**UNIVERSIDADE DE SÃO PAULO**

**INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO**

**Um Enigma das Galáxias**

**Parte 2**

**Bruno dos Santos**, **NUSP 10786170**

brunosantos.cps@usp.br

**Henrique de S. Q. dos Santos**, **NUSP 10819029**

henriquesqs@usp.br

**Paulo H. da Silva**, **NUSP 10734515**

henrique\_phs117@usp.br

**Witor M. A. de Oliveira**, **NUSP 10692190**

witor.mao@usp.br

São Carlos

2020

**Sumário**

[**1. Introdução**](#_issmqhy05e9) **3**

[**2. Mudanças**](#_f7ckr9ozqph9) **3**

[2.1 Modelagem](#_vmax5yr9uumh) 3

[2.2 Solver](#_shznyzg7ocko) 4

[**3. Descrição da modelagem**](#_kgzuugo0bi8e) **5**

[3.1 As variáveis](#_xtnwpmshem6a) 5

[3.2 As restrições](#_340qr8gwkdoj) 6

[**4. Implementação**](#_uz2dium6dxhh) **6**

[4.1 Heurística LKH](#_hl0dvaanet1j) 6

[4.2 Heurística 2-OPT](#_ga4suya29nmf) 7

[**5. Problemas resolvidos**](#_rond5klwdysi) **8**

[5.1 Problema amostra (Toy Problem)](#_7d1dkqx3we3d) 8

[5.2 Instâncias de Waterloo](#_qz0lz1t1p36b) 9

[**6. Resultados**](#_adi5t04ry2tg) **10**

[6.1 Representação gráfica das soluções](#_ium7fy3s07tj) 11

[6.1.1 Problema Amostra (Toy Problem)](#_ys598p7g01ia) 11

[6.1.2 Instâncias de Waterloo](#_eah4doqc0w4d) 11

[6.2 Resultados](#_7nfk4t9hs1uf) 14

[6.3 Análise dos resultados](#_f080d65iu65u) 16

[**7. Conclusões**](#_knd0q52h1ncg) **16**

[**8. Referências**](#_z9siye1y3qez) **17**

# **1. Introdução**

O tema do caixeiro viajante trata-se de um problema antigo que surgiu em meados do século XVII, sendo que só em 1930 os matemáticos universitários começaram a estudá-lo [5]. O problema do caixeiro viajante consiste na procura de um circuito que possua a menor distância, começando numa cidade qualquer, entre várias, visitando cada cidade precisamente uma vez e regressando à cidade inicial. Abaixo, um exemplo de grafo usado nesse problema:

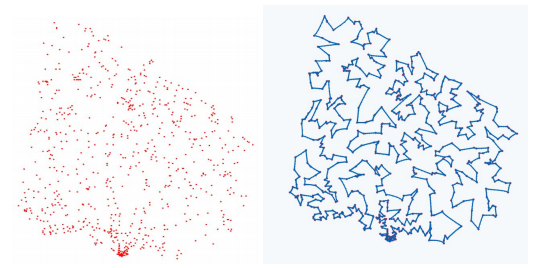


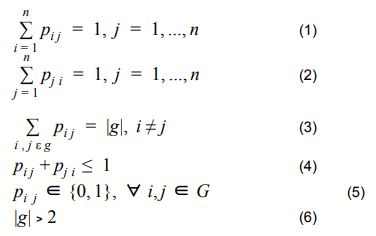
Imagem 1 - Grafo das 743 cidades do Uruguai, antes e após a resolução do caixeiro viajante.

Para esse trabalho, foi proposta a resolução do tema Enigma das Galáxias. Esse tema é uma derivação do problema original, onde a diferença é que um astrônomo observa o céu à procura de uma supernova, e para isso ele deseja mover o telescópio o mínimo possível já que ele é frágil e difícil de trabalhar. Portanto, esse astrólogo precisa ordenar as galáxias de forma a retornar o menor trajeto possível.

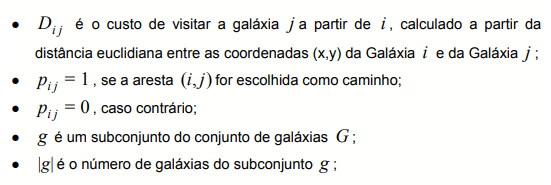
# **2. Mudanças**

## 2.1 Modelagem

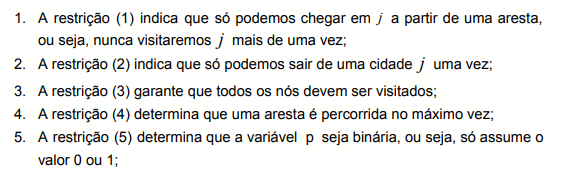
Para a primeira etapa deste trabalho, o grupo optou por uma modelagem com mais restrições, como é possível observar na imagem abaixo. Em uma breve análise por nós, concluímos que isso acabou afetando o desempenho dos *solvers*, já que, para um problema com um grande número de dados, o número de restrições crescia muito, além do número de variáveis. Além disso, uma restrição que impedia a existência de subcaminhos na solução não foi devidamente inserida na primeira etapa, sendo corrigida apenas na etapa atual.



Onde



E





## 2.2 *Solver*

Para a primeira etapa, o grupo optou por utilizar o [Pulp](https://coin-or.github.io/pulp/index.html), um modelador de problemas lineares, com o [COIN-OR *Branch-and-Cut* *solver*](https://github.com/coin-or/Cbc). Por dificuldades na busca por referências no que tange a resolução de problemas com um número maior de dados, optou-se pela remodelação do problema utilizando uma nova ferramenta - [OR-Tools](https://developers.google.com/optimization) - com o solver [SCIP](https://www.scipopt.org/). Além disso, os problemas que foram citados na [seção 2.1](#_vmax5yr9uumh) foram corrigidos nesta nova modelagem e serão aprofundados posteriormente neste documento.

# **3. Descrição da modelagem**

Dadas as circunstâncias descritas em [2.1](#_vmax5yr9uumh), a nova modelagem seguiu a formulação clássica do problema do caixeiro-viajante apresentada por [1], que define o problema em um grafo completo e direcionado com um conjunto de vértices e de arestas com uma matriz de custos assimétrica para . Além disso, para a eliminação de possíveis subcaminhos encontrados, optou-se por acrescentar a Formulação de Miller, Tucker e Zemlin (MTZ), também apresentada em [1]. Ambas estão demonstradas a seguir.

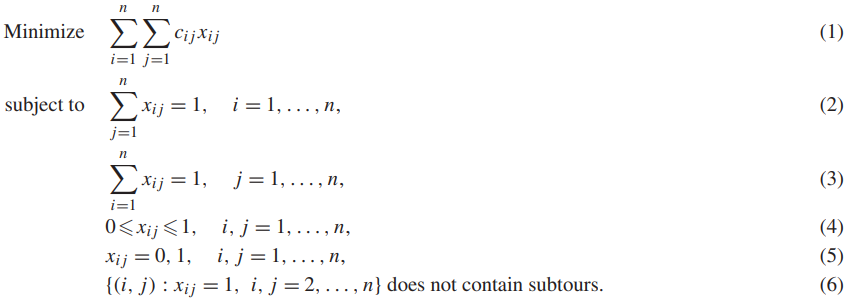


Imagem retirada de [1] que representa a modelagem clássica do problema do *TSP.*



Imagem retirada de [1] que representa a restrição de eliminação de subcaminhos proposta por (MTZ).

## 3.1 As variáveis

* custo (arredondado para o inteiro mais próximo dado pela distância euclidiana) de visitar a galáxia a partir de ;
* = 1, se a aresta for escolhida como caminho;
* = 0, caso contrário;
* = ordem que a cidade é visitada;
* = número de cidades;

## 3.2 As restrições

* As restrições (2) e (3) “garantem que cada vértice é ligado à uma aresta de saída e uma aresta de entrada” [1]. Basicamente, só existe um caminho saindo de e indo para ou vice versa;
* As restrições (4) e (5) definem o domínio das variáveis e a restrição (10) o domínio das variáveis ;
* A restrição (9) garante a eliminação de subcaminhos.

# **4. Implementação**

Como fora citado anteriormente, utilizamos a ferramenta [OR-Tools](https://developers.google.com/optimization) com o solver [SCIP](https://www.scipopt.org/) para a resolução do problema. É necessário dizer que, para fins de melhoria no desempenho do *solver*, utilizamos duas heurísticas (LKH, baseada na heurística *Lin-Kernighan*, e a 2-*opt*) detalhadas a seguir, que nos retornam uma hipótese de melhor caminho baseado em suas próprias análises e critérios e comparamos os resultados obtidos pelas duas. Utilizamos esse caminho retornado pela heurística como uma solução inicial para o *solver* através da função [*SetHint*](https://developers.google.com/optimization/reference/python/linear_solver/pywraplp#sethint). Assim, antes de iniciar a busca por uma solução, o *solver* já possuía uma inicial calculada.

A adição de uma solução inicial provou melhorar o desempenho do *solver* em alguns casos, permitindo que uma solução melhor que a solução original (sem o compartilhamento de uma solução inicial) fosse encontrada. Os resultados encontrados que comprovam a melhoria no desempenho serão exibidos mais a frente neste documento.

## 4.1 Heurística LKH

A LKH [3] é uma heurística implementada com base na heurística *Lin-Kernighan*, proposta em [2]. Para a implementação dessa heurística, nós optamos por utilizar um módulo chamado [elkai](https://github.com/filipArena/elkai), que implementa a LKH.

Em [3], é dito que a LKH possui algumas melhorias na eficiência em relação à original, principalmente por “revisões feitas nas regras heurísticas para restringir e direcionar a busca do algoritmo”, sendo tratada como a diferença mais notável em relação à versão original, mas não a única. Além desta, a LKH “usa etapas de busca maiores (e mais complexas) que a original e [...] acrescenta uma [análise sensitiva](https://en.wikipedia.org/wiki/Sensitivity_analysis) para direcionar e restringir a busca.”

De acordo com [3], a heurística *Lin-Kernighan* é geralmente considerada como sendo um dos métodos mais efetivos para gerar soluções ótimas (ou próximas da ótima) para o problema do caixeiro viajante. Ainda, essa heurística pertence à classe dos algoritmos de otimização local. “Dado um caminho viável, o algoritmo executa repetidamente trocas que reduzem o custo do caminho atual, até que encontre um caminho o qual nenhuma troca trará melhorias. Este processo pode ser repetido várias vezes para caminhos iniciais gerados de forma aleatória.” [3] Abaixo, um pseudocódigo retirado de [3] que representa a heurística *Lin-Kernighan*.

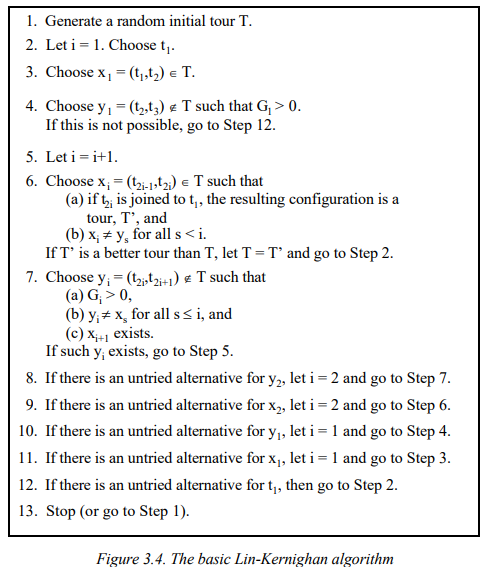


Imagem 2 - Algoritmo de *Lin-Kernighan* básico.

Considere que G é o grafo que representa o problema do *TSP*, “T” é o caminho com menor custo encontrado, () são arestas de G.

## 4.2 Heurística 2-OPT

Muito parecido com a heurística *Lin-Kernighan*, a heurística 2-OPT faz o mesmo caminho que a *Lin-Kernighan*, mas, ao invés de executar várias trocas, ela efetua apenas 2 trocas entre 2 arestas. Essas 2 trocas são feitas, principalmente, em caminhos que se cruzam, de forma a remover esse cruzamento. Além disso, a 2-opt também finaliza quando nenhuma melhoria pode ser feita. Abaixo, um pseudocódigo desta heurística, retirado de [4].

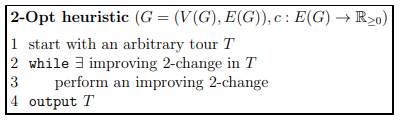


Imagem 3 - Algoritmo *2-OPT* básico.

Considere que G é o grafo que representa o problema do *TSP*, “V” é o conjunto de vértices (cidades) do problema, “E” é o conjunto de arestas (conexões entre cidades) e “T” é o caminho com menor custo encontrado.

# **5. Problemas resolvidos**

## 5.1 Problema amostra (*Toy Problem*)

Para testarmos o funcionamento do nosso modelo, criamos um exemplo contendo apenas 5 conjuntos de coordenadas (x,y) definidas no plano cartesiano, que identificamos como “cidades” enumeradas de 0 à N-1, com N sendo o número de conjuntos utilizados (5, neste caso). Como podemos chegar em qualquer coordenada a partir de outra, temos um grafo completo com 20 arestas.

Para esta amostra, buscamos o menor trajeto partindo da Cidade 0 que visitasse todas as outras e retornasse para o início (Cidade 0), como é requisitado pela especificação do trabalho. As coordenadas utilizadas encontram-se a seguir e, em seguida, suas representações no plano cartesiano:

* Cidade 0: (4,5);
* Cidade 1: (7,4);
* Cidade 2: (6,2);
* Cidade 3: (1,3);
* Cidade 4: (2,0);

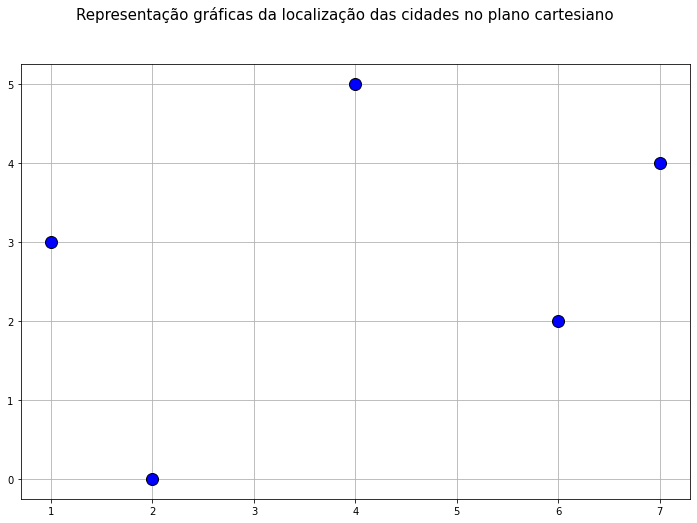


Imagem 4 - Representação das cidades do *Toy Problem*.

## 5.2 Instâncias de *Waterloo*

Para testar a eficiência do modelo apresentado neste trabalho, o grupo utilizou o modelo criado para solucionar algumas instâncias do [repositório](http://www.math.uwaterloo.ca/tsp/world/countries.html) da Universidade de *Waterloo,* Canadá. Todas as instâncias foram retiradas do conjunto de dados [*National Traveling Salesman Problems*](http://www.math.uwaterloo.ca/tsp/world/countries.html), que propõe a solução do problema do caixeiro viajante nas cidades de vários países.

Vale ressaltar aqui que, por conta da comparação dos resultados obtidos pelo nosso modelo com os resultados apresentados por *Waterloo*, nós optamos por **arredondar os valores da matriz de custo para o inteiro mais próximo**, [como fora feito por eles](http://www.math.uwaterloo.ca/tsp/world/countries.html). Essa condição foi retratada na [seção 3.1](#_xtnwpmshem6a).

Abaixo representamos graficamente 4 instâncias testadas na modelagem: *Western Sahara*, Djibouti, *Qatar* e *Uruguay* com 29, 38, 194 e 734 cidades, respectivamente.

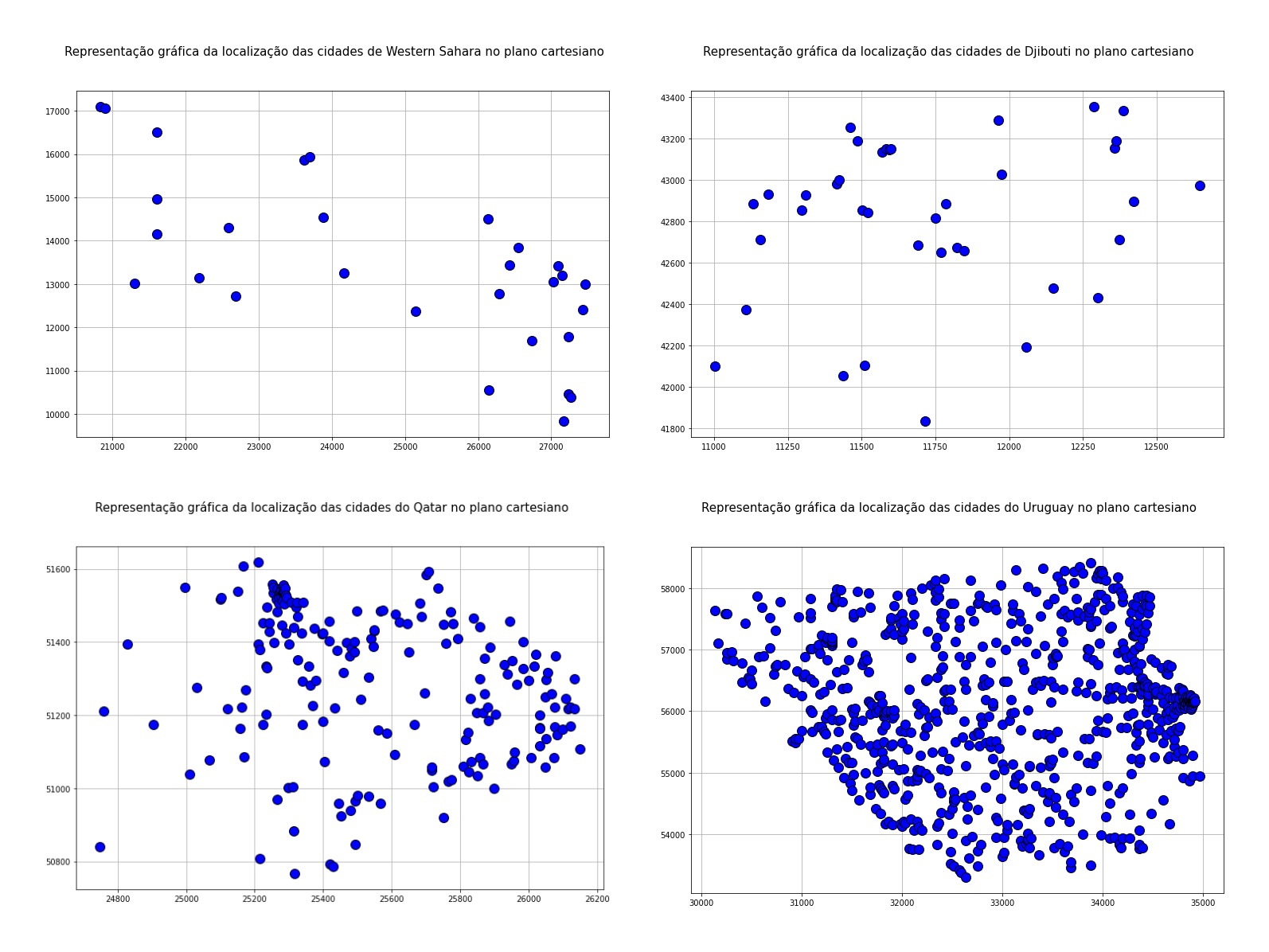


Imagem 5 - Representação gráfica das cidades dos conjuntos retirados do *Western Sahara*, *Djibouti*, *Qatar* e *Uruguay*.

# **6. Resultados**

Para uma melhor apresentação dos resultados obtidos, iremos mostrar sequencialmente uma representação gráfica no plano cartesiano bidimensional para cada instância indicando onde estão localizadas as cidades, além da resolução do *solver*, que representa o melhor caminho encontrado. Em seguida, iremos mostrar uma tabela, que contém o valor da função objetivo encontrado pelo *solver*, o GAP relativo entre a melhor solução encontrada em *Waterloo* em relação à encontrada na nossa implementação, qual o tempo limite permitido para a busca da solução (10, 30 ou 60 minutos) e qual o tempo útil utilizado e, por fim, qual a heurística foi utilizada para o caminho inicial (LKH, 2-opt ou nenhuma). No final, uma análise entre os resultados será demonstrada.

## 6.1 Representação gráfica das soluções

### 6.1.1 Problema Amostra (Toy Problem)

Para este problema específico, como o conjunto é pequeno, optamos por listar o caminho encontrado: (4,5), (7,4), (6,2) (2,0), (1,3) e (4,5) com custo 16.

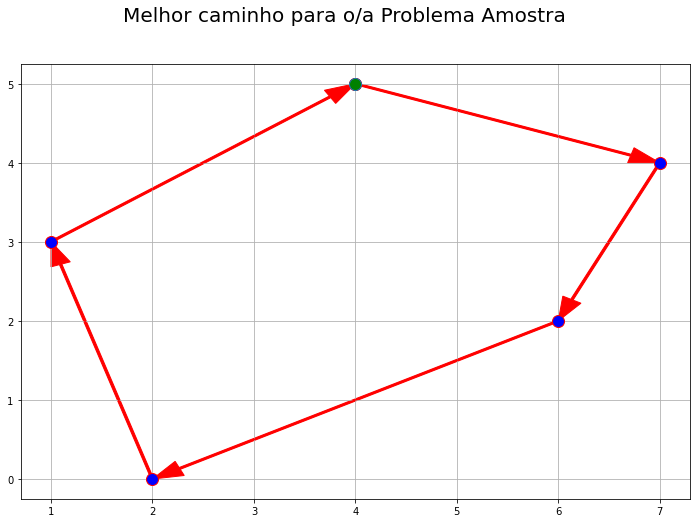


Imagem 6 - soluções encontradas para o conjunto *Toy Problem* sem heurística.

### 6.1.2 Instâncias de Waterloo

Abaixo, a solução encontrada para os conjuntos ***Western Sahara, Djibouti e Qatar*** e ***Uruguay*** sem uma dica inicial, com a heurística 2-OPT e com a LKH respectivamente com um tempo limite de 10 minutos para encontrar a solução.

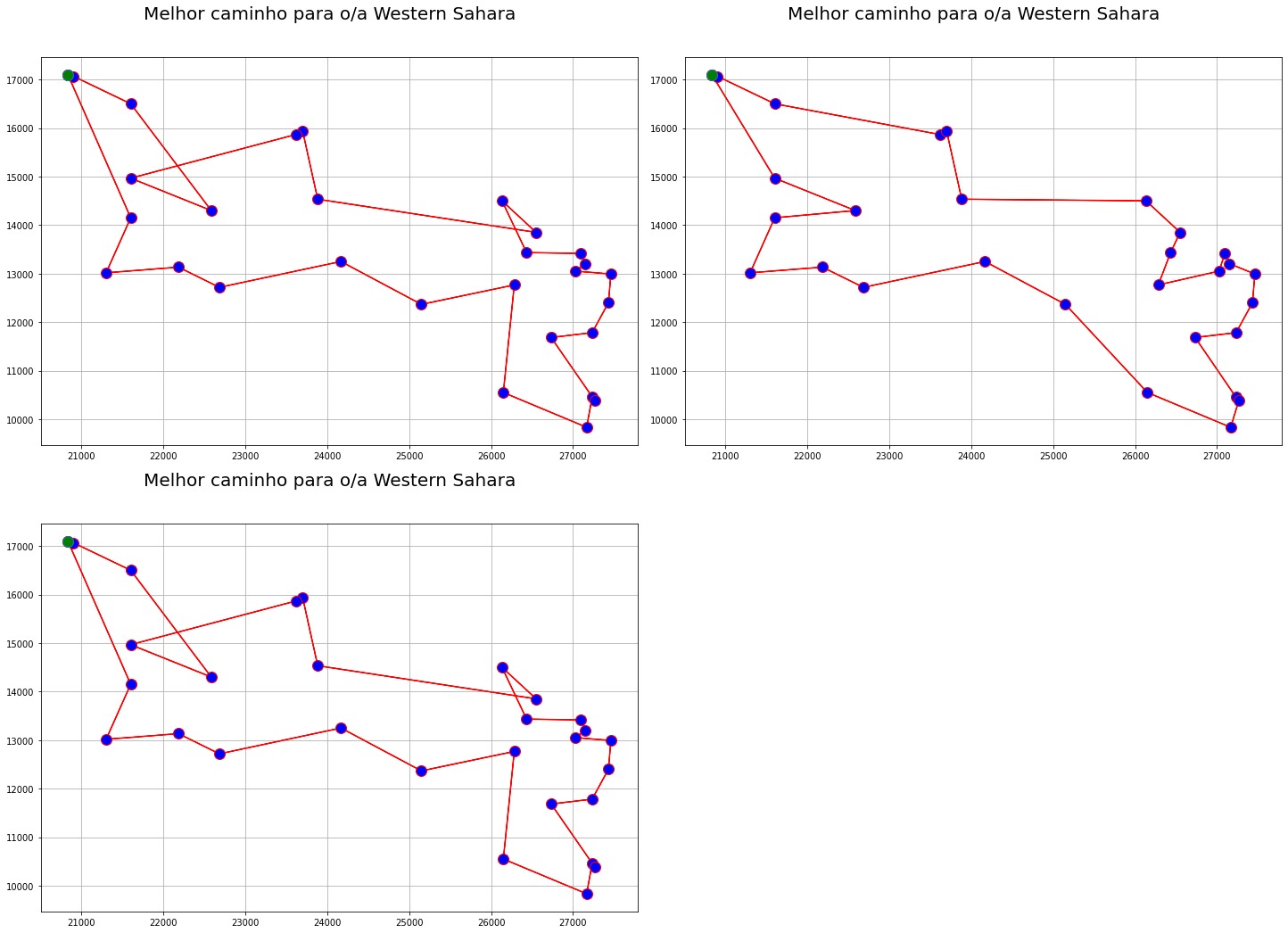


Imagem 7 - soluções encontradas para o conjunto *Western Sahara*, respectivamente, sem heurística, com 2-Opt e LKH.

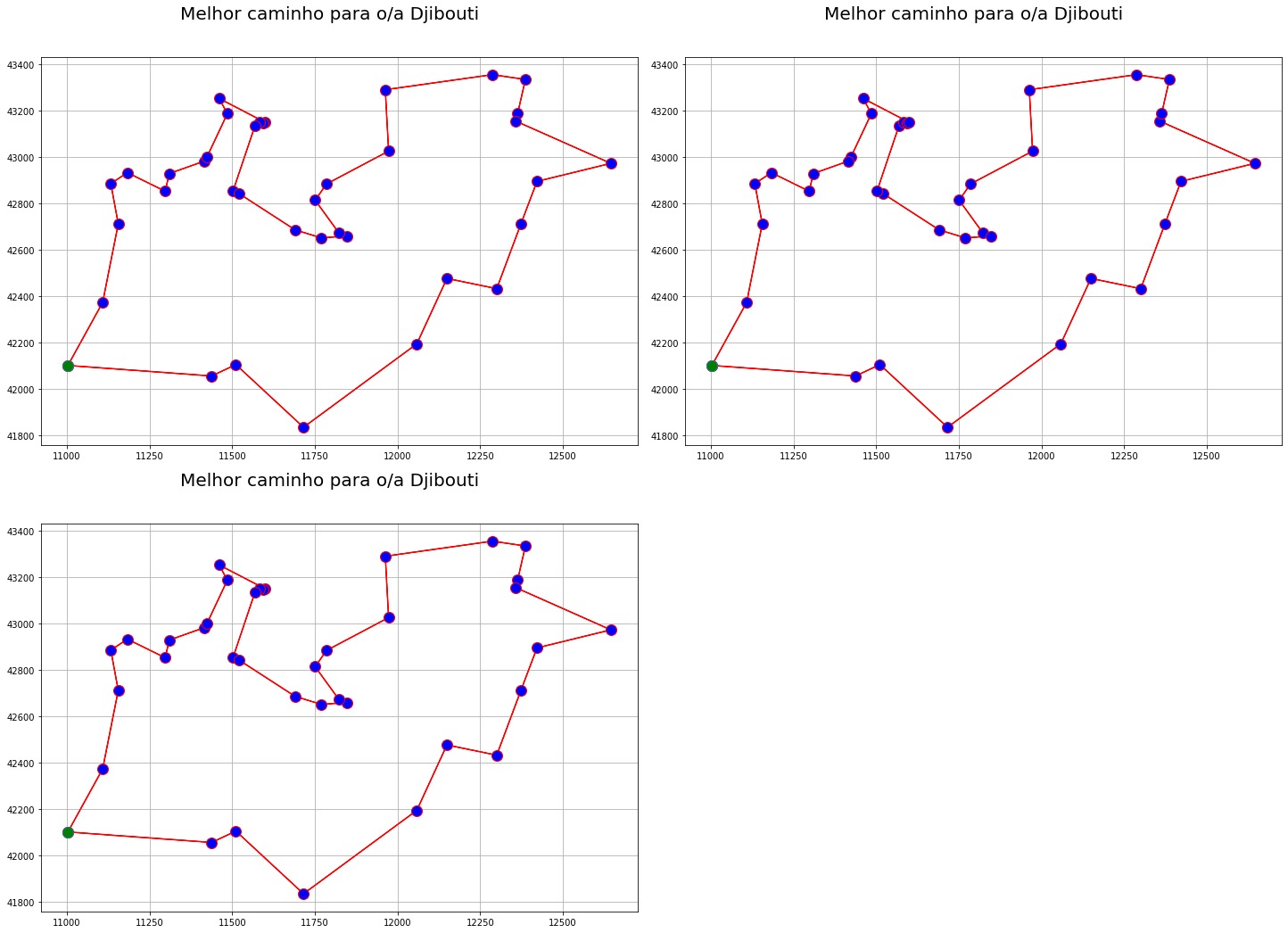


Imagem 8 - soluções encontradas para o conjunto *Djibouti*, respectivamente, sem heurística, com 2-Opt e LKH.

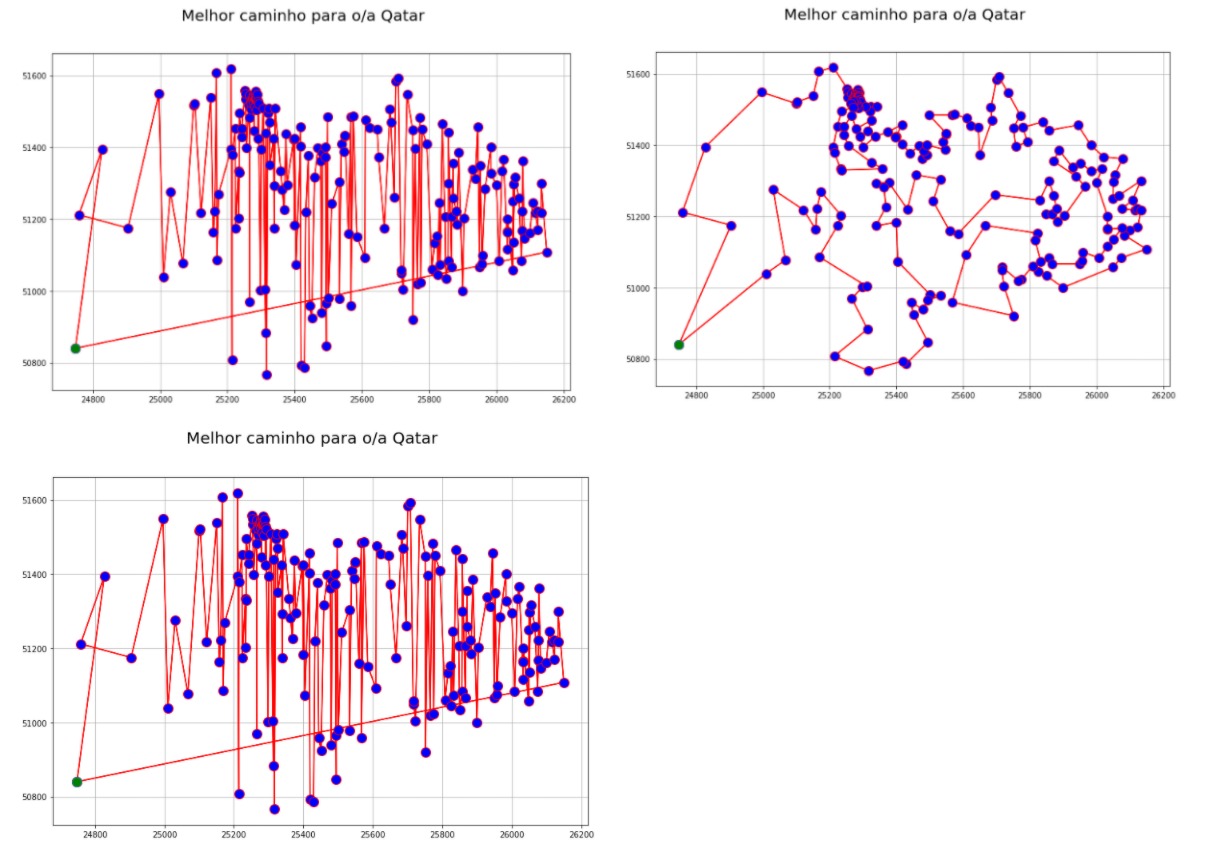


Imagem 9 - soluções encontradas para o conjunto *Qatar*, respectivamente, sem heurística, com 2-Opt e LKH.

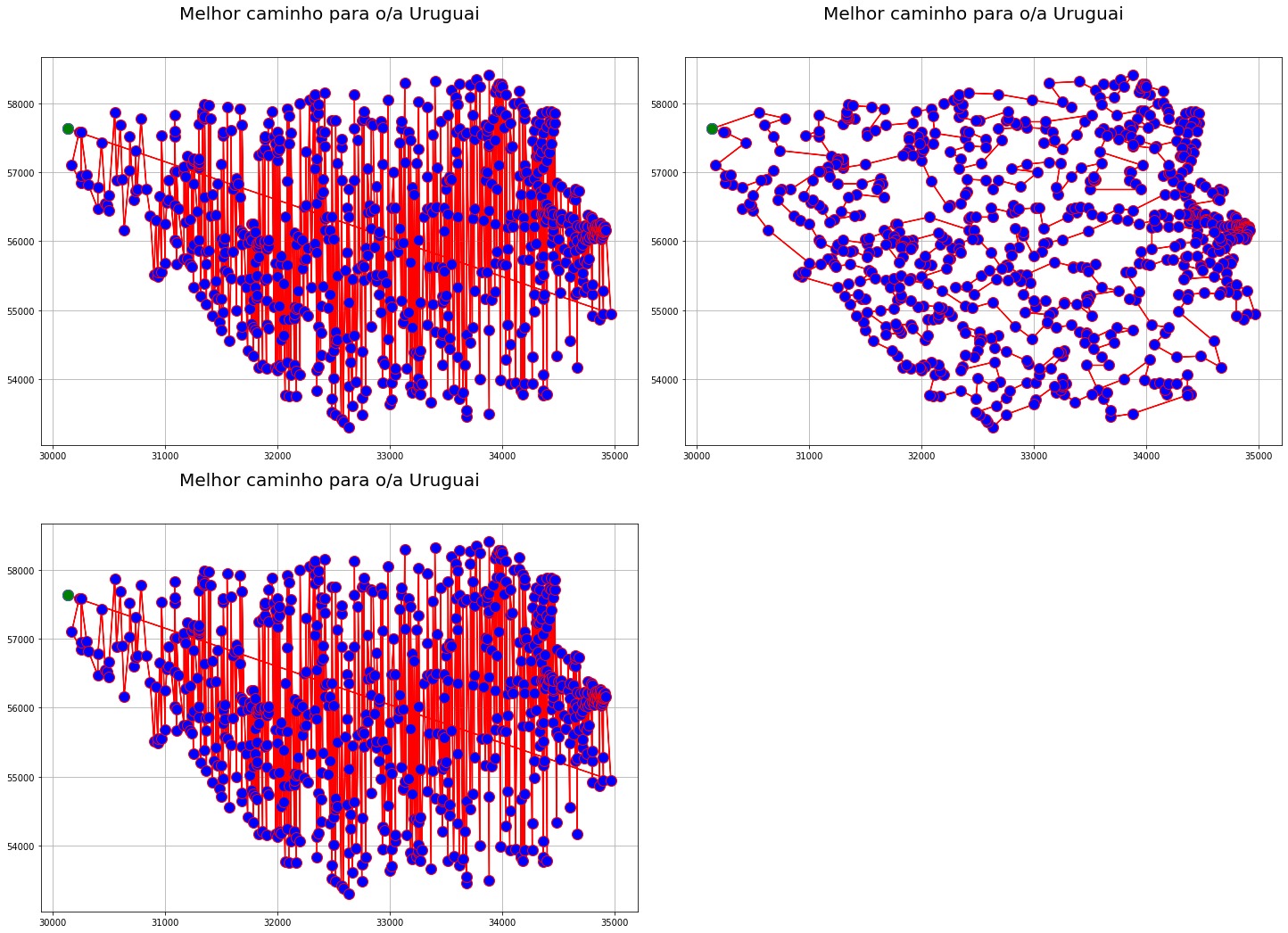


Imagem 10 - soluções encontradas para o conjunto *Uruguay*, respectivamente, sem heurística, com 2-Opt e LKH.

## 6.2 Resultados

Abaixo, listamos os resultados obtidos na mesma ordem de apresentação da [seção anterior](#_ium7fy3s07tj). Criamos uma tabela para cada instância resolvida que apresenta o **tempo limite** que o *solver* teve para encontrar uma solução, o **tempo útil** que ele utilizou, o **GAP** (distância) entre a melhor solução encontrada na execução e a solução encontrada em *Waterloo* utilizando o *software* [Concorde](http://www.math.uwaterloo.ca/tsp/concorde/index.html) e o **número de iterações** realizadas até encontrar a solução. Para as instâncias que alguma das propriedades analisadas não se enquadra, deixamos um hífen com fundo cinza indicando que não há resultado. Para o *Toy Problem* em específico, houve apenas uma execução simples sem o uso de heurísticas e não há como compará-lo com os problemas em *Waterloo* e, por tal, a tabela dele foi simplificada.

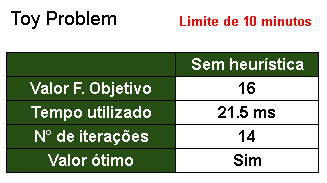


Tabela 1 - Resultados obtidos com a resolução do conjunto *Toy Problem.*

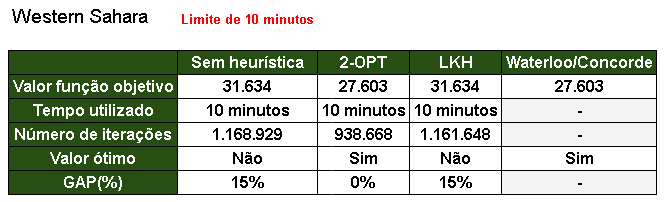


Tabela 2 - Resultados obtidos com a resolução do conjunto *Western Sahara.*

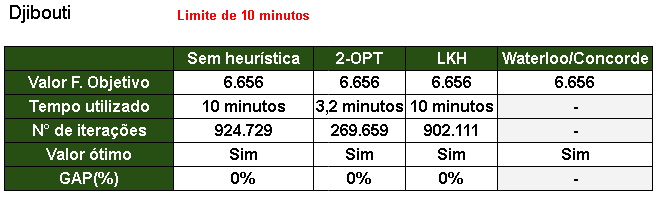


Tabela 3 - Resultados obtidos com a resolução do conjunto *Djibouti.*



Tabela 4 - Resultados obtidos com a resolução do conjunto *Qatar.*

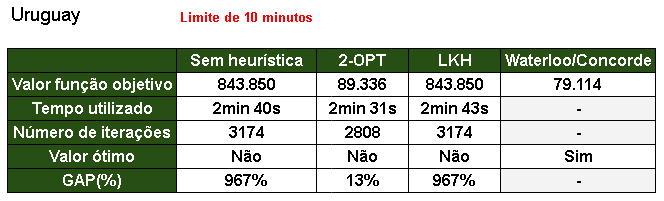


Tabela 5 - Resultados obtidos com a resolução do conjunto *Uruguay.*

## 6.3 Análise dos resultados

Numa análise geral, notamos que a resolução das instâncias com uma dica inicial gerada com a heurística 2-OPT se sobressaiu, em muito, em relação às outras execuções (sem heurística e com a heurística LKH), pois sempre apresentou o melhor GAP em relação ao resultado obtido em *Waterloo*, bem como conseguiu melhorar o desempenho temporal do *solver*.

Para os conjuntos *Western Sahara* e *Djibouti*, foi possível encontrar o mesmo valor ótimo da função objetivo encontrado por *Waterloo* ao utilizarmos a dica inicial gerada pela 2-OPT. Nos demais conjuntos, apesar de não obter o mesmo valor, conseguiu ajudar o *solver* a encontrar um valor muito próximo (menos de 14% de GAP nos dois casos). Além disso, como era de se esperar, o *solver* encontrou dificuldades na resolução das instâncias *Qatar* e *Uruguay*, pois possuem muitas cidades, o que, inclusive, nos obrigou a buscarmos soluções para tentar melhorar o desempenho nesses dois casos. Por isto, optamos pelo uso das duas heurísticas citadas neste trabalho.

No mais, é importante dizer que o *solver* acusou ter encontrado o valor ótimo da função objetivo apenas na resolução das instâncias *Western Sahara* com a heurística 2-OPT e *Djibouti* sem heurística e com as duas heurísticas. Isso se deve, provavelmente, ao *solver* não ter tido tempo suficiente para realizar todas as iterações necessárias para analisar completamente o problema nas outras instâncias, mas não garante que, se fosse permitido executar por mais tempo, iria conseguir obter o valor ótimo.

# **7. Conclusões**

Com a realização deste trabalho podemos concluir com base nos resultados observados que para o *Problema do Caixeiro Viajante* (TSP), o uso de uma boa heurística contribui para a obtenção de uma melhor solução para o problema com menos computação necessária. Vale ressaltar que a escolha da heurística também é importante para obter um melhor resultado, visto que a heurística 2-OPT em todos os conjuntos de dados se sobressaiu, nos testes feitos por nós, sobre a outra heurística utilizada - a LKH.

Além disso, podemos afirmar que o uso de heurísticas juntas ao *solver* para resolver problemas NP-Hard é recomendável e válida, por permitir que as soluções obtidas possam se aproximar da ótima além de, em alguns casos, também diminuir a computação que o *solver* deve realizar oferecendo, assim, um melhor desempenho para a aplicação.

# **8. Referências**

[1] Temel Öncan, İ. Kuban Altınel, Gilbert Laporte, **”*A comparative analysis of several asymmetric traveling salesman problem formulations”***, *Computers & Operations Research*, Volume 36, *Issue* 3, 2009, Páginas 637-654, ISSN 0305-0548, <https://doi.org/10.1016/j.cor.2007.11.008>;

[2] S. Lin & B. W. Kernighan, ***“An Effective Heuristic Algorithm for the Traveling-Salesman Problem”***, Oper. Res. 21, 498-516 (1973);

[3] Keld Helsgaun, ***“An effective implementation of the Lin–Kernighan traveling salesman heuristic”***, *European Journal of Operational Research*, Volume 126, *Issue* 1, 2000, Páginas 106-130, ISSN 0377-2217, <https://doi.org/10.1016/S0377-2217(99)00284-2>;

[4] Stefan Hougardy, Fabian Zaiser, & Xianghui Zhong. (2020). ***”The Approximation Ratio of the 2-Opt Heuristic for the Metric Traveling Salesman Problem”***.

[5] **Problema do caixeiro-viajante**. Disponível em <https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante>. Acesso em 07/11/2020.