é melhor que o melhor custo anterior.
 - Se for, atualize o melhor custo, a lista de custos para plotagem, a lista de itens transportados e a rota
- Retorne a resposta para o programa.

3.4. Branch and Bound

O algoritmo branch and bound apresentado aqui aborda o problema de encontrar a rota de entrega ideal em um cenário de logística com várias lojas e entregas. Este artigo fornece uma explicação de alto nível da lógica por trás do algoritmo sem se aprofundar em detalhes de variáveis específicas.

Dado um conjunto de lojas, entregas, uma matriz de distância e uma restrição no número de produtos que um caminhão pode transportar, o objetivo é determinar a rota mais eficiente que minimize o custo total da entrega. O algoritmo considera tanto a distância percorrida quanto o consumo de combustível como fatores na determinação da solução ótima.

Para chegar a solução ótima deve-se aplicar restrições e calcular o lower bound do custo do combustível a fim de podar ramos infrutíferos da árvore de permutações evitando fazer cálculos desnecessários para caminhos que não serão interessantes. Algumas das restrições são: ver se os produtos que tem que ser entregues ainda não foram pegos e as lojas ainda não foram visitadas, conferir se a quantidade de produtos carregados é suportada pelo caminhão, ver se o caminho atual tem rendimento melhor que o antigo, com isso os ramos serão podados de forma eficiente.

3.4.1. Visão geral do algoritmo branch and bound

Inicialização:

- Extraí os IDs das lojas dos dados de entrada, excluindo a origem e o destino.
- Define a melhor rota inicial e custo como indefinido..

- Inicializar variáveis para armazenar permutações e número de ramificações podadas

Gerar permutações:

- Gere recursivamente todas as permutações possíveis dos IDs da loja.
- Para cada permutação
 - Acrescente a origem (0) e o destino (0) para formar uma rota completa.
 - Aumente a contagem de permutações.
 - Chame a função auxiliar para calcular a viagem total para esta permutação.

Calcular Viagem Total para uma Permutação:

- Inicialize variáveis para armazenar o consumo de combustível, produtos transportados e informações de rota.
- Iterar sobre as lojas na permutação:
 - Verifique se a loja pode ser visitada com base na rota atual.
 - Se a loja pode ser visitada:
 - * Verifique se a loja atual é válida com base nos itens transportados e na lista de entregas.
 - * Se a ramificação for válida:
 - Calcule o limite inferior do consumo de combustível para o ramal atual.
 - Se o limite inferior for melhor que o melhor custo atual: explore a ramificação
 - Caso contrário, podar o galho.
 - * Caso contrário, podar o galho.
 - Caso contrário, podar o galho.

Avaliar solução:

- Se a permutação atual chegar ao destino com todos os produtos entregues:
 - Calcule o custo total da viagem.
 - Se o custo for melhor que o melhor custo anterior:
 - * Atualize o melhor custo e a rota solução.
 - * Armazene o consumo de combustível, produtos transportados e informações de rota.
- Retorne a resposta para o programa.

3.5. Branch and Bound - alternativo

O algoritmo de branch and bound por alternativo é baseado no algoritmo de branch and bound citado acima, com os mesmos testes de restrições e mesma estrutura, porém a diferença observada é no cálculo de lower bound, o qual tenta prever o futuro para executar de uma forma mais eficiente, utilizando uma heurística com base em *Cheapest Link* e *Nearby Neighbor*. Uma das partes do cálculo para o lower bound foi utilizado do somatório:

$$\frac{1}{2} \sum_{v \in V} (\text{soma dos custos das duas arestas adjacentes a } v) \quad (1)$$

Essa equação determina que, para cada loja 'v', se for considerado que duas arestas passam por ele em um caminho T e somar seus custos, é possível determinar que a soma geral para todos os vértices seria o dobro do custo do caminho T, pois são consideradas todas as arestas duas vezes. Além disso, as arestas observadas serão sempre as duas menores arestas conectadas a cada loja 'v'. Portanto, é possível afirmar que esse somatório compreende "O menor caminho teórico para entrar e sair todas as lojas restantes".

3.5.1. Visão geral do algoritmo branch and bound alternativo

Verifique o ramo:

- Verifique se a loja atual é válida com base nos itens no caminhão e nas entregas a serem feitas.
- Se a ramificação for válida:
 - Remova a loja atual do dicionário de lojas temporárias.
 - Calcule a soma das menores distâncias das lojas restantes usando a matriz de distância.
 - Calcule a estimativa do lower bound adicionando o custo atual à soma das menores distâncias.
 - Se a estimativa do lower bound estiver dentro de 95% do melhor custo atual:
 - * Continue recursivamente gerando permutações com os parâmetros atualizados.
 - Caso contrário, incremente a contagem de poda, pois a ramificação não é válida e será removida.
 - Adicione a loja atual no dicionário de lojas temporárias.
- Caso contrário, incremente a contagem de poda, pois a ramificação não é válida e será removida.

Calcular menores distâncias:

- Se houver pelo menos três lojas restantes:
 - Inicialize um dicionário para armazenar as menores distâncias para cada loja.
 - Iterar sobre cada loja:
 - * Calcule as distâncias da loja atual para todas as outras lojas restantes
 - * Classifique as distâncias em ordem crescente.
 - * Mantenha apenas as duas menores distâncias para cada loja.
 - * Armazene as menores distâncias no dicionário para o caminho atual.
 - * Calcule a soma de todas as menores distâncias para uso posterior.
 - Calcule a estimativa do lower bound, dividindo a soma de todas as distâncias menores por 20 (supondo uma eficiência de combustível de 10).
- Caso contrário, se houver duas lojas restantes:
 - Calcule a distância entre as lojas.
 - Retorne a distância dividida por 10 (assumindo uma eficiência de combustível de 10) como a estimativa do lower bound.
- Se houver apenas uma loja restante ou nenhuma loja restante, retorne 0,0 como a estimativa do lower bound, pois seu cálculo não é mais necessário.

4. Relatório de testes

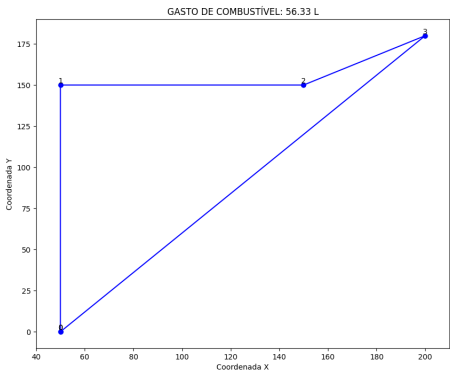
4.1. Teste 0: 3 lojas

Arquivo de entrada do teste 0:

Figure 8.

```
0 50 0
1 50 150 2 3 2
2 150 150
3 200 180
```

Figure 9. Caminho 3 Lojas



4.1.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----------------|-----------|-----------------|-----------------|
| 3,5,10 (igual) | 0.0 | [0, 1, 2, 3, 0] | 56.33 |

4.1.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.1.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-----------|-----------|-----------------|-----------------|
| 3, 5 e 10 | 0.0 | [0, 1, 2, 3, 0] | 56.33 |

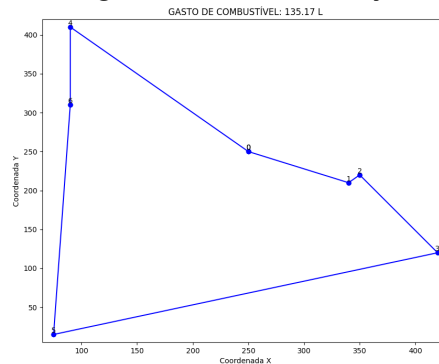
4.2. Teste 1: 6 lojas

Arquivo de entrada do teste 1:

Figure 10.

```
0 250 250
1 340 210 3 4 5
2 350 220 6
3 420 120
4 90 410
5 75 15
6 90 310
```

Figure 11. Caminho 6 lojas



4.2.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|--------------------------|-----------------|
| 3 | 0.0 | [0, 1, 3, 2, 5, 6, 4, 0] | 143.90 |
| 5, 10 | 0.0 | [0, 1, 2, 3, 5, 6, 4, 0] | 135.17 |

4.2.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.2.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|--------------------------|-----------------|
| 3 | 0.02 | [0, 1, 3, 2, 5, 6, 4, 0] | 143.90 |
| 5, 10 | 0.02 | [0, 1, 2, 3, 5, 6, 4, 0] | 135.17 |

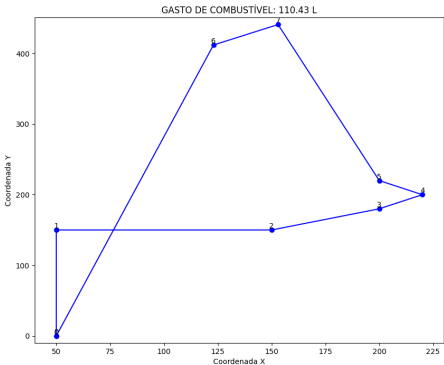
4.3. Teste 2: 7 lojas

Arquivo de entrada do teste 2:

Figure 12.

```
0 50 0
1 50 150 2 3
2 150 150 4 5
3 200 180
4 220 200
5 200 220 6 7
6 123 412
7 153 441
```

Figure 13. Caminho 7 Lojas



4.3.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|-----------------------------|-----------------|
| 3 | 0.0 | [0, 1, 2, 3, 4, 5, 7, 6, 0] | 110.42 |
| 5, 10 | 0.0 | [0, 1, 2, 3, 4, 5, 7, 6, 0] | 110.42 |

4.3.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.3.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-----------|-----------|-----------------------------|-----------------|
| 3, 5 e 10 | 0.18 | [0, 1, 2, 3, 4, 5, 7, 6, 0] | 110.42 |

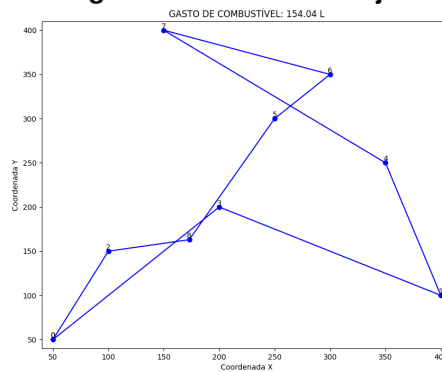
4.4. Teste 3: 8 lojas

Arquivo de entrada do teste 3:

Figure 14.

```
0 50 50
1 400 100 7 2
2 100 150
3 200 200 4
4 350 250
5 250 300 8
6 300 350
7 150 400 6 5
8 173 163
```

Figure 15. Caminho 8 Lojas



4.4.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|------|-----------|--------------------------------|-----------------|
| 3 | 0.03 | [0, 3, 1, 4, 7, 6, 5, 8, 2, 0] | 154.03 |
| 5,10 | 0.05 | [0, 3, 1, 4, 7, 6, 5, 8, 2, 0] | 154.03 |

4.4.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.4.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-----------|-----------|--------------------------------|-----------------|
| 3, 5 e 10 | 1.56 | [0, 3, 1, 4, 7, 6, 5, 8, 2, 0] | 154.03 |

4.5. Teste 4: 9 lojas

Arquivo de entrada do teste 4:

```
0 50 0
1 50 150 2 3
2 150 150
3 200 180 4
4 220 200
5 300 150 9
6 100 250
7 180 220 6
8 150 300
9 220 220 7 8
```

Figure 16.

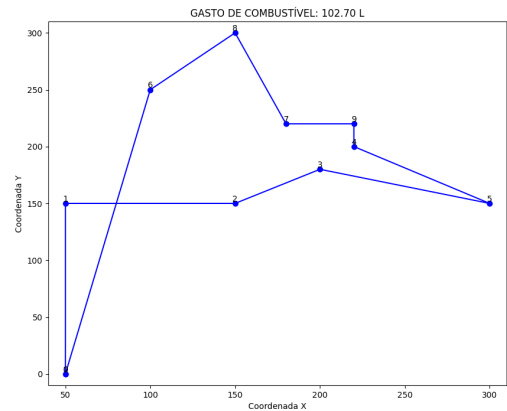


Figure 17. Caminho 9 Lojas

4.5.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|-----------------------------------|-----------------|
| 3 | 0.06 | [0, 1, 2, 3, 5, 4, 9, 7, 8, 6, 0] | 102.70 |
| 5, 10 | 0.13 | [0, 1, 2, 3, 5, 4, 9, 7, 8, 6, 0] | 102.70 |

4.5.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.5.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-----------|-----------|-----------------------------------|-----------------|
| 3, 5 e 10 | 15.86 | [0, 1, 2, 3, 5, 4, 9, 7, 8, 6, 0] | 102.70 |

4.6. Teste 5: 10 lojas

Arquivo de entrada do teste 5:

```
0 50 0
1 50 150 2 3
2 150 150
3 200 180 4
4 220 200
5 300 150 9
6 100 250
7 180 220 6
8 150 300 10
9 220 220 7 8
10 421 435
```

Figure 18.

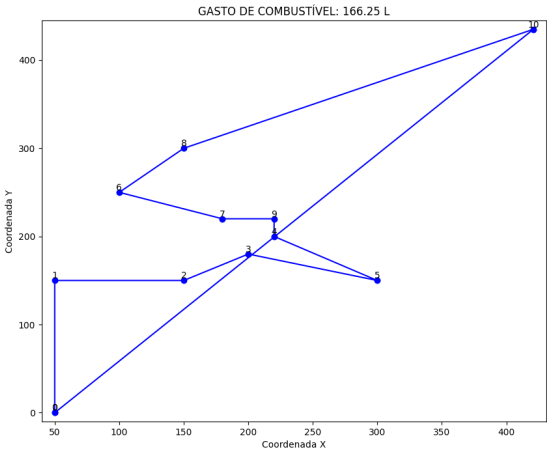


Figure 19. Caminho 10 Lojas

4.6.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|---------------------------------------|-----------------|
| 3 | 0.21 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 0] | 166.24 |
| 5, 10 | 0.66 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 0] | 166.24 |

4.6.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.6.3. Força bruta

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-----------|-----------|---------------------------------------|-----------------|
| 3, 5 e 10 | 175.01 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 0] | 166.24 |

4.7. Teste 6: 11 lojas

Arquivo de entrada do teste 6:

```
0 50 0
1 50 150 2 3
2 150 150
3 200 180 4
4 220 200
5 300 150 9
6 100 250
7 180 220 6
8 150 300 10
9 220 220 7 8
10 421 435 11
11 351 141
```

Figure 20.

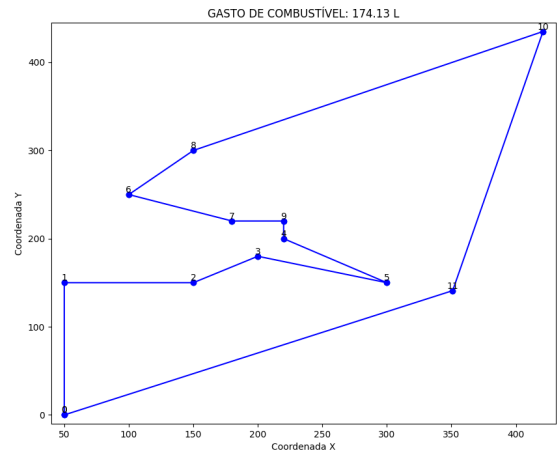


Figure 21. Caminho 11 Lojas

4.7.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|---|-----------------|
| 3 | 0.38 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 11, 0] | 174.12 |
| 5, 10 | 1.21 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 11, 0] | 174.12 |

4.7.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.7.3. Força bruta

Neste momento, torna-se impossível executar o algoritmo de força bruta em tempo viável, uma vez que tomou proporções muito grandes.

4.8. Teste 7: 12 lojas

Arquivo de entrada do teste 7:

```
0 50 0
1 50 150 2 3
2 150 150
3 200 180 4
4 220 200
5 300 150 9
6 100 250
7 180 220 6
8 150 300 10
9 220 220 7 8
10 421 435 11 12
11 351 141
12 361 114
```

Figure 22.

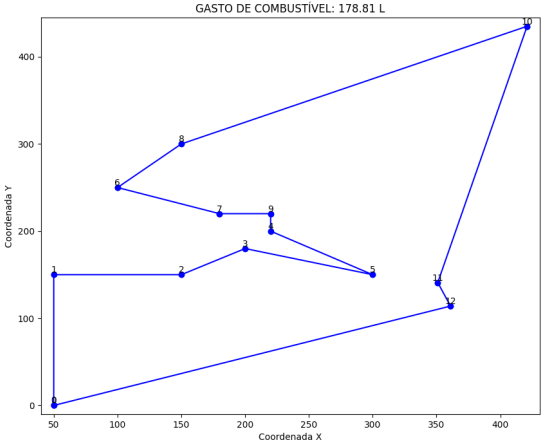


Figure 23. Caminho 12 Lojas

4.8.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|-------|-----------|---|-----------------|
| 3 | 0.63 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 11, 12, 0] | 178.81 |
| 5, 10 | 2.23 | [0, 1, 2, 3, 5, 4, 9, 7, 6, 8, 10, 11, 12, 0] | 178.81 |

4.8.2. Branch and bound - Alternativo

Neste momento, foi observada uma diferença insignificante em relação ao tempo de execução do Branch and Bound normal com o Branch and Bound utilizando alternativo. Desta forma, não é mais uma solução relevante para o problema.

4.8.3. Força bruta

Neste momento, torna-se impossível executar o algoritmo de força bruta em tempo viável, uma vez que tomou proporções muito grandes.

4.9. Teste 8: 13 lojas

Arquivo de entrada do teste 8:

```
0 50 50
1 100 500
2 150 350 11
3 200 400
4 250 100 7 6
5 300 150
6 350 200 5 1
7 500 0 13
8 450 250
9 400 300 8
10 0 450 9
11 125 475 3
12 175 325
13 225 375 12
```

Figure 24.

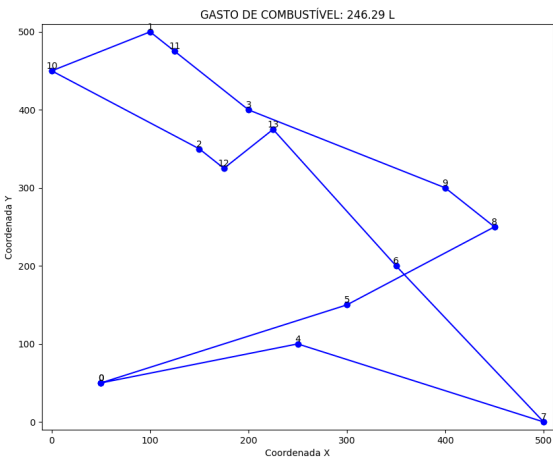


Figure 25. Caminho 13 Lojas

4.9.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 6.37 | [0, 4, 7, 6, 5, 13, 12, 2, 10, 1, 11, 3, 9, 8, 0] | 249.16 |
| 5 | 35.06 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 3, 9, 8, 5, 0] | 246.28 |
| 10 | 36.72 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 3, 9, 8, 5, 0] | 246.28 |

4.9.2. Branch and bound - Alternativo

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 2.64 | [0, 4, 7, 6, 5, 13, 12, 2, 10, 1, 11, 3, 9, 8, 0] | 249.16 |
| 5 | 8.47 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 3, 9, 8, 5, 0] | 246.28 |
| 10 | 8.57 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 3, 9, 8, 5, 0] | 246.28 |

Observa-se o mesmo resultado para combustível e caminhos, todavia com visível mudança no tempo de execução.

4.9.3. Força bruta

Neste momento, torna-se impossível executar o algoritmo de força bruta em tempo viável, uma vez que tomou proporções muito grandes.

4.10. Teste 9: 14 lojas

Arquivo de entrada do teste 9:

```
0 50 50
1 100 500
2 150 350 11
3 200 400
4 250 100 7 6
5 300 150
6 350 200 5 1
7 500 0 13
8 450 250
9 400 300 8
10 0 450 9
11 125 475 3
12 175 325
13 225 375 12 14
14 131 431
```

Figure 26.

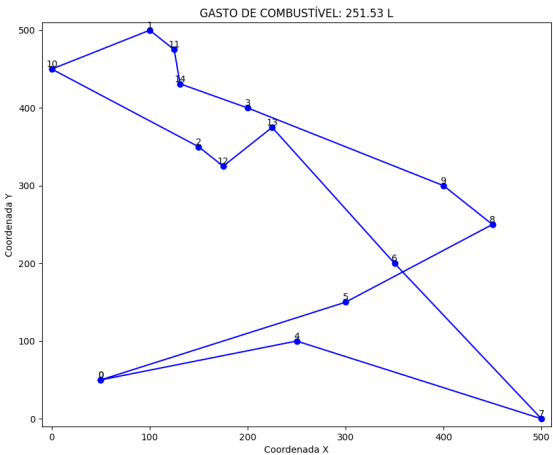


Figure 27. Caminho 14 Lojas

4.10.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 9.81 | [0, 4, 7, 6, 5, 13, 12, 2, 14, 11, 1, 10, 3, 9, 8, 0] | 254.25 |
| 5 | 117.2 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 14, 3, 9, 8, 5, 0] | 251.52 |
| 10 | 129.98 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 14, 3, 9, 8, 5, 0] | 251.52 |

4.10.2. Branch and bound - Alternativo

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 5.61 | [0, 4, 7, 6, 5, 13, 12, 2, 14, 11, 1, 10, 3, 9, 8, 0] | 254.25 |
| 5 | 31.39 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 14, 3, 9, 8, 5, 0] | 251.52 |
| 10 | 27.21 | [0, 4, 7, 6, 13, 12, 2, 10, 1, 11, 14, 3, 9, 8, 5, 0] | 251.52 |

Observa-se o mesmo resultado para combustível e caminhos, todavia com visível mudança no tempo de execução.

4.10.3. Força bruta

Neste momento, torna-se impossível executar o algoritmo de força bruta em tempo viável, uma vez que tomou proporções muito grandes.

4.11. Teste 10: 15 lojas

Arquivo de entrada do teste 10:

```
0 400 400
1 320 120
2 189 283
3 176 341 1 2
4 234 472
5 134 164
6 451 361 5
7 153 174 4
8 421 341 9 10 11
9 125 153
10 163 131 12
11 114 261 13
12 123 423
13 165 165 14 15
14 412 235
15 500 324
```

Figure 28.

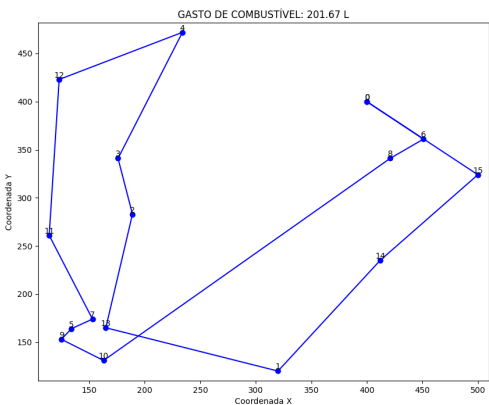


Figure 29. Caminho 9 Lojas

4.11.1. Branch and bound

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 58.08 | [0, 8, 10, 9, 11, 12, 3, 2, 13, 1, 14, 15, 6, 7, 5, 4, 0] | 257.57 |
| 5 | 540.12 | [0, 6, 8, 10, 9, 5, 7, 11, 12, 4, 3, 2, 13, 1, 14, 15, 0] | 201.67 |
| 10 | 630.81 | [0, 6, 8, 10, 9, 5, 7, 11, 12, 4, 3, 2, 13, 1, 14, 15, 0] | 201.67 |

4.11.2. Branch and bound - Alternativo

| K | TEMPO (s) | LOJAS | COMBUSTÍVEL (L) |
|----|-----------|---|-----------------|
| 3 | 9.39 | [0, 8, 10, 9, 11, 12, 3, 2, 13, 1, 14, 15, 6, 7, 5, 4, 0] | 257.57 |
| 5 | 34.70 | [0, 6, 8, 10, 9, 5, 7, 11, 12, 4, 3, 2, 13, 1, 14, 15, 0] | 201.67 |
| 10 | 79.70 | [0, 6, 8, 10, 9, 5, 7, 11, 12, 4, 3, 2, 13, 1, 14, 15, 0] | 201.67 |

Observa-se o mesmo resultado para combustível e caminhos, todavia com visível mudança no tempo de execução.

4.11.3. Força bruta

Neste momento, torna-se impossível executar o algoritmo de força bruta em tempo viável, uma vez que tomou proporções muito grandes.

4.12. MÁQUINA

A fim de fornecer informações mais detalhadas sobre o ambiente experimental utilizado, apresentamos na **Tabela 1** as especificações do computador utilizado nos testes realizados neste estudo. Os dados contidos nesta tabela incluem informações sobre o processador, quantidade de memória RAM, sistema operacional, entre outros.

| Especificações | |
|------------------|------------------------|
| PROCESSADOR | i5-10400F 2.90GHz |
| MEMÓRIA RAM | 16 GB - 2666mhz - DDR4 |
| SIS. OPERACIONAL | Windows 10 |
| PLACA DE VÍDEO | RTX 3060 - 12GB |

Table 1. Especificação da máquina usada para os testes

5. Conclusão

Com base nos testes realizados, observou-se um comportamento interessante: à medida que a quantidade de lojas aumenta, o número K de itens que o caminhão consegue carregar afeta o caminho final e aumenta o tempo de execução. Essa relação ocorre devido à existência de mais caminhos válidos no algoritmo de branch and bound, logo ocorrendo menos podas, que em consequência geram maior tempo de execução. No entanto, essa tendência não é significativa quando se trata de quantidades pequenas de lojas.

É importante ressaltar algumas considerações sobre o funcionamento dos algoritmos. O problema de entregas entre as filiais é classificado como NP, o que significa que utilizar a força bruta não é uma opção viável quando lidamos com arquivos de entrada muito grandes, como observado nos testes da seção anterior. O tempo de processamento se torna inviável, pois seria necessário explorar todas as permutações de todos os caminhos. Nesse cenário, a técnica de branch and bound surge como uma alternativa para reduzir o tempo de execução, pois permite podar ramos não promissores da árvore, economizando recursos e tempo que seriam desperdiçados com eles. A partir disso, torna-se possível lidar com entradas maiores. Além deste fator, pode-se notar também a existência de um segundo método do tipo branch and bound, que deixa o tempo ainda menor, o qual consiste em adicionar heurísticas ao código para gerar podas mais promissoras, que visivelmente tornam o código mais rápido.

Sendo assim, é importante destacar novamente que a escolha da linguagem de programação também pode afetar pequenas características da solução final, devido às suas particularidades e limitações específicas. Mesmo utilizando uma linguagem com 100x melhor desempenho (Por exemplo C++), a melhoria seria linear, assim se tornando irrelevante em próximas iterações, devido a complexidade dos algoritmos escolhidos para tratar o problema, como abordado em todo documento.

6. Bibliografia

References

- The travelling salesman problem. https://www.thechalkface.net/resources/Travelling_Salesman_England.pdf. Accessed on June 4, 2023.
- Cook, W. The traveling salesman problem. <https://math.mit.edu/~goemans/18433S15/TSP-CookCPS.pdf>. Acessado em 4 de junho de 2023.
- Rathinam, S. and Sengupta, R. (2006). Lower and upper bounds for a multiple depot uav routing problem. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5287–5292.
- Urrutia, S., Ribeiro, C. C., and Melo, R. A. (2007). A new lower bound to the traveling tournament problem. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 15–18.