Relatório
Programação Matemática

Rodrigo Cesar Arboleda - 10416722
Marcelo Isaias de Moraes Junior - 10550218
Igor Takeo Ambo de Melo - 10830054
Mateus Ferreira Gomes - 10734773
Guilherme Targon Marques Barcellos - 10724181

Modelagem

O problema consiste em encontrar um caminho com origem na primeira galáxia para percorrer todas as galáxias sendo que deve se percorrer o menor caminho e não deve passar pela mesma galáxia 2 vezes. Para resolver o problema iremos primeiramente modelar o problema.

Variáveis

x_{ij} : variável inteira binária que indica se o caminho da galáxia ***i*** à galáxia ***j*** foi escolhido.

$x_{ij} \varepsilon \{0,1\}$ para todo $i \varepsilon \{1,2,..,n\}$ e $j \varepsilon \{1,2,..,n\}$ com $i \neq j$, sendo ***n*** o número de galáxias do problema em questão. Assume o valor 1 caso o caminho seja escolhido e 0 caso contrário.

Parâmetros

d_{ij} : distância (custo) do caminho da galáxia ***i*** para a galáxia ***j*** , em que $d_{ij} > 0$ para todo $i \varepsilon \{1,2,..,n\}$ e $j \varepsilon \{1,2,..,n\}$. Além disso, a distância de ***i*** para a galáxia ***j*** é igual à distância de ***j*** à ***i***, isto é $d_{ij} = d_{ji}$.

Função objetivo

A função que queremos minimizar é a seguinte:

$$\min \sum_i \sum_j d_{ij} x_{ij}$$

Esta função irá retornar a distância percorrida para se passar por todas as galáxias.

Restrições

O problema apresenta algumas restrições. Primeiro precisamos garantir que iremos chegar em cada galáxia e também iremos sair dela, pois não podemos simplesmente “teletransportar” o telescópio para uma galáxia, além de que temos que passar por todas. Além disso, iremos chegar e sair de cada galáxia apenas uma vez.

Para garantir que sempre haverá um caminho chegando a todas as galáxias e apenas uma vez, teremos a seguinte restrição:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, ..., n, i \neq j$$

Com isso conseguimos garantir que sempre iremos chegar em uma galáxia apenas uma vez.

Para garantir que iremos sair dessa galáxia e que também faremos isso uma vez apenas teremos a seguinte restrição:

$$\sum_{j=1}^{n} x_{ji} = 1 \qquad i = 1, ..., n, i \neq j$$

Dessa forma podemos garantir essas restrições. Por fim precisamos garantir outro ponto que é que não haverá subciclos, ou seja, dois ciclos independentes. Isso é um problema pois tal fato satisfaz as restrições anteriores mas não serve para o problema pois o telescópio precisaria "teletransportar" de um ciclo para o outro.

Para resolver este problema iremos criar mais uma restrição. A restrição que iremos criar é um pouco diferente do convencional, tal restrição se dá por contradição, ou seja, quando tivermos subciclo chegaremos em uma contradição e a restrição não irá valer invalidando a resposta. Tal método foi escolhido para facilitar a implementação do problema. Segue a restrição a seguir:

$$r_i - r_j + nx_{ij} \leq n - 1 \quad \forall i, j \in \{2, ..., n\}, i \neq j$$

$$r_i \geq 0 \quad i = 1, ..., n$$

Está restrição funciona da seguinte forma: n é o número de galáxias que temos, e com isso adicionamos uma variável r, esta variável irá servir apenas para gerar uma inconsistência caso surja um ciclo. Para melhor entendimento de como a restrição funciona, segue um exemplo prático dela.

Tomamos 5 galáxias (1,2,3,4,5) onde a galáxia 1 está ligada com a 2, e a 2 com a 3, e a 3 com a 1. E por fim a galáxia 4 e 5 ligadas entre elas formando um subciclo. Ou seja, temos um ciclo em (1,2,3) e outro em (4,5).

Caso isso ocorra teremos que:

$$r_4 - r_5 + 5 * 1 \leq 4$$

$$r_5 - r_4 + 5 * 1 \leq 4$$

O que claramente é um absurdo. Esta formulação é conhecida como: Formulação de Miller-Tucker-Zemlin. Esta formulação foi retirada do documento da UFPR do Professor Volmir Eugênio Wilhelm, o documento pode ser encontrado no link: https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf (Referência [11])

Com isso nosso problema fica com a seguinte formulação:

Formulação

Função objetivo:

$$\min \sum_i \sum_j d_{ij} x_{ij}$$

Restrições:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, ..., n \qquad (1)$$

$$\sum_{j=1}^{n} x_{ji} = 1 \quad i = 1, ..., n \qquad (2)$$

$$r_i - r_j + n x_{ij} \leq n - 1 \quad \forall \; i, j \in \{2, ..., n\}, \; i \neq j \qquad (3)$$

$$r_i \geq 0 \quad i = 1, ..., n \qquad (4)$$

$$x_{ij} \in \{0, 1\} \qquad \forall \text{ i,j} \qquad (5)$$

Resolvendo um problema (toy problem)

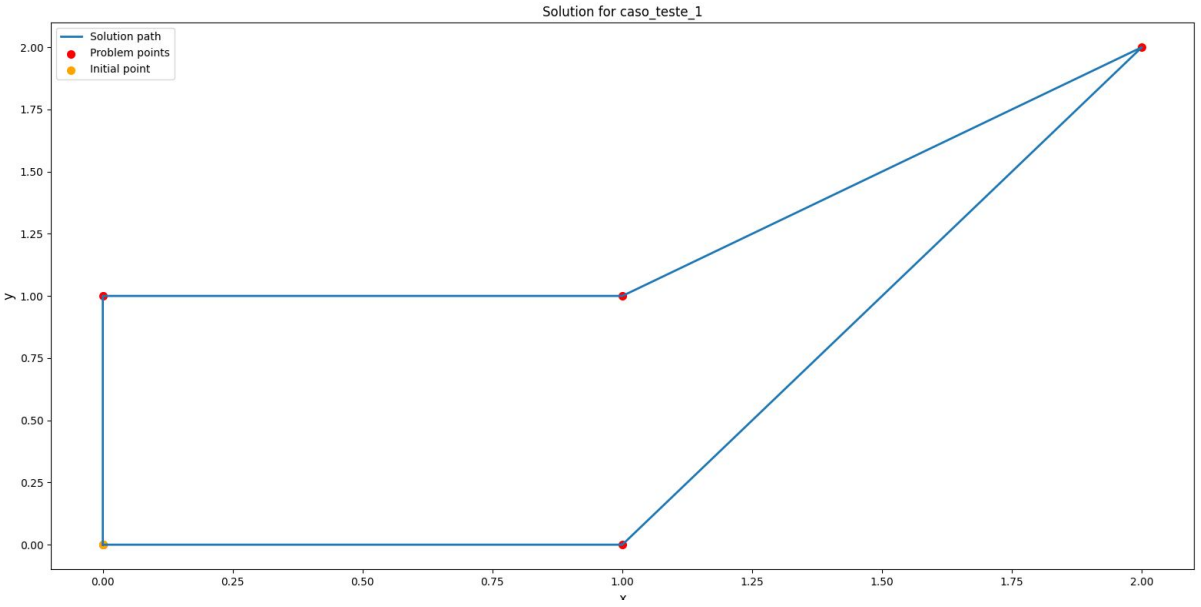
Um problema simples para ser resolvido seria um problema com 5 galáxias (apesar de ter um número reduzido de galáxias, o problema exige muitos cálculos para gerar a resposta correta por conta da complexidade do problema).

Um problema real seria com as seguintes galáxias:

- Galáxia 1: (0.0;0.0)
- Galáxia 2: (1.0;0.0)
- Galáxia 3: (0.0;1.0)
- Galáxia 4: (1.0;1.0)
- Galáxia 5: (2.0;2.0)

Este problema ao ser executado no algoritmo que desenvolvemos que irá definir o modelo que criamos e buscar uma solução para o problema, obtemos o seguinte resultado:

1 → 2 → 5 → 4 → 3 → 1, distância percorrida: 6.65.



Distance travelled: 6.650000

Podemos verificar de que fato esta solução é uma solução básica viável e ótima. Para isso basta verificar que satisfaz todas as restrições o que é trivial de verificar. Além disso, podemos aplicar o método *branch and cut* e verificar que não existe como melhorar esta solução, que como visto em aula, tal condição garante que encontramos solução ótima.

Outros exemplos

Também foram adicionados outros exemplos para serem executados. A lista com o nome dos arquivos com exemple segue a seguir:

- teste-1.tsp
- teste-2.tsp
- teste-3.tsp
- teste-4.tsp
- teste-djibouti.tsp
- teste-western-sahara.tsp
- teste-qatar.tsp
- teste-uruguay.tsp

Heurísticas

A fim de buscar melhor o desempenho do *Solver* para resolver o problema das galáxias, vamos fazer uso de heurísticas para encontrar uma solução inicial. Essa solução será passada ao solver que irá utilizá-la para tentar otimizar a resolução.

Heurísticas Construtivas

Heurística Gulosa

Esta heurística foi baseada na aula da Professora Franklina Maria Bragion de Toledo ICMC-USP. A heurística consiste em, dado uma galáxia inicial, ir sempre para a galáxia mais próxima ainda não visitada. Com isso, podemos calcular um caminho inicial.
Exemplo:

Vamos aplicar tal método ao nosso *toy problem* para ter um entendimento melhor da heurística.

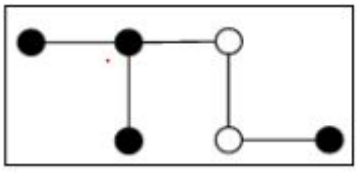
- Começando pela galáxia 1, o algoritmo pode tanto seguir para a galáxia 2 quanto para 3, pois a distância é a menor (1). Vamos assumir que ele irá para a 2.
- Estando na galáxia 2, o algoritmo irá para a galáxia 4 obrigatoriamente, pois a galáxia 1 já foi visitada e a 4 apresenta a menor distância (1)
- Estando na galáxia 4, ele irá para a 3, pois tem a menor distância (1), e como antes, será obrigatoriamente a 3 pois as outras com distância 1 já foram visitadas.
- Estando na galáxia 3 só teremos uma opção, a galáxia 5.
- Agora como todos já foram visitados voltamos para o começa, ou seja galáxia 1.
- Com isso a heurística nos dará a resposta: 1→2→4→3→5→1

Algoritmo de Christofides

O Algoritmo de Christofides trabalha como uma heurística construtiva para encontrar uma solução inicial para o problema do caixeiro viajante. Com isso, tal algoritmo acaba se tornando muito útil para a resolução dos problemas do trabalho. Segue a explicação com imagens retiradas de (referência [11]): PO_II_12_TSP.pdf (ufpr.br) do Professor Volmir Eugênio Wilhelm da UFPR, para facilitar a compreensão do procedimento.

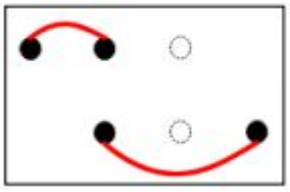
Esse algoritmo produz um caminho cujo custo excede o ótimo em menos de 50%, ou seja, não necessariamente gera a solução ótima.

Primeiramente, monta-se uma árvore geradora mínima para o problema inicial dado, usando as galáxias como nó desse grafo. Chamamos esse grafo de A.



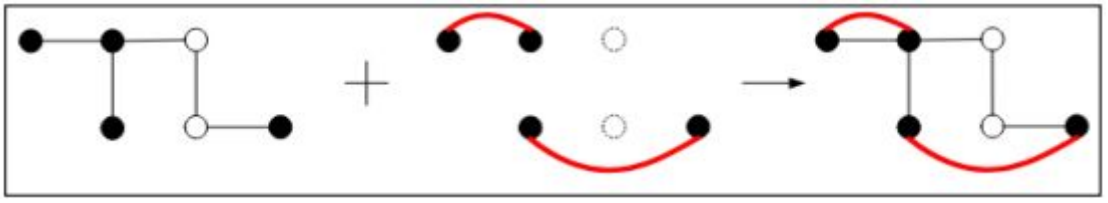
Grafo A

A seguir, selecionamos os nós de grau ímpar e estabelecemos conexões entre eles, de forma a conseguir ligações que reduzem nosso custo, criando um grafo desconexo com vértices 2 a 2, descobrindo assim um emparelhamento perfeito(subconjunto de arcos onde cada nó é o ponto final de uma aresta) entre os nós do grafo, denominando esse grafo desconexo de B.



Grafo B

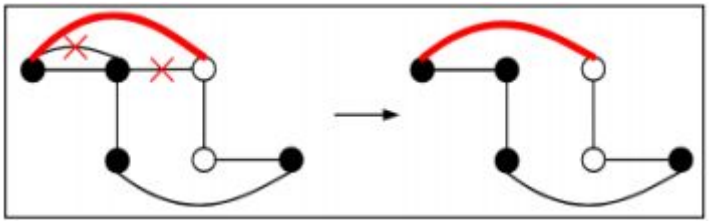
Agora, suponhamos uma C como sendo A U B (C = A U B) sendo assim, teremos um grafo conexo onde cada nó necessariamente tem grau par. Se todos os nós do grafo C apresentam grau 2, o algoritmo termina e a solução encontrada é dada pelo grafo C. Caso contrário, será preciso realizar um "shortcut" que vai gerar um subgrafo conexo com grau 2 em todos os nós.



Grafo C = A U B

O passo denominado shortcut funciona da seguinte maneira: Seja v um nó qualquer do nosso grafo C , com grau no mínimo 4, então existem arestas (u, v) e (v, w) que devem

ser substituídas por uma única aresta (u, w). Após realizar esse passo, o subgrafo permanece conexo e com grau par em todos os nós. Seja N um número par que representa o grau do nó v, após realizar o shortcut o novo grau de v é igual a N-2. Devemos repetir o processo até que todos os nós tenham grau 2.



Fim do Procedimento

Dessa forma se conclui o algoritmo de Christofides para buscar uma solução inicial construtiva para nosso problema.

Heurísticas de Melhoramento

Two-Opt

O algoritmo Two-Opt é, na maioria das vezes, aplicado com a finalidade de melhorar soluções do problema do Caixeiro Viajante. Para a nossa realidade, é possível aplicar tal algoritmo de maneira semelhante à sua função principal - e coincidente com a resolução do Caixeiro Viajante. Para isso, a maneira inicial com que o algoritmo trabalha de forma a desfazer alguns cruzamentos, (pensando num plano cartesiano, onde a aresta que une os vértices tem um tamanho proporcional ao seu custo, tais os cruzamentos normalmente aumentam o custo da rota).

Numa rota original dada e que passa pelas galáxias, e, de modo a realizar tal tarefa, ele troca as rotas duas a duas, e visualiza se a solução encontrada provém alguma otimização ao caminho. Caso essa otimização ocorra, o algoritmo segue trocando as rotas, até que ela pare de ocorrer. Um exemplo de execução do algoritmo (inspirado no exemplo do Wikipedia (Referência [10]) pode ser dado da seguinte forma:
Primeiramente, suponhamos uma rota dada, como:

A → B → C → D → E → F → G

Agora, o algoritmo cria uma nova rota e acrescenta a ela as ‘i’ primeiras galáxias, nesse caso, para i = 2, acrescenta-se A e B. Logo, a nova rota e a antiga se comporão assim:

- Nova rota: A → B
- Antiga rota: A → B → C → D → E → F → G → A

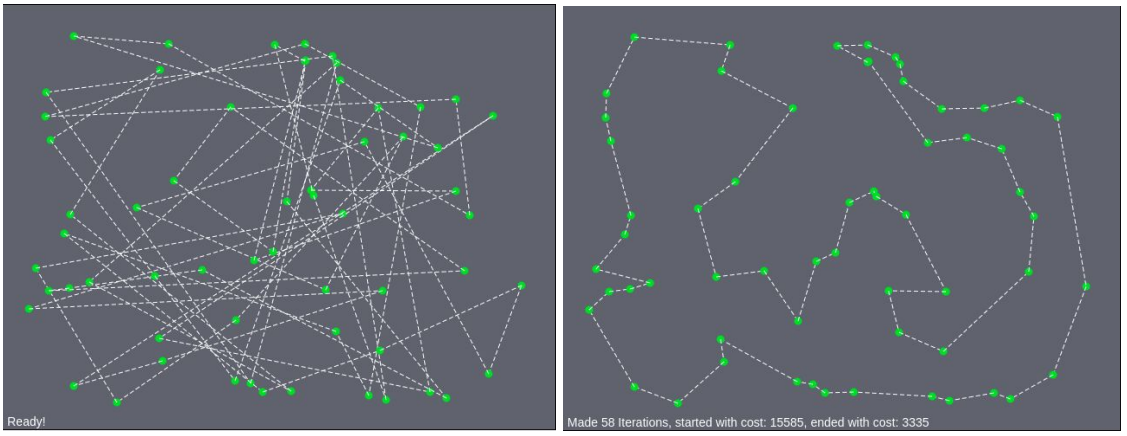
A seguir, o algoritmo usa do trecho que inicia em i+1 e vai à galáxia j, tomemos j = 6 (considerando o código como 1-based, j = 6 equivale a galáxia F) e inverte, obtendo:

- Nova rota: A → B → F → E → D → C
- Antiga rota: A → B → C → D → E → F → G → A

Por fim, o algoritmo acrescenta ao final da nova rota, os vértices de j+1 ao fim da rota antiga, isto é:

- Nova rota: A → B → F → E → D → C → G → A
- Antiga rota: A → B → C → D → E → F → G → A

Se tal procedimento reduziu o custo, continua-se. Caso contrário, há a interrupção.
Dessa forma, segue a ilustração do método Two-Opt:



Pelas imagens acima adicionadas (referência [9]), é possível perceber o grande impacto na redução de custo para traçar um caminho mais curto que passe por todos os nós ao se eliminar os cruzamentos anteriormente existentes.
É importante ressaltar que não há garantia de que esse método encontrará, sempre, a melhor solução.

Focos na Exploração de Nós da Árvore

Além das heurísticas, foram utilizados 2 focos diferentes no *Solver,* estes focos são implementados diretamente pela biblioteca. Por ser uma implementação da biblioteca, não temos detalhes específicos de como ele implementa essas estratégias para explorar a árvore.

Foco na Solução Factível

O primeiro foco que utilizamos é um foco na solução factível, ou seja, o *Solver* irá usar uma estratégia para explorar os nós da árvore buscando encontrar uma solução factível mais rapidamente. Isso pode ajudar a achar uma solução factível em problemas muito grandes onde não é encontrada nenhuma solução

Foco na Solução Ótima

O outro foco que utilizamos é um foco na solução ótima, ou seja, o *Solver* irá usar uma estratégia para explorar os nós da árvore buscando provar que a solução encontrada é ótima. Pode ajudar quando o *Solver* encontra a solução ótima rapidamente.

Testando heurísticas e focos

Vamos testar cada heurística e cada foco agora para tentar resolver os problemas: Western Sahara, Djibouti, Qatar e Uruguay. Iremos mostrar Número de nós explorados na árvore do método *branch and cut*, Número de interações no *simplex* executado dentro do método *branch and cut*, a melhor distância encontrada e a diferença (GAP) para a melhor estimativa para o problema que o método chegou, assim como a diferença (GAP) para a melhor solução conhecida (Waterloo).

Para garantir uma boa comparação entre as heurísticas e focos em um determinado problema, e como o intuito não é a comparação entre os diferentes problemas, pois cada um tem um característica diferente. Então, foi somente necessário que cada teste para um mesmo problema fosse executado em uma mesma máquina. Exemplificando, Western Sahara e todos os testes de heurística e foco foram executados em uma máquina X, enquanto o problema Uruguay e todos os testes em cima dele em uma máquina Y. (Tal método foi usado pois cada exemplo pode chegar a 30 minutos de execução, o que levaria a um tempo enorme para execução de todos os testes) . Segue a lista de cada teste e qual processador foi utilizado.

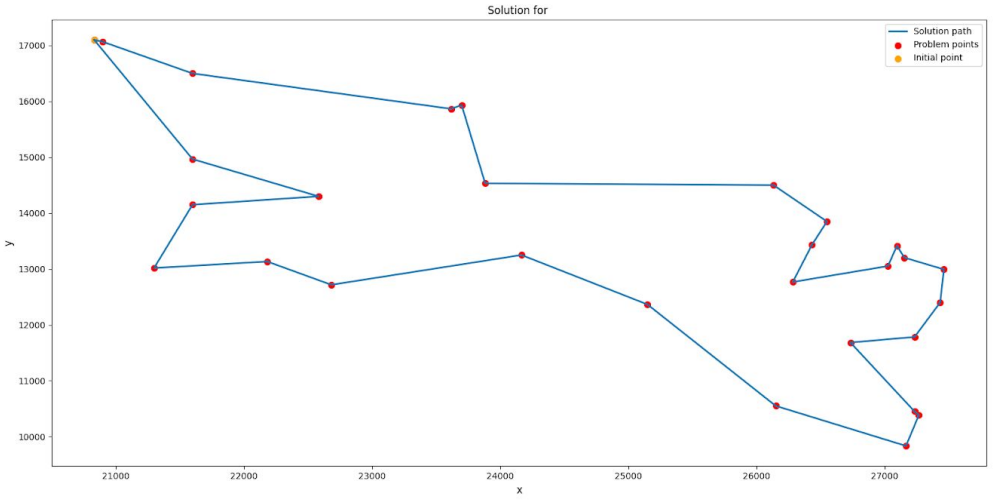
- Western Sahara - Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- Djibouti - Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- Qatar - Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- Uruguay - Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz

Iremos também apresentar os dados em forma gráfica para uma melhor visualização dos dados e fazer uma análise dos resultados.

Western Sahara

Sem heurística

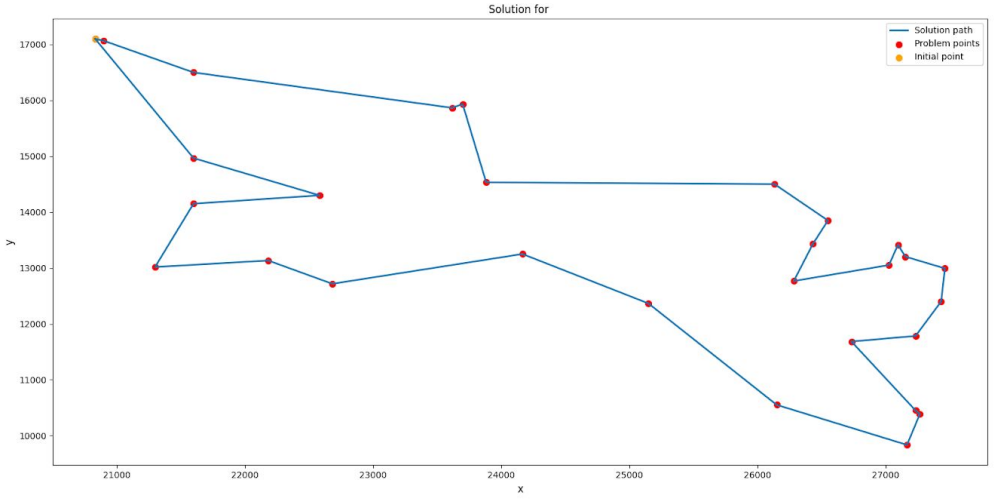
Nós explorados	1115375
Interações Simplex	9224374
Caminho encontrado - Tam.	27601,17377449
Best Bound	27600,54919432
GAP	0,0023%
GAP - Waterloo	-0,00661604%



Caminho: `[1, 2, 6, 10, 11, 12, 15, 19, 18, 17, 21, 22, 23, 29, 28, 26, 20, 25, 27, 24, 16, 14, 13, 9, 7, 3, 4, 8, 5, 1]`

Heurística Gulosa

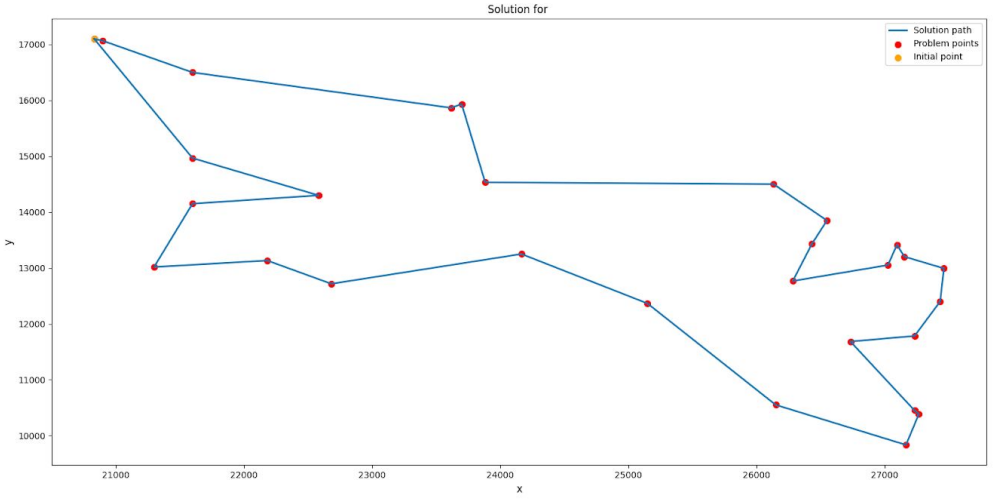
Nós explorados	1428816
Interações Simplex	11787904
Caminho encontrado - Tam.	27601,17377449
Best Bound	27601,17359389
GAP	0%
GAP - Waterloo	-0,00661604%



Caminho: `[1, 2, 6, 10, 11, 12, 15, 19, 18, 17, 21, 22, 23, 29, 28, 26, 20, 25, 27, 24, 16, 14, 13, 9, 7, 3, 4, 8, 5, 1]`

Heurísticas Gulosa + Two-Opt

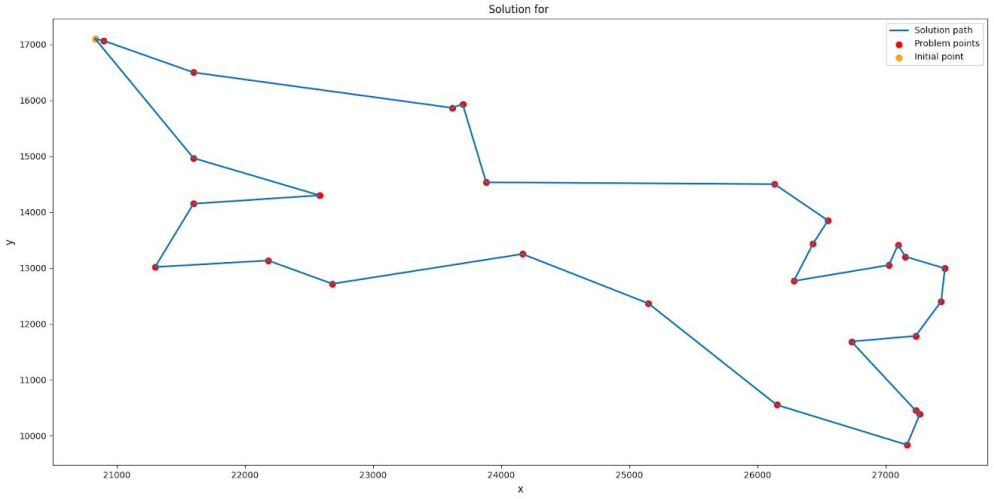
Nós explorados	2052936
Interações Simplex	18444228
Caminho encontrado - Tam.	27601,17377449
Best Bound	27600,2766188
GAP	0,0033%
GAP - Waterloo	-0,00661604%



Caminho: `[1, 2, 6, 10, 11, 12, 15, 19, 18, 17, 21, 22, 23, 29, 28, 26, 20, 25, 27, 24, 16, 14, 13, 9, 7, 3, 4, 8, 5, 1]`

Algoritmo de Christofides

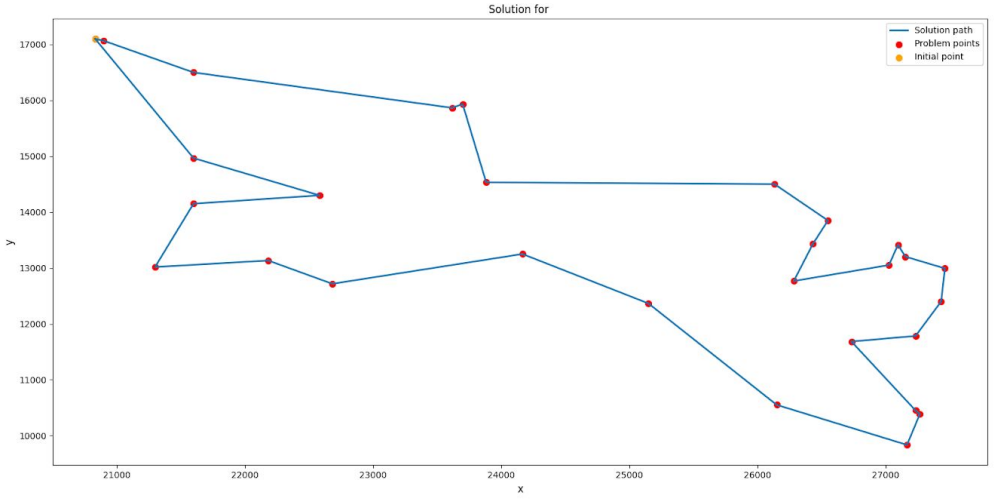
Nós explorados	2144121
Interações Simplex	21947963
Caminho encontrado - Tam.	27601,17377449
Best Bound	27600,2766188
GAP	0,0091%
GAP - Waterloo	-0,00661604%



Caminho: `[1, 5, 8, 4, 3, 7, 9, 13, 14, 16, 24, 27, 25, 20, 26, 28, 29, 23, 22, 21, 17, 18, 19, 15, 12, 11, 10, 6, 2, 1]`

Foco em Melhorar a Solução Primal

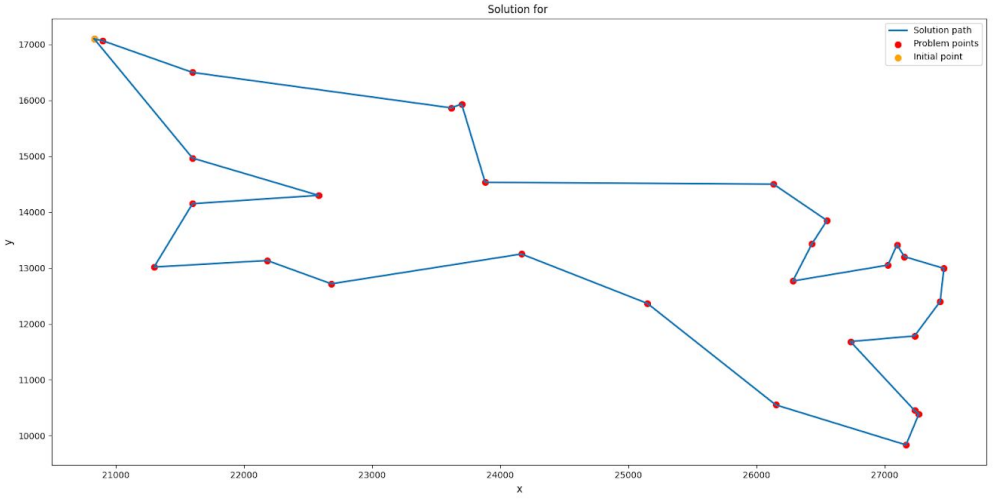
Nós explorados	9940641
Interações Simplex	71662801
Caminho encontrado - Tam.	27601,17377449
Best Bound	27101,80145735
GAP	1,8092%
GAP - Waterloo	-0,00661604%



Caminho: `[1, 5, 8, 4, 3, 7, 9, 13, 14, 16, 24, 27, 25, 20, 26, 28, 29, 23, 22, 21, 17, 18, 19, 15, 12, 11, 10, 6, 2, 1]`

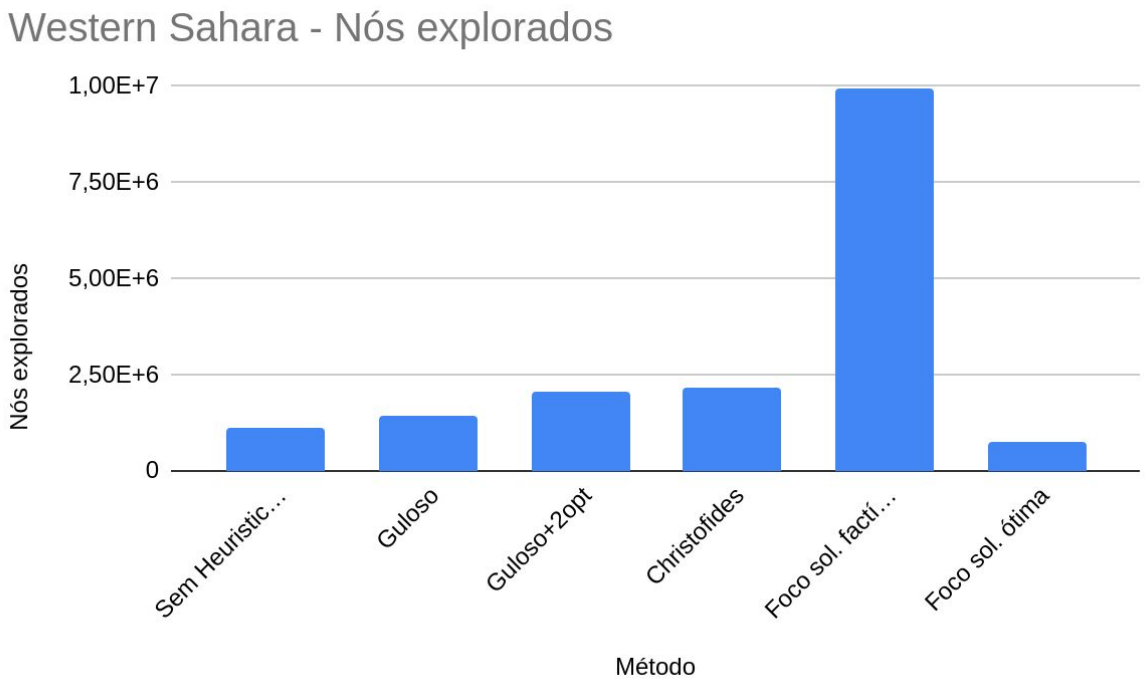
Foco em Encontrar a Solução Ótima

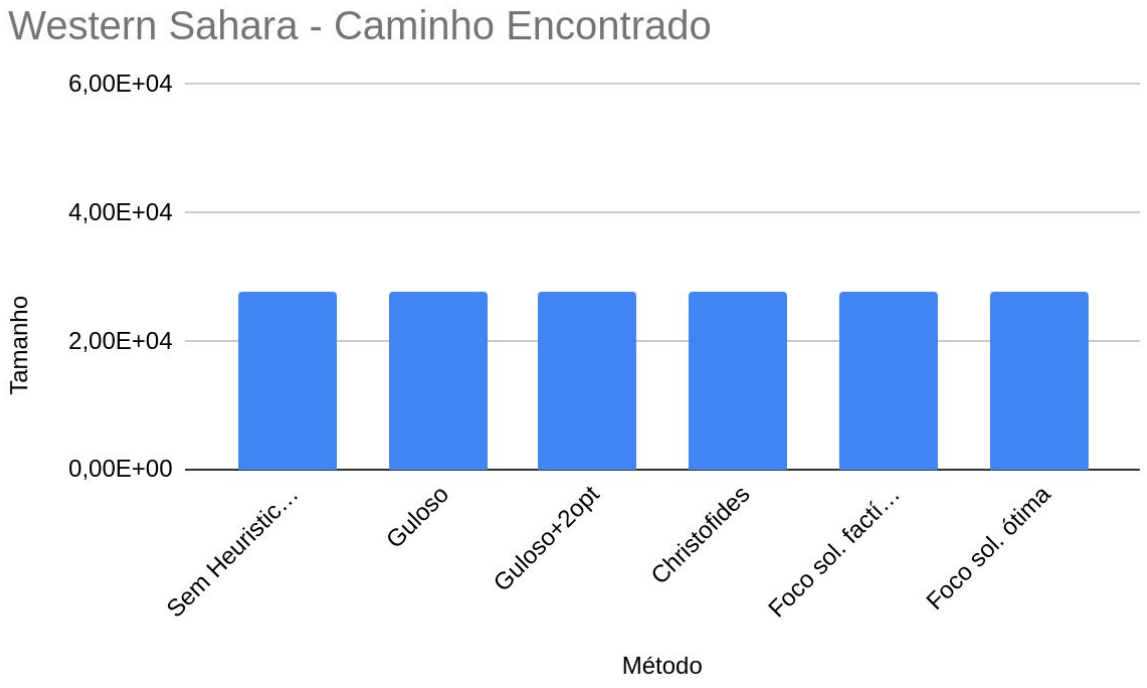
Nós explorados	779588
Interações Simplex	6671188
Caminho encontrado - Tam.	27601,17377449
Best Bound	27601,17229526
GAP	0%
GAP - Waterloo	-0,00661604%

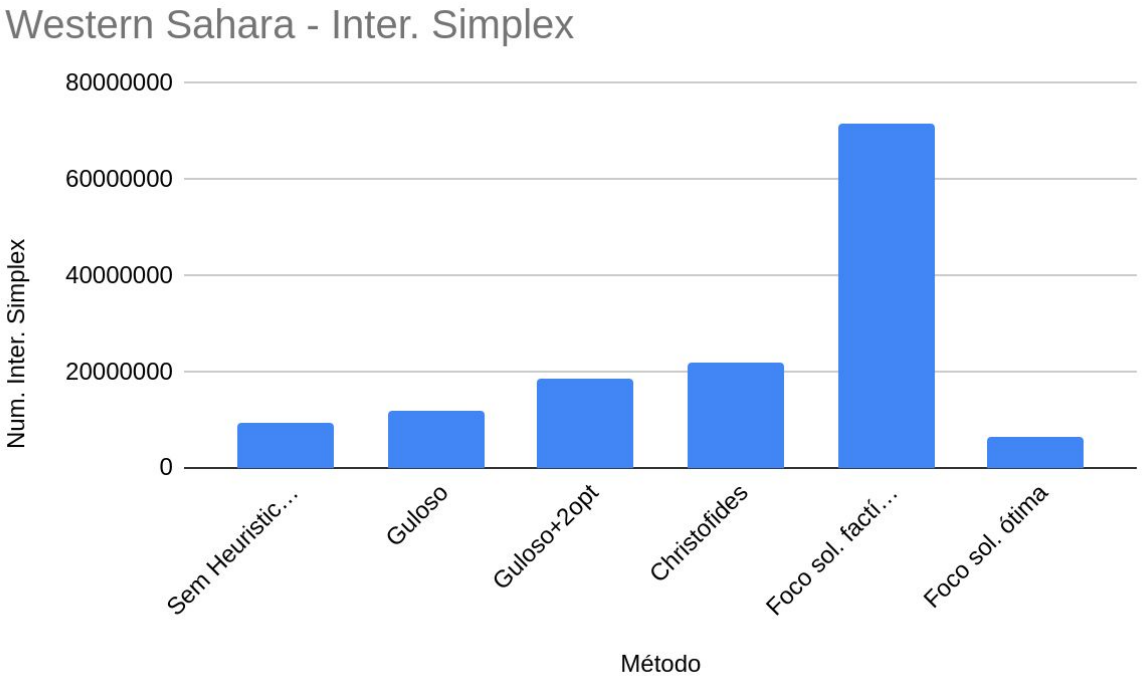


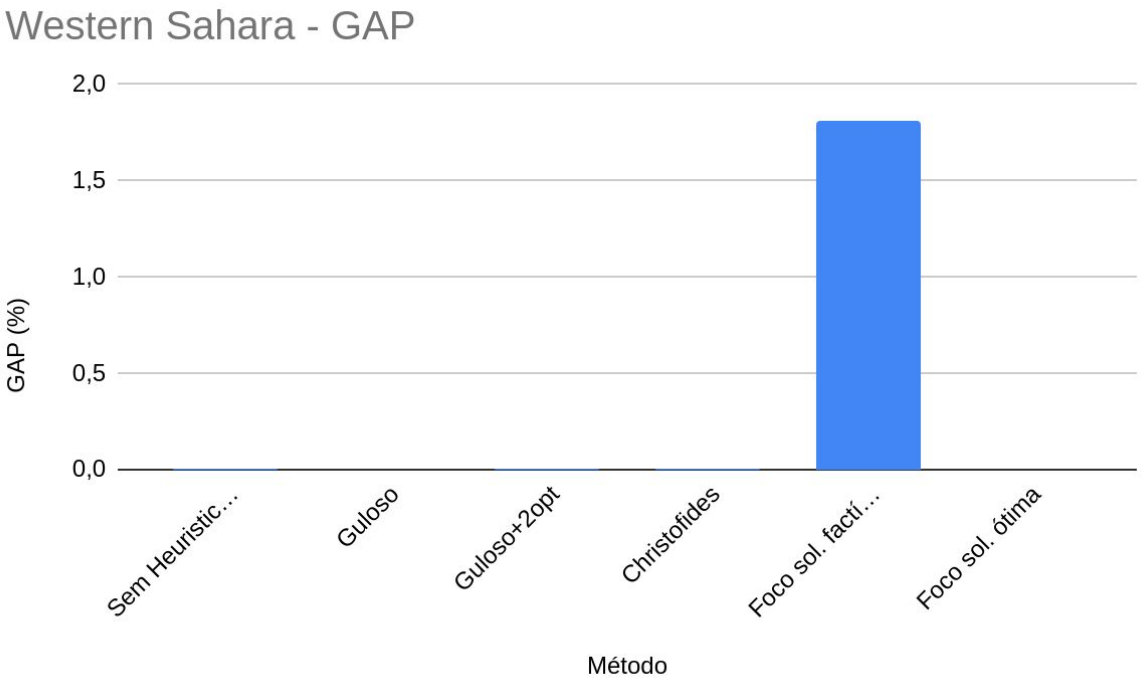
Caminho: [1, 2, 6, 10, 11, 12, 15, 19, 18, 17, 21, 22, 23, 29, 28, 26, 20, 25, 27, 24, 16, 14, 13, 9, 7, 3, 4, 8, 5, 1]

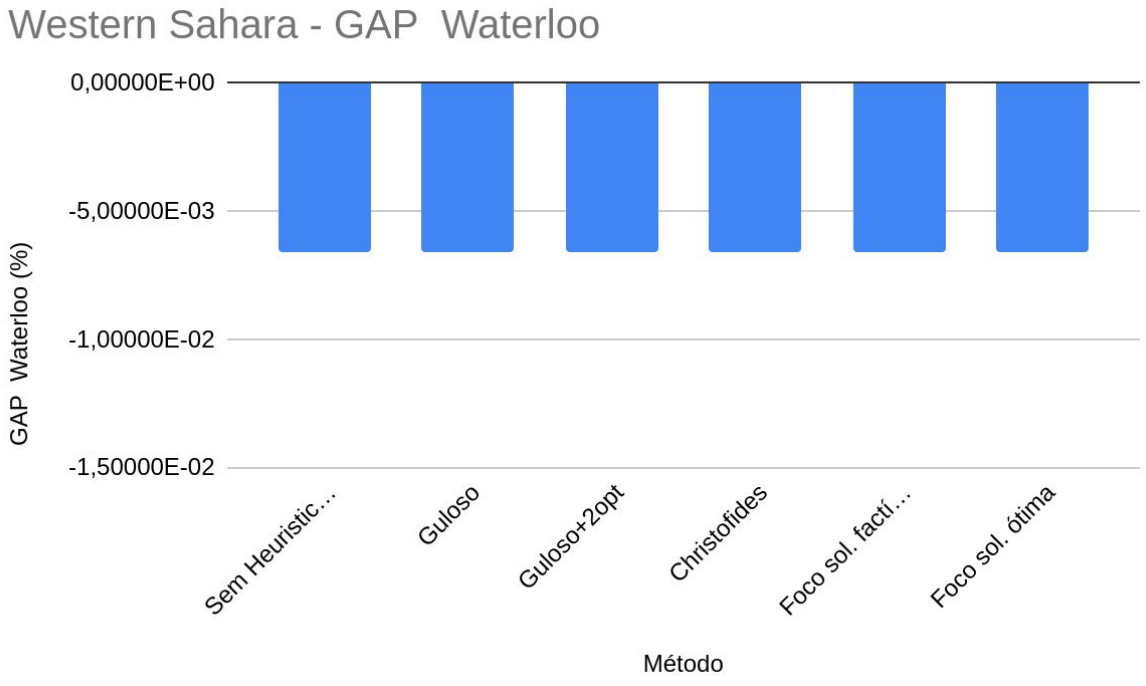
Gráficos

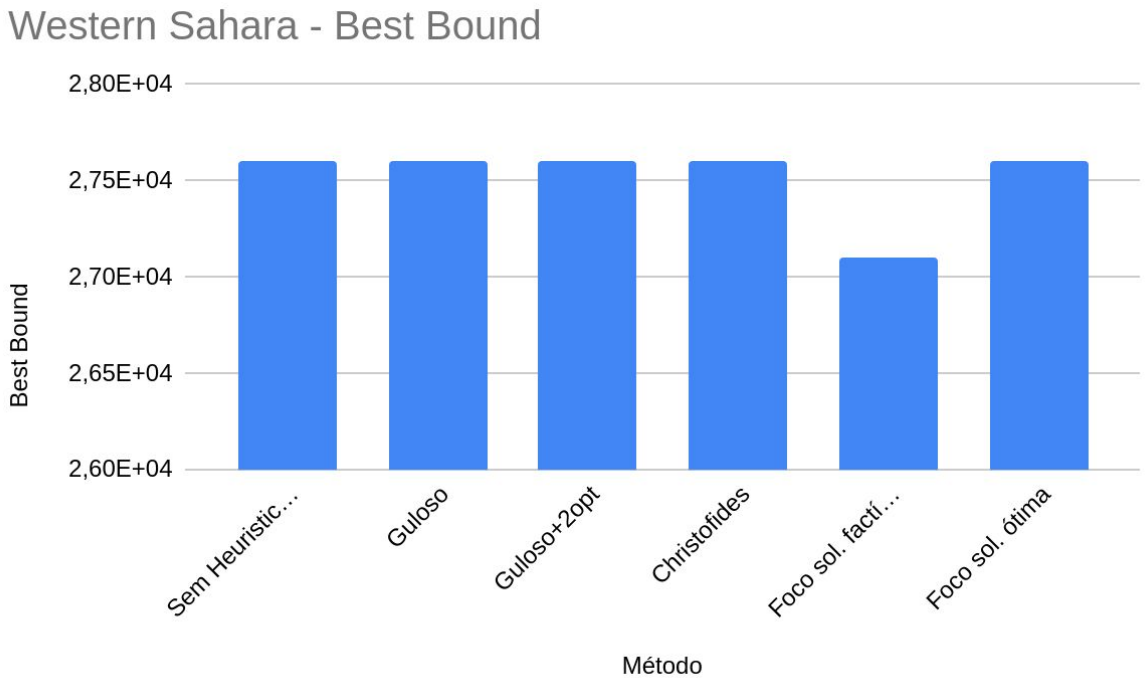

Western Sahara - Nós explorados
1,00E+7
7,50E+6
5,00E+6
2,50E+6
0
Nós explorados
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método

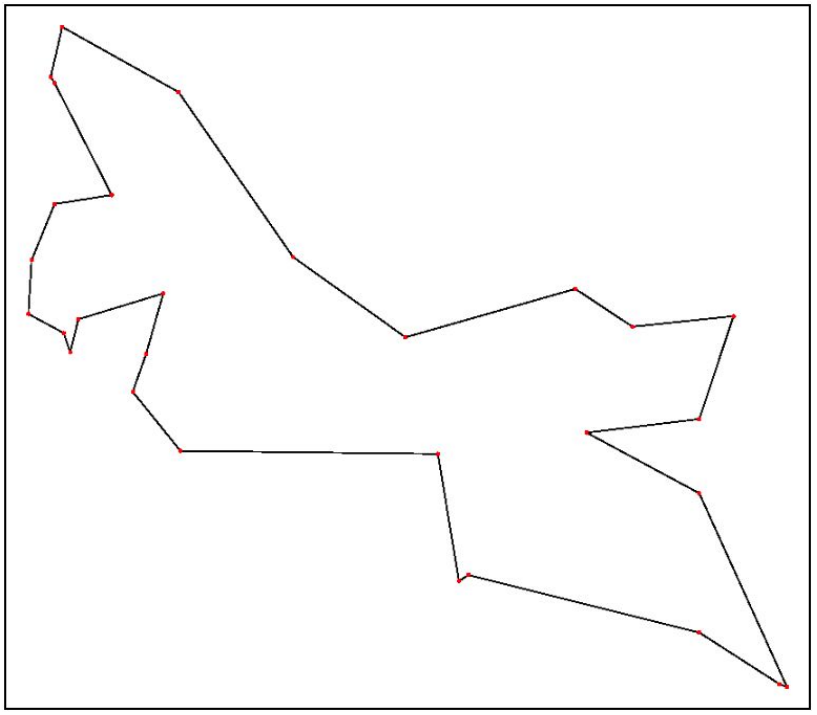

Western Sahara - Caminho Encontrado
6,00E+04
4,00E+04
2,00E+04
0,00E+00
Tamanho
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Western Sahara - Inter. Simplex
80000000
60000000
40000000
20000000
0
Num. Inter. Simplex
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Western Sahara - GAP
2,0
1,5
1,0
0,5
0,0
GAP (%)
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Western Sahara - GAP Waterloo
0,00000E+00
-5,00000E-03
-1,00000E-02
-1,50000E-02
GAP Waterloo (%)
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Western Sahara - Best Bound
2,80E+04
2,75E+04
2,70E+04
2,65E+04
2,60E+04
Best Bound
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método



Solução do Waterloo

Como podemos ver, é bem claro que o foco em solução factível gerou um problema muito maior para o *Solver.* Podemos afirmar isto pois a solução ótima foi encontrada em todos os casos, mas somente neste caso o *Solver* não conseguiu provar a otimalidade da solução, deixando assim um GAP de 1.8%. Isso aconteceu mesmo neste caso tendo explorado muitos mais nós e realizado muito mais interações do simplex.

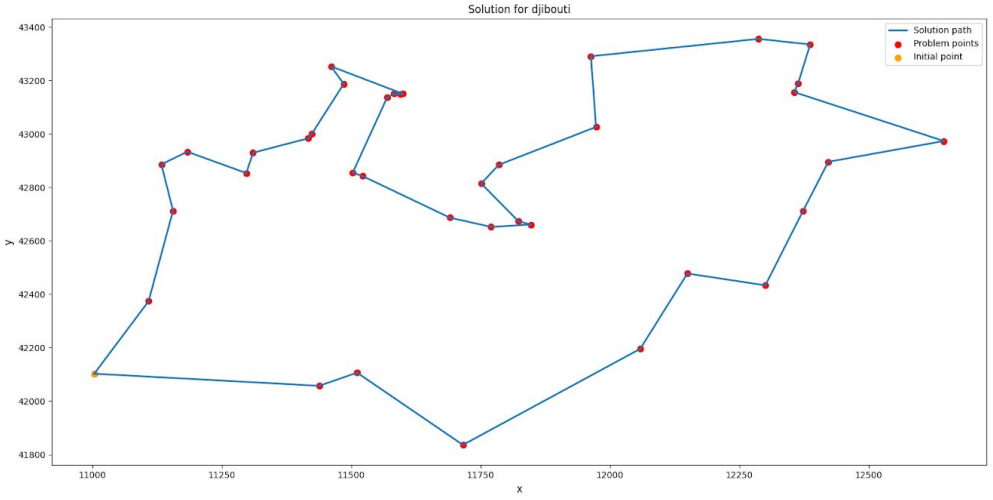
Outro ponto interessante de se notar é que todas as heurísticas pioraram o cenário. Apesar de todas apresentarem uma solução ótima e provar a otimalidade, todas fizeram o *Solver* ter um esforço maior . Isto pode ser visto no primeiro gráfico, onde sem heurística e sem foco apresentou um número menor de nós explorados antes de achar a resposta. Mas o foco na solução ótima conseguiu gerar uma solução ótima diminuindo significativamente o número de nós e interações se comparado a sem heurística e foco, se mostrando ser a melhor opção para este problema.

Outro ponto que pode parecer estranho à primeira vista é o GAP com a melhor solução do Waterloo. Pois, apesar de achar a resposta ótima, ele apresenta um GAP negativo, mas era esperado 0 de GAP uma vez que no Waterloo também tem uma resposta ótima. Isto ocorre pois não temos precisão infinita para realizar cálculos no computador forçando a própria linguagem a realizar arredondamentos para que o número caiba dentro do máximo que a arquitetura permite (isso é uma questão da forma como os computadores funcionam). Isso faz com que, dependendo da maneira como se lida com este problema, terá uma resposta diferente, mas claro que a diferença é quase 0. É possível ver que de fato a diferença é quase 0, pois temos um GAP para o Waterloo de apenas -0,00661604% em todos os casos, ou seja, desprezível neste caso.

Djibouti

Sem heurística

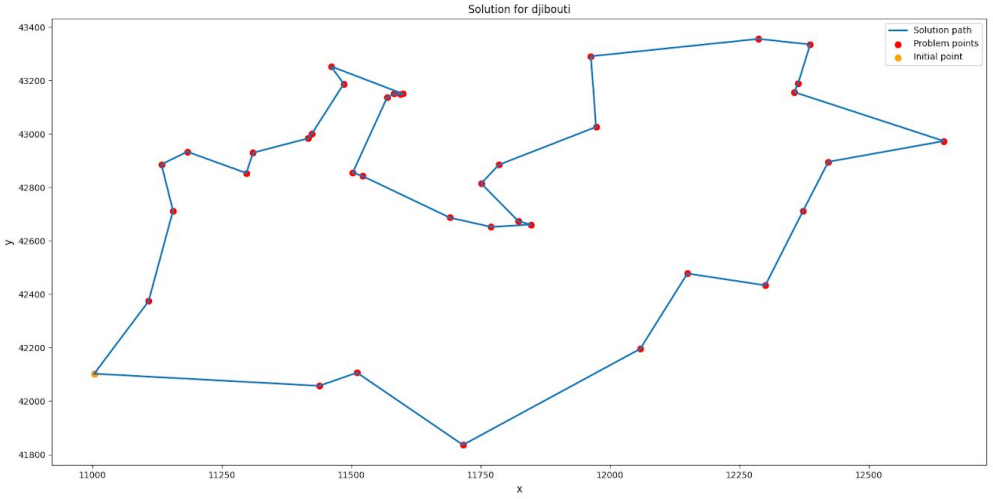
Nós explorados	7740
Interações Simplex	120016
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%



Caminho: *[1, 10, 14, 21, 29, 30, 32, 35, 37, 38, 33, 34, 36, 31, 27, 28, 24, 22, 25, 26, 23, 20, 15, 13, 16, 17, 18, 19, 11, 12, 9, 8, 7, 6, 5, 3, 4, 2, 1]*

Heurística Gulosa

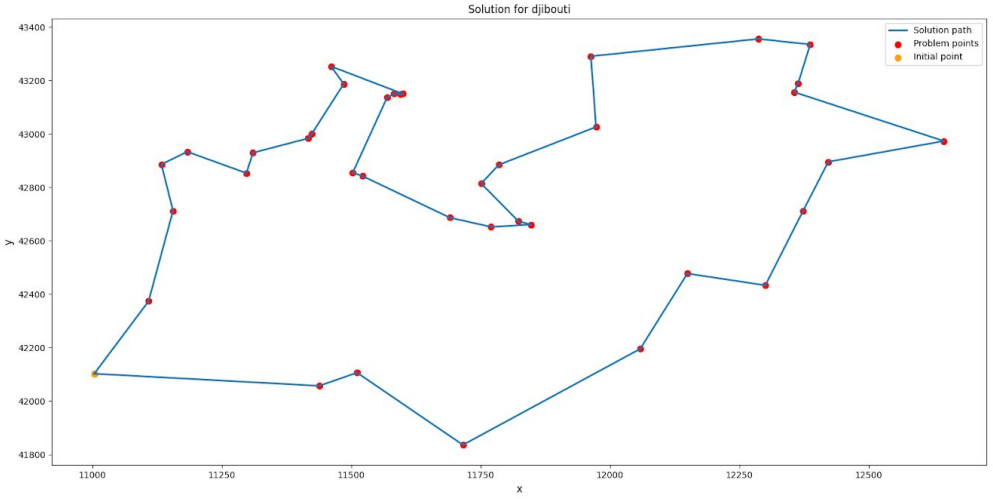
Nós explorados	8293
Interações Simplex	136413
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%



Caminho: `[1, 10, 14, 21, 29, 30, 32, 35, 37, 38, 33, 34, 36, 31, 27, 28, 24, 22, 25, 26, 23, 20, 15, 13, 16, 17, 18, 19, 11, 12, 9, 8, 7, 6, 5, 3, 4, 2, 1]`

Heurísticas Gulosa + Two-Opt

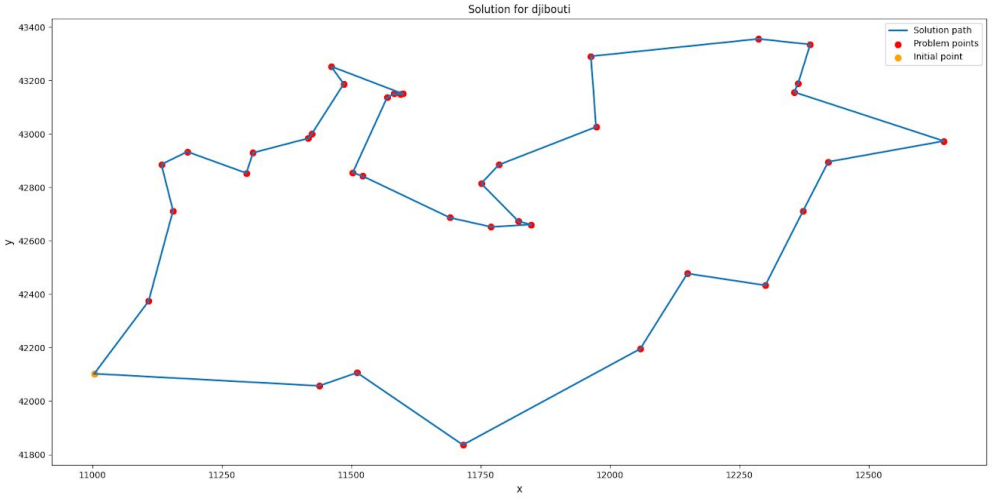
Nós explorados	4609
Interações Simplex	75917
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%



Caminho: `[1, 2, 4, 3, 5, 6, 7, 8, 9, 12, 11, 19, 18, 17, 16, 13, 15, 20, 23, 26, 25, 22, 24, 28, 27, 31, 36, 34, 33, 38, 37, 35, 32, 30, 29, 21, 14, 10, 1]`

Algoritmo de Christofides

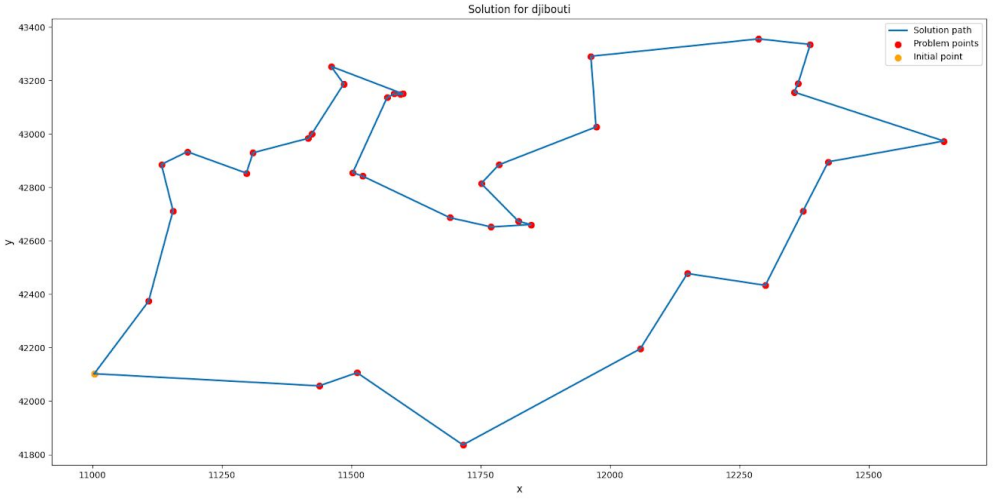
Nós explorados	9034
Interações Simplex	162867
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%



Caminho: `[1, 10, 14, 21, 29, 30, 32, 35, 37, 38, 33, 34, 36, 31, 27, 28, 24, 22, 25, 26, 23, 20, 15, 13, 16, 17, 18, 19, 11, 12, 9, 8, 7, 6, 5, 3, 4, 2, 1]`

Foco em Melhorar a Solução Primal

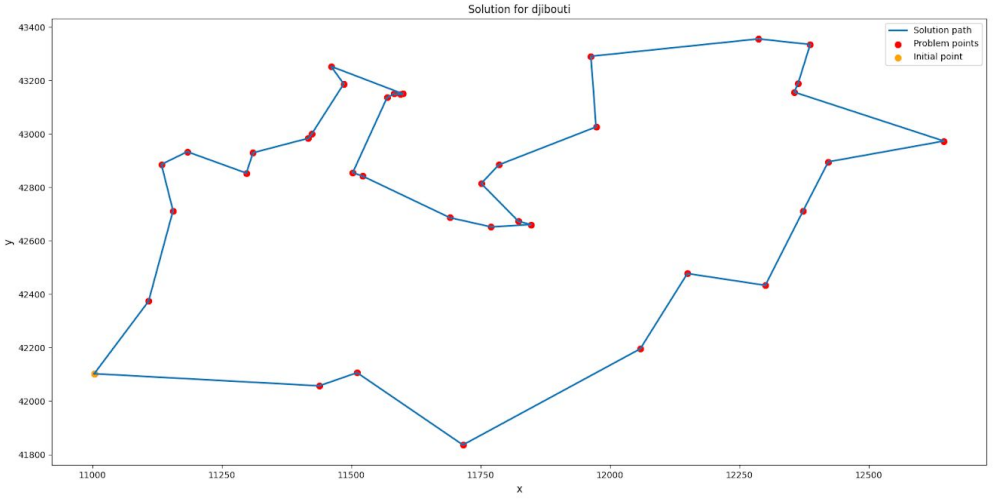
Nós explorados	33856
Interações Simplex	409873
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%



Caminho: `[1, 10, 14, 21, 29, 30, 32, 35, 37, 38, 33, 34, 36, 31, 27, 28, 24, 22, 25, 26, 23, 20, 15, 13, 16, 17, 18, 19, 11, 12, 9, 8, 7, 6, 5, 3, 4, 2, 1]`

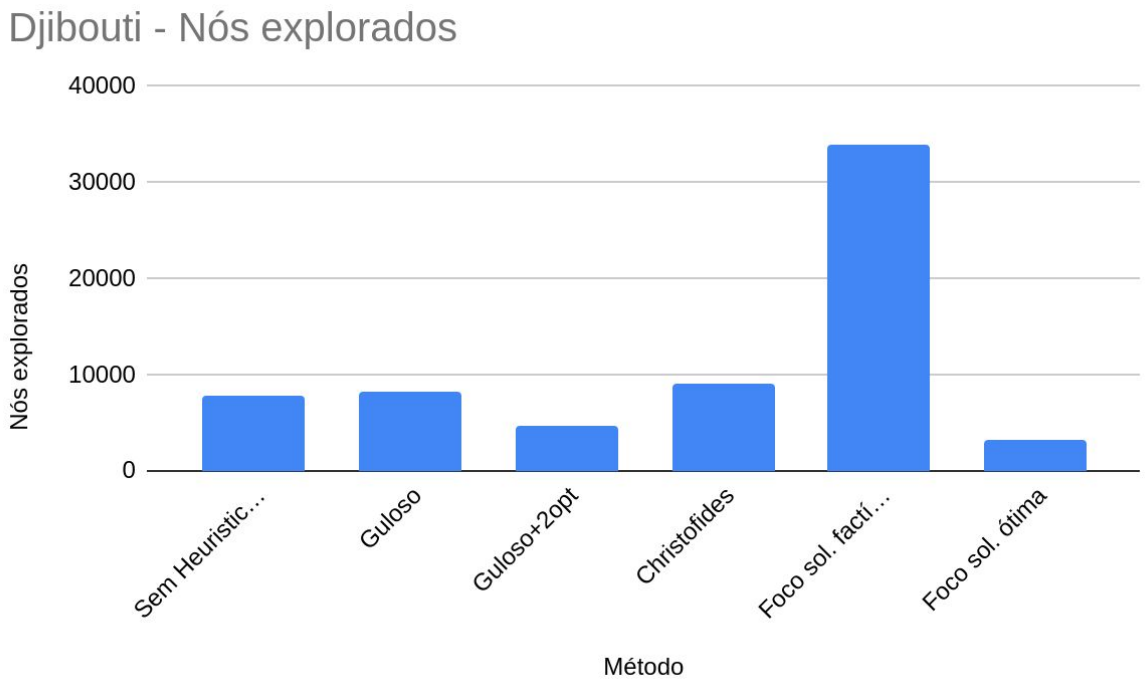
Foco em Encontrar a Solução ótima

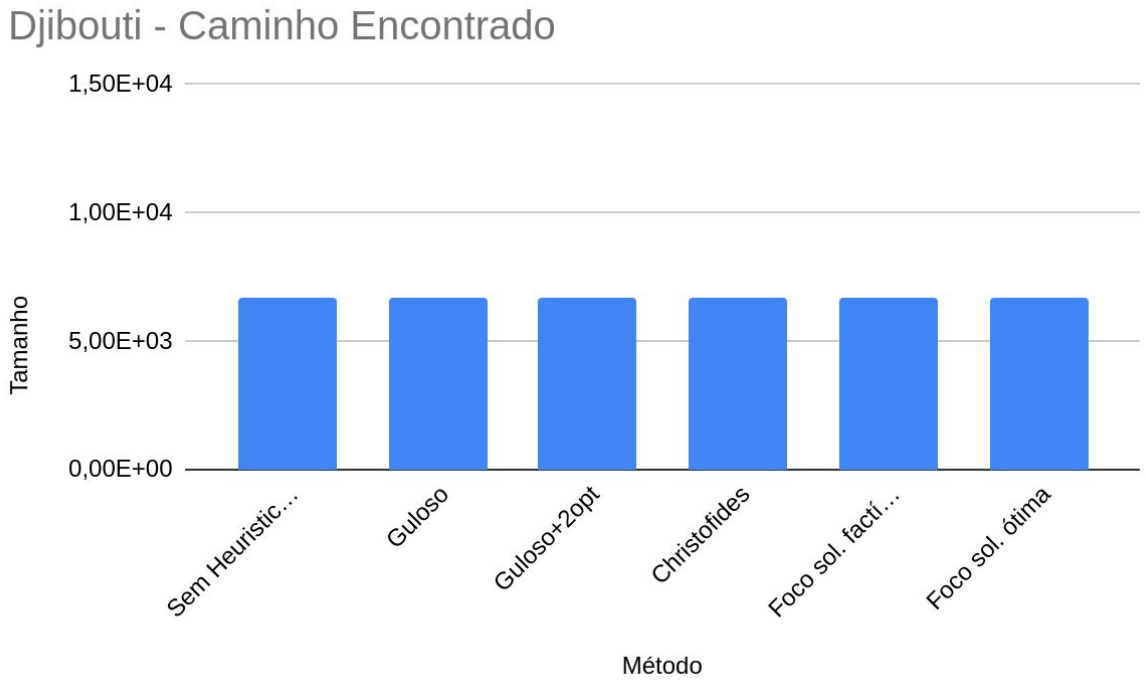
Nós explorados	3301
Interações Simplex	54992
Caminho encontrado - Tam.	6659,431532931
Best Bound	6659,431532931
GAP	0%
GAP - Waterloo	0,05155548%

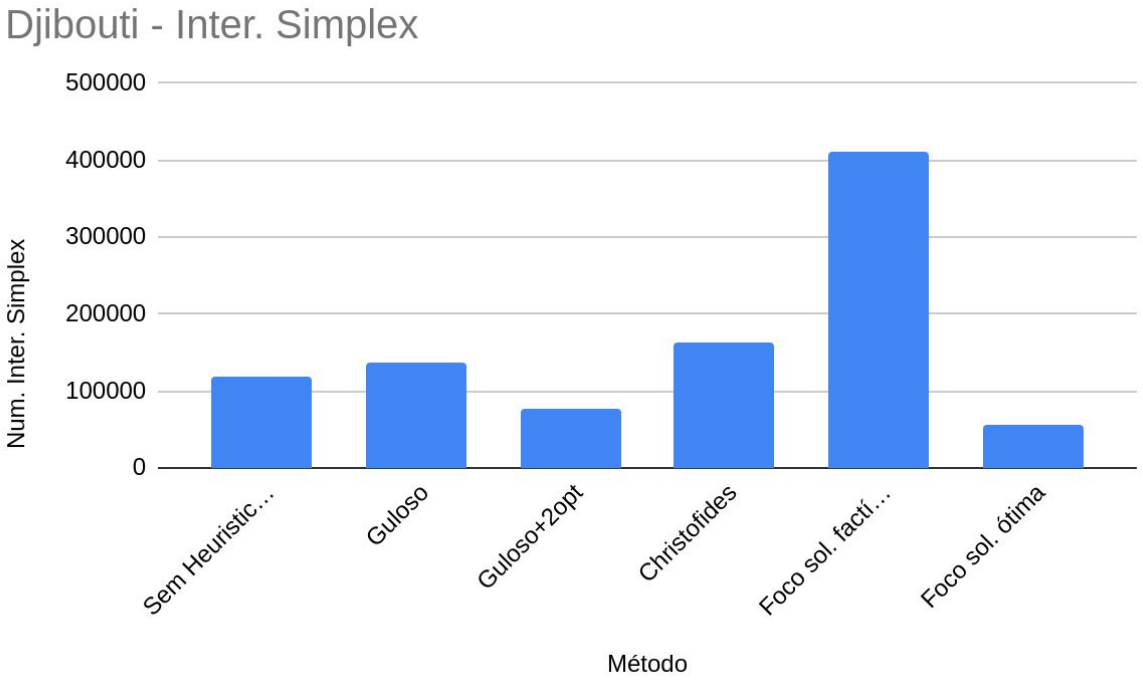


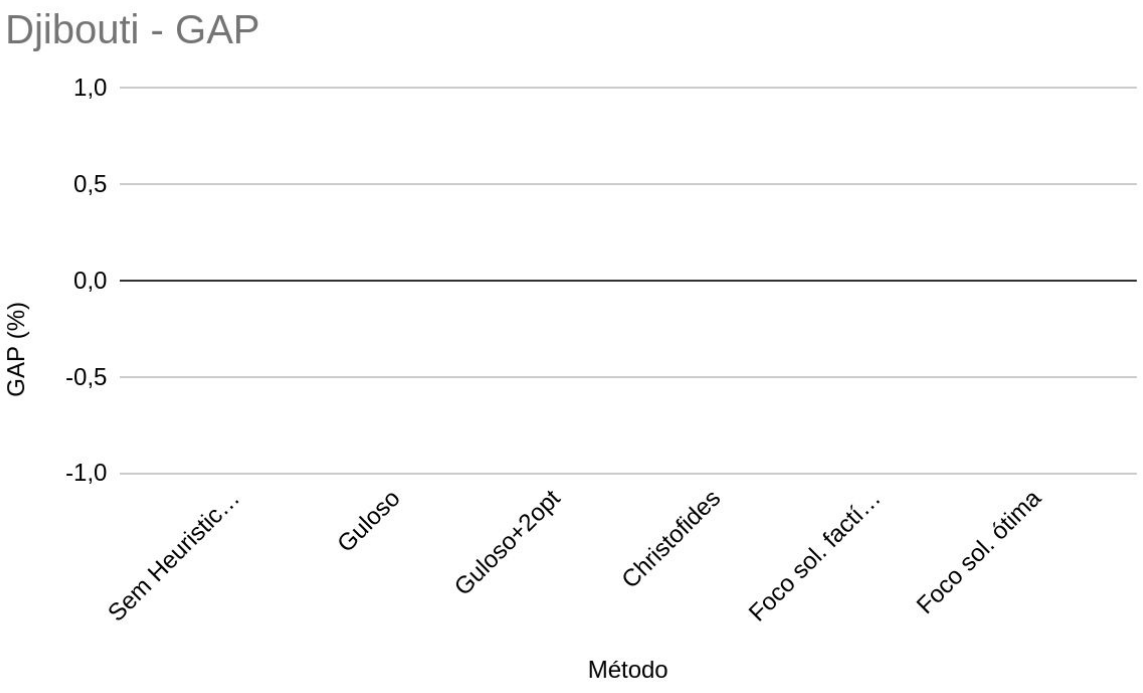
Caminho: `[1, 10, 14, 21, 29, 30, 32, 35, 37, 38, 33, 34, 36, 31, 27, 28, 24, 22, 25, 26, 23, 20, 15, 13, 16, 17, 18, 19, 11, 12, 9, 8, 7, 6, 5, 3, 4, 2, 1]`

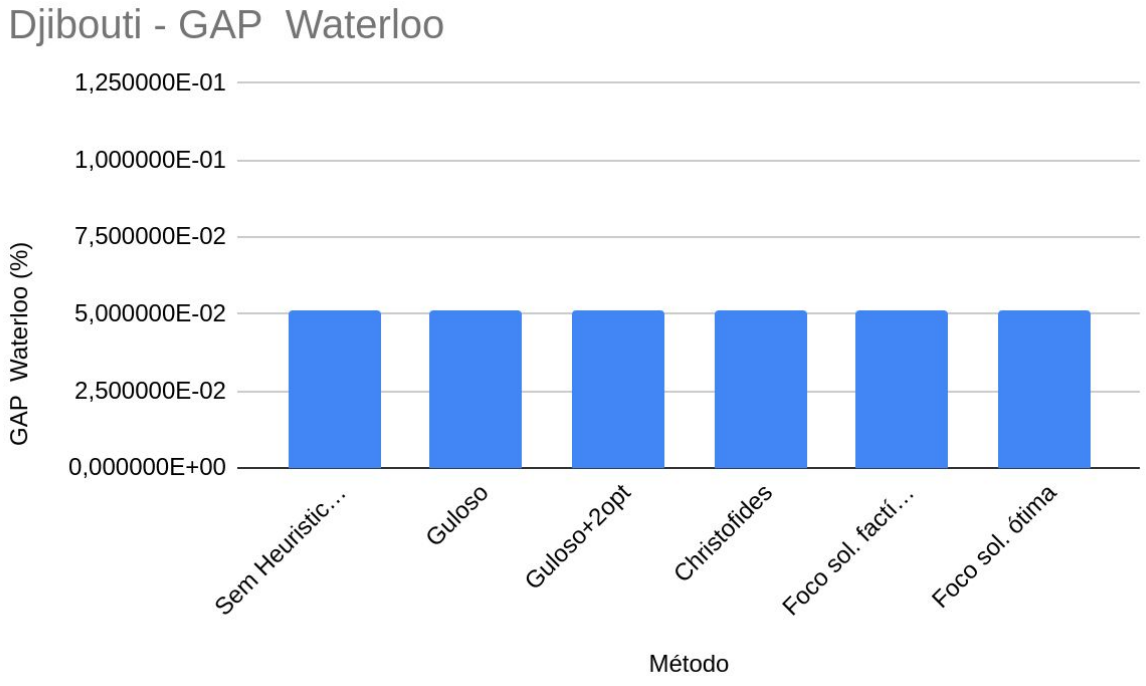
Gráficos

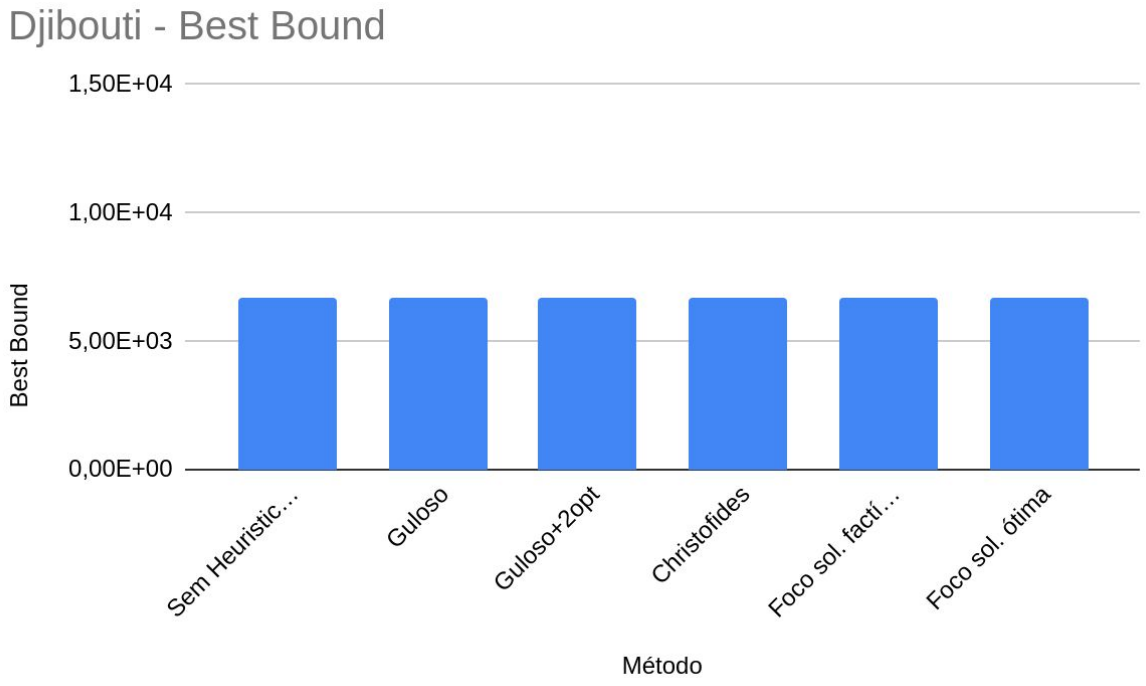

Djibouti - Nós explorados
40000
30000
20000
10000
0
Nós explorados
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método

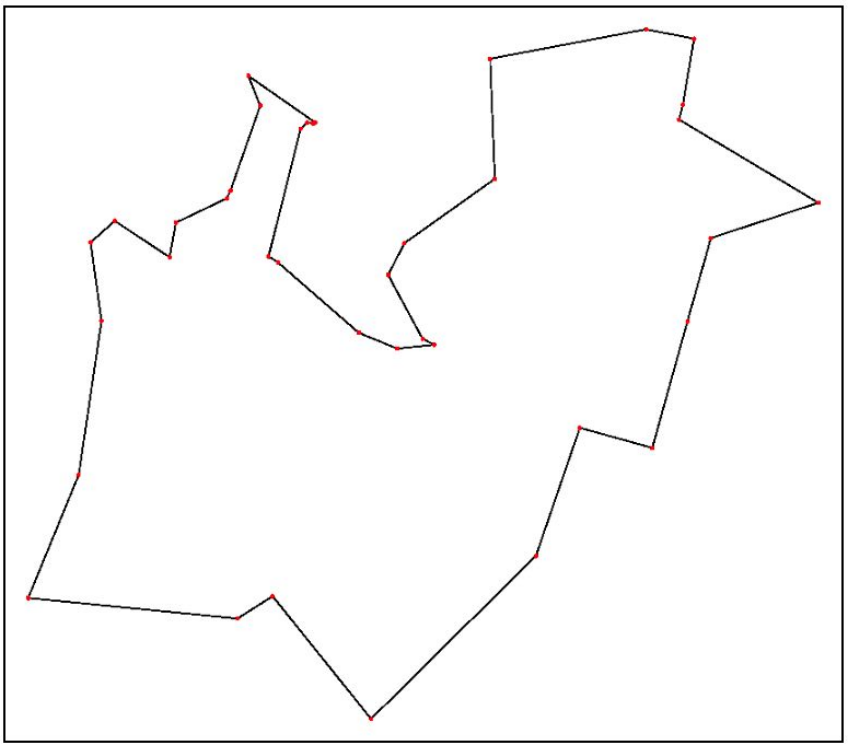

Djibouti - Caminho Encontrado
1,50E+04
1,00E+04
5,00E+03
0,00E+00
Tamanho
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Djibouti - Inter. Simplex
500000
400000
300000
200000
100000
0
Num. Inter. Simplex
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Djibouti - GAP
1,0
0,5
0,0
-0,5
-1,0
GAP (%)
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Djibouti - GAP Waterloo
1,250000E-01
1,000000E-01
7,500000E-02
5,000000E-02
2,500000E-02
0,000000E+00
GAP Waterloo (%)
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Djibouti - Best Bound
1,50E+04
1,00E+04
5,00E+03
0,00E+00
Best Bound
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método



Solução do Waterloo

Este problema apresentou resultados semelhantes ao anterior (Western Sahara), onde o foco em solução factível apresentou uma dificuldade para o *Solver* de provar a otimalidade. Mas diferente do caso anterior, apesar de precisar explorar muito mais nós para isto, o *Solver* conseguiu provar a otimalidade. Isto pode ser verificado no gráfico do GAP que apresenta GAP de 0, ou seja, a diferença entre o Primal e o Dual é 0. Vale dizer que todos os testes encontraram a solução ótima e provaram a otimalidade neste *Dataset*.

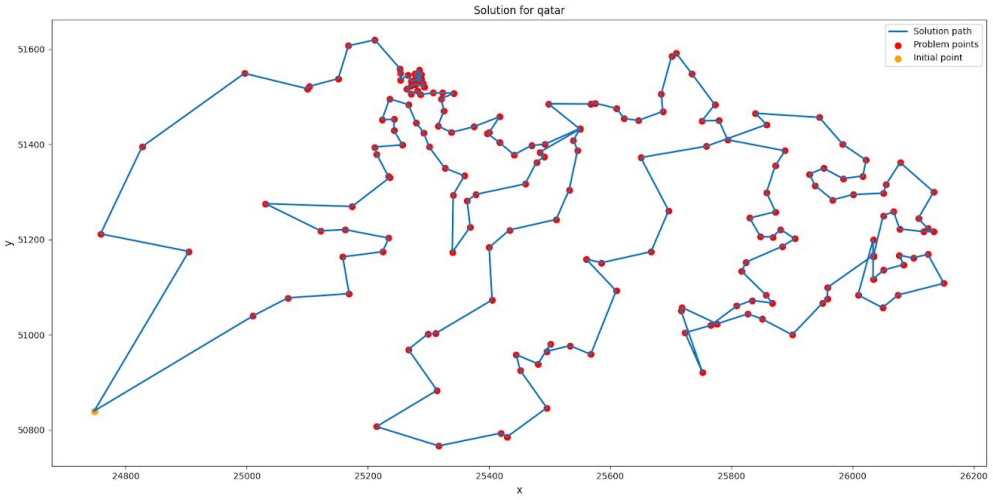
Outro ponto interessante é que a heurística conseguiu ajudar o *Solver* desta vez. Apesar do Guloso e do Christofides precisarem de um pouco mais de nós e interações do *Simplex* comprado ao *Solver* sem nada para resolver o problema, o Guloso com o Two-Opt conseguiu diminuir os 2, nós exploramos e interações. Mas apesar disto, o foco na solução ótima se mostrou novamente a melhor opção para este *Dataset*, pois foi o que apresentou melhor resultado.

Podemos ver que assim como no caso anterior, apesar de achar a solução ótima e o Waterloo também possuir uma solução ótima, o GAP para o Waterloo não é 0. Isto ocorre pelo mesmo motivo e como podemos ver o GAP para o Waterloo é quase 0 também, com um valor de apenas 0,05155548% para todos os casos.

Qatar

Sem heurística

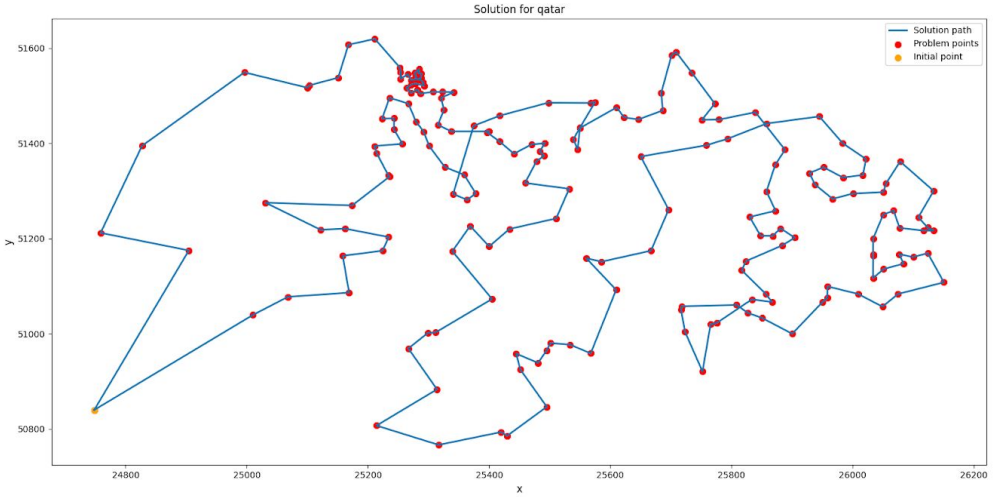
Nós explorados	21226
Interações Simplex	707218
Caminho encontrado - Tam.	9968,005145461
Best Bound	9273,210439157
GAP	6,9702%
GAP - Waterloo	6,58688137%



Caminho: *[1, 4, 2, 3, 5, 9, 10, 12, 15, 19, 30, 32, 31, 35, 38, 41, 46, 48, 54, 49, 44, 55, 50, 42, 52, 53, 56, 58, 43, 40, 34, 39, 47, 51, 61, 67, 73, 66, 68, 64, 70, 77, 84, 81, 79, 83, 88, 92, 97, 108, 105, 106, 103, 102, 87, 80, 82, 62, 59, 36, 63, 20, 65, 85, 86, 98, 90, 89, 94, 101, 99, 104, 111, 114, 109, 113, 119, 122, 118, 131, 148, 143, 160, 166, 171, 170, 167, 162, 158, 159, 165, 168, 178, 180, 185, 193, 188, 191, 192, 189, 184, 181, 177, 174, 172, 179, 186, 183, 187, 190, 194, 182, 176, 169, 175, 173, 164, 163, 161, 156, 145, 140, 134, 126, 125, 130, 127, 132, 137, 142, 149, 146, 138, 139, 154, 157, 153, 150, 144, 141, 152, 147, 151, 155, 136, 135, 129, 133, 128, 124, 123, 120, 121, 117, 116, 115, 112, 110, 100, 107, 95, 96, 93, 91, 78, 75, 76, 71, 72, 74, 69, 60, 57, 45, 37, 27, 22, 29, 28, 33, 18, 21, 24, 26, 17, 7, 11, 14, 25, 23, 13, 16, 8, 6, 1]*

Heurística Gulosa

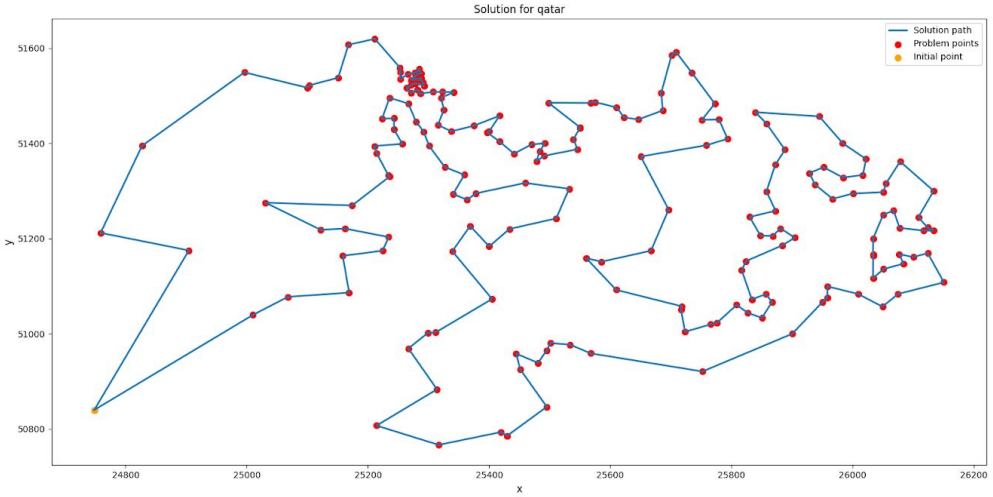
Nós explorados	15425
Interações Simplex	610233
Caminho encontrado - Tam.	9626,508875652
Best Bound	9282,184335728
GAP	3,5768%
GAP - Waterloo	2,93529593%



Caminho: *[1, 4, 2, 3, 5, 9, 10, 12, 15, 19, 30, 32, 31, 35, 38, 41, 46, 44, 42, 50, 49, 55, 54, 48, 52, 53, 56, 58, 43, 40, 34, 39, 47, 51, 61, 67, 73, 66, 68, 64, 70, 81, 79, 83, 88, 92, 97, 95, 96, 93, 91, 103, 102, 87, 80, 76, 71, 82, 62, 59, 36, 63, 20, 65, 85, 86, 98, 90, 89, 94, 99, 101, 104, 111, 114, 109, 113, 119, 122, 118, 131, 136, 148, 160, 166, 171, 170, 167, 162, 158, 159, 165, 168, 178, 180, 185, 193, 188, 191, 192, 189, 184, 181, 177, 175, 173, 174, 172, 179, 186, 183, 187, 190, 194, 182, 176, 169, 164, 163, 161, 156, 145, 140, 137, 126, 125, 127, 130, 132, 134, 142, 149, 146, 138, 139, 154, 157, 153, 150, 144, 141, 152, 147, 151, 155, 143, 135, 129, 133, 128, 124, 123, 120, 121, 117, 116, 115, 107, 106, 105, 108, 112, 110, 100, 84, 77, 72, 75, 78, 74, 69, 60, 57, 45, 37, 27, 22, 29, 28, 33, 18, 21, 24, 26, 17, 7, 11, 14, 25, 23, 13, 16, 8, 6, 1]*

Heurísticas Gulosa + Two-Opt

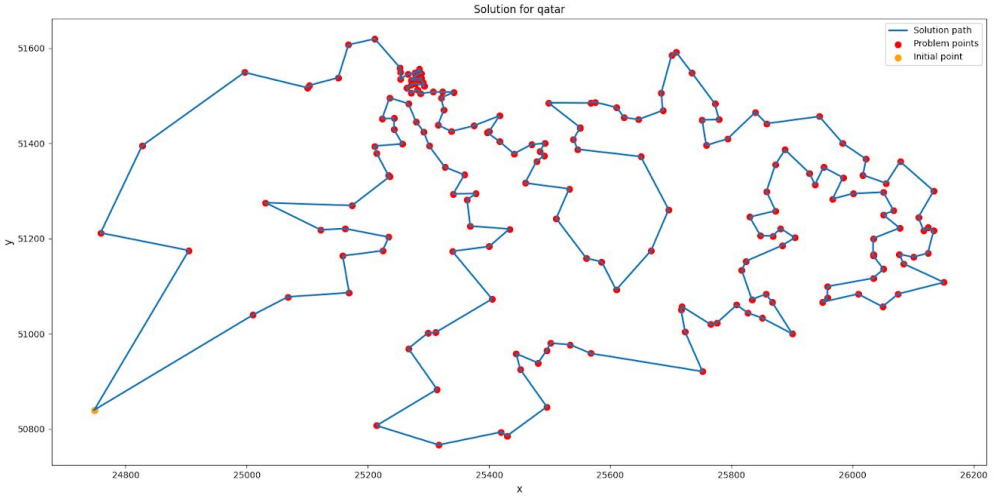
Nós explorados	21415
Interações Simplex	1651941
Caminho encontrado - Tam.	9407,405036515
Best Bound	9282,263080774
GAP	1,3302%
GAP - Waterloo	0,592440510%



Caminho: *[1, 4, 2, 3, 5, 9, 10, 12, 15, 19, 30, 32, 31, 35, 42, 50, 55, 49, 44, 38, 41, 46, 48, 54, 52, 53, 56, 58, 43, 40, 34, 39, 47, 51, 61, 67, 73, 66, 68, 64, 70, 77, 84, 81, 79, 83, 88, 92, 97, 95, 93, 96, 106, 105, 107, 108, 100, 110, 112, 115, 116, 117, 121, 120, 123, 124, 128, 133, 129, 135, 136, 131, 118, 122, 119, 113, 109, 114, 126, 125, 127, 132, 134, 137, 140, 145, 149, 146, 142, 138, 139, 154, 157, 153, 150, 144, 141, 152, 147, 151, 155, 148, 143, 160, 166, 171, 170, 167, 162, 158, 159, 165, 168, 178, 180, 185, 193, 188, 191, 192, 189, 184, 181, 177, 175, 173, 174, 172, 179, 186, 183, 187, 190, 194, 182, 176, 169, 164, 163, 161, 156, 130, 111, 104, 101, 99, 94, 89, 90, 98, 86, 85, 65, 20, 63, 36, 59, 62, 82, 71, 76, 80, 87, 102, 103, 91, 78, 75, 72, 74, 69, 60, 57, 45, 37, 27, 22, 29, 28, 33, 18, 21, 24, 26, 17, 7, 11, 14, 25, 23, 13, 16, 8, 6, 1]*

Algoritmo de Christofides

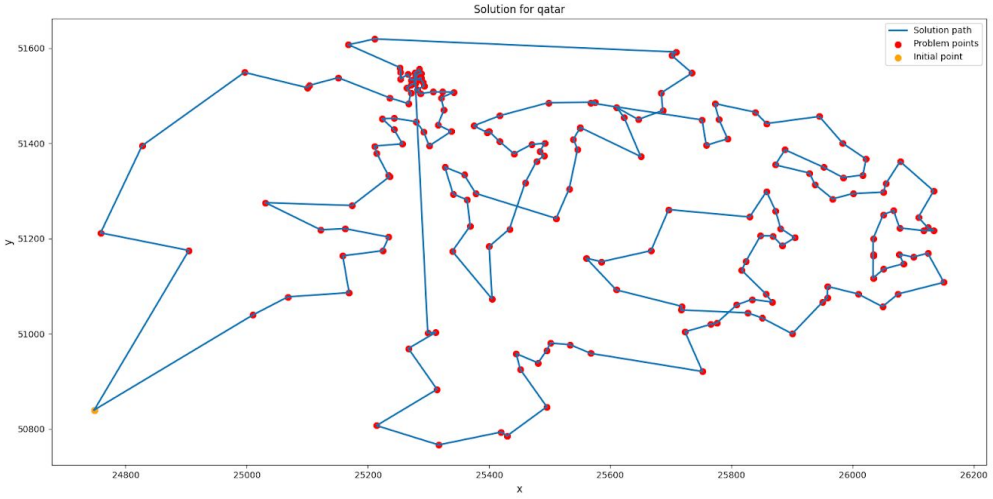
Nós explorados	22923
Interações Simplex	1894073
Caminho encontrado - Tam.	9389,034793904
Best Bound	9288,008839274
GAP	1,076%
GAP - Waterloo	0,396009345%



Caminho: *[1, 4, 2, 3, 5, 9, 10, 12, 15, 19, 30, 32, 31, 35, 42, 50, 55, 49, 44, 38, 41, 46, 48, 54, 52, 53, 56, 58, 43, 40, 34, 39, 47, 51, 61, 67, 73, 66, 68, 64, 70, 77, 84, 81, 79, 83, 88, 92, 97, 95, 96, 93, 91, 103, 102, 109, 113, 114, 119, 122, 118, 106, 105, 107, 108, 100, 110, 112, 115, 116, 117, 121, 120, 123, 124, 128, 133, 135, 129, 131, 136, 143, 148, 160, 166, 171, 170, 180, 185, 193, 188, 189, 191, 192, 190, 187, 183, 186, 194, 182, 176, 169, 161, 163, 164, 172, 179, 174, 173, 175, 184, 177, 181, 178, 168, 165, 167, 162, 159, 158, 155, 151, 147, 152, 141, 144, 150, 153, 157, 154, 139, 138, 142, 146, 149, 156, 145, 140, 137, 134, 132, 126, 125, 127, 130, 111, 104, 101, 99, 94, 89, 90, 98, 86, 85, 65, 20, 63, 36, 59, 62, 82, 71, 80, 87, 76, 75, 78, 72, 74, 69, 60, 57, 45, 37, 27, 22, 29, 28, 33, 18, 21, 24, 26, 17, 7, 11, 14, 25, 23, 13, 16, 8, 6, 1]*

Foco em Melhorar a Solução Primal

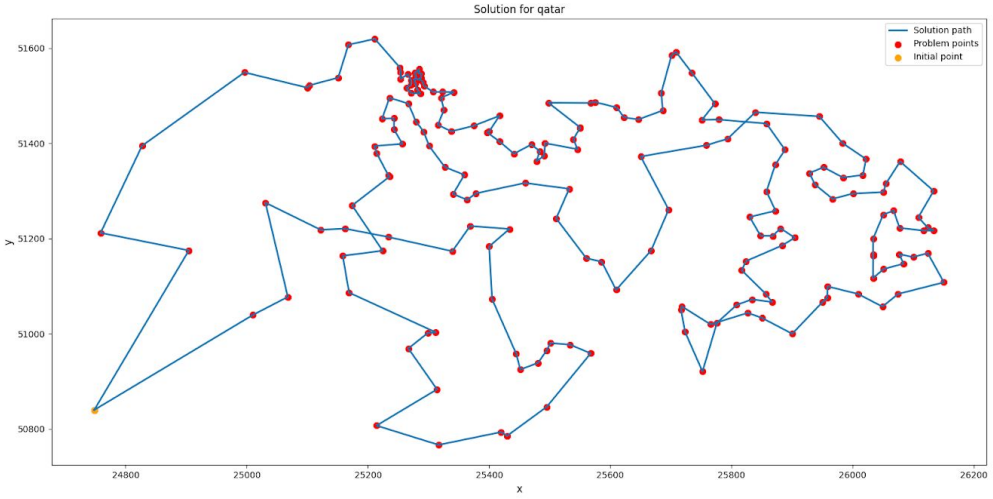
Nós explorados	22582
Interações Simplex	482367
Caminho encontrado - Tam.	10817,82712865
Best Bound	9282,184335728
GAP	14,8217%
GAP - Waterloo	15,6739428%



Caminho: *[1, 6, 8, 16, 13, 23, 25, 14, 11, 7, 17, 26, 24, 21, 18, 33, 28, 22, 29, 45, 57, 60, 70, 64, 68, 66, 73, 67, 61, 51, 47, 58, 56, 53, 52, 54, 48, 46, 44, 49, 55, 50, 42, 59, 62, 36, 63, 20, 65, 85, 86, 98, 90, 89, 94, 99, 101, 104, 111, 130, 127, 132, 134, 137, 142, 149, 146, 138, 139, 144, 150, 154, 157, 153, 152, 147, 141, 122, 119, 113, 109, 114, 126, 125, 140, 145, 156, 161, 163, 164, 169, 176, 182, 194, 190, 187, 183, 186, 179, 172, 174, 173, 175, 177, 181, 184, 189, 192, 191, 188, 193, 185, 180, 178, 168, 165, 159, 158, 151, 155, 162, 167, 170, 171, 166, 160, 148, 143, 133, 135, 136, 131, 129, 110, 112, 100, 84, 77, 79, 81, 83, 88, 92, 97, 95, 96, 93, 91, 87, 80, 82, 71, 76, 75, 72, 69, 74, 78, 102, 103, 106, 105, 107, 108, 118, 116, 115, 117, 121, 120, 128, 123, 124, 19, 15, 30, 32, 31, 35, 38, 41, 43, 40, 34, 39, 37, 27, 12, 10, 9, 5, 3, 2, 4, 1]*

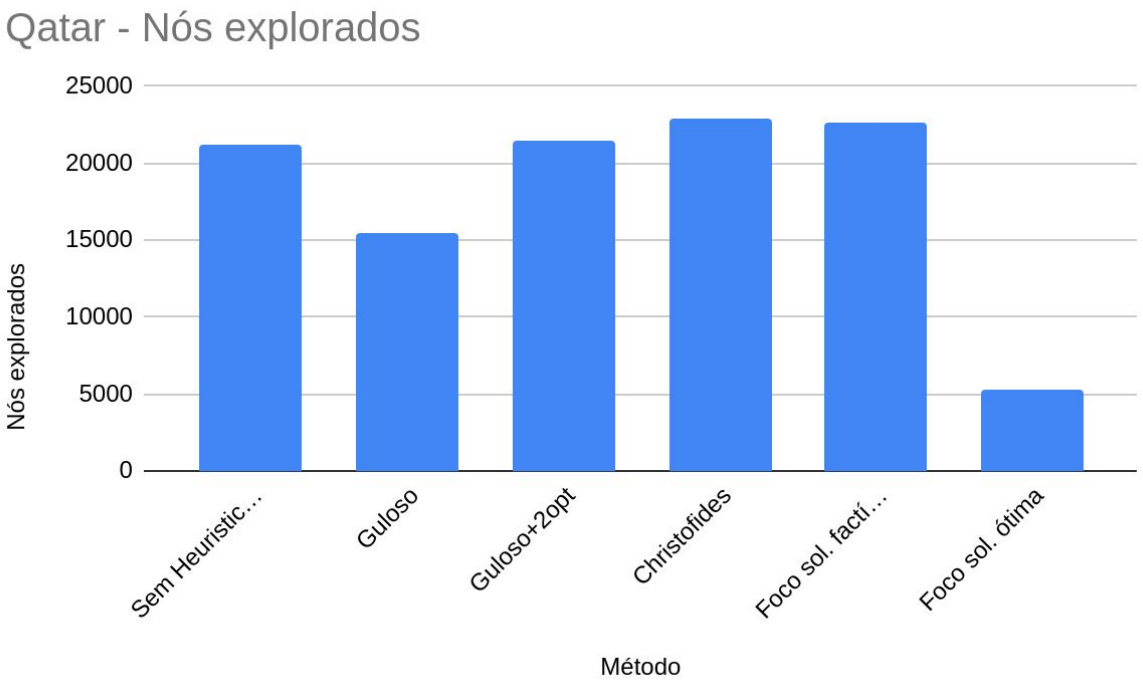
Foco em Encontrar a Solução Ótima

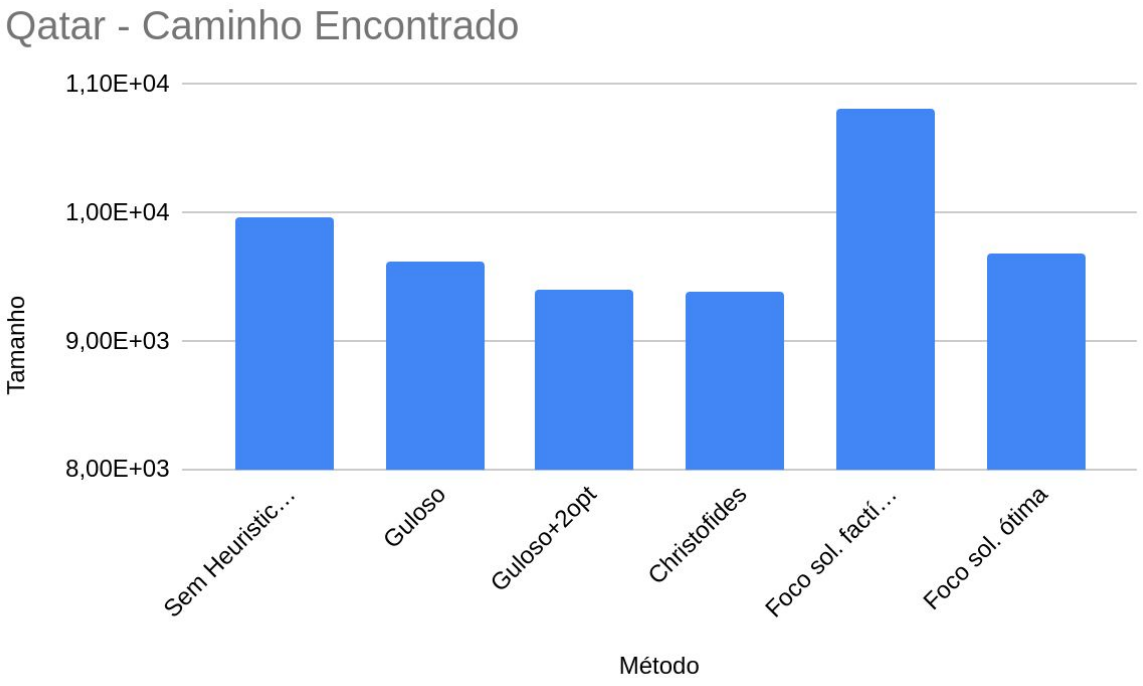
Nós explorados	5320
Interações Simplex	149703
Caminho encontrado - Tam.	9687,371156437
Best Bound	9267,343342367
GAP	4,3358%
GAP - Waterloo	3,58609021%

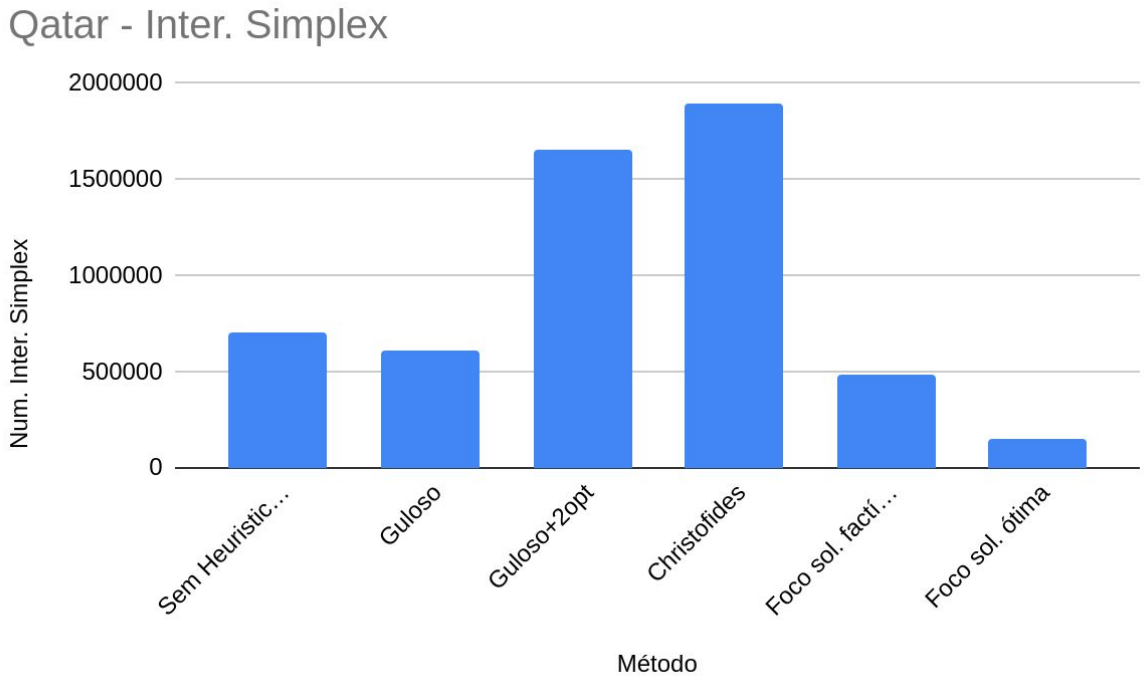


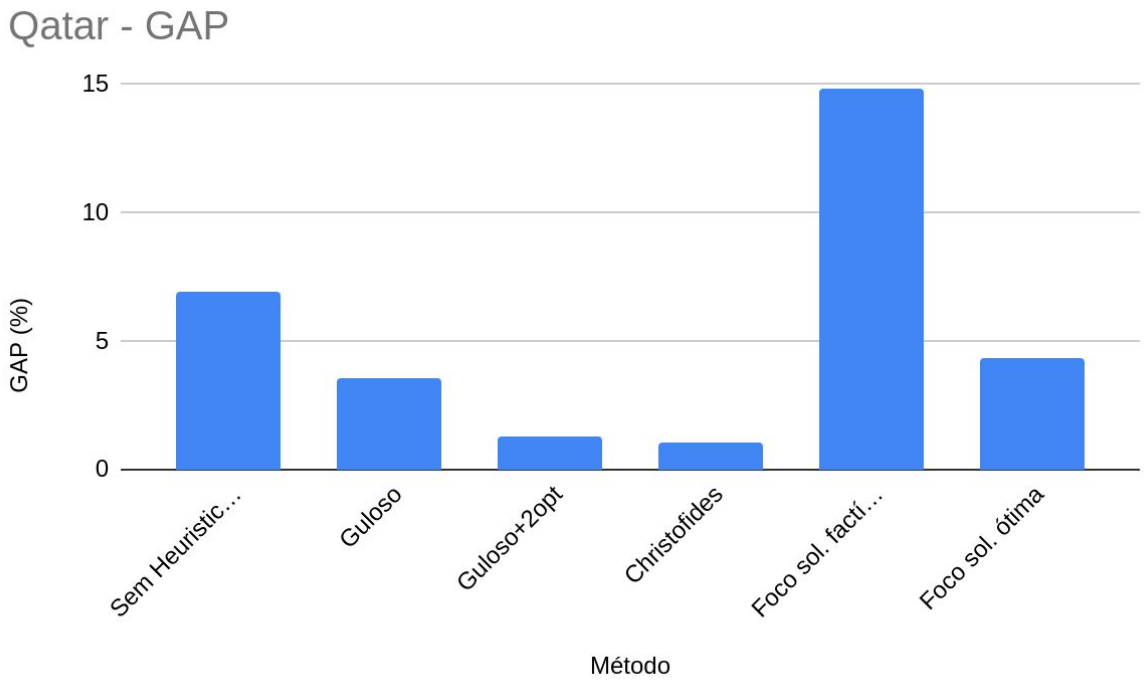
Caminho: *[1, 6, 8, 7, 11, 14, 25, 71, 76, 87, 80, 82, 89, 90, 94, 99, 101, 104, 111, 98, 86, 85, 65, 20, 63, 36, 59, 62, 16, 13, 23, 17, 26, 24, 21, 18, 33, 28, 29, 22, 27, 37, 45, 57, 60, 69, 74, 72, 75, 78, 91, 103, 102, 109, 113, 114, 119, 122, 118, 131, 136, 143, 160, 166, 171, 170, 167, 162, 158, 159, 165, 168, 178, 180, 185, 193, 188, 191, 192, 189, 184, 181, 177, 175, 173, 174, 172, 179, 186, 183, 187, 190, 194, 182, 176, 169, 164, 163, 161, 156, 145, 140, 132, 126, 125, 127, 130, 134, 137, 142, 149, 146, 138, 139, 154, 157, 153, 150, 144, 141, 152, 147, 151, 155, 148, 135, 129, 133, 128, 124, 123, 120, 121, 117, 116, 115, 112, 110, 100, 108, 107, 105, 106, 97, 96, 93, 95, 92, 88, 83, 79, 81, 84, 77, 70, 64, 68, 66, 73, 67, 61, 58, 56, 53, 52, 48, 54, 55, 49, 50, 42, 44, 46, 41, 43, 47, 51, 39, 34, 40, 38, 35, 31, 32, 30, 19, 15, 12, 10, 9, 5, 3, 2, 4, 1]*

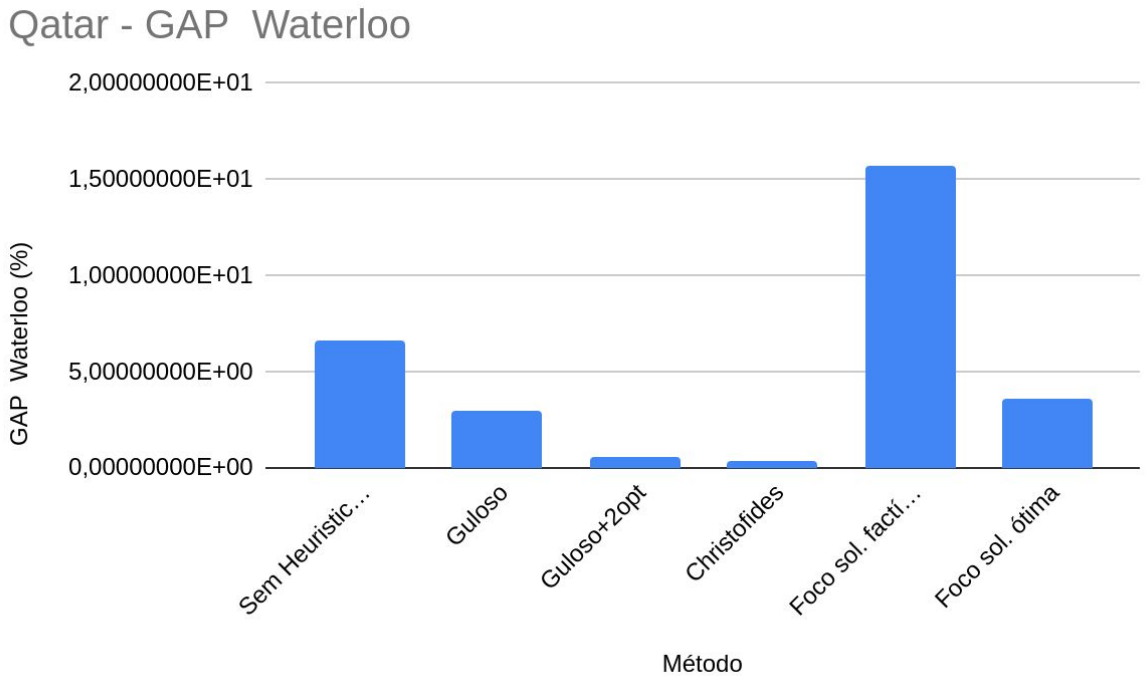
Gráficos

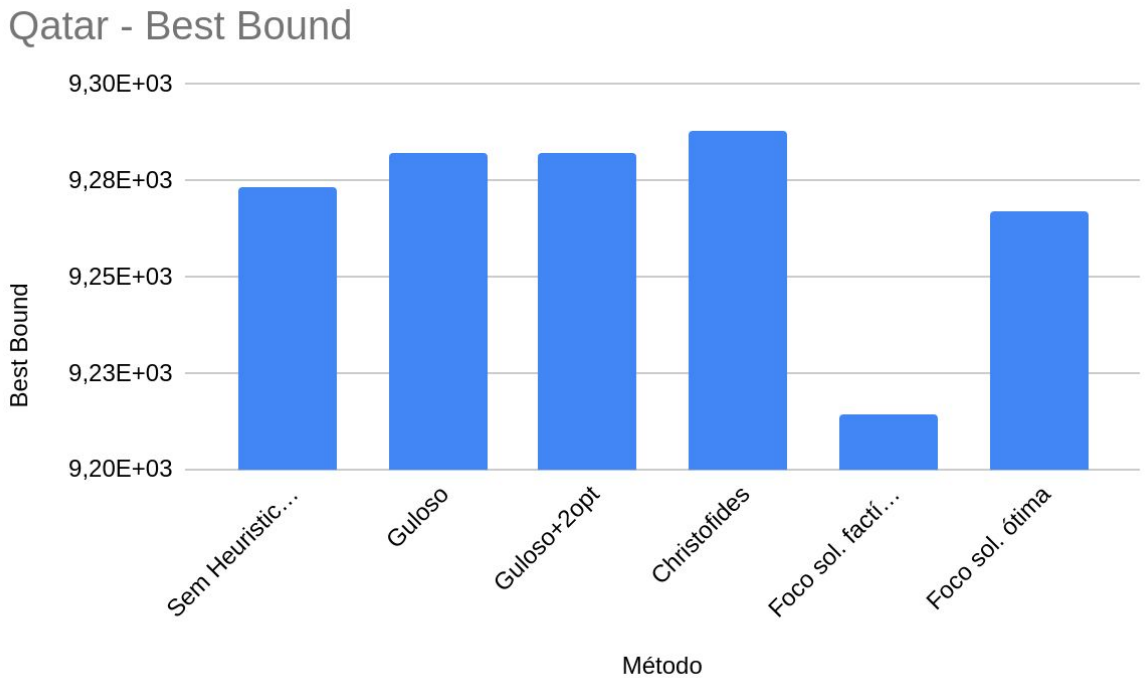

Qatar - Nós explorados
25000
20000
15000
10000
5000
0
Nós explorados
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método

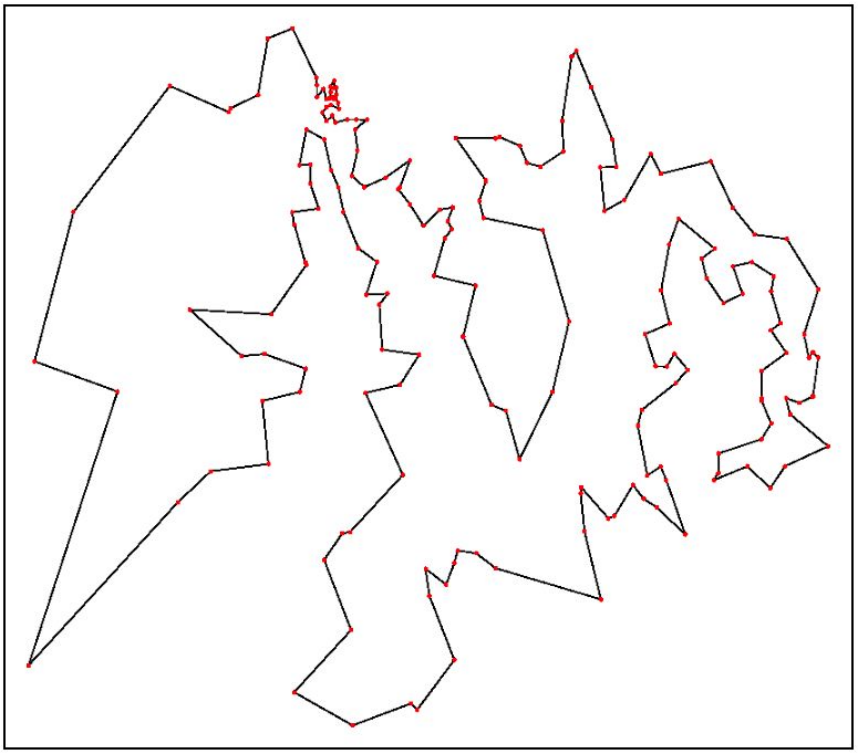

Qatar - Caminho Encontrado
1,10E+04
1,00E+04
9,00E+03
8,00E+03
Tamanho
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Qatar - Inter. Simplex
2000000
1500000
1000000
500000
0
Num. Inter. Simplex
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Qatar - GAP
15
10
5
0
GAP (%)
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Qatar - GAP Waterloo
2,00000000E+01
1,50000000E+01
1,00000000E+01
5,00000000E+00
0,00000000E+00
GAP Waterloo (%)
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Qatar - Best Bound
9,30E+03
9,28E+03
9,25E+03
9,23E+03
9,20E+03
Best Bound
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método



Solução do Waterloo

Como podemos ver, neste caso não conseguimos encontrar a solução ótima em nenhum caso, mas as heurísticas apresentaram uma melhora considerável no caminho encontrado. Como podemos ver, o Christofides foi o que conseguiu encontrar o melhor caminho com um GAP para a solução do Waterloo de somente 0,396009345%.

Outro ponto interessante é que desta vez o foco na solução ótima, apesar de melhorar o caminho encontrado, ele não foi o melhor para resolver o problema. E enquanto o foco em soluções factíveis só apresentou piora no caminho, se mostrando mais uma vez ruim.

Desta vez, como todos os testes rodaram até o tempo limite e nenhum encontrou uma solução ótima, um baixo número de nós explorados, não significa que ele se saiu melhor que os outros. Desta vez o que se saiu melhor de longe foi o Christofides que apresentou o melhor caminho e GAP também. Isso é esperado pois ele apresenta uma solução inicial muito boa, e como não conseguimos resolver o problema, é esperado que isto melhore o caminho encontrado.

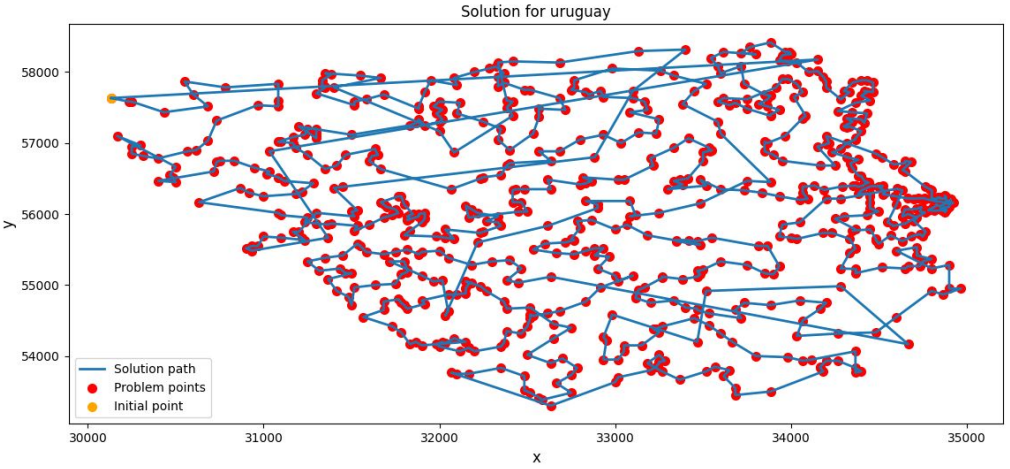
Uruguay

Sem heurística

Nós explorados	1
Interações Simplex	15095
Caminho encontrado - Tam.	Não encontrou nenhum caminho
Best Bound	74400,99751127
GAP	100%
GAP - Waterloo	Waterloo não possui solução

Heurística Gulosa

Nós explorados	1
Interações Simplex	15743
Caminho encontrado - Tam.	102594,3579305
Best Bound	74443,68951009
GAP	27,4388%
GAP - Waterloo	Waterloo não possui solução



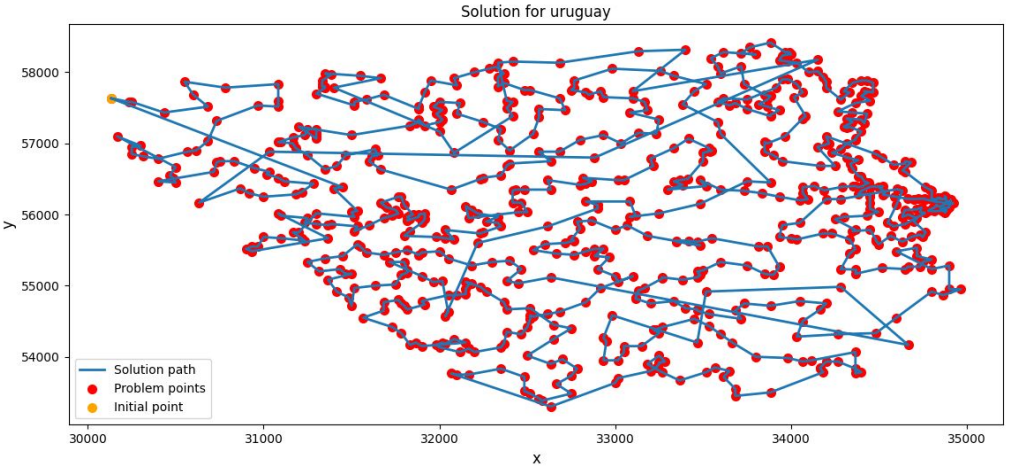
Caminho: [1, 3, 6, 11, 22, 18, 16, 27, 47, 46, 45, 36, 25, 21, 19, 17, 10, 8, 4, 5, 7, 2, 15, 12, 13, 14, 9, 23, 24, 26, 28, 34, 40, 39, 43, 51, 69, 60, 55, 38, 31, 29, 20, 42, 49, 65, 62, 70, 89, 90,

112, 107, 119, 131, 137, 142, 140, 134, 132, 125, 143, 147, 151,
163, 161, 157, 160, 156, 167, 168, 179, 178, 176, 159, 150, 186,
199, 207, 200, 233, 235, 251, 243, 239, 244, 231, 227, 220, 224,
274, 284, 277, 267, 268, 272, 280, 305, 301, 330, 336, 340, 332,
364, 368, 392, 390, 411, 424, 447, 446, 448, 451, 439, 438, 422,
416, 410, 418, 408, 442, 436, 458, 475, 486, 507, 530, 538, 533,
534, 541, 552, 573, 592, 599, 606, 615, 619, 624, 616, 610, 604,
588, 586, 607, 577, 581, 562, 571, 578, 600, 601, 611, 584, 579,
575, 589, 580, 612, 617, 621, 590, 567, 602, 630, 613, 622, 618,
632, 633, 623, 614, 591, 576, 585, 572, 558, 559, 551, 543, 539,
527, 518, 509, 513, 519, 522, 520, 515, 501, 481, 473, 483, 463,
449, 455, 460, 472, 476, 490, 500, 494, 485, 493, 508, 499, 498,
479, 470, 465, 454, 462, 480, 504, 512, 517, 526, 536, 523, 537,
535, 514, 492, 489, 488, 503, 511, 548, 564, 542, 556, 560, 557,
637, 642, 650, 664, 665, 671, 659, 663, 687, 691, 697, 696, 695,
704, 705, 709, 708, 720, 721, 715, 719, 718, 728, 730, 731, 729,
732, 727, 717, 714, 707, 703, 702, 700, 698, 690, 699, 711, 722,
716, 706, 725, 726, 733, 713, 710, 701, 688, 685, 678, 672, 666,
651, 644, 645, 646, 640, 643, 631, 609, 605, 629, 628, 627, 625,
638, 620, 574, 563, 598, 587, 626, 636, 635, 639, 648, 656, 670,
680, 684, 689, 673, 677, 669, 657, 660, 674, 682, 681, 686, 662,
649, 647, 654, 658, 655, 570, 555, 529, 510, 506, 521, 525, 550,
561, 583, 597, 582, 569, 603, 596, 641, 661, 668, 679, 683, 694,
675, 676, 653, 693, 724, 723, 734, 712, 692, 652, 634, 566, 524,
532, 547, 554, 528, 496, 474, 469, 471, 445, 428, 429, 420, 409,
389, 376, 377, 380, 383, 405, 406, 417, 430, 433, 431, 437, 457,
477, 487, 502, 505, 491, 484, 450, 432, 423, 414, 413, 434, 386,
369, 360, 339, 328, 329, 321, 324, 316, 298, 291, 310, 323, 334,
342, 351, 341, 355, 344, 359, 363, 373, 347, 333, 311, 299, 290,
287, 286, 283, 275, 258, 254, 250, 228, 222, 214, 188, 185, 193,
197, 206, 216, 172, 164, 158, 144, 136, 116, 128, 129, 141, 145,
148, 152, 177, 173, 202, 211, 217, 218, 223, 225, 219, 232, 238,
255, 259, 288, 308, 320, 307, 282, 303, 315, 327, 319, 309, 318,
297, 293, 285, 278, 279, 249, 221, 210, 204, 302, 358, 361, 388,
393, 396, 397, 407, 398, 385, 403, 415, 440, 452, 464, 461, 468,
467, 495, 549, 545, 546, 553, 565, 594, 608, 593, 595, 540, 531,
516, 482, 466, 456, 444, 425, 404, 394, 391, 399, 382, 367, 366,
362, 349, 354, 350, 356, 427, 441, 568, 667, 304, 273, 260, 276,
264, 241, 226, 203, 189, 175, 166, 155, 139, 130, 117, 114, 111, 97,
82, 64, 78, 96, 99, 103, 87, 94, 101, 102, 106, 122, 138, 146, 153,
154, 149, 135, 174, 183, 194, 201, 198, 230, 300, 345, 346, 335,
371, 378, 370, 400, 435, 478, 497, 459, 453, 419, 426, 443, 412,
402, 357, 326, 325, 322, 337, 338, 348, 353, 374, 384, 387, 372,
401, 395, 381, 365, 352, 331, 312, 294, 306, 265, 261, 252, 236,
234, 205, 126, 120, 121, 124, 123, 100, 95, 83, 66, 56, 53, 50, 44,
54, 63, 67, 68, 76, 75, 74, 73, 72, 58, 105, 162, 165, 169, 180,
191, 195, 187, 196, 184, 192, 215, 212, 237, 253, 245, 266, 292,
295, 296, 317, 313, 289, 281, 256, 246, 247, 257, 242, 240, 229,
209, 213, 182, 181, 171, 170, 133, 118, 110, 109, 77, 84, 85, 79,

80, 81, 91, 86, 115, 127, 93, 190, 208, 269, 262, 270, 263, 248, 271, 314, 379, 421, 375, 343, 98, 92, 113, 104, 108, 71, 57, 52, 59, 61, 48, 37, 35, 33, 30, 32, 88, 41, 544, 1]

Heurísticas Gulosa + Two-Opt

Nós explorados	29
Interações Simplex	20467
Caminho encontrado - Tam.	98928,05390867
Best Bound	74443,68951009
GAP	24,7042%
GAP - Waterloo	Waterloo não possui solução

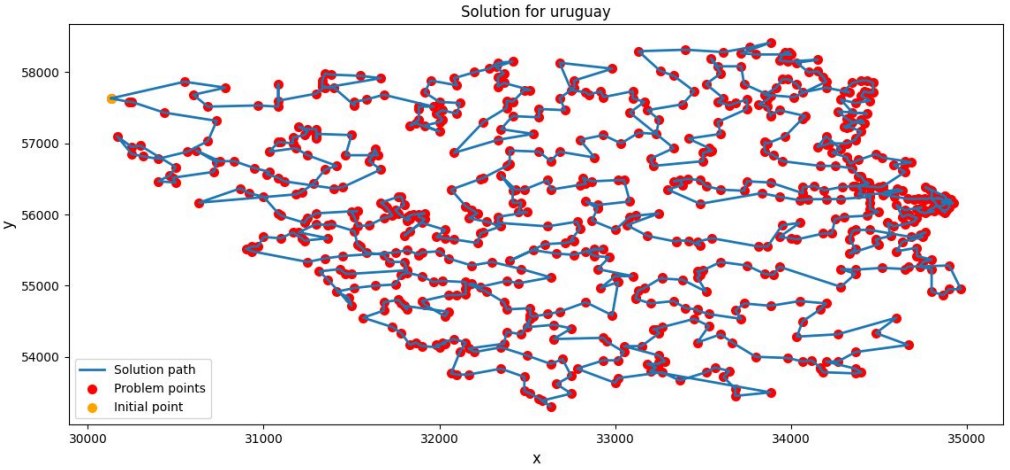


Caminho: [1, 3, 6, 11, 22, 18, 16, 27, 47, 46, 45, 36, 25, 21, 19, 17, 10, 8, 4, 5, 7, 2, 15, 12, 13, 14, 9, 23, 24, 26, 28, 34, 40, 39, 43, 51, 69, 60, 55, 38, 31, 29, 20, 41, 343, 544, 375, 421, 379, 314, 271, 248, 263, 270, 262, 269, 208, 190, 93, 127, 115, 86, 91, 81, 80, 79, 85, 84, 77, 109, 110, 118, 133, 170, 171, 181, 182, 213, 209, 229, 240, 242, 257, 247, 246, 256, 281, 289, 313, 317, 296, 295, 292, 266, 245, 253, 237, 212, 215, 192, 184, 196, 187, 195, 191, 180, 169, 165, 162, 105, 58, 72, 73, 74, 75, 76, 68, 67, 63, 54, 44, 50, 53, 56, 66, 83, 95, 100, 123, 124, 121, 120, 126, 205, 234, 236, 252, 261, 265, 306, 294, 312, 331, 352, 365, 381, 395, 401, 372, 387, 384, 374, 353, 348, 338, 337, 322, 325, 326, 357, 402, 412, 443, 426, 419, 453, 459, 497, 478, 435, 400, 370, 378, 371, 335, 346, 345, 300, 230, 198, 201, 194, 183, 174, 135, 149, 154, 153, 146, 138, 122, 106, 102, 101, 94, 87, 103, 99, 96, 78, 64, 82, 97, 111, 114, 117, 130, 139, 155, 166, 175, 189, 203, 226, 241, 264, 276, 260, 273, 304, 667, 568, 441, 427, 356, 350, 354, 349, 362, 366, 367, 382, 399, 391, 394, 404, 425, 444, 456, 466, 482, 516, 531, 540, 595, 593, 608, 594, 565, 553, 546, 545, 549, 495, 467, 468, 461, 464, 452, 440, 415, 403, 385, 398, 407, 397, 396, 393, 388, 361, 358, 302, 204, 210, 221, 249, 279, 278, 285, 293,

297, 318, 309, 319, 327, 315, 303, 282, 307, 320, 308, 288, 259,
255, 238, 232, 219, 225, 223, 218, 217, 211, 202, 173, 177, 152,
148, 145, 141, 129, 128, 116, 136, 144, 158, 164, 172, 216, 206,
197, 193, 185, 188, 214, 222, 228, 250, 254, 258, 275, 283, 286,
287, 290, 299, 311, 333, 347, 373, 363, 359, 344, 355, 341, 351,
342, 334, 323, 310, 291, 298, 316, 324, 321, 329, 328, 339, 360,
369, 386, 434, 413, 414, 423, 432, 450, 484, 491, 505, 502, 487,
477, 457, 437, 431, 433, 430, 417, 406, 405, 383, 380, 377, 376,
389, 409, 420, 429, 428, 445, 471, 469, 474, 496, 528, 554, 547,
532, 524, 566, 634, 652, 692, 712, 734, 723, 724, 693, 653, 676,
675, 694, 683, 679, 668, 661, 641, 596, 603, 569, 582, 597, 583,
561, 550, 525, 521, 506, 510, 529, 555, 570, 655, 658, 654, 647,
649, 662, 686, 681, 682, 674, 660, 657, 669, 677, 673, 689, 684,
680, 670, 656, 648, 639, 635, 636, 626, 587, 598, 563, 574, 620,
638, 625, 627, 628, 629, 605, 609, 631, 643, 640, 646, 645, 644,
651, 666, 672, 678, 685, 688, 701, 710, 713, 733, 726, 725, 706,
716, 722, 711, 699, 690, 698, 700, 702, 703, 707, 714, 717, 727,
732, 729, 731, 730, 728, 718, 719, 715, 721, 720, 708, 709, 705,
704, 695, 696, 697, 691, 687, 663, 659, 671, 665, 664, 650, 642,
637, 557, 560, 556, 542, 564, 548, 511, 503, 488, 489, 492, 514,
535, 537, 523, 536, 526, 517, 512, 504, 480, 462, 454, 465, 470,
479, 498, 499, 508, 493, 485, 494, 500, 490, 476, 472, 460, 455,
449, 463, 483, 473, 481, 501, 515, 520, 522, 519, 513, 509, 518,
527, 539, 543, 551, 559, 558, 572, 585, 576, 591, 614, 623, 633,
632, 618, 622, 613, 630, 602, 567, 590, 621, 617, 612, 580, 589,
575, 579, 584, 611, 601, 600, 578, 571, 562, 581, 577, 607, 586,
588, 604, 610, 616, 624, 619, 615, 606, 599, 592, 573, 552, 541,
534, 533, 538, 530, 507, 486, 475, 458, 436, 442, 408, 418, 410,
416, 422, 438, 439, 451, 448, 446, 447, 424, 411, 390, 392, 368,
364, 332, 340, 336, 330, 301, 305, 280, 272, 268, 267, 277, 284,
274, 224, 220, 227, 231, 244, 239, 243, 251, 235, 233, 200, 207,
199, 186, 150, 159, 176, 178, 179, 168, 167, 156, 160, 157, 161,
163, 151, 147, 143, 125, 132, 134, 140, 142, 137, 131, 119, 107,
112, 90, 89, 70, 62, 65, 49, 42, 88, 32, 30, 33, 35, 37, 48, 61, 59,
52, 57, 71, 108, 104, 113, 92, 98, 1]

Algoritmo de Christofides

Nós explorados	29
Interações Simplex	22041
Caminho encontrado - Tam.	88840,16176869
Best Bound	74443,68951009
GAP	16,1853%
GAP - Waterloo	Waterloo não possui solução



Caminho: [1, 16, 27, 18, 22, 36, 45, 47, 46, 77, 85, 79, 80, 81, 86, 91, 115, 127, 84, 93, 109, 110, 118, 133, 184, 171, 170, 169, 162, 165, 190, 180, 191, 195, 187, 196, 212, 215, 192, 181, 182, 213, 209, 229, 240, 248, 271, 242, 257, 247, 246, 256, 289, 281, 270, 263, 262, 237, 208, 245, 292, 253, 269, 295, 296, 317, 313, 325, 326, 357, 314, 322, 337, 338, 348, 353, 375, 374, 384, 372, 411, 390, 392, 438, 439, 451, 448, 446, 447, 424, 459, 453, 479, 480, 470, 465, 454, 462, 443, 460, 449, 455, 472, 476, 500, 494, 485, 493, 498, 499, 508, 537, 535, 514, 492, 489, 488, 503, 511, 548, 564, 588, 604, 610, 616, 624, 619, 599, 592, 573, 552, 541, 534, 533, 497, 478, 475, 458, 442, 436, 422, 410, 416, 418, 408, 435, 486, 507, 530, 538, 555, 570, 628, 627, 625, 638, 620, 574, 563, 561, 550, 525, 521, 506, 529, 510, 491, 484, 450, 432, 434, 423, 414, 413, 386, 369, 400, 378, 370, 360, 339, 345, 335, 346, 371, 368, 364, 340, 332, 336, 330, 305, 301, 280, 272, 252, 268, 267, 277, 274, 284, 244, 243, 300, 328, 329, 321, 324, 316, 298, 264, 291, 310, 323, 342, 351, 334, 341, 355, 344, 373, 359, 347, 363, 356, 333, 311, 299, 290, 287, 288, 259, 255, 238, 223, 225, 232, 260, 273, 304, 276, 241, 226, 203, 189, 175, 166, 155, 139, 97, 82, 64, 32, 30, 33, 35, 37, 48, 52, 59, 61, 88, 57, 70, 89, 90, 107,

111, 114, 117, 130, 135, 149, 154, 103, 99, 96, 78, 87, 102, 101, 94, 106, 122, 138, 146, 153, 174, 183, 194, 219, 218, 217, 211, 202, 173, 177, 198, 201, 148, 152, 145, 141, 129, 128, 116, 136, 144, 158, 164, 172, 188, 185, 197, 193, 206, 216, 214, 204, 210, 221, 249, 279, 278, 285, 293, 302, 297, 318, 309, 319, 315, 303, 282, 250, 228, 222, 254, 258, 275, 286, 283, 308, 320, 307, 350, 354, 366, 362, 349, 327, 358, 361, 403, 415, 440, 452, 464, 461, 468, 467, 495, 388, 393, 385, 396, 397, 407, 398, 367, 382, 399, 394, 391, 404, 425, 444, 427, 456, 466, 482, 516, 531, 540, 546, 545, 549, 593, 608, 594, 553, 565, 595, 667, 634, 652, 566, 524, 532, 547, 554, 528, 496, 474, 469, 471, 445, 428, 429, 420, 409, 389, 376, 377, 380, 383, 406, 405, 417, 441, 430, 433, 431, 437, 457, 477, 487, 502, 505, 568, 596, 569, 603, 641, 661, 668, 724, 734, 723, 712, 692, 693, 679, 683, 694, 675, 676, 653, 656, 684, 680, 689, 670, 626, 598, 587, 583, 597, 582, 639, 635, 636, 648, 647, 649, 662, 673, 677, 669, 657, 654, 660, 682, 674, 681, 686, 699, 690, 698, 700, 702, 701, 695, 705, 704, 709, 708, 719, 710, 713, 718, 729, 717, 714, 707, 703, 706, 716, 711, 722, 725, 726, 727, 732, 733, 728, 730, 731, 720, 721, 715, 687, 691, 697, 696, 688, 685, 678, 672, 666, 651, 644, 658, 655, 646, 645, 629, 605, 609, 606, 615, 631, 640, 643, 663, 659, 650, 664, 671, 665, 642, 637, 607, 586, 577, 581, 562, 571, 557, 542, 556, 560, 578, 600, 601, 611, 584, 579, 575, 612, 617, 589, 580, 567, 621, 590, 572, 602, 630, 613, 618, 622, 632, 633, 623, 614, 591, 576, 585, 558, 536, 526, 517, 512, 504, 490, 523, 559, 551, 543, 539, 527, 544, 518, 509, 513, 515, 520, 519, 522, 483, 473, 501, 481, 463, 421, 379, 402, 412, 426, 419, 387, 401, 395, 381, 365, 352, 331, 343, 312, 306, 294, 266, 265, 261, 236, 234, 205, 220, 227, 231, 224, 239, 251, 235, 233, 230, 207, 199, 186, 200, 176, 159, 150, 178, 167, 168, 179, 163, 161, 157, 160, 156, 142, 140, 151, 147, 143, 125, 132, 134, 137, 131, 119, 112, 108, 104, 113, 71, 65, 62, 49, 42, 38, 31, 29, 20, 55, 60, 69, 83, 95, 66, 56, 41, 44, 50, 53, 54, 63, 67, 58, 68, 76, 75, 72, 73, 74, 105, 100, 124, 123, 121, 120, 126, 98, 92, 51, 43, 39, 40, 34, 28, 26, 19, 24, 23, 13, 14, 9, 12, 15, 7, 5, 2, 4, 8, 10, 17, 21, 25, 11, 6, 3, 1]

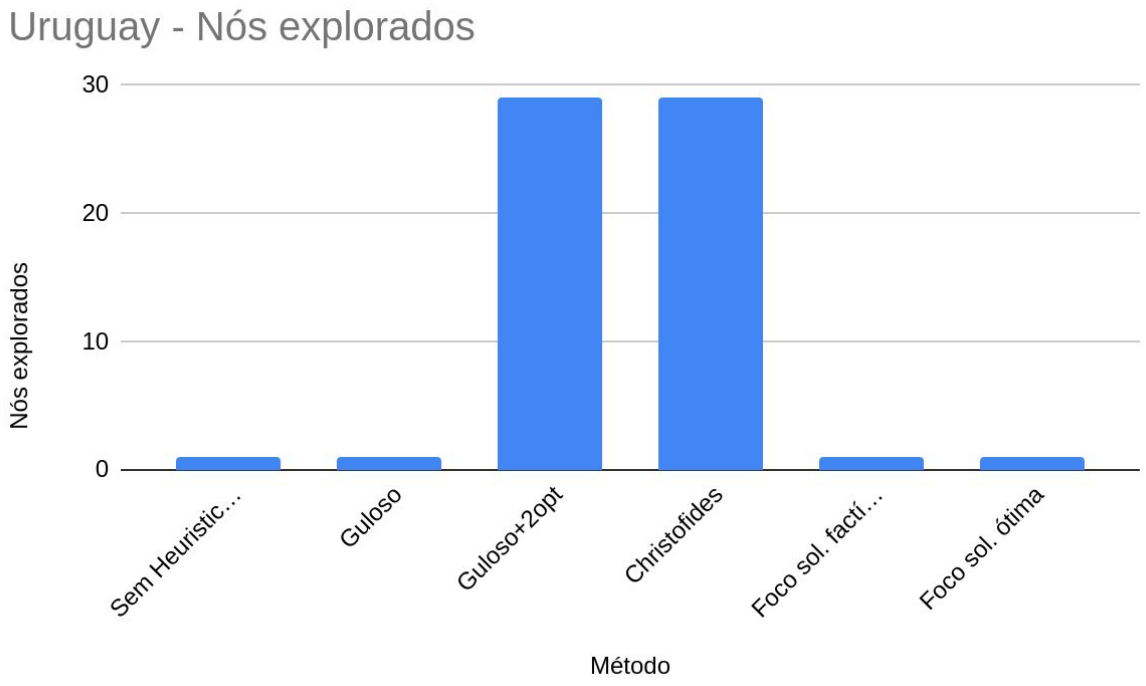
Foco em Melhorar a Solução Primal

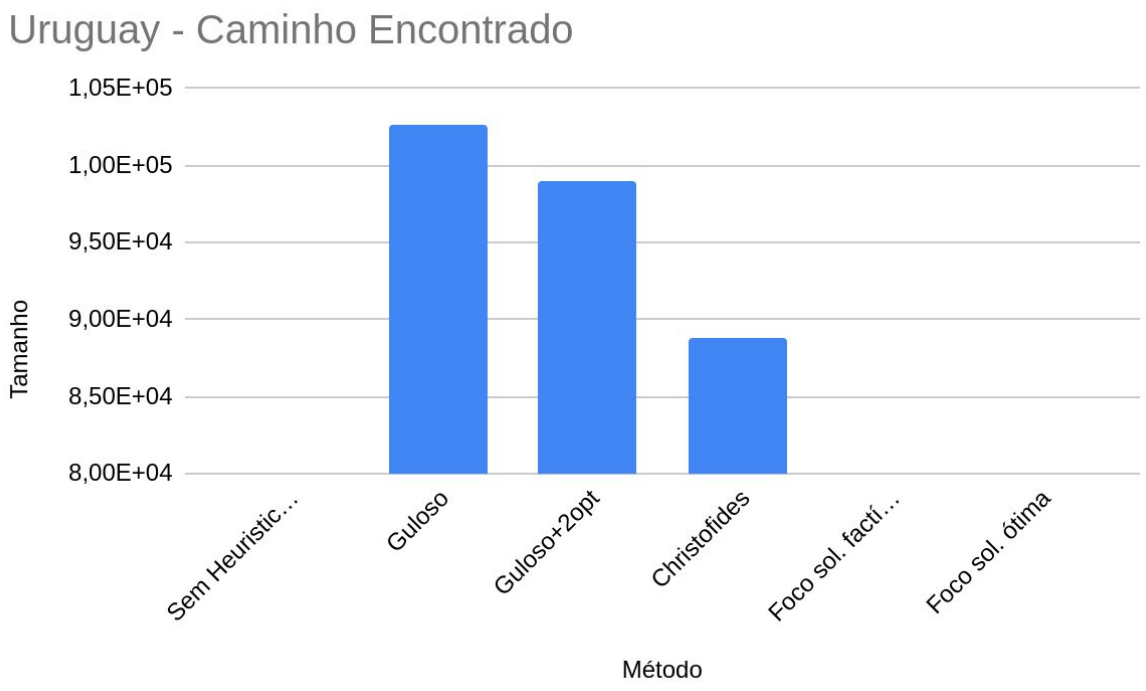
Nós explorados	1
Interações Simplex	4936
Caminho encontrado - Tam.	Não encontrou nenhum caminho
Best Bound	69988,37335624
GAP	100%
GAP - Waterloo	Waterloo não possui solução

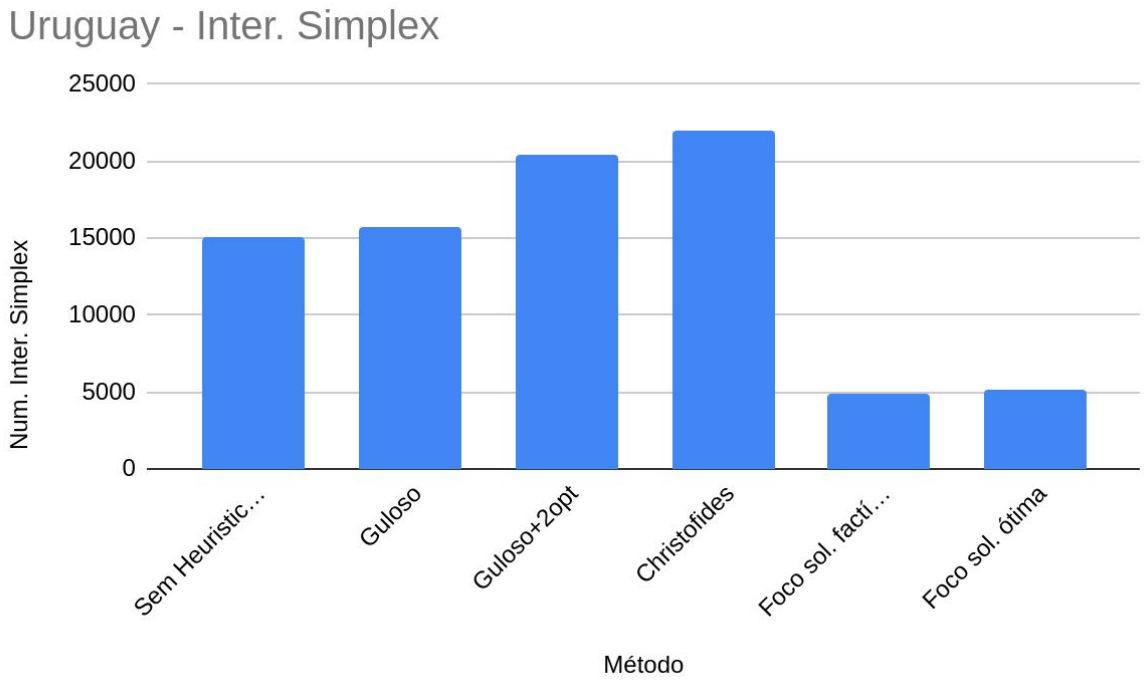
Foco em Encontrar a Solução Ótima

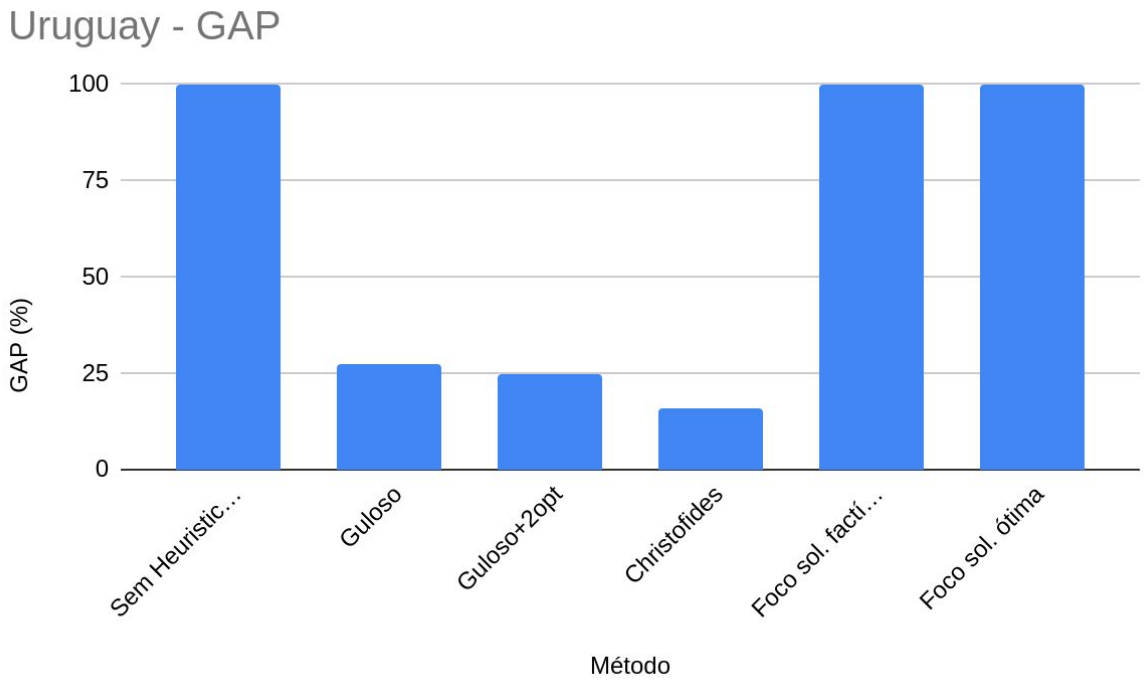
Nós explorados	1
Interações Simplex	5104
Caminho encontrado - Tam.	Não encontrou nenhum caminho
Best Bound	70014,29063296
GAP	100%
GAP - Waterloo	Waterloo não possui solução

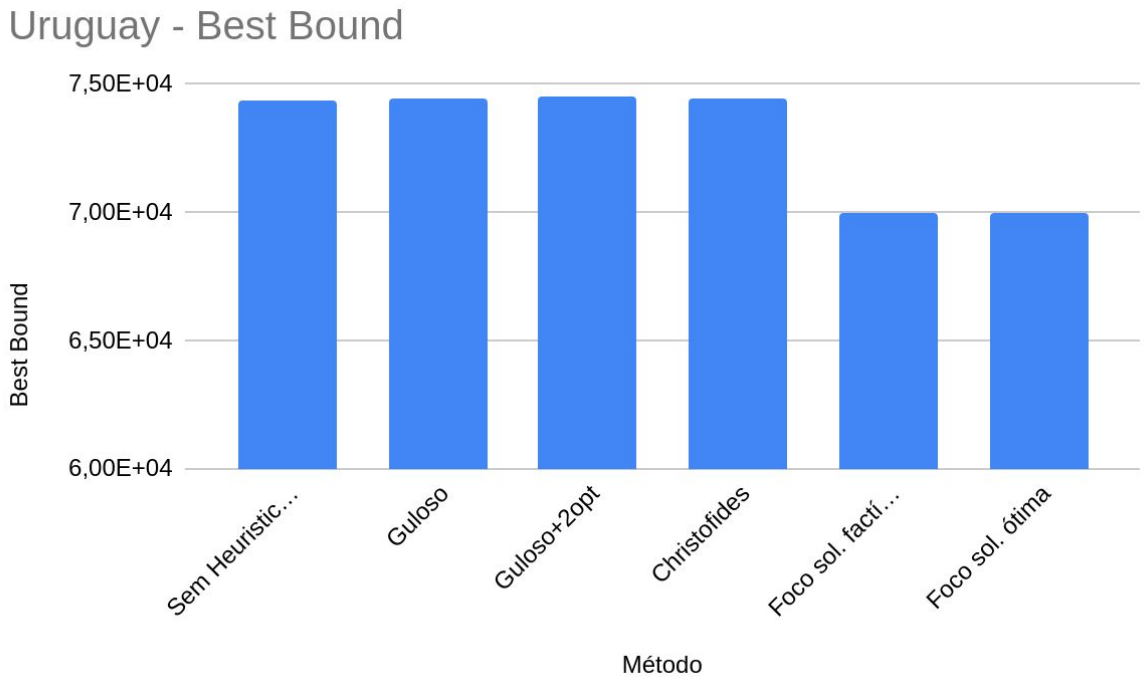
Gráficos

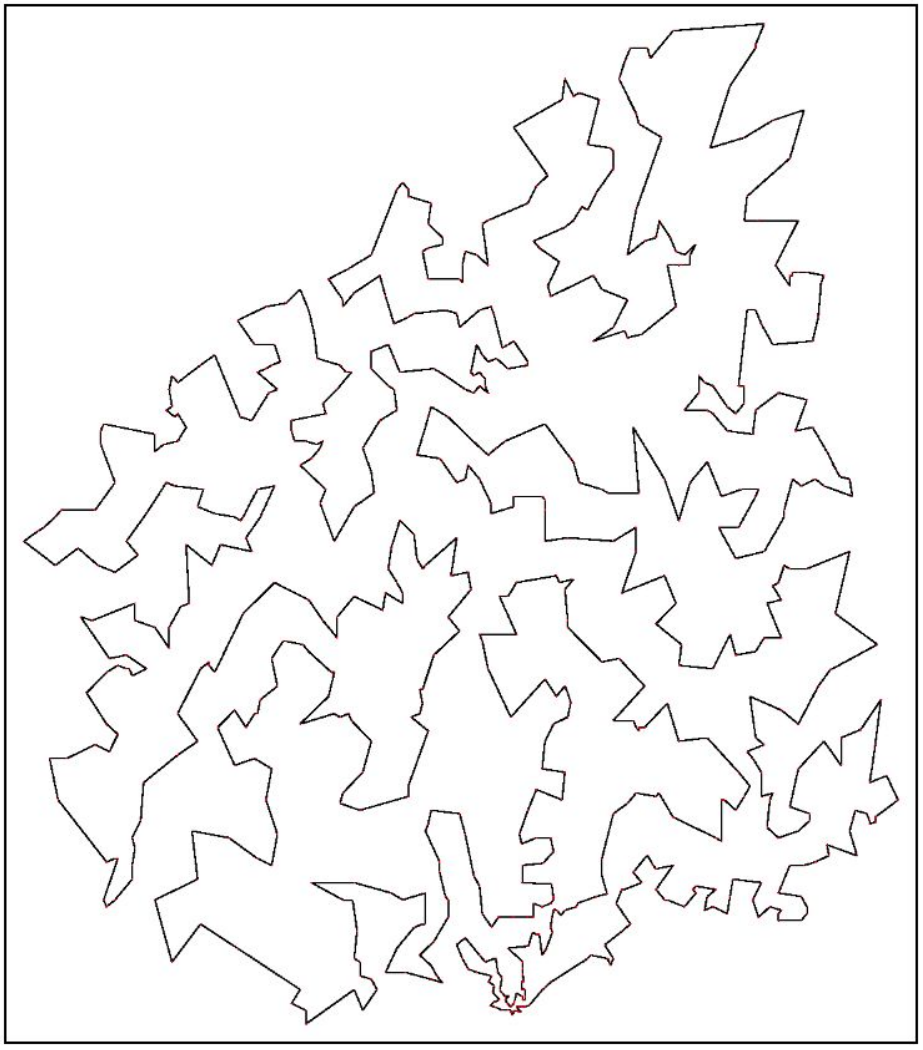

Uruguay - Nós explorados
30
20
10
0
Nós explorados
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Uruguay - Caminho Encontrado
1,05E+05
1,00E+05
9,50E+04
9,00E+04
8,50E+04
8,00E+04
Tamanho
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método


Uruguay - Inter. Simplex
25000
20000
15000
10000
5000
0
Num. Inter. Simplex
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Uruguay - GAP
100
75
50
25
0
GAP (%)
Sem Heuristic...
Guloso
Guloso+2opt
Christofides
Foco sol. factí...
Foco sol. ótima
Método


Uruguay - Best Bound
7,50E+04
7,00E+04
6,50E+04
6,00E+04
Best Bound
Sem Heuristic…
Guloso
Guloso+2opt
Christofides
Foco sol. factí…
Foco sol. ótima
Método



Solução do Waterloo

Este problema apresentou uma diferença muito grande para os demais, isso se deu ao seu tamanho, como podemos ver, caso não se utilize uma heurística para a solução inicial, o *Solver* não consegue nem sequer achar uma solução factível. O que já nos dá de primeira que as heurísticas são a melhor estratégia para este problema.

Analisando as heurísticas, temos que a de Christofides conseguiu apresentar o melhor caminho entre as outras. Além disso, o GAP dela foi o menor também. É interessante notar que somente ele e o Guloso com o Two-Opt conseguem explorar mais de um nó. Isso se deu pois estas 2 heurísticas apresentam uma boa solução inicial.

Não é possível fazer uma análise com o GAP com o Waterloo, pois o Waterloo não apresenta o tamanho do caminho ótimo encontrado.

Conclusão

No fim, após analisar todos os dados, podemos concluir que dependendo do problema uma estratégia é melhor que outra. Caso tenhamos um problema não muito grande no qual conseguimos chegar na solução ótima no tempo pré-determinado, usar a estratégia de focar em encontrar a solução ótima e provar que ela é ótima é a estratégia mais vantajosa. Entretanto, caso o problema seja muito grande e começamos a ter problemas para achar soluções, o uso de heurísticas para encontrar uma solução inicial é a melhor estratégia, sendo dentre as heurísticas, a heurística de Christofides se mostrando como a melhor dentre as testadas.

Implementação

O algoritmo foi implementado em Python 3 utilizando a biblioteca Gurobi para a modelagem do problema assim como para resolver o modelo.

Leitura de entradas

As entradas foram definidas da mesma forma como os exemplos passados, por meio dos arquivos .tsp. Segue o exemplo do arquivo do *toy problem*.

```
NAME: caso_teste_1.tsp
COMMENT : Um caso teste simples feito pelo o grupo para o projeto
TYPE: TSP
DIMENSION: 5
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 0.0 0.0
2 1.0 0.0
3 0.0 1.0
4 1.0 1.0
5 2.0 2.0
```

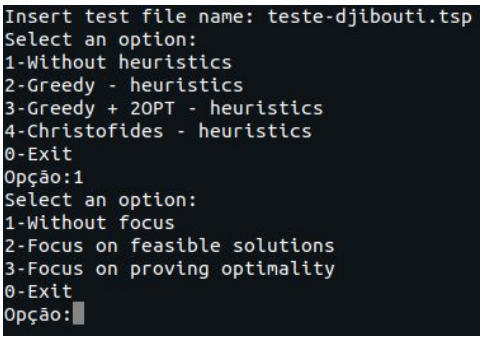
Ao executar o programa será pedido o nome do arquivo, basta digitar o nome do arquivo com a extensão que o programa irá carregar os dados do arquivo.
ATENÇÃO: O arquivo de entrada deve estar na pasta intitulada **Test-Cases** na mesma pasta do código a ser executado.

Métodos de resolução do problema

Logo após a leitura dos dados o programa irá abrir um menu com opções de heurísticas para a resolução do problema, as opções são: 1 - Sem Heurística, 2 - Heurística Gulosa, 3 - Heurística Gulosa com 2-opt e 4 - Heurística de Christofides.

Além disso, logo após a opção de heurística ser digitada, será mostrada opções de exploração dos nós da árvore. Existem 3 opções diferentes que podem ser selecionadas para realizar a exploração, 1 - Sem nenhum foco, 2 - Foco na solução factível e 3 - Foco na solução ótima.

Segue abaixo uma imagem do menu:



Matriz de distâncias

Ao receber as entradas, o programa cria uma matriz de distâncias na qual na linha ***i*** coluna ***j*** temos a distância entre a galáxia ***i*** e ***j.*** A distância é calculada pela distância euclidiana e **não** é feito arredondamentos para tornar as distâncias inteiras (Esse arredondamento pode ser encontrado em várias implementações para otimizar o código, pois operações inteiras são mais rápidas para serem executadas no computador).

Definindo o problema

O problema é definido utilizando a biblioteca Gurobi. Segue a definição de cada restrição do problema:

Restrição (1)
```
model.addConstr(sum([x[i][j] for j in range(n) if j != i]) == 1)
```

Restrição (2)
```
model.addConstr(sum([x[j][i] for j in range(n) if j != i]) == 1)
```

Restrição (3)
```
model.addConstr((y[i] - n * x[i][j]) >= y[j] - (n - 1))
```

Detalhes das variáveis como ***x*** só podendo assumir valor 0 ou 1 são definidos no momento de declarar a variável na biblioteca. Também foi limitado o tempo de execução do método para tentar resolver o problema, no máximo, em 30 minutos. Outros detalhes de implementação podem ser vistos diretamente no código fornecido.

Heurística Gulosa

A heurística gulosa é a heurística mais simples que utilizamos neste trabalho. A função “greedy” recebe como parâmetro uma matriz com as distâncias entre as galáxias e retorna uma matriz de adjacência, que representa o caminho encontrado pela execução e na última linha da matriz estão os valores das variáveis auxiliares do problema (os r). Além disso, a função também retorna um vetor que vai ser utilizado no solver (vai ser usado como resposta inicial), este vetor nada mais é que a matriz de adjacência em forma de vetor.

Definição da função

def greedy(matrix_ori): onde matrix_ori é a matriz de distâncias.

A partir da galáxia inicial, o algoritmo escolhe a próxima galáxia procurando o nó com a menor distância possível entre os nós ainda não visitados. Para que o algoritmo não escolha o nó atual como nó destino(distância = 0, portanto menor distância possível) e também não escolha nenhum nó já visitado, marcamos essas distâncias como “infinito” para que não seja escolhida novamente:

```
for i in range(n):          #Para que nó atual != nó destino
    matrix[i,i] = INF
```

```
for i in range(n):          #Marca distância de todos nós para um nó visitado como INF,
      matrix[i,local] = INF      #assim nenhum outro nó vai escolhê-lo como destino.
```

Após decidir qual o próximo nó a ser visitado, é importante marcar na matriz de adjacências para registrar qual o caminho está sendo tomado:

```
matrix_aux[local,ind] = 1          #Marca na matriz adjacências, nó local → nó ind
local = ind                        #Agora precisamos decidir para onde ir a partir de ind
```

O algoritmo segue com a execução até que todas as galáxias tenham sido visitadas e então retorna:

```
return vector, m          #vector vai ser usado como resposta inicial no solver e m é
                          #a matriz de adjacências com o caminho percorrido.
```

Heurística Two-Opt

Ao implementar a heurística de melhoria Two Opt utiliza-se o grafo como uma matriz de adjacências, recebendo também o caminho inicial para otimizá-lo. Para começar, define-se o caminho inicial como o melhor caminho e um laço de repetição atua para melhorar o caminho dado.

Enquanto o caminho é aprimorado, o código continua. Para definir se houve aprimoramento ou não, o novo caminho calculado copia o melhor caminho até agora e, a seguir, realiza o swap do two-opt, trocando a ordem dos nós para tentar obter uma nova rota da seguinte forma: $new\_path[i:j] = initial\_path[j-1:i-1:-1]$. Com isso, a nova

função $cost(route, matrix)$ realiza o cálculo do custo do novo caminho e compara com o custo do que já existia, caso haja melhora no custo do caminho atualizado, este permanece e o algoritmo segue rodando, se não há, então o algoritmo para.

Há casos em que o algoritmo não conseguirá identificar o aprimoramento dentro de 5 minutos dependendo da complexidade do problema, caso o aprimoramento não ocorra dentro desse intervalo de tempo, finalizam-se as tentativas e retorna o melhor caminho encontrado até o momento.

Heurística De Christofides

Na implementação da heurística construtiva de Christofides foi usado o grafo da biblioteca networkx, foi feito uma transformação de uma matriz de distâncias do numpy para um grafo da networkx, com a função: `G = nx.from_numpy_array(matrix),` sendo `G` o grafo e `matrix` a matriz de distâncias. Foi encontrada a árvore geradora mínima com a função `G_MST = nx.minimum_spanning_tree(G)`, sendo `G_MST` a árvore geradora mínima.

Foi armazenado em uma lista chamada `odd_vertices` todos os vértices que contém grau ímpar na árvore geradora mínima, foi criado a `matrix_sub` que é a matriz de adjacências que contém apenas os vértices de grau não par. Para encontrar o matching perfeito de menor custo, foi usado o complemento da `matrix_sub.`

Logo após a função `nx.max_weight_matching()` retornar um conjunto contendo os pares perfeitos, eles foram inseridos no grafo da árvore geradora mínima (`G_MST`).

Foi verificado se todos os vértices, do grafo resultante da união anterior, continham grau 2, se sim, é criado uma matriz de adjacência à partir de um ciclo euleriano encontrado no grafo e o algoritmo é finalizado, se não, é criado uma fila de prioridades, usando a biblioteca heapq, é feito a inserção da dupla `(degree_node, node)` e realizado o shortcut para transformar todos os vértices com grau igual a 2.

O loop do shortcut é realizado enquanto o topo da fila de prioridades possui um nó com grau maior que 2. Se o grau for superior a 2, o algoritmo pega este vértice e realiza todas as combinações possíveis de shortcut com os vizinhos desse nó, quando ele achar uma dupla de vértices que é possível realizar o shortcut, ou seja, satisfazer a desigualdade triangular e permanecer conexo. Quando o loop é finalizado é criada a matriz de adjacências que contém o caminho encontrado.

Mostrando respostas

A resposta do problema é mostrada utilizando a biblioteca Matplotlib, na qual desenha as galáxias em um plano cartesiano assim como o caminho a ser percorrido pelo telescópio. Além disso, também é mostrado no terminal a ordem dos pontos assim como a distância total percorrida e outras métricas apresentadas pela própria biblioteca como o número de nós explorados pelo *Solver* no *branch and cut*, assim como o número de interações no *simplex* utilizado dentro do *branch and cut.*

É apresentado também, no terminal, a ordem das galáxias a ser percorrida na resposta encontrada.

Executando o código

Bibliotecas

- matplotlib
- numpy
- gurobi
- heapq
- networkx

Para instalar as biblioteca basta utilizar o comando:
`$ pip3 install <biblioteca>`

OBS: A biblioteca Gurobi tem uma forma diferente para ser instalada e pode ser consultado a melhor forma para o seu caso diretamente no site biblioteca: https://www.gurobi.com/

Exemplo de execução (Linux)

Para executar o código, basta chamar o interpretador Python 3 para o arquivo de código main.py. Após isso, basta seguir as instruções do programa. Segue um exemplo.
ATENÇÃO: O arquivo de entrada deve estar na pasta intitulada **Test-Cases** na mesma pasta do código a ser executado.

```
$ python3 main.py
Insert test file name: teste-1.tsp
Select an option:
1-Without heuristics
2-Greedy - heuristics
3-Greedy + 2OPT - heuristics
4-Christofides - heuristics
0-Exit
Opção:1
Select an option:
1-Without focus
2-Focus on feasible solutions
3-Focus on proving optimality
0-Exit
Opção:1
Using license file /home/rodrigo/gurobi.lic
Academic license - for non-commercial use only - expires 2021-01-31
Changed value of parameter TimeLimit to 1800.0
   Prev: inf  Min: 0.0  Max: inf  Default: inf
Gurobi Optimizer version 9.1.0 build v9.1.0rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 22 rows, 30 columns and 76 nonzeros
Model fingerprint: 0x65e81eac
Variable types: 0 continuous, 30 integer (0 binary)
Coefficient statistics:
```

```
  Matrix range     [1e+00, 5e+00]
  Objective range  [1e+00, 3e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+00]
Found heuristic solution: objective 7.6568542
Presolve removed 0 rows and 6 columns
Presolve time: 0.00s
Presolved: 22 rows, 24 columns, 76 nonzeros
Variable types: 0 continuous, 24 integer (20 binary)

Root relaxation: objective 6.405697e+00, 14 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    6.40570    0    6    7.65685    6.40570  16.3%     -    0s
H    0     0                       7.4787087    6.40570  14.3%     -    0s
H    0     0                       6.6502815    6.40570  3.68%     -    0s
     0     0     cutoff    0         6.65028    6.65028  0.00%     -    0s

Cutting planes:
  Learned: 1
  MIR: 4

Explored 1 nodes (19 simplex iterations) in 0.02 seconds
Thread count was 8 (of 8 available processors)

Solution count 3: 6.65028 7.47871 7.65685

Optimal solution found (tolerance 1.00e-04)
Best objective 6.650281539873e+00, best bound 6.650281539873e+00, gap 0.0000%
Solution path (Points visited):
[1, 3, 5, 4, 2, 1]
```

Em caso de dúvida para executar o código, enviar um e-mail para algum membro do grupo.

Referências

- https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html [1]
- https://en.wikipedia.org/wiki/Travelling_salesman_problem#Miller%E2%80%93Tucker%E2%80%93Zemlin_formulation [2]
- https://www.gurobi.com/documentation/9.1/refman/index.html [3]
- https://www.gurobi.com/documentation/ [4]
- http://www.math.uwaterloo.ca/tsp/world/countries.html#WI [5]
- https://towardsdatascience.com/how-to-solve-the-traveling-salesman-problem-a-comparative-analysis-39056a916c9f [6]
- http://pedrohfsd.com/2017/08/09/2opt-part1.html [7]
- https://towardsdatascience.com/around-the-world-in-90-414-kilometers-ce84c03b8552 [8]
- http://pedrohfsd.com/2017/08/11/2opt-part2.html [9]
- https://en.wikipedia.org/wiki/2-opt [10]
- https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf [11]