UNIVERSIDADE FEDERAL DE UBERLÂNDIA – UFU
FACULDADE DE ADIMINISTRAÇÃO, CIÊNCIAS CONTÁBEIS, ENGENHARIA DE PRODUÇÃO E SERVIÇO SOCIAL – FACES
GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

BRUNA ROBERTO DEBORTOLLI MAYRINK
JAQUELINE MASSINI

ESTUDO DE UM MÉTODO HEURÍSTICO PARA RESOLVER O PROBLEMA DE ENTREGAS DE CARTAS

ITUIUTABA-MG
JULHO DE 2018

BRUNA ROBERTO DEBORTOLLI MAYRINK

JAQUELINE MASSINI

ESTUDO DE UM MÉTODO HEURÍSTICO PARA RESOLVER O PROBLEMA DE ENTREGAS DE CARTAS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Uberlândia, da Faculdade de Administração, Ciências Contábeis, Engenharia de Produção, Serviço Social, como requisito parcial para obtenção do título de Bacharel em Engenharia de Produção.

Orientador: Prof. Dr. José Laércio Dorício

ITUIUTABA-MG

JULHO DE 2018

BRUNA ROBERTO DEBORTOLLI MAYRINK

JAQUELINE MASSINI

Estudo de um método Heurístico para resolver o problema de entrega de cartas

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Uberlândia, da Faculdade de Administração, Ciências Contábeis, Engenharia de Produção, Serviço Social, como requisito parcial para obtenção do título de Bacharel em Engenharia de Produção.

Banca Examinadora:

_____

Prof. Dr. José Laércio Dorício – UFU

Presidente

_____

Prof. Dr. Fernando de Araujo – UFU

Membro

_____

Prof. Dr. Moises Rodrigues Cirilo do Monte – UFU

Membro

Ituiutaba (MG), 02 de julho de 2018

DEDICATÓRIA

Dedico este trabalho a Deus, por sempre está ao meu lado, me fazendo acreditar que tudo é possível. Aos meu pais Geraldo Magela Mayrink e Zilda Antônia Roberto Mayrink, que estiveram sempre me apoiando e orientando em minhas escolhas. Aos meus tios e tias que me ajudaram no decorrer dessa caminhada, em especial Lourdes Roberto que esteve sempre presente, me auxiliando nos momentos de dificuldade.

Bruna Roberto Debortolli Mayrink

Dedico este trabalho de conclusão de curso a Deus e aos meus pais que sempre estiveram comigo ao longo desta caminhada.

Jaqueline Massini

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus por ter sido o meu alicerce durante a graduação, pois a Fé que tenho nele me fez enxergar que Deus não escolhe os capacitados, mas capacita os escolhidos.

Aos meus pais, que são um exemplo de humildade e honestidade. Pois, com eles desenvolvi valores pessoais que me levam a atingir os meus objetivos, com dignidade, caráter, fé, comprometimento e coragem.

A todos os meus amigos que tive a oportunidade de conhecer no decorrer da minha graduação. Em especial, agradeço ao Lúcio e familiares que me acolheram na minha chegada a Ituiutaba – MG. Pois, me ampararam no momento em que mais precisei.

A Eleuza, uma pessoa muito especial, que se tornou não só uma amiga, mas também uma segunda mãe.

Aos professores do curso de Engenharia de Produção, pelos ensinamentos.

E por fim agradeço ao Professor José Laércio Dorício que nos auxiliou em todo o trabalho e esteve presente como mestre e amigo.

Bruna Roberto Debortolli Mayrink

Primeiramente agradeço a Deus que sempre me ajudou e esteve ao meu lado. Esteve ao meu lado durante todo o tempo.

Aos meus pais, Levi Massini e Marcia Regina Brizotto Massini, que fizeram de tudo para que eu conseguisse chegar até aqui. Sempre me mostraram que eu era capaz e me apoiaram em todos momentos. Sem eles, nada disso seria possível!

A todos os professores pois estiveram ao meu lado durante toda esta caminhada, em especial ao meu orientador e amigo José Laercio Dorício.

Jaqueline Massini

“Nem olhos viram, nem ouvidos ouviram, nem jamais penetrou em coração humano o que Deus tem preparado para aqueles que o amam”.

(1 Coríntios 2.9)

RESUMO

Este trabalho propõe um método heurístico para calcular a rota de um carteiro de maneira a otimizar o tamanho da rota. A otimização de rotas é de alta relevância para qualquer tipo de empresa, pois permite economia de tempo e dinheiro na entrega das correspondências. Atualmente, dentre todos os custos logísticos, o transporte representa o maior percentual na ordem de 60% e por esta razão as organizações buscam sua redução por meio de métodos heurísticos cuja a finalidade é obter o menor custo possível. Contudo, a aplicabilidade do método desenvolvido é ampla, sendo possível sua aplicação à problemas de engenharia elétrica, na otimização de circuitos, entre outros. A heurística foi construída como uma mesclagem de técnicas de otimização: 2-3 OPT, algoritmo guloso, Dijkstra, busca Tabu e técnicas heurísticas de controle. Para testar o método, uma região aleatória da cidade de Ituiutaba – MG foi escolhida, para que dentro da mesma, fossem criadas as rotas de entregas de maneira a minimizar os custos logísticos.

Palavras-chave: Caixeiro viajante, Controle de Busca, Dijkstra , 2/3-OPT, Roteamento.

ABSTRACT

This job is a heuristic method to calculate the route in order to optimize the size of the route. The optimization of routes is of high relevance for any type of company because it allows the saving of time and money in the delivery of the mail. Currently, among logistic costs, transport represents the highest percentage in order of 60% and for this reason the organizations seeks to reduce it by means of heuristic methods in order to obtain the lowest possible cost. However, the application of the method is ample; its application being possible for problems of electrical engineering, in the optimization of circuits, among others. The Heuristic was constructed, as a mixture of optimization techniques: 2-3 OPT, greedy algorithm, dijkstra, Tabu search and heuristic control techniques. To test the method, a random region of the city of Ituitaba – MG was chosen, so that it was the same way, to be created as the delivery routes in order to minimize logistic costs.

Keywords: Traveler Clerk, Search Control, Dijkstra, 2/3 -OPT, Routing.

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicativos
DLA	Diferença de Latitude
DLO	Diferença de Longitude
GUI	Interface Gráfica do Usuário
ITA	Instituto Tecnológico de Aeronáutica
MD	Matriz de distâncias
MR	Matriz de caminho
NM	Milhas Nauticas
OPT	Optmized Production Tecnology
PCV	Problema do Caixeiro Viajante
PO	Pesquisa Operacional
TSP	*Traveling Salesman Problem*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Logística vem se tornando um princípio de suma importância nas organizações visando alcançar qualidade em seus produtos e serviços oferecidos. Por essa razão, a mesma integra todo o processo produtivo, desde o armazenamento dos insumos até a distribuição do produto acabado. Assim, a logística desencadeia diversos custos, porém o que mais se destaca é o custo de transporte, representando cerca de 60%. Em vista disso, o presente trabalho busca minimizar as rotas e reduzir tal percentual.

O Problema do Caixeiro Viajante, que busca resolução do problema de roteamento aplicado à entrega de cartas, é um problema de classe do tipo NP-Hard da área de otimização combinatorial, onde se aplica a questões de alta complexidade. Por pertencer a essa classe de problemas, não há algoritmos ou técnicas analíticas que sejam eficientes para resolvê-lo na exatidão. Nesse projeto, buscou-se uma aplicação desse tipo de problema ao roteamento da entrega de cartas a uma região da cidade de Ituiutaba, contudo a técnica pode ser aplicada a qualquer cidade do mundo ou mesmo a projeto de circuitos elétricos, onde a otimização da distância percorrida pelos sinais elétricos é importante. A cidade de Ituiutaba foi escolhida como teste por se tratar de um projeto local e da importância de se fazer aplicações regionais das pesquisas realizadas na universidade.

O problema consiste em atender a um conjunto de clientes que precisam receber cartas a partir de um depósito central. A ideia é visitar todos os clientes, entregar suas correspondências, respeitando algumas restrições como, por exemplo, o tempo da viagem dos veículos e a capacidade de carga dos mesmos (que nesse trabalho foi desconsiderado por simplificação), e garantindo que toda a entrega seja feita ao menor custo possível. Este é um problema com alta ligação à área logística, pois a diminuição de custos de transportes é uma pressão constante nas cadeias de suprimentos. Além disso, possui alto grau de aplicabilidade pois as diferentes versões do problema refletem de maneira muito real as decisões que precisam ser tomadas diariamente por transportadoras. Esse problema de roteamento pode ser formulado como um problema de programação linear inteira. Os clientes são numerados de 1, ... , n e define-se:

$$x_{ij} = \begin{cases} 1 & \text{a rota vai do cliente i para o cliente j} \\ 0 & \text{caso contrário} \end{cases}$$

Para $i, j \in \{1, 2, \ldots, n\}$, $i \neq j$ e c_{ij} a distância do cliente i ao cliente j, onde n é o número de clientes a ser roteado, o problema é formulado como:

Função Objetivo:

$$Min \sum_{i=0}^{n} \sum_{i \neq j \,;\, j=1}^{n} c_{ij}\, x_{ij}$$

Sujeito às restrições:

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

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

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \,, \forall\, V \subset S \neq 0 \,;$$

$$x_{ij} \in \{0 \,;\, 1\} \qquad i, j = 1, \dots, n.$$

Onde S é um subconjunto do conjunto $\{1,2, \dots, n\}$ e o símbolo $| \cdot |$ representa a cardinalidade do conjunto. Desta forma, a função objetivo é a representação matemática da minimização do somatório das distâncias entre os nós da rota, considerando somente os vértices que pertencem à rota específica (x_{ij} = 1). A primeira e a segunda restrição garantem que cada nó *i* e *j* só participem da rota uma única vez como conexão de saída e uma única vez como conexão de chegada. A terceira restrição garante que não haja subrotas, ou seja, uma rota que não inclua todos os nós que devem ser visitados e a quarta restrição garante, por sua vez, que x_{ij} deve ser uma variável binária, ou seja, que só admita valores 0 ou 1.

1.1 Objetivo

Objetivo deste trabalho é a criação de um algoritmo que é capaz de calcular uma rota de maneira a otimizar o tamanho da mesma. Os pontos de entregas para as cartas são escolhidos de forma aleatória e através deles, o algoritmo faz o cálculo para que as cartas sejam entregues

no menor tempo possível. Também foi construído uma interface visual em *Java script* para a visualização dos resultados sobre o mapa na web e na montagem de malha de roteamento.

1.2 Estrutura do Trabalho

O trabalho possui 5 capítulos. No primeiro capítulo encontra-se a introdução. O segundo capítulo, trata do referencial teórico, que possui 7 partes, na primeira parte define-se Pesquisa Operacional. Na segunda, é exposto o conceito de heurística e suas melhorias de rotas. A terceira parte trata-se da meta-heurística e busca Tabu. A quarta apresenta o método exato conhecido como Dijkstra. A quinta parte refere-se a formulação de problemas clássicos como Caixeiro Viajante e o Carteiro Chinês. Na sexta, temos uma breve explicação sobre grafos e a sétima parte engloba conceitos logísticos e os custos relacionados.

No terceiro capítulo, é apresentada a metodologia utilizada na pesquisa. No quarto capítulo são apresentados os resultados. No quinto capítulo é realizado uma análise dos resultados obtidos. E o sexto remete-se as considerações finais.

2 REFERENCIAL TEÓRICO

2.1 Pesquisa Operacional

Pesquisa Operacional (PO) origina-se do termo *Operational Research*. Este termo surgiu devido à invenção do Radar na Inglaterra durante a Segunda Guerra Mundial e desde então é estudada até os dias de hoje. A Pesquisa Operacional iniciou no Brasil na década de 1960, sendo o primeiro simpósio em 1968 no ITA, em São José dos Campos, SP.

Arenales (2007, p.3) diz que a "Pesquisa Operacional consiste no desenvolvimento de métodos científicos de sistemas complexos, com a finalidade de prever e comparar estratégias ou decisões alternativas".

A PO é uma abordagem cientifica bastante difundida no âmbito das Engenharias que utiliza modelos matemáticos, estatísticos e de algoritmos na ajuda à tomada de decisão. É uma ferramenta multidisciplinar que possui uma área vasta de oportunidades de carreiras e de trabalhos. Um profissional em PO está apto para compreender e solucionar problemas, através do uso de métodos analíticos e o foco em resultados. Muito utilizada nas tomadas de decisões que pode ser usada em diversas decisões em nosso dia-a-dia, principalmente no mundo empresarial, onde as decisões são um fator-chave.

2.2 Heurística

A heurística consiste em procedimentos, também conhecidos por "busca cega" que visam encontrar a solução de problemas através das diversas possibilidades de buscas que podem existir. Ou seja, corresponde a um método que parte de uma solução viável, e por intermédio de comparações, é direcionado a encontrar resultados próximos ao ponto ótimo. Contudo, a heurística não descobre respostas precisas, mas busca-se localizar os resultados mais satisfatórios.

Em outras palavras, podemos dizer que as técnicas heurísticas, condizem a uma busca ininterrupta e baseada em comparação, com diversos ótimos locais, no qual o resultado encontrado é excelente em relação às condições estabelecidas.

2.2.1 Heurística 2 e 3 OPT

As heurísticas de melhorias de rota conhecidas por 2-OPT e 3-OPT consistem em permutar arcos em uma rota inicial factível, buscando encontrar uma rota de custo menor. Na heurística 2-OPT dois arcos são desligados e substituídos outros dois de modo que a distância total na nova rota formada seja menor que na rota inicial. Analogamente ocorre na heurística 3-OPT onde três arcos são permutados. Estes procedimentos terminam geralmente em um ótimo local, e são considerados métodos eficientes para resolver o problema do caixeiro viajante. (MICHALEWICZ; FOGEL, 2004)

2.3 Meta-heurística

A meta-heurística corresponde a um procedimento empregado para solucionar problemas do tipo NP-hard, cujo objetivo é localizar a solução viável mais eficiente, ou seja, o resultado mais aproximado da solução ótima. Tal técnica pode ser implementada tanto para maximizar ou minimizar uma dada função, com suas respectivas restrições.

> [...]. Uma meta-heurística é um método de resolução geral que fornece tanto uma estrutura quanto diretrizes de estratégia gerais para desenvolver um método heurístico específico que se ajuste a um tipo de problema particular. A meta-heurística se tornou uma das mais importantes técnicas na caixa de ferramentas dos profissionais da PO. (HILLIER; LIEBERMAN. 2010, p. 599)

2.3.1 Busca Tabu

Em meados de 1986 a Busca Tabu foi instituída por Fred W. Glover, com o propósito de realizar a otimização através da busca por soluções ótimas. Tabu é uma palavra pertencente ao idioma da Polinésia, conhecido como Tongan, em que seu significado corresponde a um objeto sagrado que não podia ser tocado.

> A mais importante associação com o uso tradicional da palavra tabu, entretanto, vem do fato que tabus são normalmente transmitidos através da memória social, que está sujeita a mudanças ao longo do tempo. Isto cria a conexão fundamental para o significado de "tabu" em Busca Tabu: os elementos proibidos da Busca Tabu recebem seus status pela dependência em uma memória evolutiva, que

> permite a este status mudar de acordo com o momento e circunstância. (GOMES, ANDRÉ. 2009, p. 5)

Portanto, podemos declarar que a Busca Tabu, corresponde a uma meta heurística no qual auxilia um algoritmo de busca a ultrapassar a otimização local, almejando localizar uma solução ótima ou que esteja próxima do ótimo global. Por essa razão, a presente ferramenta busca não regressar a um ótimo local que já havia sido visitado.

2.4 Dijkstra

Edsger Wybe Dijkstra foi um pesquisador holandês da cidade de Roterdã, conhecido por suas contribuições na área computacional, em que se destaca o problema de caminho mais curto, que pode ser solucionado por intermédio do algoritmo de Dijkstra. Tal algoritmo tem a finalidade de estabelecer o mínimo trajeto entre o nó inicial e nó final de uma rede, considerando que os custos dos arcos sejam positivos, assegurando que a solução ótima seja identificada. No algoritmo Dijkstra, um nó k é dito como rotulado ou fechado, no momento em que se depara com percurso mínimo entre o nó fonte até o nó atual. Por sua vez, os nós que não detectaram o caminho mínimo são classificados como não rotulados ou abertos. Por conseguinte, “em vez de minimizar a distância total percorrida, pode-se minimizar também o custo total ou o tempo total de viagem” (BELFIORE; FÁVERO, 2013, p. 316).

2.5 Formulação de Problemas Clássicos

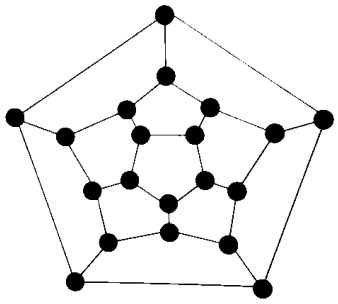
Em meados de 1980, as linguagens algébricas foram instituídas sistematicamente, por meio de símbolos e letras que tinham a função de representar os números, com a finalidade de solucionar diversos problemas, impactando principalmente o campo de otimização. Essas expressões mais robustas, proporcionam aos usuários uma maior facilidade de escrita dos modelos de programação, sendo similar a notação algébrica. Ademais, os dados são arquivados em documentos independentes, viabilizando a redução de erros, aumentando a eficiência no desenvolvimento, concentração e clareza do modelo. Em seguida, temos alguns exemplos de problemas clássicos e suas importâncias.

2.5.1 Problema do Caixeiro Viajante

O problema do caixeiro-viajante (PCV) deriva-se do termo *Traveling Salesman Problem* (TSP), caracterizando um problema de otimização, proposto por Willian Rowan Hamilton, um astrônomo, físico e matemático irlandês, cuja a ideia central era determinar caminhos conhecidos como hamiltonianos. Em vista disso, Willian R. Hamilton "[..] propôs um jogo cujo desafio consistia em encontrar uma rota por meio dos nós de um dodecaedro (sólido regular com 20 nós, 30 arcos e 12 faces) de tal modo que a rota iniciasse e terminasse no mesmo nó, sem nunca repetir uma visita" (BELFIORE; FÁVERO, 2013).

De acordo com Arenales (2007), o presente problema faz alusão a questões de roteamento em nós, e são indicados por meio de grafos, que por sua vez podem ser classificados como orientados ou não orientados. Ademais o PCV possui como finalidade, abranger um grupo de cidades, onde o caixeiro parte de uma cidade inicial ou armazém, com o objetivo de percorrer todas as cidades ou uma parte delas apenas uma vez, retornando logo em seguida para a cidade inicial, podendo assim otimizar uma ou mais metas estabelecidas. A Figura 1 faz referência a uma possível solução para o jogo hamiltoniano.

Figura 1. Grafo do Problema hamiltoniano



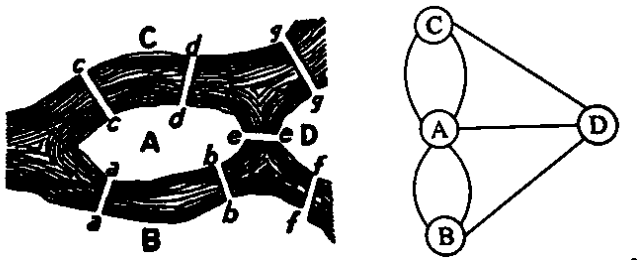
Fonte: Arenales (2007)

2.5.2 Problema do Carteiro Chinês

Segundo Arenales (2007), os problemas de carteiro chinês concernem à categoria de roteamento em arcos, que podem ser divido em grafos orientados ou não orientados. Além disso, o objetivo primordial é definir o menor custo entre um grupo especifico de arestas pertencentes a um grafo, tendo ou não restrições. Estes problemas estão presentes em diversas atividades do cotidiano, como a entrega e coleta de cartas, vigilância de ruas pela polícia, roteamento de transporte escolar e até mesmo no processo de coleta de lixo.

Um dos primeiros problemas a respeito de roteamento de arcos, ocorreu na cidade de Konigsberg que era de posse da Alemanha e atualmente pertence aos Russos, onde adquiriu uma nova nomenclatura, denominada de Kaliningrad. O grande obstáculo que existia, era identificar se havia algum caminho fechado que passasse nas pontes apenas uma vez, como mostrado na Figura 2. Todavia, em 1736, por intermédio de diversas pesquisas Leonhard Euler se deparou com alguns requisitos para que um caminho fechado pudesse existir, e assim evidenciou que no problema citado anteriormente não há a existência de tal caminho. Por conseguinte sua conquista foi de grande importância para os princípios de grafos.

Figura 2. Pontes de Konigsberg e Grafo associado



Fonte: Arenales (2007)

2.6 Grafos

"Um grafo é uma estrutura G = (V, A), onde V é um conjunto discreto e A é uma família cujos elementos (não vazios) são definidos em função dos elementos de V [...]" (NETTO, 2006). Em outras palavras, podemos afirmar que V corresponde aos vértices e A faz referência as arestas.

A origem dos grafos se deu através das pesquisas desenvolvidas por Leonhard Euler em 1736, acerca do problema das pontes de Konigsberg. A Teoria dos Grafos é vista na atualidade como um princípio de grande relevância para a matemática discreta. E além disso, vem sendo empregada em inúmeras áreas como informática, sociologia, economia, entre outros, visto que um grafo é composto por um modelo matemático ideal, utilizado para analisar a ligação de diversos objetos discretos.

2.7 Logística

Logística originou-se das atividades militares, pois na guerra cuidava do planejamento, armazenamento, distribuição e manutenção de materiais, como armas, roupas, além de alimentos, saúde, transportes e entre outros.

Até pouco tempo, a Logística era vista somente pelo seu lado clássico, exercendo funções de transporte, armazenagem e disponibilidade de bens para os processos de transformação e consumo. Nos dias de hoje, é fundamental para obtenção de vantagem competitiva entre as organizações através de seu caráter estratégico.

Segundo Lambert et al (1998), o objetivo da Logística é entregar "produto certo, no lugar certo, no momento certo, nas condições certas e pelo custo certo".

De acordo com Lambert et al (1998), a logística empresarial cresce mais a cada dia graças à busca frequente das empresas em assegurar que seus produtos e serviços tenham qualidade, pois é ela que administra os recursos financeiros, planeja a produção, o armazenamento, o transporte e a distribuição dos materiais.

2.7.1 Custos Logísticos

Custos logísticos são os custos relacionados com a logística de uma empresa, dentre eles temos: os custos de armazenagem, de transporte, de embalagem, de tecnologia, entre outros. Os custos logísticos estão entre os custos mais importantes, pois só perdem para os custos da própria mercadoria. Por esta razão, gerir os custos de maneira eficaz é crucial para a sobrevivência da empresa.

2.7.2 Custos de Transporte

O transporte é considerado como um dos processos mais importantes da logística, pois envolve o deslocamento dos materiais de maneira tanto interna quanto externa. Além de transportar é necessário que tudo chegue no momento certo sem que tenha nenhum tipo de avaria. Além de cumprir os prazos é necessário que o cliente esteja satisfeito com sua entrega, permitindo então que a empresa tenha um diferencial competitivo no mercado.

Os principais fatores que influenciam os custos de transportes são: distância, volume, densidade, facilidade de acondicionamento, facilidade de manuseio e os fatores ligados ao mercado.

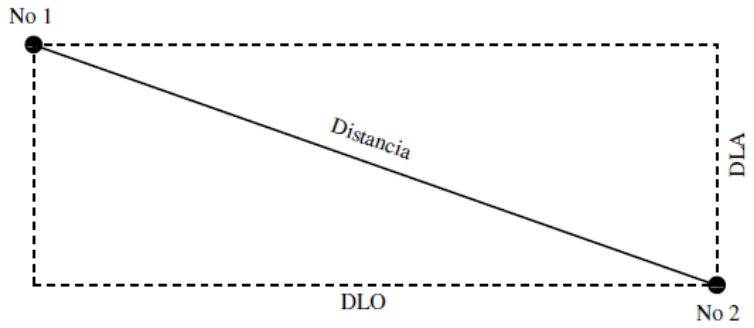
3 METODOLOGIA

3.1 Conversão de Unidades

As coordenadas geográficas, latitudes e longitudes, de cada nó *i* foram convertidas para quilômetros ou metros. Na resolução do roteamento foi necessário converter a distância angular para metros. Assim, como é conhecido, 1 grau é aproximadamente 60 NM (milhas náuticas) e 1 NM equivale a 1852 metros. Portanto, $60' = 60$ NM logo $1' = 1$ NM. Analogamente, $1' = 60''$, assim pode-se concluir que $60'' = 1$ NM, ou seja, $1'' = 1/60$ NM. Com estas equivalências é fácil transformar qualquer unidade angular para milhas, quilômetros ou metros. Considere o seguinte exemplo: converter $23^o 30' 36''$ em quilômetros. Basta fazer o cálculo a seguir:

$$\begin{aligned} 23^o \times 60 &= 1.380 \\ 30' \times 1 &= 30 \\ 36'' \div 60 &= 0,6 \\ \hline 1.380 + 30 + 0,6 &= 1.410,6 \times 1,852 = 2.612,4\ km. \end{aligned}$$

Figura 3. Distância entre dois nós.



Fonte: Autores

No sistema de coordenadas geográficas pode-se utilizar o sistema cartesiano para indicar localidades se as distâncias forem suficientemente pequenas permitindo-se desprezar a curvatura da Terra. Portanto, para calcular a distância entre dois pontos faz-se uma aplicação direta do teorema de Pitágoras, considerando a curvatura da terra desprezível cujos nós *i* e *j* estão suficientemente próximos. Assim, a distância entre dois nós nada mais é do que a hipotenusa de um triângulo retângulo onde a diferença de latitude (DLA) e de longitude (DLO) são os catetos. Conforme mostra a Figura 3. Portanto, considerando D como a distância do nó *i* ao nó *j*, tem-se:

$$D^2 = DLA^2 + DLO^2$$

Considere como exemplo o cálculo da distância entre os nós 1 e 2 com coordenadas dadas pela Tabela 1.

Tabela 1. Latitude e Longitude dos nós

Nó *i*	Latitude do nó *i*	Longitude do nó *j*
0	$18^o58'53.58''S$	$49^o28'17.15''O$
1	$18^o58'57.26''S$	$49^o27'57.13''O$
2	$18^o58'58.07''S$	$49^o28'0.77''O$
3	$18^o58'54.38''S$	$49^o28'1.58''O$
4	$18^o58'53.63''S$	$49^o27'57.81''O$
5	$18^o58'55.18''S$	$49^o28'5.19''O$
6	$18^o58'58.78''S$	$49^o28'4.44''O$

Fonte: Autores

Sejam X_1e Y_1 a latitude e a longitude, respectivamente, do nó 1 em km e analogamente para o nó 2, então:

$$X_1 = \left(18 \times 60 + 58 + \frac{57,26}{60}\right) \times 1,852 = 2109,3434 \text{ km}$$

$$Y_1 = \left(49 \times 60 + 27 + \frac{57,13}{60}\right) \times 1,852 = 5496,6474 \text{ km}$$

$$X_2 = \left(18 \times 60 + 58 + \frac{58,07}{60}\right) \times 1,852 = 2109,3684 \text{ km}$$

$$X_1 = \left(49 \times 60 + 28 + \frac{0,77}{60}\right) \times 1,852 = 5496,7598 \text{ km}$$

$$DLA = X_2 - X_1 \rightarrow 2109,3684 - 2109,3434 = 0,025 \text{ km}$$

$$DLO = Y_2 - Y_1 \rightarrow 5496,7598 - 5496,6474 = 0,1124 \text{ km}$$

$$D = \sqrt{0,025^2 + 0,1124^2} = \sqrt{0,000625 + 0,012633760} = 0,11515 \text{ km} \rightarrow 115,15 \text{ m}$$

Desta forma foi possível montar uma matriz de distância linear denominada Mdists armazenando as distâncias entre os nós *i* e *j* dos conjuntos de nós da malha.

3.2 Interface Gráfica

O roteamento dos clientes é feito sobre uma malha composta por um conjunto de pontos denominados vértices ou nós, que representam a conexão entre cada rua e/ou avenida da cidade,

e por um conjunto de arestas ou segmentos de reta orientados, que representam a direção de percurso para cada rua e/ou avenida. A Figura 4 ilustra um conjunto de nós e arestas orientadas escolhidas sobre o mapa da cidade de Ituiutaba – MG.

Figura 4. Malha utilizada no roteamento.

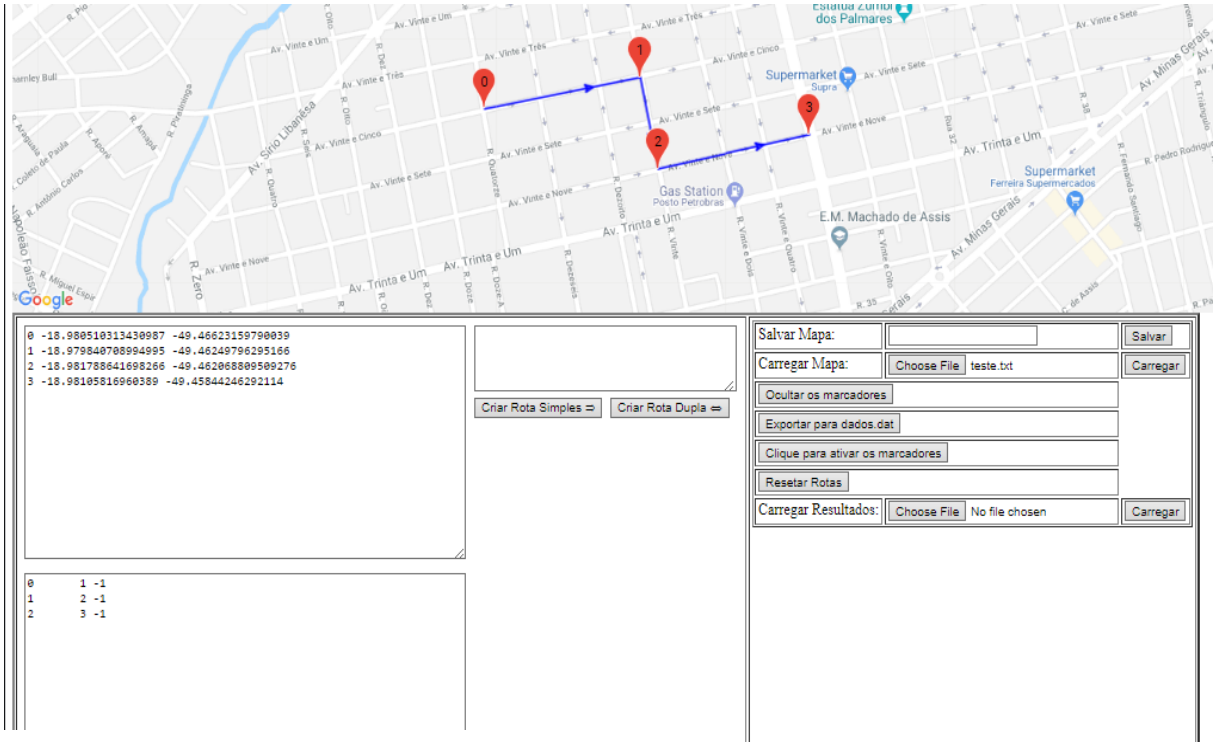
Fonte: API do Google Maps

Na Figura 4, cada balão vermelho (também chamado de marcador) representa um nó (ou cruzamento entre ruas e avenidas) e as arestas em azul representam as ruas e/ou avenidas. As setas indicam a direção do percurso, por exemplo, o nó 63 está ligado ao nó 64 por uma aresta indicando que a direção de percurso é do nó 63 para o nó 64. A esse conjunto de nós e arestas podemos associar um grafo orientado G = (V, A), onde V(G) é o conjunto de vértices do grafo e A(G) é o conjunto de arestas do grafo. Sobre esse grafo será realizado o roteamento dos nós (vértices) a serem visitados para entrega de correspondências.

O grafo G = (V, A), representando a malha do roteamento, foi construído utilizando uma interface gráfica de usuário - GUI, implementada utilizando linguagem de programação *JavaScript* (muito parecida com a linguagem de programação C^{++}) através de uma API (Interface de Programação de Aplicativos) do *Google Maps*. A Google fornece o licenciamento gratuitamente para fins de pesquisa ao acesso a sua API. Essa API permite a implementação de aplicativos utilizando o banco de dados de mapas do *Google Maps*. A grande vantagem da interface gráfica desenvolvida é que ela pode ser aberta em qualquer computador com um

navegador (com suporte a html 5) e acesso à internet sem necessidade de instalação de nenhum software adicional. A Figura 5 mostra a interface gráfica desenvolvida.

Figura 5. Interface Gráfica



Fonte: API do Google Maps

A interface gráfica apresentada na Figura 5, é constituída de um quadro superior contendo o mapa importado pela API do Google (que pode estar em modo mapa ou satélite). Sobre o mapa foram implementados os balões vermelhos representando os nós e as arestas orientadas em azul representando as conexões entre os nós.

A primeira coluna à esquerda abaixo do mapa possui dois quadros na vertical. O primeiro quadro mostra a lista de coordenadas geográficas de cada nó em latitude e longitude, onde minutos e segundos foram convertidos em graus. Para latitude/longitude, o sinal de menos representa latitude/longitude sul/oeste, e o sinal de mais representa latitude/longitude norte/leste. A latitude é a coordenada geográfica ou geodésica definida na esfera, no elipsoide de referência ou na superfície terrestre, que é o ângulo entre o plano do equador e a normal à superfície de referência. A latitude mede-se a partir do equador, varia entre 90º sul, no Polo Sul (ou polo antártico – negativa), e 90º norte, no Polo Norte (ou polo ártico – positiva). A longitude, por sua vez, descreve a localização de um lugar na Terra medido em graus, de zero a 180º para leste (positivo) ou para oeste (negativo), a partir do Meridiano de Greenwich.

Ainda na primeira coluna, mostra-se um segundo quadro contendo a lista de adjacências do grafo G = (V, A). No exemplo da Figura 5, o nó 0 incide sobre o nó 1, o nó 1 incide sobre o nó 2 e o nó 2 incide sobre o nó 3. O número -1 indica o final da lista de vizinhos de um dado nó *i* em V. A segunda coluna contém botões para gerar as arestas de um nó *i* para um nó *j,* usando orientação de mão simples ou dupla. A terceira coluna contém botões para gerenciar o mapa, salvar e carregar os dados para utilização no roteador ou para visualização das rotas geradas pelo roteador.

Os nós são gerados automaticamente clicando sobre o mapa e as coordenadas geográficas do nó gerado são armazenadas em uma lista e mostradas no quadro 1 referente a Figura 5. Para gerar a aresta, seleciona-se um nó *i* clicando sobre o mesmo, em seguida seleciona-se um segundo nó *j* clicando sobre o mesmo e clica-se no botão Criar Rota Simples ou Criar rota dupla. Com esse procedimento, aparece desenhado no mapa, em azul, a aresta conectando os nós *i* e *j* com uma seta indicando a orientação e no quadro 2 referente a Figura 5 é atualizada a lista de adjacências criando-se uma conexão entre os nós *i* e *j*.

Após a criação da malha de roteamento pela interface gráfica, os dados podem ser exportados para o arquivo *dados.dat* que está formatado para leitura pelo roteador, que foi desenvolvido em linguagem de programação C^{++}.

3.3 Algoritmo de Roteamento

Dado um conjunto de vértices C ∈ V, representando o conjunto de clientes cujas correspondências deverão ser entregues – ou o conjunto de nós a rotear – e os parâmetros iniciais, como o número total de iterações, período tabu, etc. O algoritmo de roteamento pode ser dividido em:

- Etapa de leitura dos dados e parâmetros;
- Primeira etapa do roteamento: construção das matrizes de distância e listas de caminhos entre todo nó i e j em G = (V, A);
- Segunda etapa do roteamento: construção inicial e processo iterativo com heurísticas de melhoria e controle de busca.

Leitura dos Dados e Parâmetros: Nessa etapa os dados gerados pela interface gráfica desenvolvida são lidos do arquivo *dados.dat*. Desse arquivo lê-se as coordenadas geográficas de cada nó e a lista de adjacências do grafo G = (V, A). Nessa etapa converte-se graus para metros conforme a seção 3.1. Os parâmetros utilizados pelo roteador são lidos de um arquivo

texto denominado *parametros.dat*, esse arquivo contém o valor do período tabu, o número máximo de iterações do algoritmo, a tolerância em metros para o processo de controle de busca e velocidade média em km/h (utilizada para estimar o tempo de percurso).

Primeira Etapa do Roteamento – construção de uma matriz de rotas MR e de uma matriz de distâncias MD que ligam os nós *i* e *j* em V pelo melhor caminho no grafo. A distância de se percorrer o caminho do nó *i* ao nó *j* sobre o grafo G = (V, A) em geral é diferente da distância de se percorrer o caminho do nó *j* ao nó *i*, isto é a matriz de distâncias MD não é simétrica. Isso se dá ao fato das distâncias e rotas entre dois nós não ser linear e porque as ruas e avenidas são orientadas. Assim, para se percorrer um percurso de um dado ponto A a um dado ponto B é necessário contornar quarteirões e seguir a correta orientação das ruas e avenidas que torna a matriz MD não simétrica.

A construção dessas matrizes foi feita utilizando o algoritmo de Dijkstra, concebido pelo cientista da computação holandês Edsger Dijkstra em 1956. Esse algoritmo é um método exato que permite a construção de uma rota dentro de um grafo orientado G = (V, A) de forma a se obter o menor caminho de um nó *i* a um nó *j* com tempo de computação na ordem O (m + n log(n)), onde m representa o número de arestas e n representa o número de vértices. Dado um nó inicial do grafo G = (V, A), e considerando que a distância marcada em um nó B em V(G) é a distância do nó B ao nó inicial, o algoritmo de Dijkstra consiste em:

1. Marcar todos os nós do conjunto V(G) como não visitados. Criar um conjunto de todos os nós não visitados denominado conjunto não visitado S_u.
2. Atribuir inicialmente a todos os nós uma distância da seguinte forma: atribuir zero para o nó inicial e infinito para todos os outros nós e atribuir o nó inicial como o atual.
3. Para o nó atual, pegar todos os seus vizinhos não visitados e calcular a distância entre eles até o nó atual. Comparar cada distância calculada com o valor correntemente atribuído e atualizar o valor corrente com a menor delas.

Por exemplo, se o nó atual é o nó A e está marcado com distância 7 até o nó inicial, e supondo que a aresta conectando o nó atual com o nó B tem comprimento 3, então a distância de B até A será 7 + 3 = 10 que representa a distância do nó B ao nó inicial. Se o B foi previamente marcado com uma distância maior que 10 então troque-a por 10, caso contrário mantenha o valor marcado em B.

4. Quando todos os vizinhos do nó atual foram analisados, marcar o nó atual como visitado e removê-lo do conjunto não visitado S_u. Assim, um nó já visitado nunca será checado novamente.

5. Pegar o próximo nó, que será o atual, e que está no conjunto de não visitados S_u que possui menor distância ao nó inicial e repetir os passos acima até checar todos os vizinhos, após isso marcar o nó atual como visitado.

6. Se o nó destino foi marcado como visitado (para problemas onde se busca a distância de um nó inicial *i* a um nó final *j*) ou se a menor distância dos nós no conjunto S_u é infinito (ocorre quando não há conexão entre o nó atual e um nó não visitado) então parar. O algoritmo terminou.

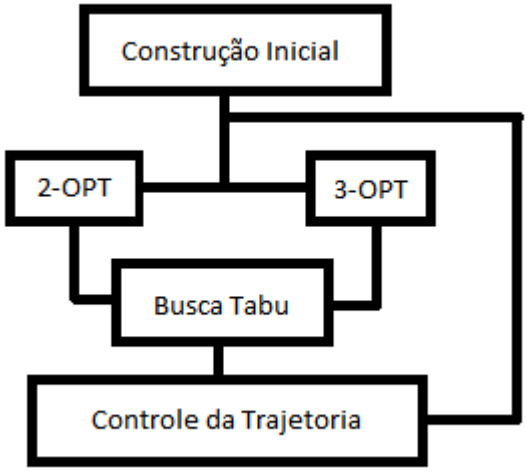
7. Caso contrário, atribuir o nó não visitado com a menor distância como o novo nó atual e voltar ao passo 3.

De acordo com o Pseudocódigo abaixo, dado um grafo G = (V, A) e um nó de origem A, a distância do nó A a todos os nós v do conjunto V(G) é dado por dist[*v*]. Dessa forma é possível montar a matriz *MD[i, j]* que guarda a distância não linear do nó *i* ao no *j*. Além disso utilizando a estrutura prev[ ] (que guarda o nó anterior no caminho ótimo) é possível construir a matriz de caminho *MR[i, j]* que guarda o caminho de menor custo do nó *i* ao nó *j*.

```
1  função Dijkstra( Grafo G, origem A ):
2
3    crie um conjunto de vértices Q (Q é um tipo lista)
4
5    Para cada vértice v no Grafo G :
6      dist[v] ← INFINITO (lista: distâncias desconhecidas da origem até v)
7      prev[v] ← UNDEF    (lista: guarda o nó anterior no caminho ótimo)
8      adicione v ao conjunto Q  (nós não visitados)
9
10   dist[A] ← 0
11
12   Enquanto Q não está vazia:
13     u ← vértice em Q com min(dist[u])  (Nó com a menor distância
14                                         será escolhido primeiro)
15     remova u de Q
16
17     Para cada vizinho v de u:          (Onde v está em Q)
18       alt ← dist[u] + distância(u, v)
19       if alt < dist[v]:                (Menor caminho encontrado)
20         dist[v] ← alt
21         prev[v] ← u
22
23   retorne dist[], prev[]
```

Segunda Etapa do Roteamento – O algoritmo de roteamento desenvolvido é composto por um agregado de heurísticas de busca local (2-OPT e 3-OPT), meta-heurística baseada em esquemas de inteligência artificial, como é o caso da meta-heurística de Busca Tabu, e meta-heurística de controle da trajetória da solução dentro do espaço de busca. Uma iteração do algoritmo consiste em executar um movimento de melhoria local 2-OPT e outro 3-OPT, tornar os nós envolvidos como tabu se houve melhoria no tamanho da rota, analisar o padrão da trajetória no espaço de busca e tomar decisão sobre qual regra usar no processo de escolha de solução. A estrutura geral do algoritmo de roteamento é dada pelo esquema apresentado na Figura 6.

Figura 6. Estrutura geral de um algoritmo de roteamento



Fonte: Autores

Heurística Gulosa para Construção Inicial

A construção de uma rota inicial para o conjunto C foi feita utilizando uma heurística gulosa. Essa heurística consiste em: dado o primeiro nó $i = 0$ (o nó zero foi considerado como o centro de distribuição de cartas, é onde a rota se origina e termina), o próximo nó será o nó j que está a menor distância do nó i dentre todos os nós do conjunto V(G), utilizando a matriz *MD* construída anteriormente. O procedimento se repete sucessivamente até se esgotar todos os nós do conjunto V(G). A Figura 7 ilustra a heurística de construção inicial.

Figura 7. Construção inicial (a) e (b)

a)

b)

Fonte: API Google Maps

As linhas azuis são as arestas do grafo G = (V, A) e as linhas em preto são as rotas de um no *i* a um nó *j*. Vamos construir a rota inicial que interliga os nós do conjunto de clientes C = {0, 5, 9 e 11}, ver Figura 7 (a). O resultado da construção inicial é visitar os nós na ordem 0 – 9 – 11 – 5 – 0, assim o conjunto solução é n-upla ordenada S = (0, 9, 11, 5, 0). Para visitar os nós nessa ordem é necessário fazer o percurso 0 – 8 – 9 – 10 – 11 – 7 – 6 – 5 – 2 – 1 – 0 conforme mostrado na figura 7 (b). Nesse exemplo, podemos observar que a heurística de construção inicial busca entre os nós do conjunto C, o nó mais próximo ao depósito, representado pelo nó 0. Assim obtivemos:

- O nó mais próximo ao nó 0 é o nó 9 através o caminho 0 – 8 – 9.
- O próximo nó será o mais próximo ao nó 9 em C que não foi visitado, nesse caso é o nó 11 através do caminho 9 – 10 – 11.
- Analogamente, o próximo nó da rota inicial será o mais próximo ao nó 11 em C que não foi visitado, nesse caso é o nó 5 através do caminho 11 – 7 – 6 – 5.
- Finalmente após visitados todos os nós em C, o ultimo nó é o nó zero, que representa a volta ao depósito através do caminho 5 – 2 – 1 – 0.

Essa heurística é um método robusto de construção inicial que em geral não gera o mínimo global do roteamento dos clientes do conjunto C, porém dá uma boa rota inicial para se utilizar as heurísticas de melhoramento durante o processo de otimização.

O Pseudocódigo a seguir, faz referência a construção inicial utilizando uma heurística de construção gulosa. O roteamento inicial é guardado em *S* e o tamanho da rota é guardado em *tam*.

```
 1  função ConstrInicial( Nós a rotear C ):
 2
 3    crie uma lista Q (Lista de vértices não visitados )
 4    A ← 0            ( Primeiro nó: depósito é atribuído como nó zero )
 5    tam ← 0          ( Tamanho da rota: distancia total da rota )
 6
 7    Para cada nó v em C :
 8      adicione v ao conjunto Q
 9      S[] ← UNDEF               ( Rota inicial)
10
11    S[0] ← 0
12    i ← 0
```

```
13
14    Enquanto Q não está vazia:
15      u ← vértice em Q com min(dist[A])   ( Nó não visitado com a menor
16                                            distância até o nó A )
17      remova u de Q
18      tam ← tam + dist[u, A]
19      incremente i
20      S[i] ← u
21
22    incremente i
23    S[i] ← 0                    ( Adicionando o último nó como depósito )
24    tam ← tam + dist[u, 0]
25    retorne S[], tam
```

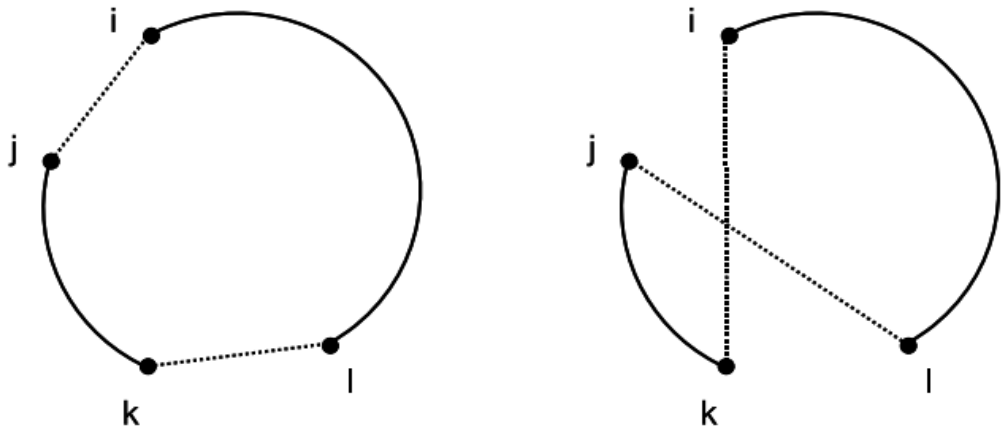
Heurísticas de Melhoria Local 2-OPT e 3-OPT

A primeira fase da segunda etapa é a construção da rota inicial S, para posterior melhoramento da mesma. O melhoramento da rota é feito através da heurística de melhoria 2-OPT e 3-OPT.

A heurística 2-OPT é um algoritmo de busca local simples proposto por Croes, 1958 para resolver o problema do caixeiro viajante. A ideia principal desse algoritmo é pegar uma rota com arestas que se cruzam (duas arestas se cruzam) e reorganizá-la de forma que não se cruzem.

A Figura 8 mostra as possíveis combinações de duas arestas para o caso 2-OPT, onde as arestas *ji* e kl são desconectadas e reconectadas como novas arestas *j*l e *i*k ou vice-versa. Se o movimento das arestas resultar em menor custo ele é realizado. Em geral, as rotas cruzadas têm tamanho maior do que as rotas que não se cruzam.

Figura 8. Esquema da heurística 2-OPT



Fonte: Autores

De acordo com o pseudocódigo abaixo, dados aleatoriamente dois nós ou vértices não consecutivos *i* e *j* da lista de clientes C, o algoritmo 2-OPT é dado por:

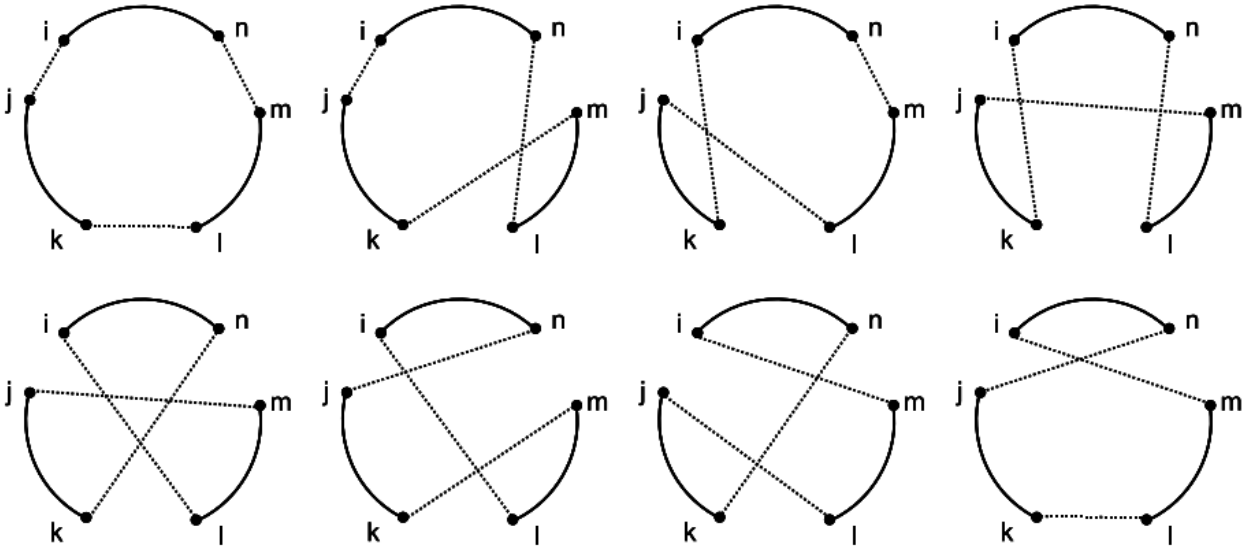
```
1  função 2-OPT( Rota inicial S, Nó i, Nó j ):
2
3     crie uma lista S2OPT (Guarda a nova rota gerada pelo 2-OPT)
4
5     ordene o conjunto {i, j} de forma que i < j
6
7     tome os nós S[0] até S[i] e adicione-os em ordem a S2OPT
8     tome os nós S[i+1] até S[j] e adicione-os em ordem reversa a S2OPT
9     tome os nós S[j+1] até o fim de S e adicione-os em ordem a S2OPT
10
11    Se o tam(S2OPT) < tam(S) retorne S2OPT
12    Senão retorne S
```

O algoritmo 2-OPT é aplicado escolhendo-se aleatoriamente, para cada iteração, duas arestas *i* e *j* que não sejam tabu. Caso o movimento resulte em melhoria, as arestas *i* e *j* (particularmente na lista S, cada nó está conectado a outro nó por apenas uma aresta, desta forma assumiu-se que a aresta que começa no nó i e termina no nó *i*+1 recebe o nome de aresta *i*) são declaradas tabu e não poderão mais ser alteradas durante um dado intervalo de iterações. Nesse ponto a meta-heurística tabu é aplicada, essa meta-heurística será descrita mais adiante nesse trabalho.

A heurística 3-OPT segue a mesma ideia da 2-OPT, contudo a análise consiste em alterar 3 arestas não consecutivas ao invés de duas como é feito no 2-OPT. A Figura 9 mostra todas as possíveis combinações de 3 arestas para o caso 3-OPT. É importante notar que os casos em que não há alteração de uma das arestas recai sobre o caso 2-OPT tratado anteriormente e, portanto, esses casos podem ser desconsiderados da análise.

O método desenvolvido para resolver o roteamento dos clientes do conjunto C em nosso problema utilizou a aplicação das heurísticas 2-OPT e 3-OPT consecutivamente como heurísticas de melhoria local. Dessa forma realiza-se um movimento 2-OPT sobre duas arestas aleatórias *i* e *j*. Em seguida aplica-se o movimento 3-OPT em três arestas aleatórias *i, j* e *k* possivelmente diferentes das arestas *i* e *j* utilizadas durante o processo de melhoria 2-OPT.

Figura 9. Esquema da heurística 3-OPT



Fonte: Autores

Segundo o pseudocódigo a seguir, dados aleatoriamente três nós ou vértices não consecutivos *i*, *j* e *k* da lista de clientes C, o algoritmo 3-OPT para os 4 casos em que as 3 arestas são alteradas é dado por:

```
1  função 3-OPT(Rota inicial S, Nó i, Nó j, Nó k):
2
3    crie listas S3OPT1, S3OPT2, S3OPT3, S3OPT4
4
5    ordene o conjunto {i, j, k} de forma que i < j < k
6
7    S3OPT1 ← S   ( S3OPTi guarda o roteamento de cada caso com i=1,2,3,4 )
```

```
 8    S3OPT2 ← S
 9    S3OPT3 ← S
10    S3OPT4 ← S
11
12    Caso 1:
13
14    tome os nós S[0] até S[i] e adicione-os em ordem a S3OPT1
15    tome os nós S[j+1] até S[k] e adicione-os em ordem reversa a S3OPT1
16    tome os nós S[i+1] até S[j] e adicione-os em ordem a S3OPT1
17    tome os nós S[k+1] até o fim de S e adicione-os em ordem a S3OPT1
18
19    Caso 2:
20
21    tome os nós S[0] até S[i] e adicione-os em ordem a S3OPT2
22    tome os nós S[j+1] até S[k] e adicione-os em ordem a S3OPT2
23    tome os nós S[i+1] até S[j] e adicione-os em ordem a S3OPT2
24    tome os nós S[k+1] até o fim de S e adicione-os em ordem a S3OPT2
25
26    Caso 3:
27
28    tome os nós S[0] até S[i] e adicione-os em ordem a S3OPT3
29    tome os nós S[j+1] até S[k] e adicione-os em ordem a S3OPT3
30    tome os nós S[i+1] até S[j] e adicione-os em ordem reversa a S3OPT3
31    tome os nós S[k+1] até o fim de S e adicione-os em ordem a S3OPT3
32
33    Caso 4:
34
35    tome os nós S[0] até S[i] e adicione-os em ordem a S3OPT4
36    tome os nós S[i+1] até S[j] e adicione-os em ordem reversa a S3OPT4
37    tome os nós S[j+1] até S[k] e adicione-os em ordem reversa a S3OPT4
38    tome os nós S[k+1] até o fim de S e adicione-os em ordem a S3OPT4
39
40    Se i ← min{ tam(S3OPTi) } retorne S3OPTi se tam(S3OPTi) < tam(S)
41
```

Analogamente ao 2-OPT, o algoritmo 3-OPT é aplicado escolhendo-se aleatoriamente, para cada iteração, três arestas *i*, *j* e *k* que não sejam tabu. Caso o movimento resulte em melhoria, as arestas *i*, *j* e *k* são declaradas tabu e não poderão ser utilizadas durante um dado intervalo de iterações em outro movimento de melhoria local.

Meta-heurística Busca Tabu

Durante o procedimento de melhoria, utilizou-se uma meta-heurística denominada Busca Tabu. A Busca Tabu, foi criada por Fred W. Glover em 1986 e é uma meta-heurística que emprega métodos de busca local usados para otimização matemática. As buscas locais (de vizinhança) tomam uma solução potencial para um problema e checam seus vizinhos imediatos (ou seja, soluções que são semelhantes, exceto por pequenos detalhes) na esperança de encontrar uma solução melhorada. Os métodos de busca local tendem a ficar presos em regiões sub-ótimas ou em platôs onde muitas soluções são igualmente boas.

A Busca Tabu melhora o desempenho da busca local, relaxando sua regra básica. Inicialmente, a cada passo os movimentos de piora podem ser aceitos se nenhum movimento de melhoria estiver disponível (como quando a busca é travada em um mínimo local estrito). Além disso, as proibições (daqui em diante o termo tabu será usado) são introduzidas para desencorajar a busca de voltar às soluções anteriormente visitadas.

A implementação da busca tabu usa estruturas de memória de longo ou curto prazo que descrevem as soluções visitadas ou conjuntos de regras fornecidos pelo usuário. Por exemplo, se uma solução potencial foi anteriormente visitada dentro de um determinado período de curto prazo ou se violou uma regra, ela é marcada como "tabu" (proibida) para que o algoritmo não considere essa possibilidade repetidamente.

Em nosso algoritmo, quando uma solução boa é encontrada durante o processo de movimento de troca de arestas pelos métodos de melhoria local 2-OPT e 3-OPT, as arestas envolvidas no processo são marcadas como tabu para que estas não sejam utilizadas novamente em outro movimento de melhoria local. Para tal, as arestas são guardadas em uma lista cíclica denominada *TabuList* de tamanho 3t, onde t é o período tabu. Dessa forma, a cada iteração do algoritmo 3 arestas são adicionadas à *TabuList* e os 3 valores mais antigos da mesma são excluídos, assim, um dado nó *i* permanece na lista durante um período de t iterações do algoritmo.

No caso 3-OPT, as arestas *i*, *j* e *k* são adicionadas caso haja melhoria. No caso 2-OPT as arestas *i, j*, -1 são adicionadas caso haja melhoria. No caso 2-OPT, o número -1 é utilizado para ocupar o espaço da terceira aresta ausente. Durante o processo de melhoria, verifica-se se as arestas envolvidas estão na *TabuList*, caso uma das arestas esteja o movimento de melhoria não é realizado. Assim, somente é permitido movimentos em que todas as arestas envolvidas não sejam tabu. Abaixo está um exemplo do pseudocódigo utilizado durante o processo de otimização.

O pseudocódigo abaixo, faz referência a Busca Tabu.

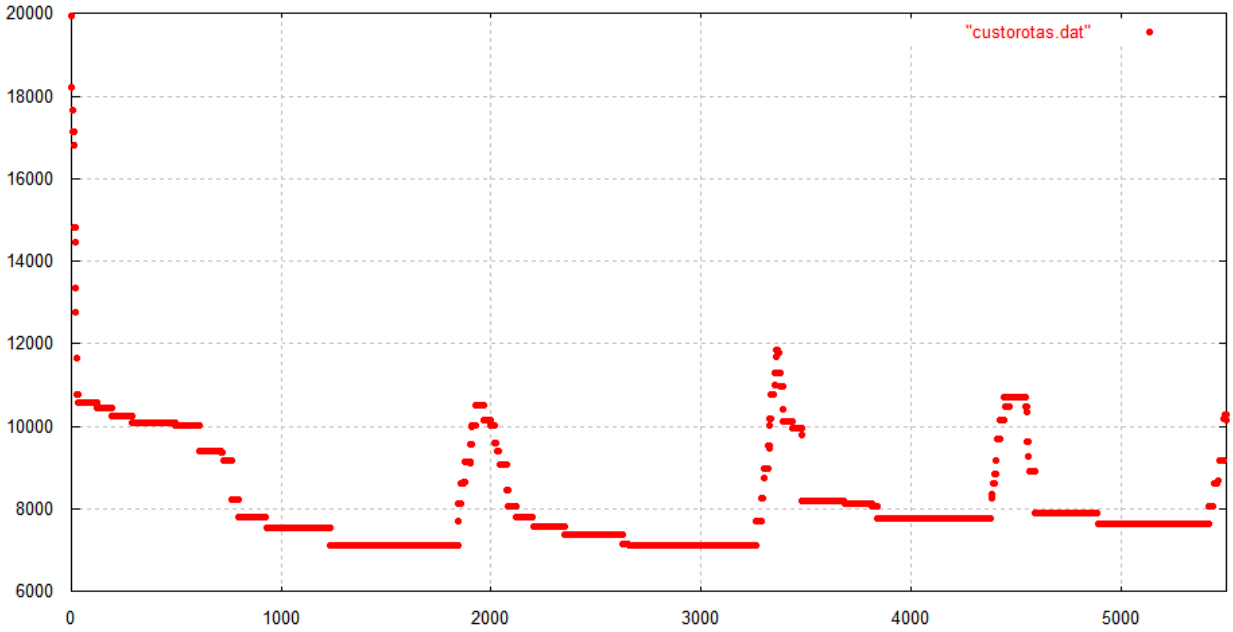
```
1  função BuscaTabu( ):
2
3    Caso 2-OPT(S, i, j) resulte em melhoria
4      TabuList ← {i, j, -1}
5
6    Caso 3-OPT(S, i, j, k) resulte em melhoria
7      TabuList ← {i, j, k}
8
9
```

Meta-heurística de Controle de Busca

Nosso algoritmo também permite a realização de um movimento que causa o aumento da distância percorrida no tamanho de uma rota, piorando a mesma, seguindo uma regra definida pela heurística de controle da busca. O objetivo desse controle é evitar que o algoritmo se prenda a mínimos locais e passe a vasculhar outras regiões do espaço de busca, ver Dorício, (2011).

O controle da busca utiliza um mecanismo de memória de longo prazo simplificado, que guarda o tamanho de cada rota S obtido a cada iteração do algoritmo de busca formando uma lista de custos denominada *CustoRotas* como mostrado pelo gráfico no exemplo da Figura 10.

Figura 10. Memória das soluções encontradas



Fonte: Autores

Na Figura 10 podemos observar que as soluções se apresentam como uma trajetória de descida até aproximadamente a iteração 1250. Entre 1250 e 1850 a trajetória se apresenta estagnada (ou presa) em um mínimo local. E entre 1850 e 1950 a trajetória é crescente, fugindo do mínimo local. Esse padrão de busca se repete ao longo do algoritmo e é controlado pela heurística de controle.

Durante a execução dos algoritmos de busca local 2-OPT (mostrado pelo pseudocódigo) utiliza-se a seguinte regra para se aceitar uma nova solução:

```
Se o tam(S2OPT) < tam(S) retorne S2OPT
```

Essa linha é alterada para implementar o controle de busca usando um parâmetro *E*:

```
Se o tam(S2OPT) < (tam(S) + E) retorne S2OPT
```

Onde *E* é um parâmetro de controle escolhido pelo usuário. Se *E* for igual a zero o movimento de melhoria é executado somente se a nova rota for melhor que a rota antiga, ou seja, apenas movimentos que resulte em melhoria são aplicados. Contudo, se *E* for maior que zero permite-se que a nova rota seja aceita mesmo que esta seja pior que a rota antiga, porém dentro de uma faixa de controle (-∞, E) em torno do *tam(S)*. A mesma metodologia se aplica

ao algoritmo de busca local 3-OPT. Trabalhando em conjunto com a meta-heurística de busca tabu, esse movimento de piora é mantido durante o período tabu de t iterações.

O controle de busca analisa o padrão da trajetória de busca guardado em *CustoRotas* e determina qual regra irá utilizar no critério de aceitação de um movimento 2-OPT ou 3-OPT. Para tal, analisa-se o comportamento dos *N* últimos valores da lista *CustoRotas.* Há 3 tipos de comportamentos que são considerados:

- **Trajetória decrescente:** Nesse caso a busca está descendo para um ótimo local, portanto mantém-se $E = 0$.
- **Trajetória horizontal:** Nesse caso a busca está presa em um ótimo local, altera-se o valor de *E* para algum número positivo dado pelo usuário com o objetivo de fazer a busca fugir do ótimo local.
- **Trajetória crescente:** Nesse caso a busca está fugindo do ótimo local para outra região no espaço de busca, altera-se o valor de *E* para zero para que um novo ótimo local seja localizado.

Esse procedimento em conjunto com a Busca Tabu se mostrou uma meta-heurística eficiente para localização de mínimos, Dorício (2011). O padrão da trajetória pode ser calculado, por exemplo, utilizando-se o método dos mínimos quadrados para determinar o coeficiente angular da reta que melhor se aproxima dos valores da lista *CustoRotas*.

Outro critério de controle utilizado a cada 10% de execução do algoritmo consiste em embaralhar aleatoriamente os elementos de *S.* Procedendo dessa forma o algoritmo permite que a busca avance para uma região aleatória do espaço de busca, possivelmente não explorada. Esse procedimento permite paralelização futura do algoritmo baseado em esquemas heurísticos como os desenvolvidos por Brandão, (2015); que têm se mostrado bastante eficientes na determinação de mínimos para problemas de otimização de grande porte.

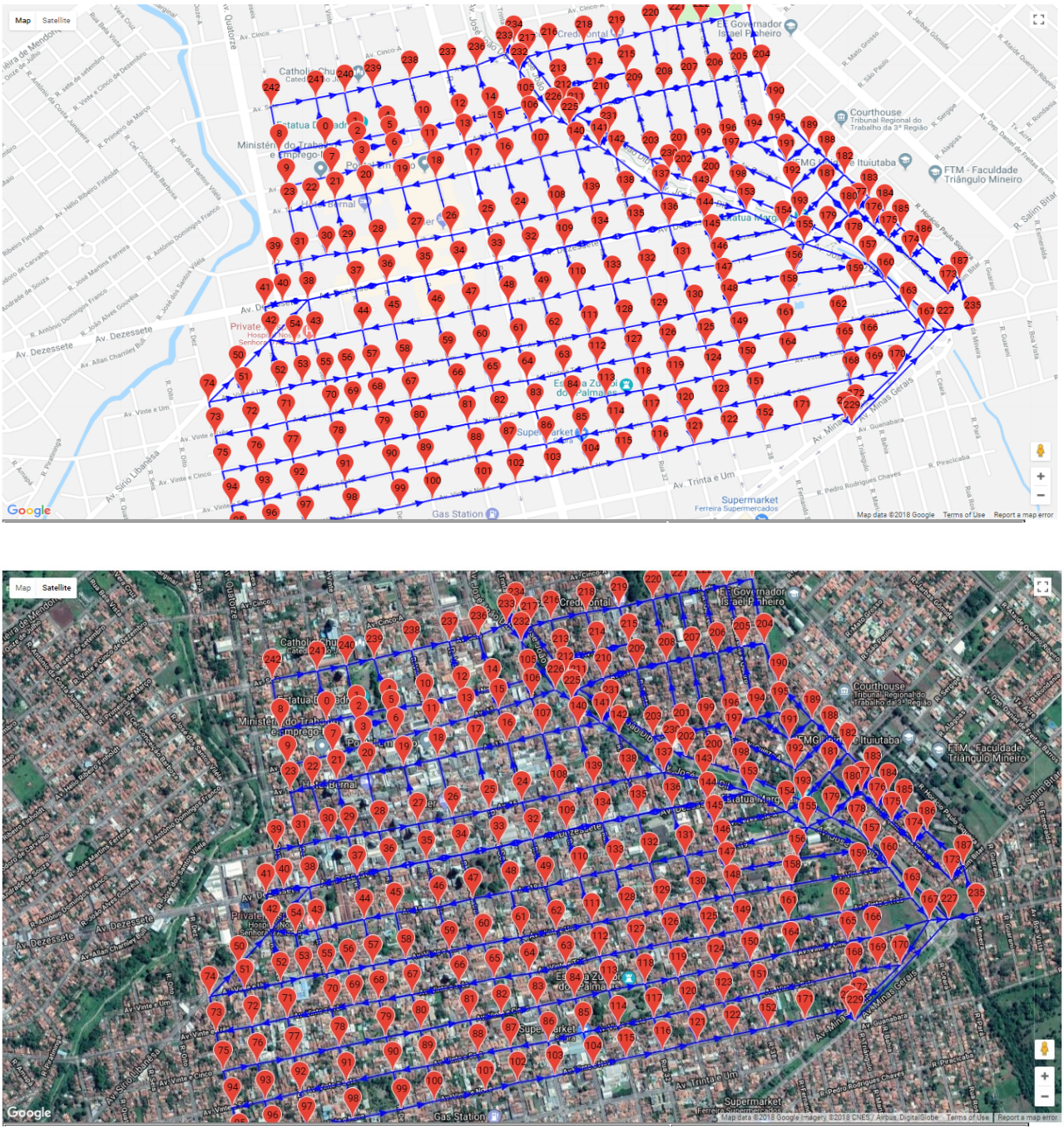
Critério de Parada:

Devido ao padrão de busca ser dinâmico, utilizou-se um critério de parada baseado no número de iterações do algoritmo. A cada iteração a solução ótima é atualizada da seguinte forma: se $tam(S) < tam(Sotimo)$ atualiza-se *Sotimo* com *S*. Dessa forma um critério de parada baseado num intervalo de iterações onde não haja alteração do valor de *Sotimo* também poderia ser utilizado como critério de parada.

4 RESULTADOS

A meta-heurística utilizada para o roteamento proposto foi testada sobre uma malha construída pela interface gráfica, onde foram capturados os dados de 243 vértices e a lista de adjacências para cada um deles. A malha pode ser visualizada na Figura 11 e os valores numéricos estão tabelados no Anexo A.

Figura 11. Malha utilizada no roteamento. Vista em modo mapa e em modo satélite respectivamente.



Fonte: API Google Maps

O algoritmo de roteamento requer a utilização de alguns parâmetros como critério de parada, período tabu, etc. Para todos os casos testados utilizou-se os parâmetros a seguir:

- Número de iterações do critério de parada: 400.000 iterações.
- Período Tabu: 5 iterações (utilizou-se memória de curto prazo).
- Parâmetro de controle *E* para aceitar soluções piores: 300 metros.
- Parâmetro *N* da lista *CustoRotas*: 10.000 últimos valores.
- Velocidade Média: 25 Km/h (para estimar o tempo de percurso).
- Embaralhamento da rota: 3 vezes durante o roteamento.

O algoritmo foi testado utilizando 20 instâncias geradas aleatórias, constituídas de 4 conjuntos de 10, 20, 40 e 80 clientes que foram extraídos da malha do roteamento mostrada na Figura 11. As tabelas abaixo mostram os vértices utilizados em cada instância que compõem a lista *C* de clientes que receberão suas correspondências. A lista *C* é a lista que contém os clientes que serão roteados pelo algoritmo desenvolvido nesse trabalho.

Inst. com 10 clientes	Lista *C* de clientes
Instância 01	103 45 227 208 39 186 205 10 46 14
Instância 02	133 48 34 84 29 152 14 30 167 176
Instância 03	204 228 81 121 69 50 58 210 224 1
Instância 04	70 159 14 59 69 154 186 108 198 43
Instância 05	19 162 206 94 33 129 213 75 187 37

Inst. com 20 clientes	Lista *C* de clientes
Instância 06	43 186 20 22 16 26 202 154 15 95 207 184 149 117 166 206 7 11 67 210
Instância 07	187 96 196 92 186 47 15 203 109 236 51 214 188 184 21 4 206 228 19 142
Instância 08	229 77 2 145 33 51 149 237 37 84 170 23 32 26 62 19 208 27 14 41
Instância 09	88 121 105 134 9 184 118 209 104 124 236 27 161 227 76 26 12 146 58 151
Instância 10	122 125 185 227 194 15 41 111 123 187 177 38 181 137 53 202 155 63 44 232

Inst. com 40 clientes	Lista *C* de clientes
Instância 11	208 74 87 13 226 75 189 19 21 185 4 180 134 72 130 2 181 105 64 53 174 209 166 82 206 217 138 221 227 178 147 203 61 223 16 241 18 190 216 114
Instância 12	238 72 203 43 131 228 221 5 120 156 19 171 199 152 184 143 155 87 206 48 211 136 57 202 59 180 116 195 240 38 79 218 60 114 29 108 110 130 51 50

Instância 13	79 173 132 188 62 180 164 163 6 161 226 201 146 157 160 234 31 149 48 236 141 218 18 21 57 92 38 177 56 186 91 102 151 23 106 137 81 42 214 156
Instância 14	103 44 79 15 136 179 69 131 50 28 150 29 221 208 183 153 226 216 224 155 45 13 121 218 67 110 82 3 191 58 124 112 217 97 231 40 56 120 48 115
Instância 15	93 51 58 165 29 218 139 18 185 124 30 61 214 167 89 169 110 117 75 170 135 106 70 149 8 125 161 126 62 240 223 180 195 137 81 152 60 65 181 144

Inst. com 80 clientes	Lista *C* de clientes
Instância 16	233 76 197 88 5 72 190 165 98 1 109 60 87 50 131 39 42 203 90 227 83 213 169 105 178 138 128 166 102 51 113 126 65 167 198 149 2 236 157 114 136 30 189 162 67 151 182 75 220 123 91 173 176 121 28 32 218 142 144 84 115 135 205 74 156 212 187 175 47 153 199 3 196 229 133 59 139 36 207 147
Instância 17	186 92 174 81 156 124 41 229 88 106 3 37 217 97 93 77 59 147 159 231 151 161 15 9 22 205 191 218 111 98 75 27 112 188 68 47 101 1 108 87 136 133 239 82 189 78 117 222 72 67 2 32 193 129 107 225 153 7 192 190 99 126 53 40 86 197 110 141 149 113 96 220 181 172 8 95 240 227 223 23
Instância 18	159 208 178 160 165 93 22 180 66 61 9 202 128 234 47 38 7 143 124 5 205 74 157 77 57 21 27 167 67 50 214 73 146 211 150 218 92 191 16 28 147 222 230 112 24 48 217 60 238 135 133 179 70 226 183 158 212 29 98 3 12 196 94 23 13 184 31 186 129 33 236 142 100 84 181 26 63 216 65 109
Instância 19	219 185 64 175 113 110 149 229 98 116 14 6 56 36 160 43 85 125 230 84 241 57 105 32 127 68 135 134 61 17 103 9 145 196 29 142 217 158 73 40 147 155 207 201 212 238 148 139 5 193 159 86 59 189 166 153 206 37 171 114 19 87 90 182 195 33 204 91 225 51 38 213 34 176 53 102 128 78 200 130
Instância 20	134 205 159 119 158 84 186 106 193 128 145 6 42 12 49 148 132 86 74 66 137 142 194 206 129 54 68 196 58 41 238 152 178 62 28 226 222 228 25 91 232 190 163 11 224 122 7 103 166 237 229 141 52 118 161 45 13 217 85 135 31 156 117 19 136 21 183 236 64 200 48 29 160 75 155 100 2 209 188 173

O algoritmo de roteamento desenvolvido nesse trabalho foi implementado em linguagem de programação C^{++} e executado para cada uma das 20 instâncias apresentadas acima. As Figuras 12 a 40 mostram os resultados obtidos para o roteamento dos clientes das instâncias 01 a 20. As rotas para cada caso são mostradas pelas tabelas abaixo, onde os clientes são postos na ordem que devem ser visitados pelo carteiro, partindo e chegando ao centro de distribuição. Os resultados obtidos são:

Instâncias com 10 clientes:

Rota da instância 01:	0 - 39 - 46 - 45 - 103 - 227 - 186 - 205 - 208 - 14 - 10 - 0
Distância percorrida:	7538.34 metros.
Tempo de percurso:	18:5 [min:seg], a 25 km/h.

Rota da instância 02:	0 - 30 - 29 – 34 - 48 - 84 - 152 - 167 - 176 - 133 - 14 - 0
Distância percorrida:	6742 metros.
Tempo de percurso:	16:10 [min:seg], a 25 km/h.

Rota da instância 03:	0 - 69 - 50 - 58 - 81 - 121 - 228 - 204 - 224 - 210 - 1 - 0
Distância percorrida:	7995.59 metros.
Tempo de percurso:	19:11 [min:seg], a 25 km/h.

Rota da instância 04:	0 - 43 - 69 - 70 - 59 - 108 - 154 - 159 - 186 - 198 - 14 - 0
Distância percorrida:	7298.71 metros.
Tempo de percurso:	17:31 [min:seg], a 25 km/h.

Rota da instância 05:	0 - 94 - 75 - 37 - 33 - 129 - 162 - 187 - 206 - 213 - 19 – 0
Distância percorrida:	8681.01 metros.
Tempo de percurso:	20:50 [min:seg], a 25 km/h.

Instâncias com 20 clientes:

Rota da instância 06:	0 - 43 - 67 - 95 - 117 - 149 - 166 - 186 - 184 - 154 - 202 - 206 - 207 - 210 - 16 - 15 - 11 - 26 - 20 - 22 - 7 – 0
Distância percorrida:	10522.4 metros.
Tempo de percurso:	25:15 [min:seg], a 25 km/h.

Rota da instância 07:	0 - 236 - 15 - 214 - 206 - 228 - 187 - 186 - 184 - 188 - 196 - 203 - 142 - 109 - 47 - 92 - 96 - 51 - 19 - 4 - 21 – 0
Distância percorrida:	10487.5 metros.
Tempo de percurso:	25:10 [min:seg], a 25 km/h.

Rota da instância 08:	0 - 23 - 41 - 77 - 51 - 37 - 27 - 26 - 33 - 32 - 62 - 84 - 149 - 170 - 229 - 145 - 208 - 237 - 14 - 19 - 2 - 0
Distância percorrida:	10203.2 metros.
Tempo de percurso:	24:29 [min:seg], a 25 km/h.

Rota da instância 09:	0 - 9 - 27 - 26 - 58 - 76 - 88 - 104 - 118 - 121 - 124 - 151 - 161 - 227 - 184 - 146 - 134 - 209 - 105 - 236 - 12 – 0
Distância percorrida:	10730.1 metros.

Tempo de percurso:	25:45 [min:seg], a 25 km/h.

Rota da instância 10:	0 - 15 - 232 - 137 - 202 - 194 - 181 - 177 - 185 - 187 - 227 - 155 - 122 - 123 - 125 - 111 - 63 - 44 - 53 - 41 - 38 – 0
Distância percorrida:	9141.85 metros.
Tempo de percurso:	21:56 [min:seg], a 25 km/h.

Instâncias com 40 clientes:

Rota da instância 11:	0 - 241 - 21 - 72 - 75 - 74 - 53 - 61 - 64 - 82 - 87 - 114 - 130 - 147 - 166 - 227 - 178 - 180 - 185 - 174 - 181 - 189 - 190 - 206 - 221 - 223 - 208 - 209 - 203 - 105 - 226 - 217 - 216 - 138 - 134 - 16 - 13 - 18 - 19 - 2 - 4 – 0
Distância percorrida:	13437.3 metros.
Tempo de percurso:	32:14 [min:seg], a 25 km/h.

Rota da instância 12:	0 - 240 - 238 - 19 - 5 - 38 - 43 - 72 - 51 - 50 - 79 - 57 - 59 - 60 - 87 - 114 - 116 - 120 - 152 - 171 - 228 - 184 - 180 - 155 - 156 - 130 - 131 - 211 - 218 - 221 - 206 - 195 - 199 - 202 - 203 - 143 - 136 - 108 - 110 - 48 - 29 - 0
Distância percorrida:	13958 metros.
Tempo de percurso:	33:29 [min:seg], a 25 km/h.

Rota da instância 13:	0 - 23 - 31 - 38 - 42 - 57 - 56 - 91 - 92 - 79 - 81 - 102 - 62 - 48 - 132 - 146 - 149 - 151 - 164 - 161 - 156 - 157 - 160 - 163 - 173 - 186 - 177 - 180 - 188 - 201 - 137 - 214 - 218 - 141 - 106 - 236 - 234 - 226 - 18 - 6 - 21 - 0
Distância percorrida:	12946 metros.
Tempo de percurso:	31:4 [min:seg], a 25 km/h.

Rota da instância 14:	0 - 3 - 13 - 15 - 226 - 217 - 216 - 218 - 221 - 224 - 191 - 183 - 179 - 155 - 131 - 136 - 153 - 208 - 231 - 28 - 45 - 58 - 67 - 56 - 69 - 50 - 40 - 97 - 79 - 82 - 103 - 115 - 121 - 124 - 150 - 120 - 112 - 110 - 48 - 44 - 29 - 0
Distância percorrida:	13691.9 metros.
Tempo de percurso:	32:51 [min:seg], a 25 km/h.
Rota da instância 15:	0 - 8 - 30 - 29 - 58 - 89 - 93 - 75 - 51 - 70 - 81 - 60 - 65 - 61 - 62 - 110 - 139 - 135 - 137 - 144 - 149 - 125 - 126 - 117 - 124 - 152 -

	161 - 165 - 170 - 169 - 167 - 185 - 180 - 181 - 195 - 223 - 218 - 214 - 106 - 18 - 240 – 0
Distância percorrida:	14208.4 metros.
Tempo de percurso:	34:6 [min:seg], a 25 km/h.

Instâncias com 80 clientes:

Rota da instância 16:	0 - 1 - 3 - 5 - 2 - 59 - 60 - 83 - 84 - 113 - 114 - 115 - 121 - 123 - 151 - 162 - 165 - 166 - 169 - 229 - 178 - 156 - 157 - 167 - 227 - 187 - 173 - 175 - 176 - 182 - 189 - 190 - 205 - 207 - 220 - 142 - 138 - 135 - 133 - 139 - 109 - 32 - 105 - 236 - 233 - 218 - 212 - 213 - 203 - 199 - 196 - 197 - 198 - 153 - 144 - 131 - 136 - 147 - 149 - 126 - 128 - 47 - 65 - 102 - 87 - 88 - 67 - 90 - 91 - 98 - 75 - 76 - 72 - 74 - 51 - 50 - 42 - 39 - 30 - 28 - 36 - 0
Distância percorrida:	20346.7 metros.
Tempo de percurso:	48:49 [min:seg], a 25 km/h.

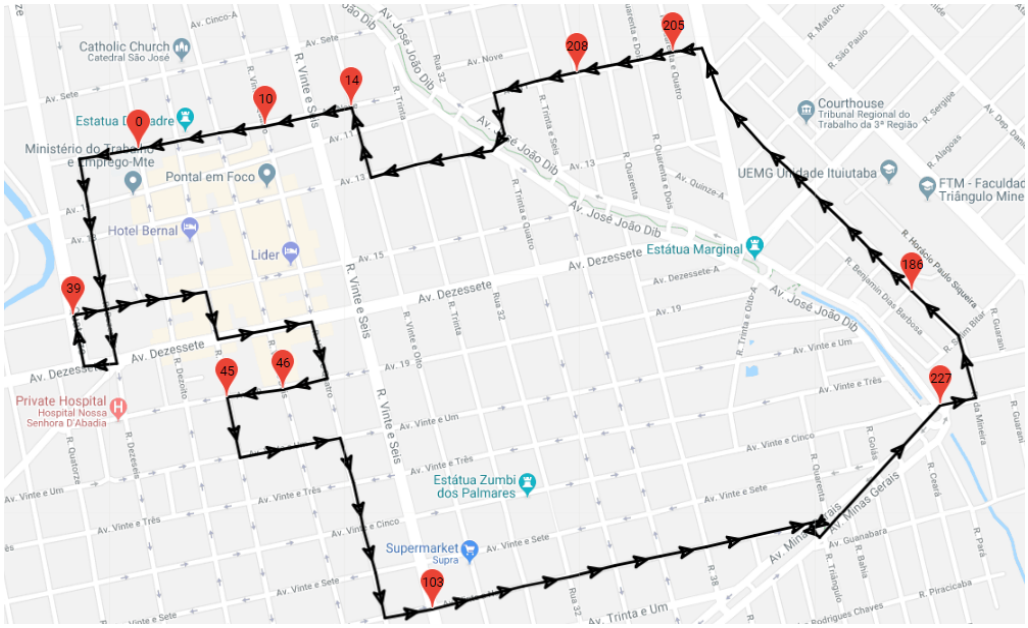
Rota da instância 17:	0 - 240 - 1 - 239 - 15 - 106 - 217 - 218 - 220 - 222 - 223 - 205 - 190 - 189 - 191 - 192 - 193 - 156 - 159 - 161 - 149 - 147 - 129 - 126 - 133 - 110 - 111 - 112 - 113 - 117 - 124 - 151 - 172 - 229 - 227 - 186 - 174 - 181 - 188 - 197 - 153 - 136 - 231 - 141 - 225 - 107 - 108 - 32 - 37 - 22 - 23 - 40 - 41 - 53 - 67 - 68 - 72 - 75 - 95 - 96 - 93 - 97 - 92 - 77 - 78 - 98 - 99 - 101 - 81 - 82 - 86 - 87 - 88 - 59 - 47 - 27 - 2 - 3 - 7 - 8 - 9 - 0
Distância percorrida:	19490.8 metros.
Tempo de percurso:	46:46 [min:seg], a 25 km/h.

Rota da instância 18:	0 - 9 - 23 - 31 - 38 - 29 - 28 - 27 - 26 - 5 - 238 - 236 - 234 - 217 - 216 - 218 - 214 - 211 - 212 - 226 - 142 - 135 - 109 - 33 - 48 - 47 - 60 - 65 - 66 - 57 - 67 - 70 - 73 - 74 - 50 - 77 - 92 - 93 - 94 - 98 - 100 - 61 - 112 - 63 - 84 - 128 - 129 - 124 - 150 - 147 - 146 - 143 - 230 - 202 - 208 - 222 - 205 - 196 - 191 - 157 - 160 - 167 - 178 - 179 - 180 - 186 - 184 - 183 - 181 - 158 - 159 - 165 - 133 - 24 - 16 - 13 - 12 - 3 - 22 - 21 - 7 - 0
Distância percorrida:	20741.7 metros.
Tempo de percurso:	49:46 [min:seg], a 25 km/h.

Rota da instância 19:	0 - 9 - 241 - 238 - 14 - 17 - 19 - 6 - 5 - 37 - 40 - 38 - 43 - 73 - 51 - 53 - 56 - 68 - 78 - 91 - 98 - 90 - 57 - 59 - 61 - 64 - 87 - 102 - 103 - 86 - 85 - 84 - 113 - 114 - 116 - 127 - 128 - 130 - 125 - 148 - 147 - 145 - 155 - 158 - 159 - 160 - 149 - 171 - 166 - 229 - 185 - 175 - 176 - 182 - 189 - 193 - 153 - 200 - 230 - 201 - 196 - 195 - 204 - 206 - 207 - 219 - 213 - 212 - 217 - 105 - 225 - 142 - 135 - 134 - 139 - 110 - 32 - 33 - 34 - 36 - 29 – 0
Distância percorrida:	20610.2 metros.
Tempo de percurso:	49:27 [min:seg], a 25 km/h.

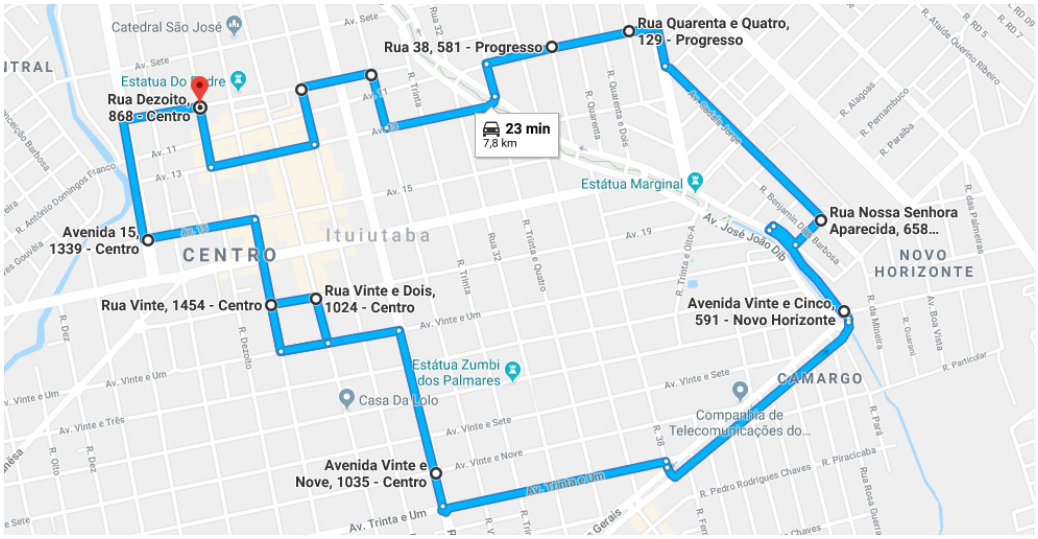
Rota da instância 20:	0 - 238 - 237 - 236 - 217 - 232 - 226 - 106 - 141 - 142 - 135 - 134 - 132 - 129 - 128 - 118 - 117 - 122 - 152 - 166 - 228 - 229 - 178 - 193 - 155 - 156 - 158 - 161 - 159 - 160 - 163 - 173 - 186 - 183 - 188 - 190 - 194 - 200 - 196 - 205 - 206 - 222 - 224 - 209 - 137 - 136 - 145 - 148 - 119 - 85 - 84 - 62 - 66 - 68 - 91 - 75 - 74 - 42 - 52 - 54 - 41 - 31 - 29 - 28 - 45 - 58 - 100 - 103 - 86 - 64 - 49 - 48 - 25 - 13 - 12 - 11 - 19 - 6 - 2 - 21 - 7 – 0
Distância percorrida:	19875.3 metros.
Tempo de percurso:	47:42 [min:seg], a 25 km/h.

Figura 12. Resultado do roteamento da Instância 01.



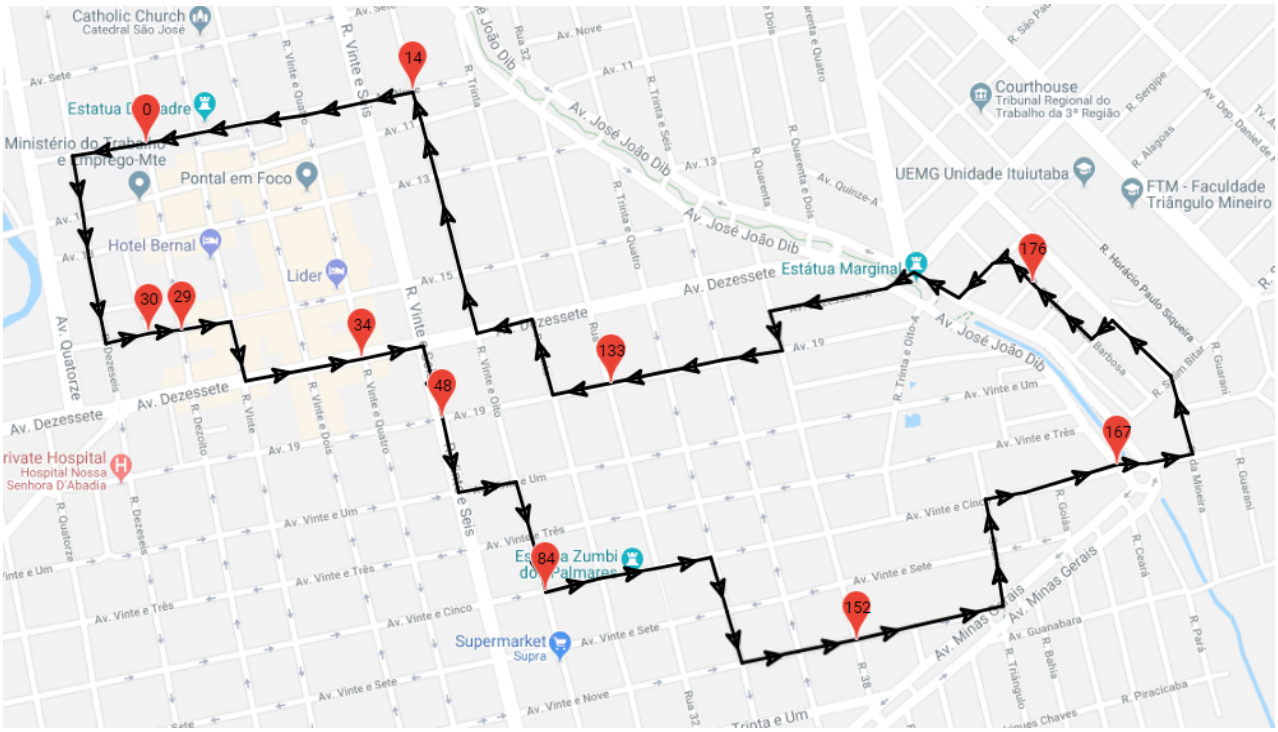
Fonte: API Google Maps

Figura 13. Resultado do roteamento da Instância 01.



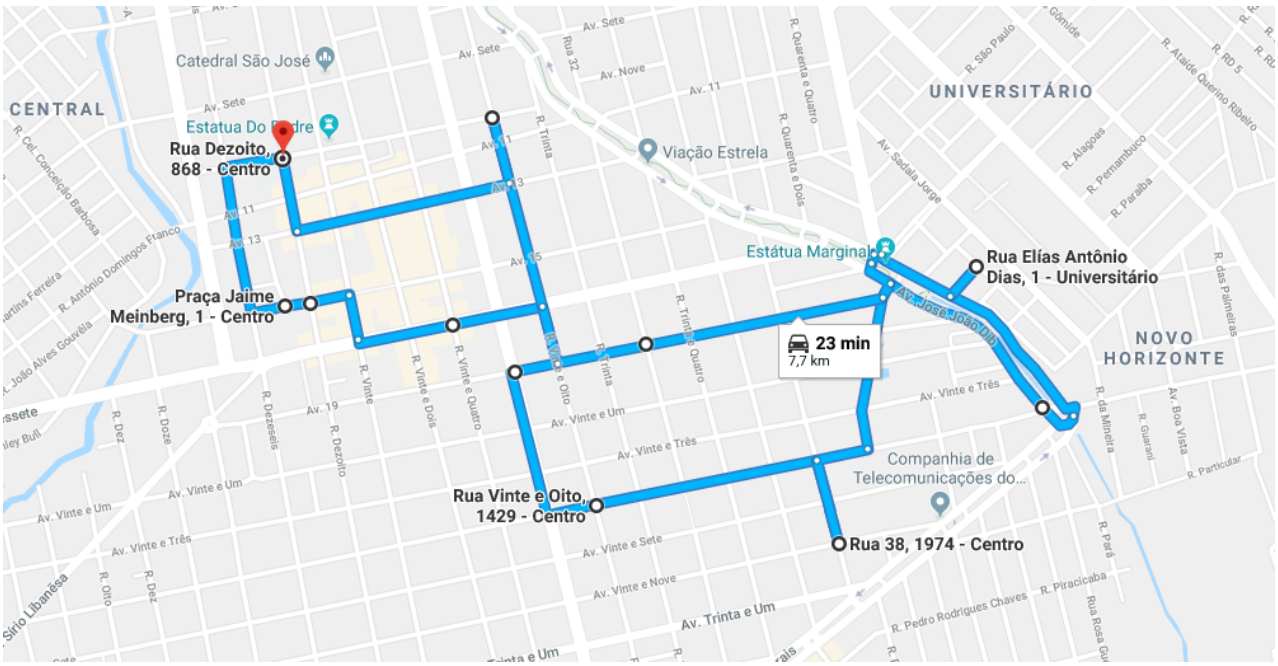
Fonte: Google Maps

Figura 14. Resultado do roteamento da Instância 02.



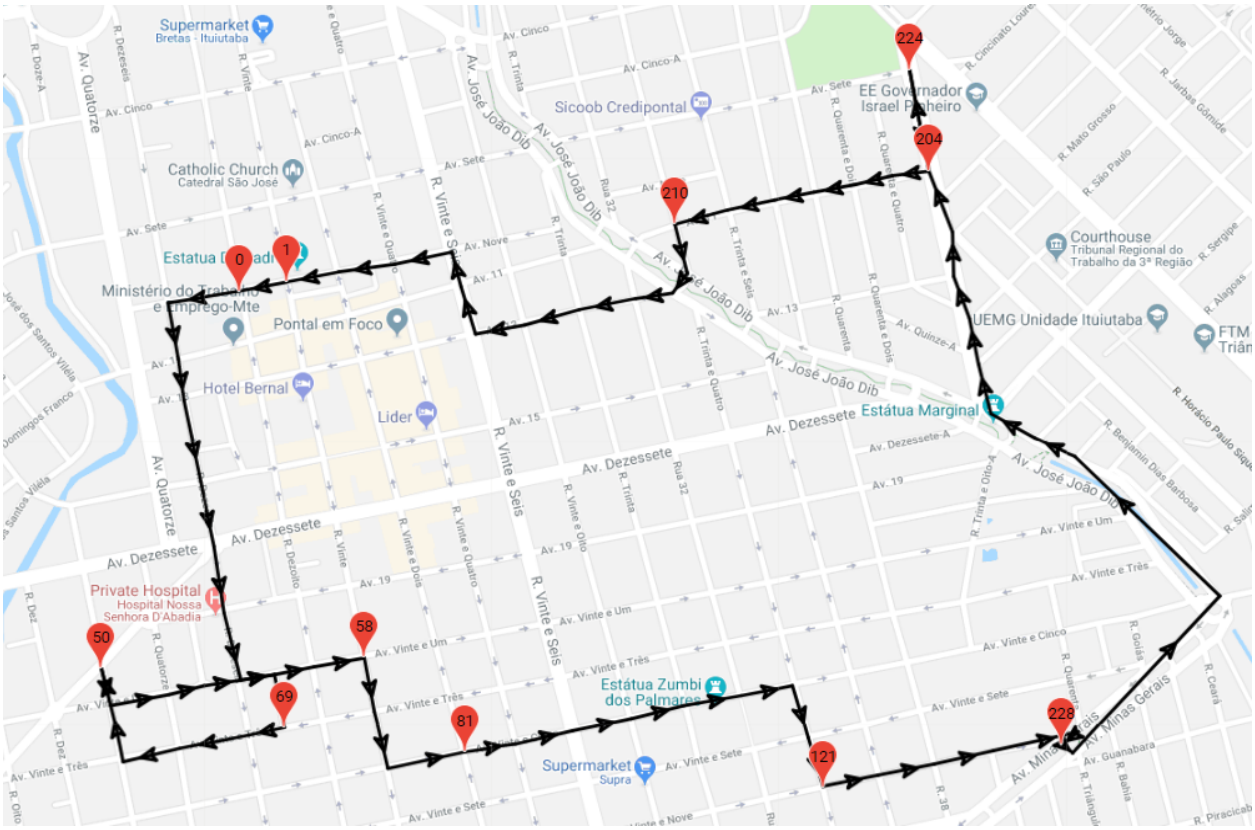
Fonte: API Google Maps

Figura 15. Resultado do roteamento da Instância 02.



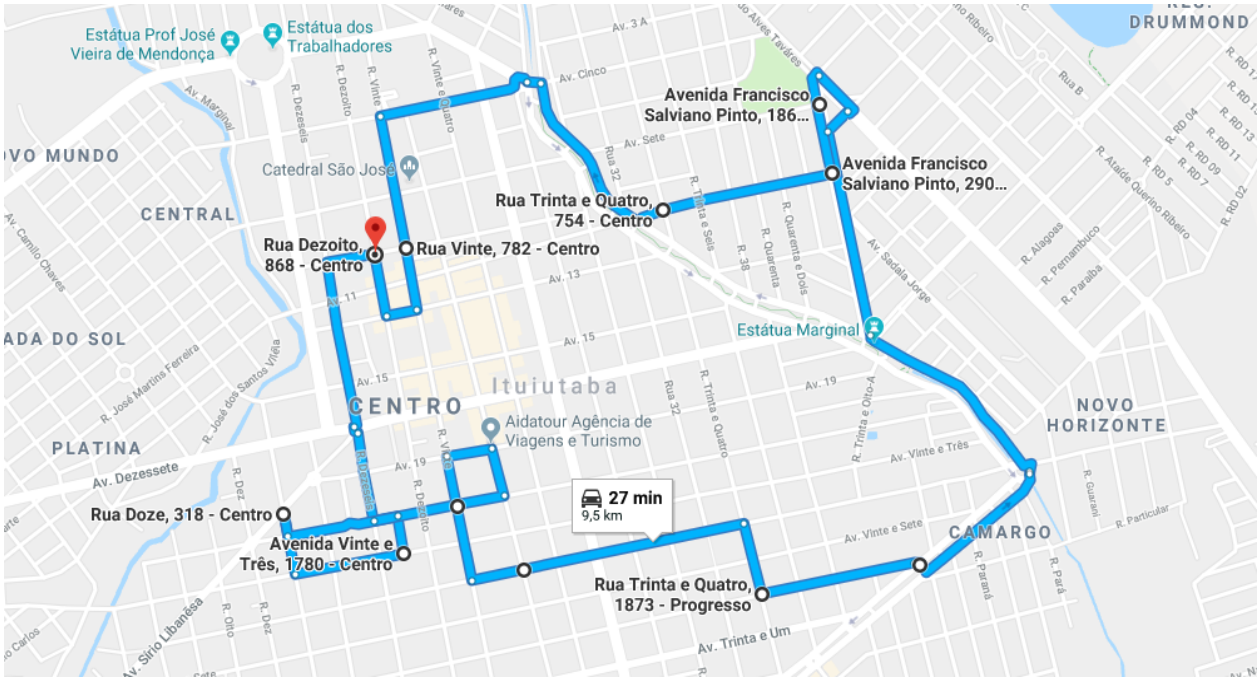
Fonte: Google Maps

Figura 16. Resultado do roteamento da Instância 03.



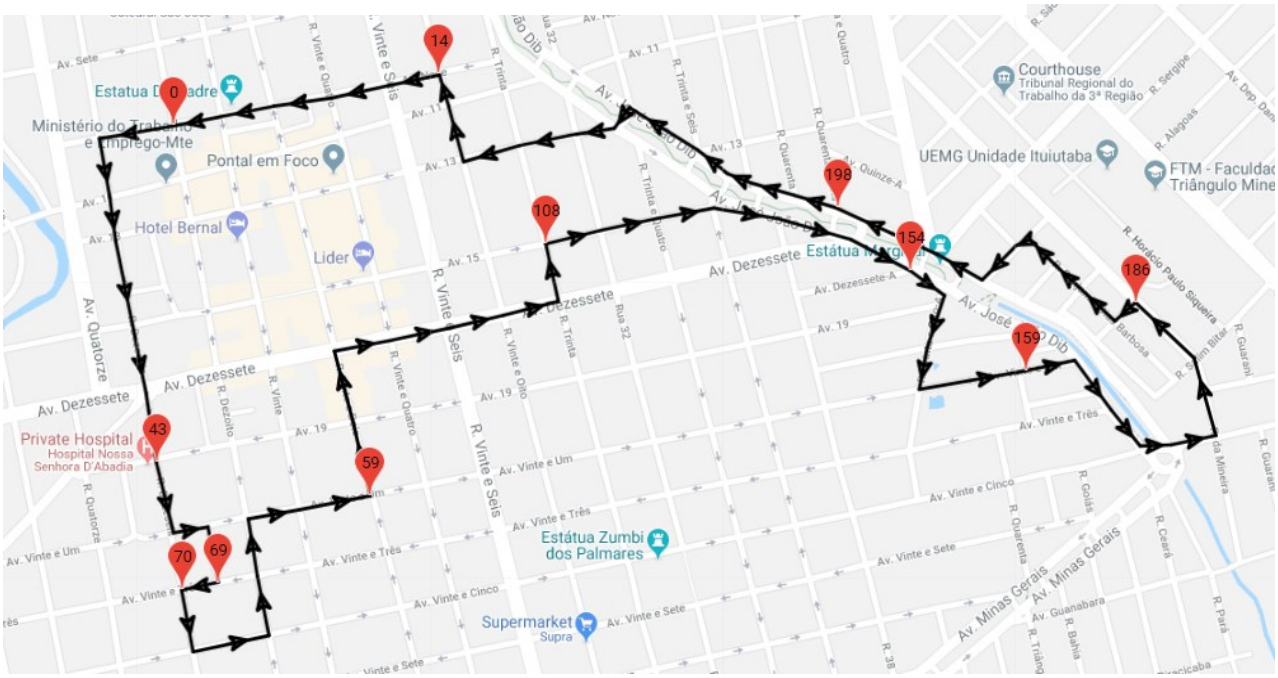
Fonte: API Google Maps

Figura 17. Resultado do roteamento da Instância 03.



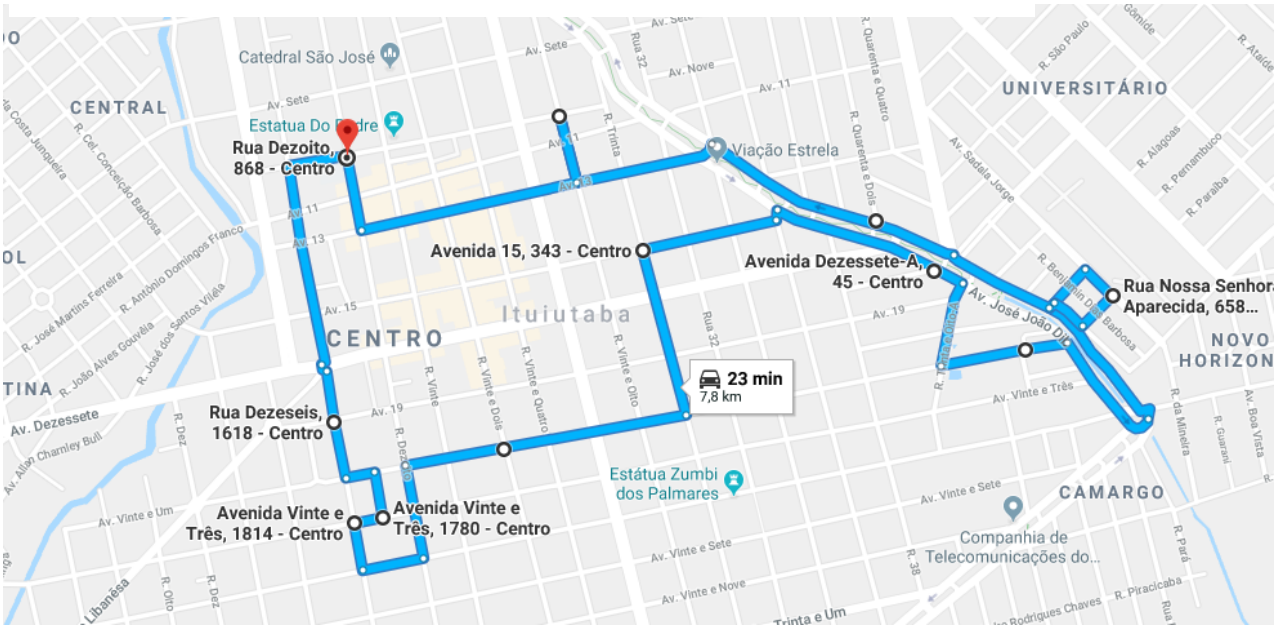
Fonte: Google Maps

Figura 18. Resultado do roteamento da Instância 04.



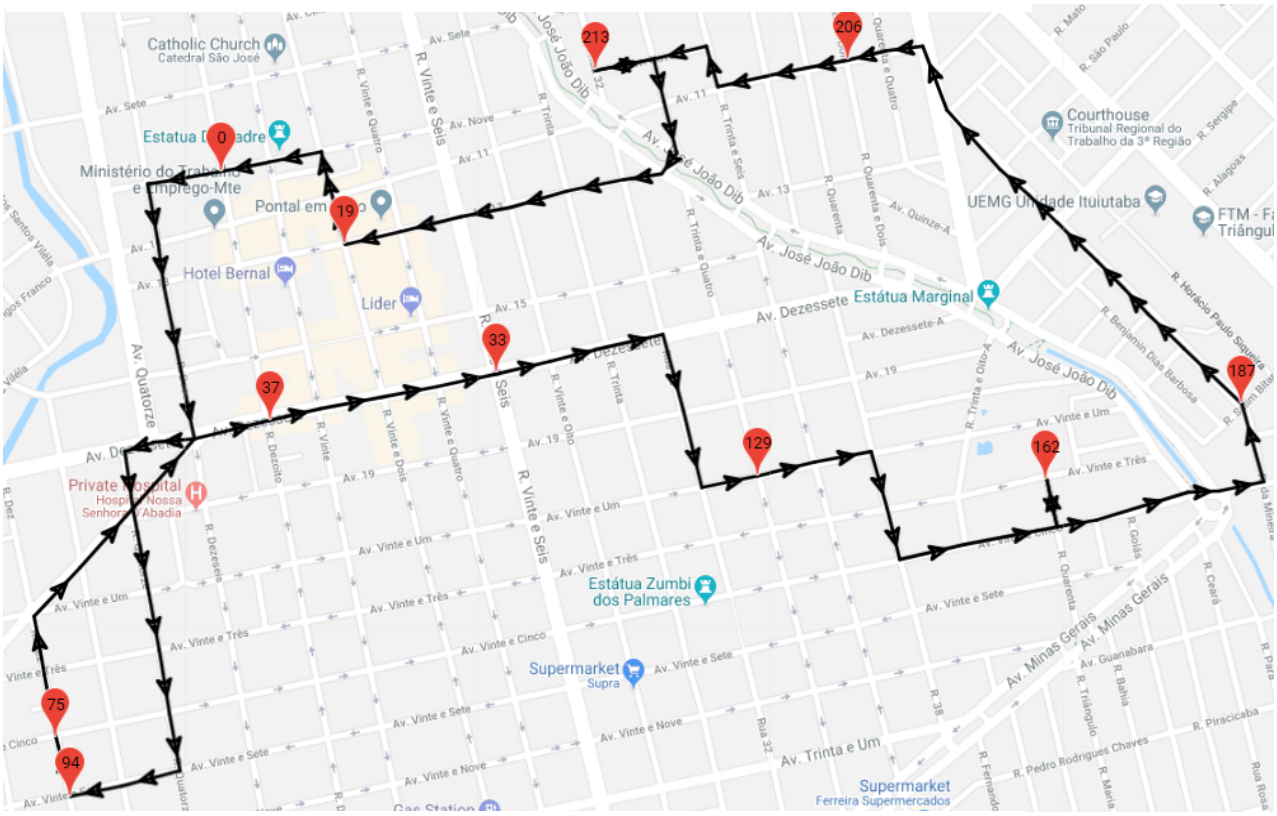
Fonte: API Google Maps

Figura 19. Resultado do roteamento da Instância 04.



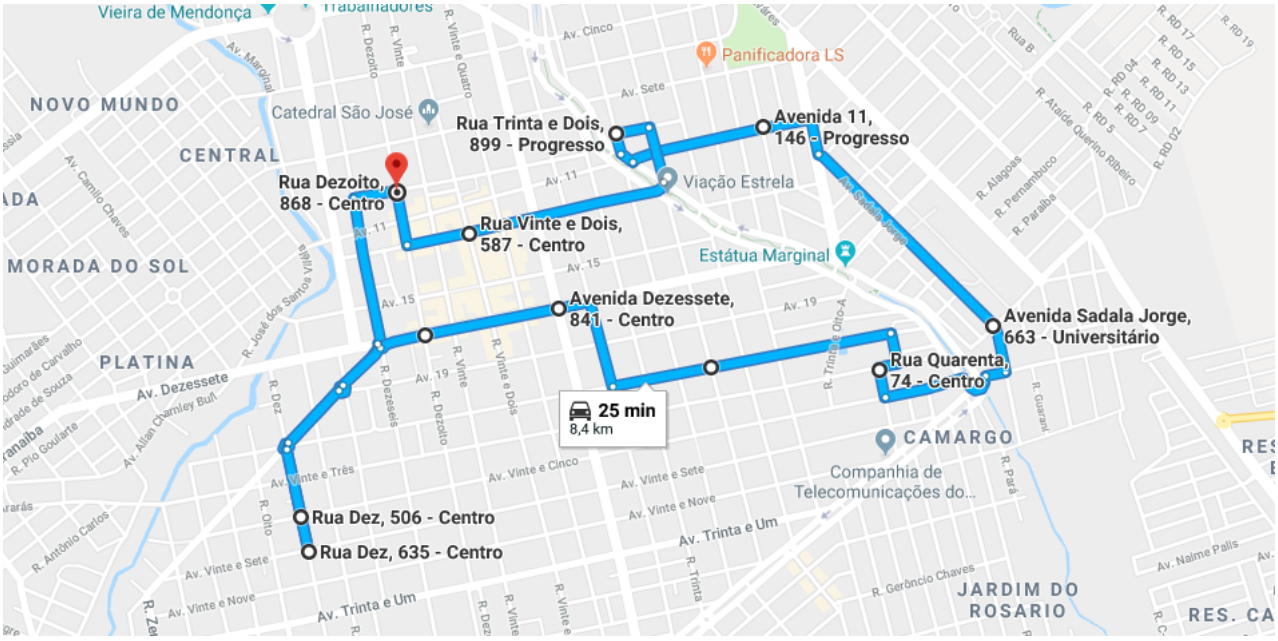
Fonte: Google Maps

Figura 20. Resultado do roteamento da Instância 05.



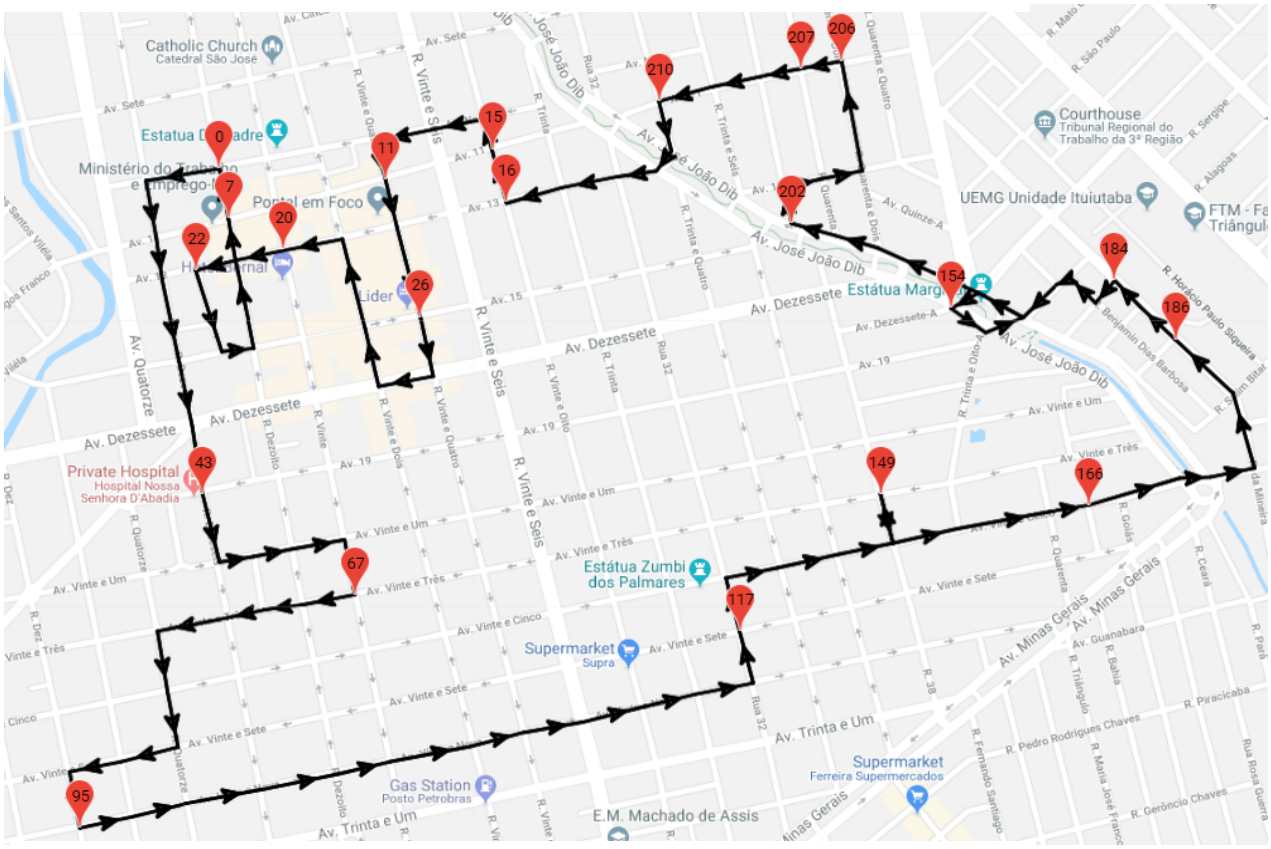
Fonte: API Google Maps

Figura 21. Resultado do roteamento da Instância 05.



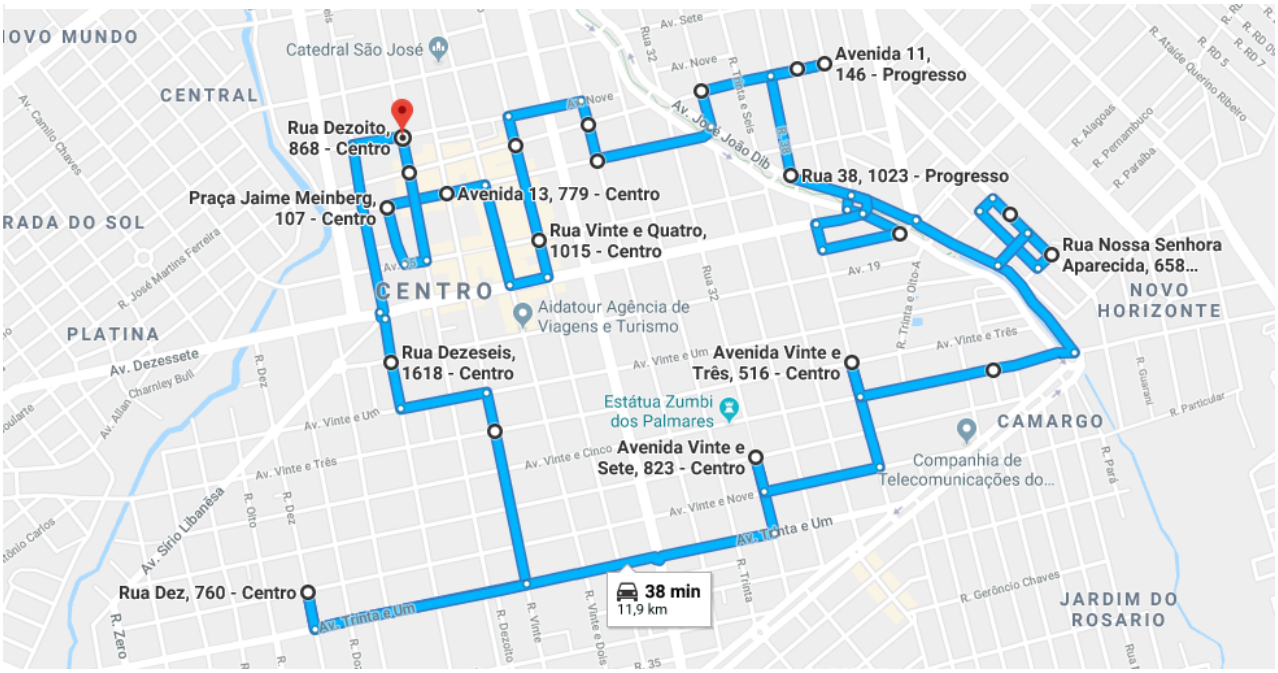
Fonte: Google Maps

Figura 22. Resultado do roteamento da Instância 06.



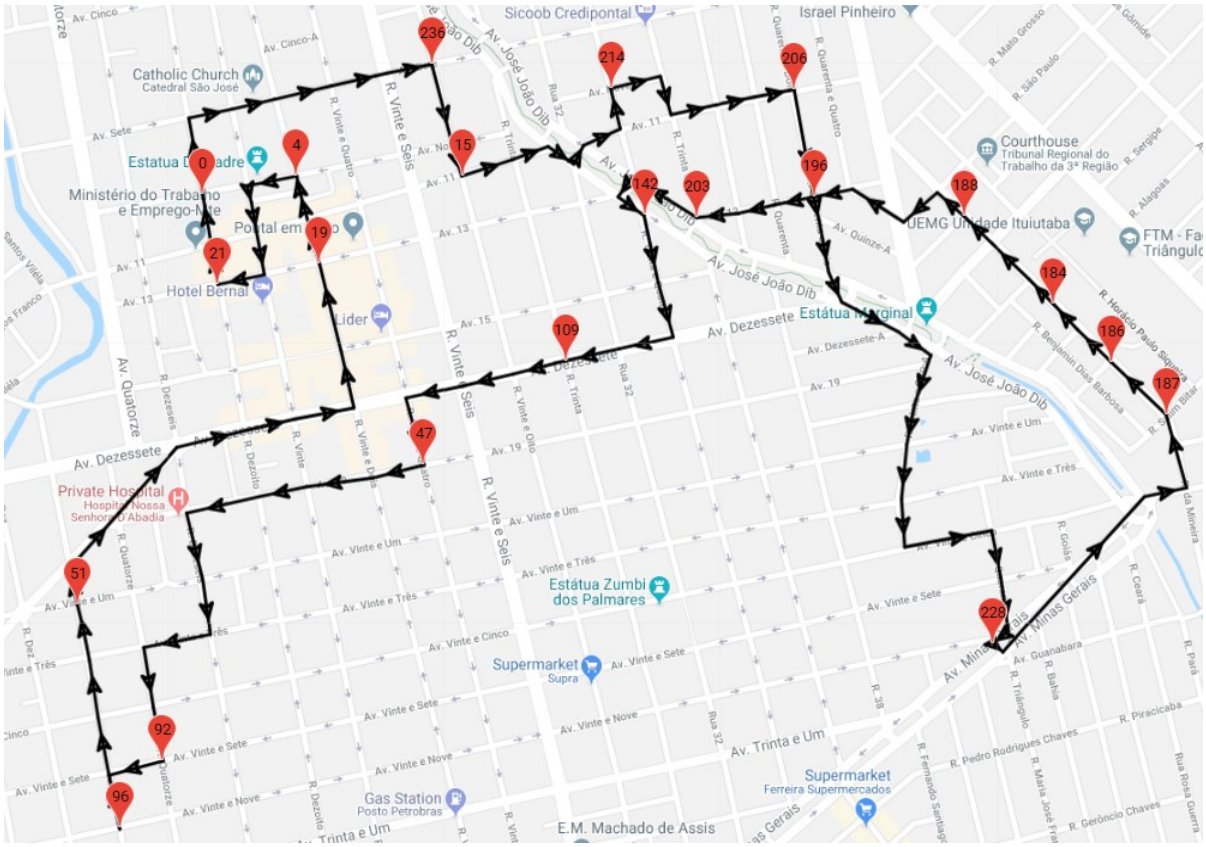
Fonte: API Google Maps

Figura 23. Resultado do roteamento da Instância 06.



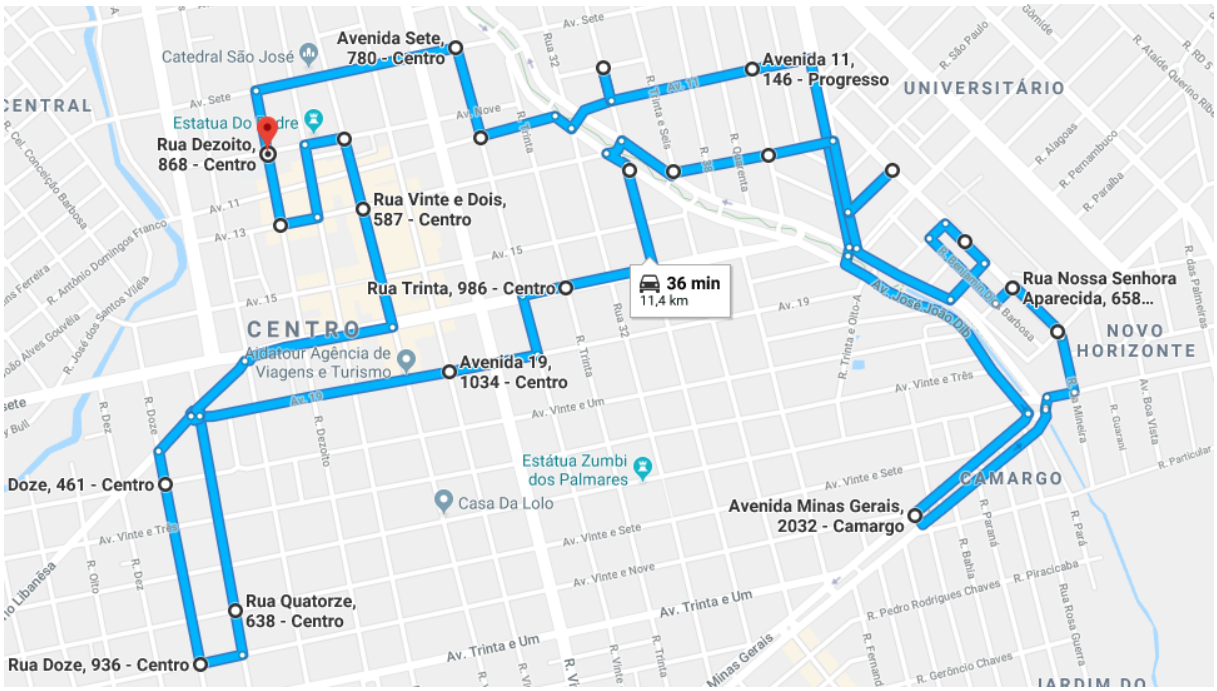
Fonte: Google Maps

Figura 24. Resultado do roteamento da Instância 07.



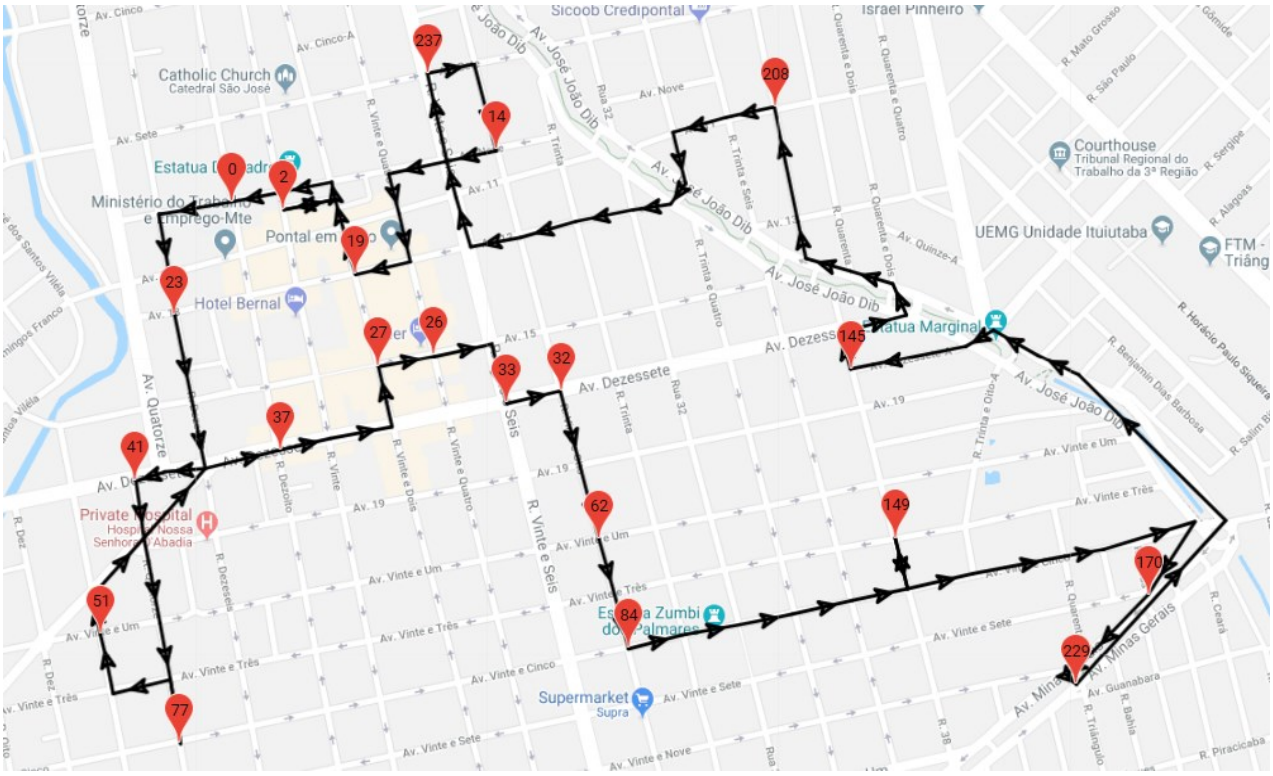
Fonte: API Google Maps

Figura 25. Resultado do roteamento da Instância 07.



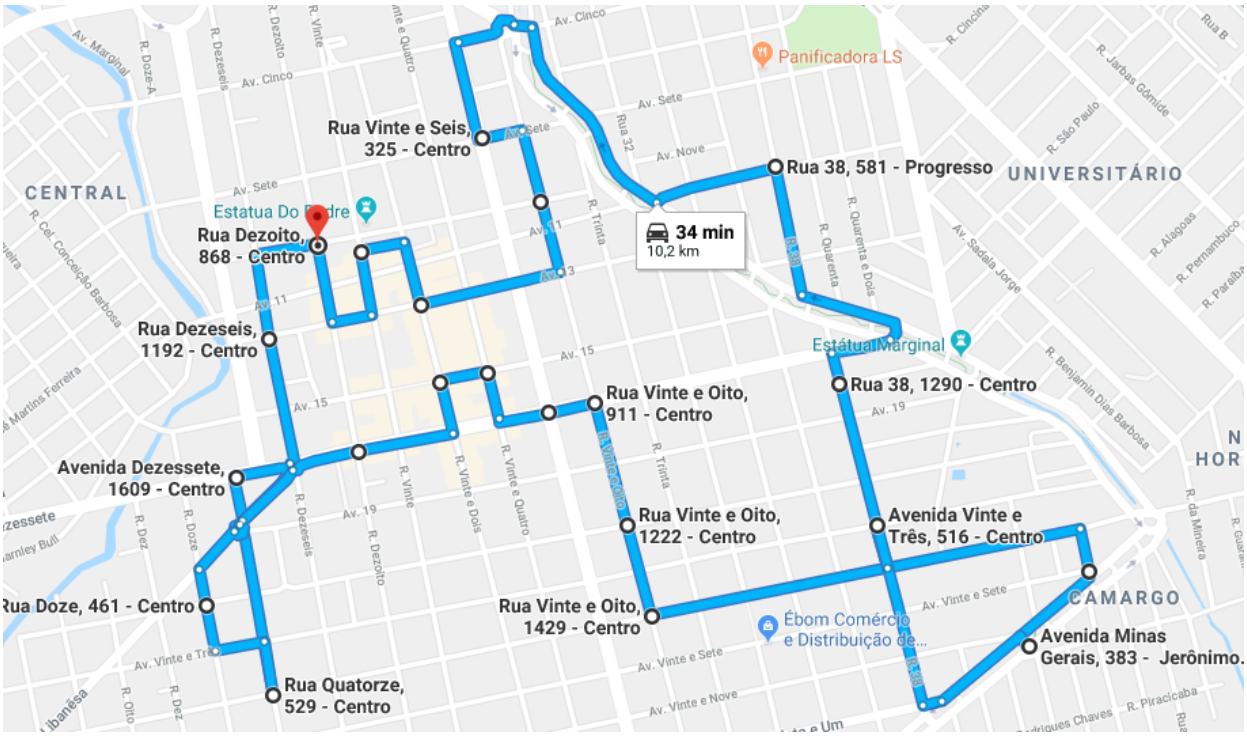
Fonte: Google Maps

Figura 26. Resultado do roteamento da Instância 08.



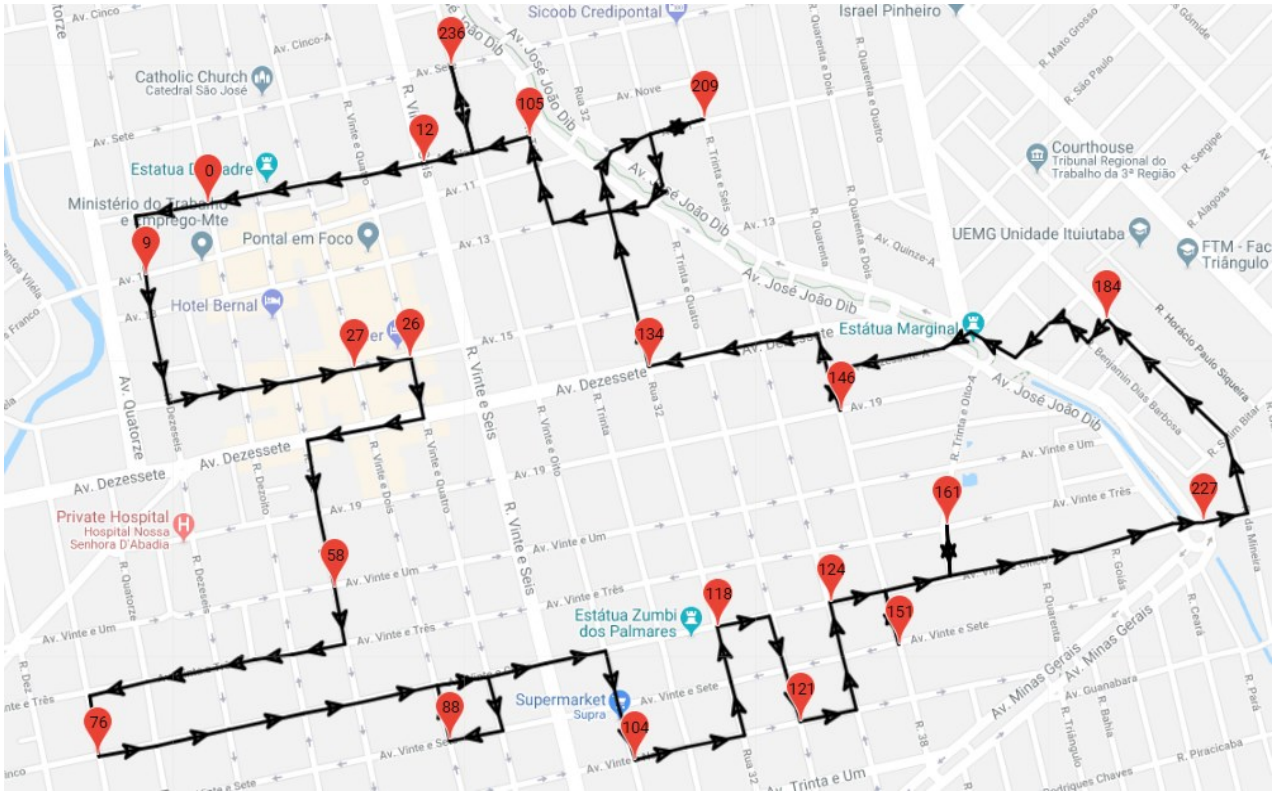
Fonte: API Google Maps

Figura 27. Resultado do roteamento da Instância 08.



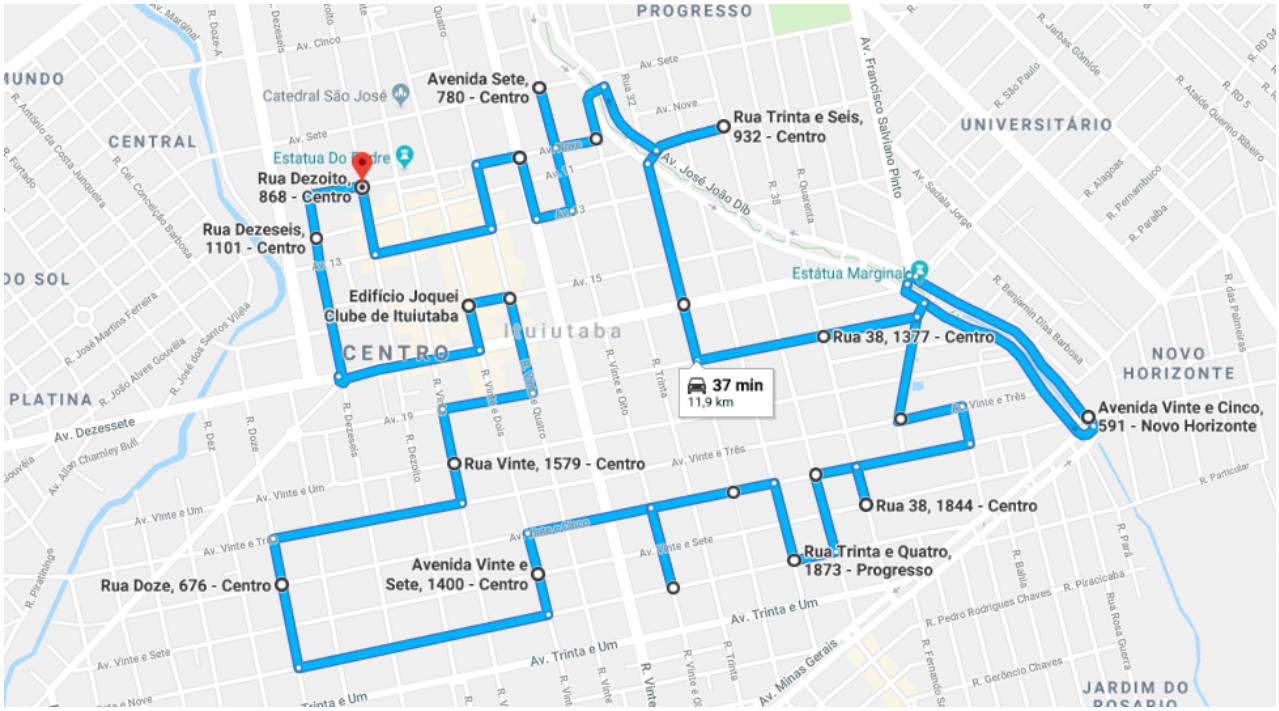
Fonte: Google Maps

Figura 28. Resultado do roteamento da Instância 09.



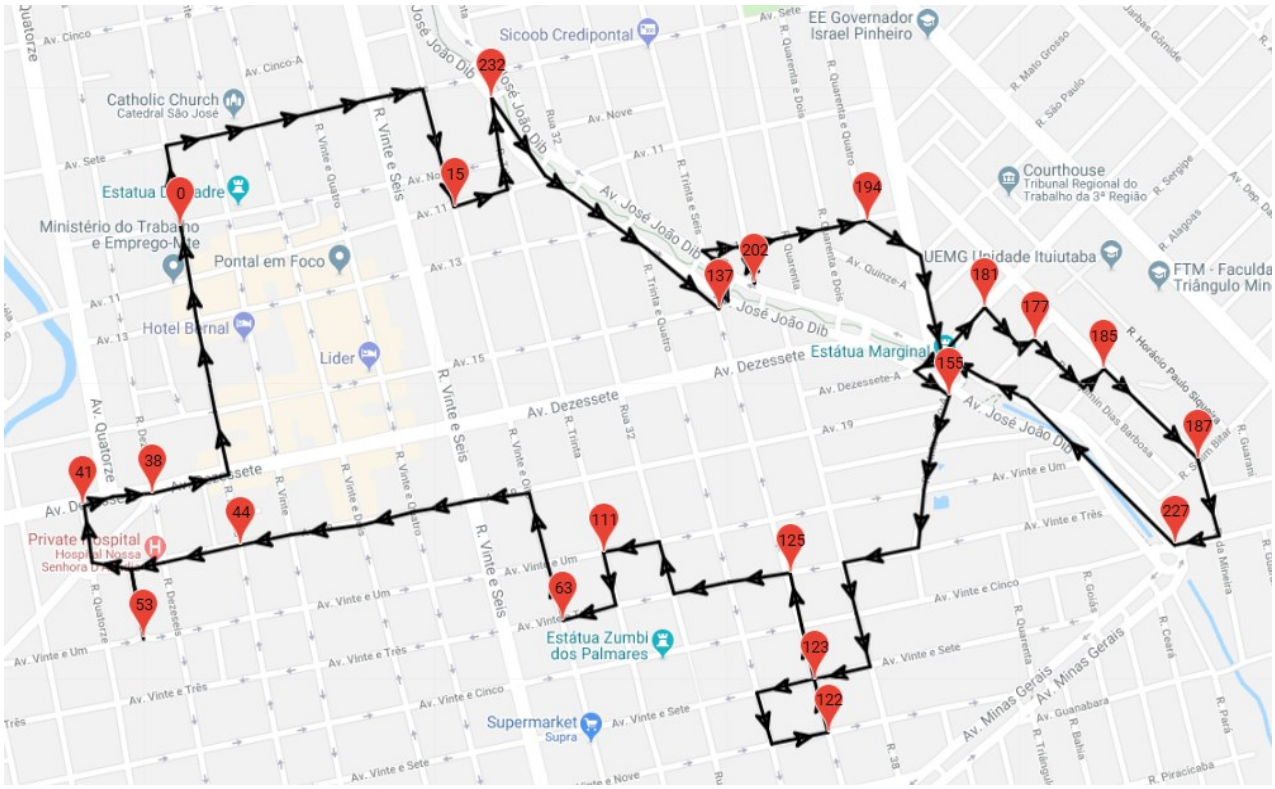
Fonte: API Google Maps

Figura 29. Resultado do roteamento da Instância 09.



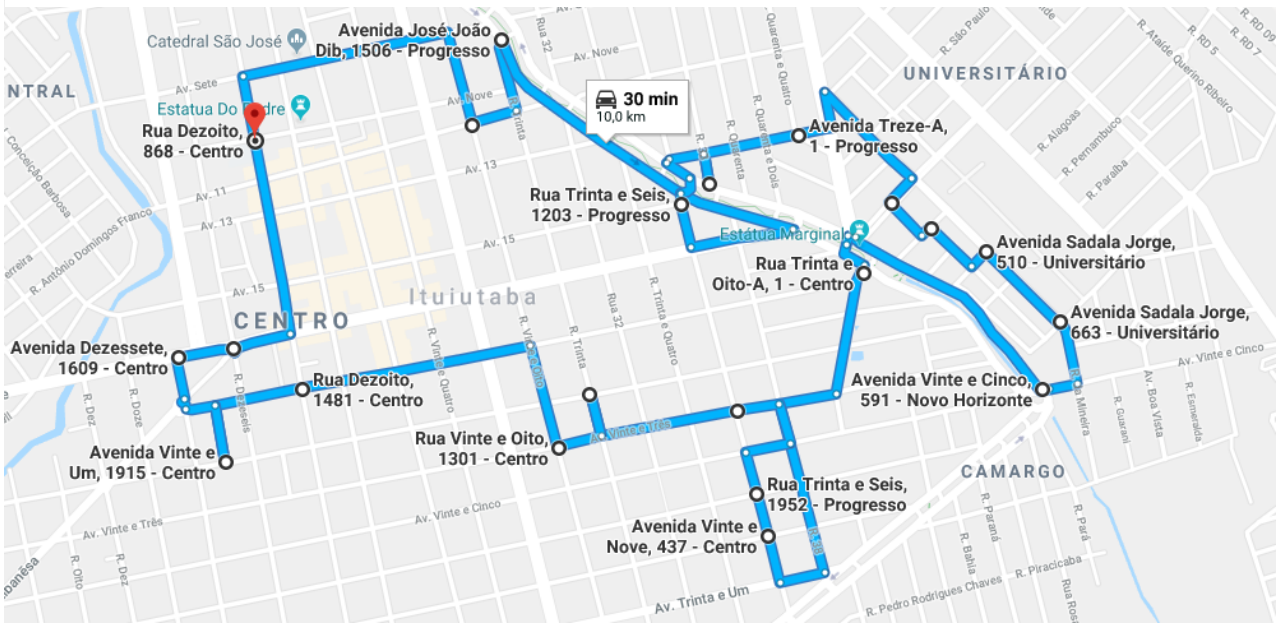
Fonte: Google Maps

Figura 30. Resultado do roteamento da Instância 10.



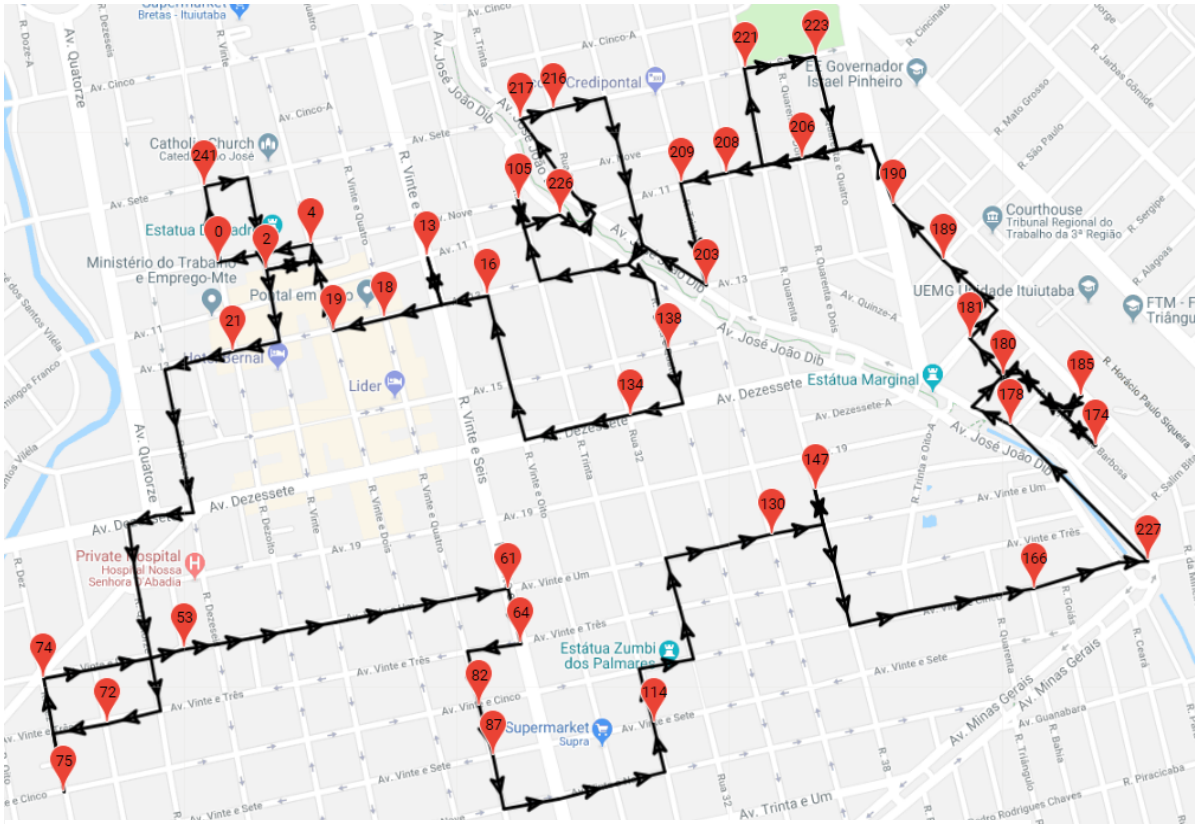
Fonte: API Google Maps

Figura 31. Resultado do roteamento da Instância 10.



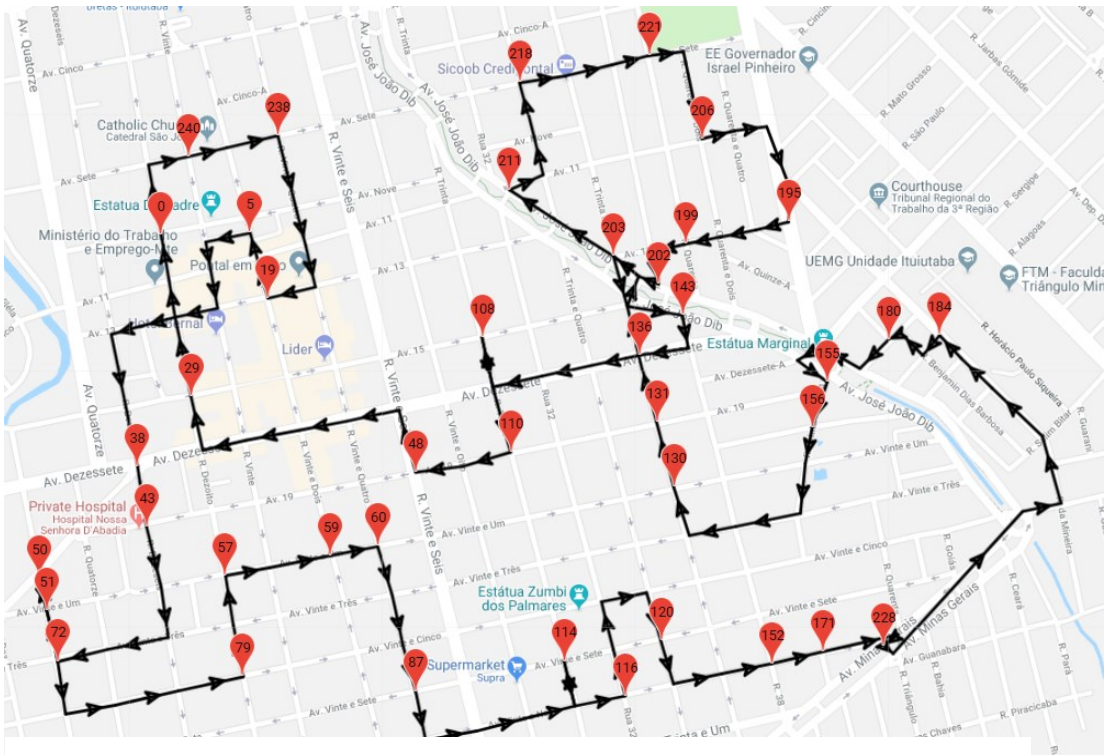
Fonte: Google Maps

Figura 32. Resultado do roteamento da Instância 11



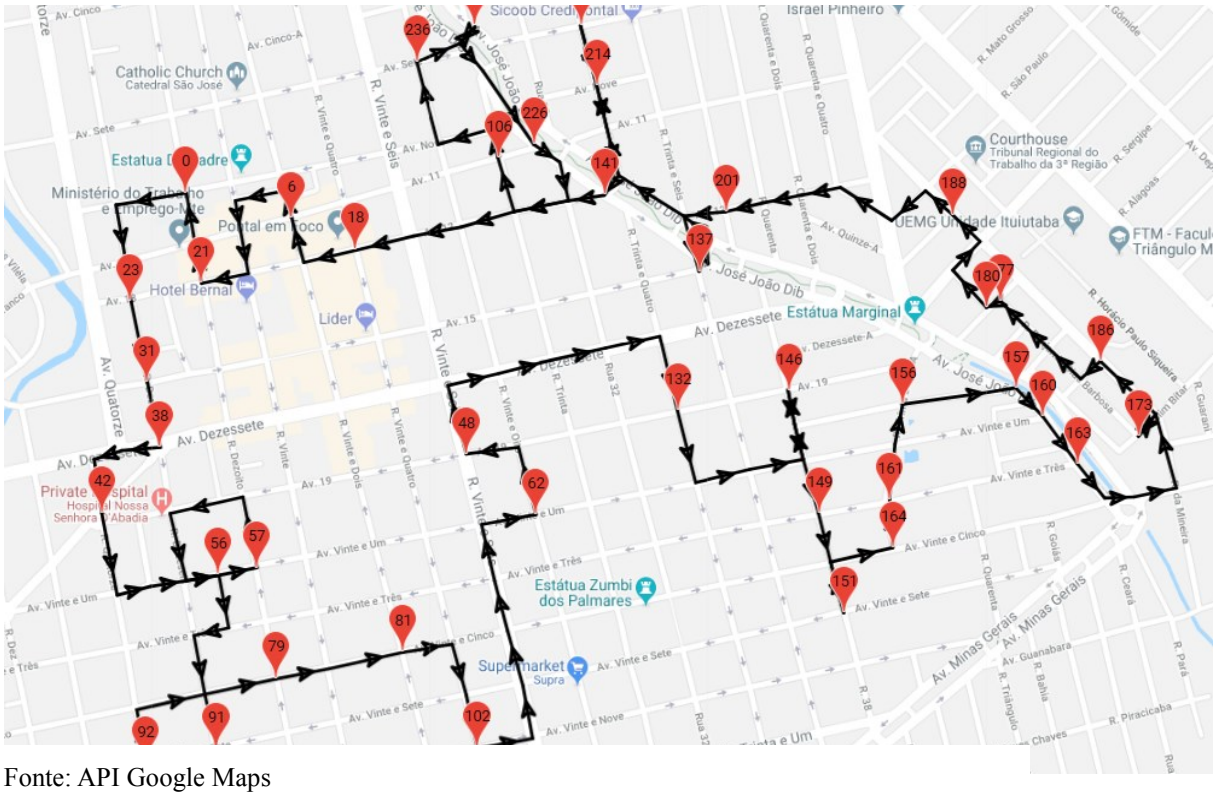
Fonte: API Google Maps

Figura 33. Resultado do roteamento da Instância 12



Fonte: API Google Maps

Figura 34. Resultado do roteamento da Instância 13

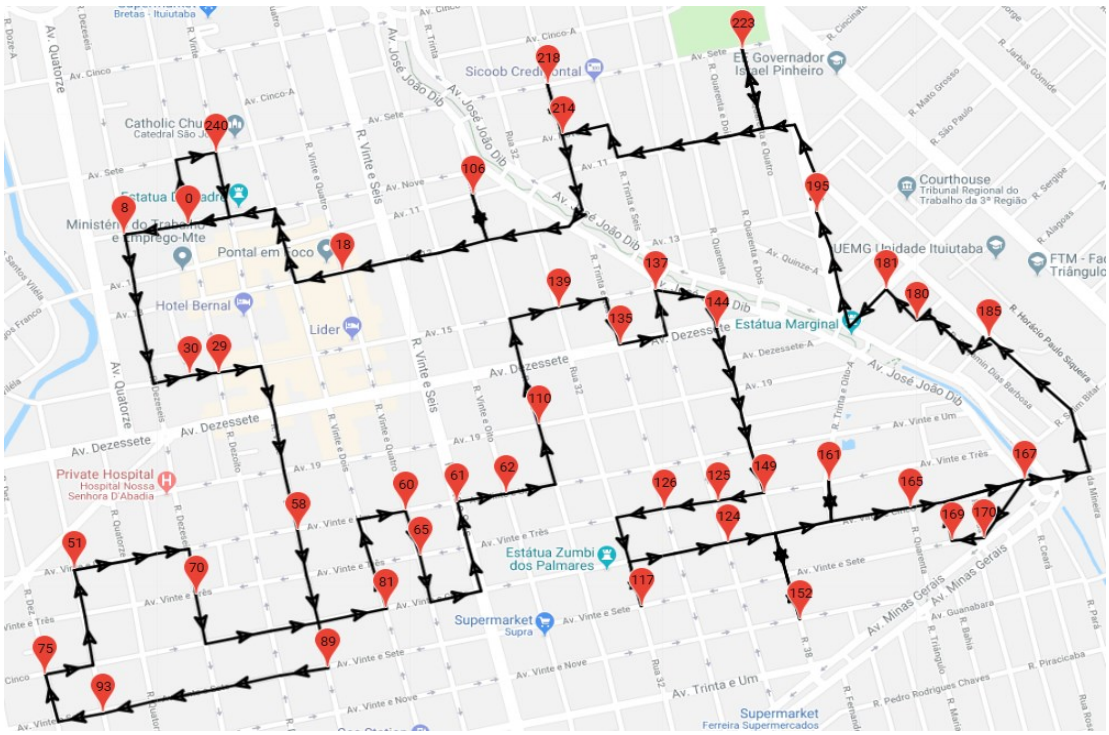


Fonte: API Google Maps

Figura 34. Resultado do roteamento da Instância 14

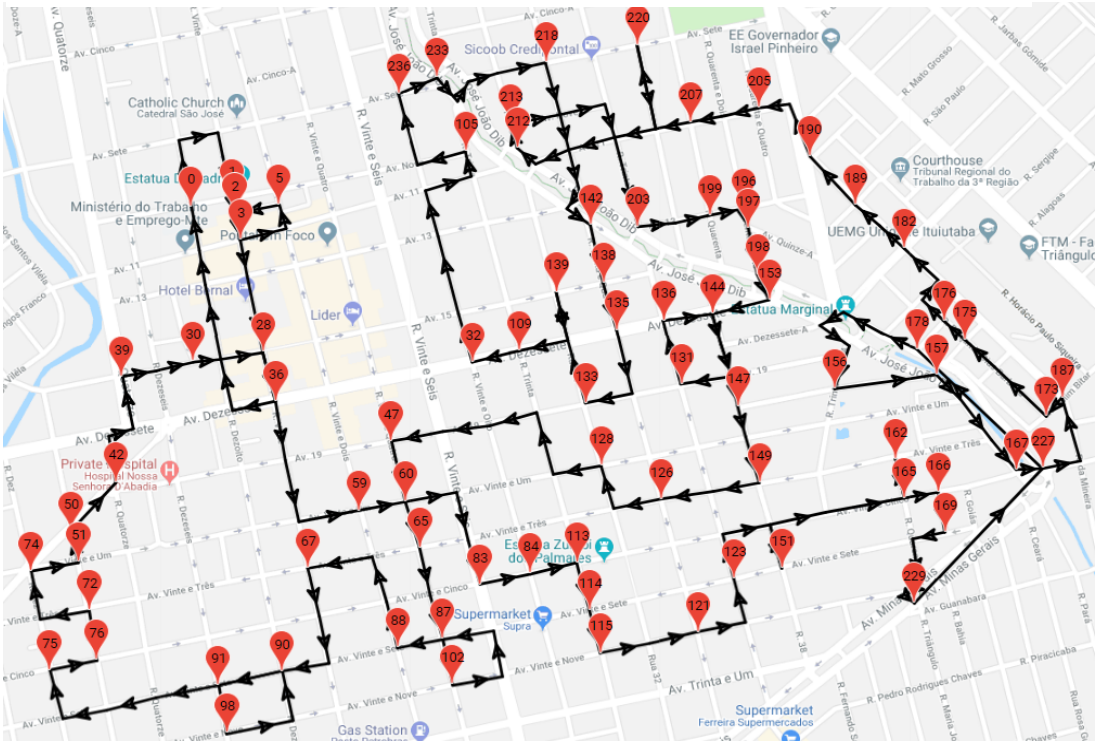
Fonte: API Google Maps

Figura 35. Resultado do roteamento da Instância 15



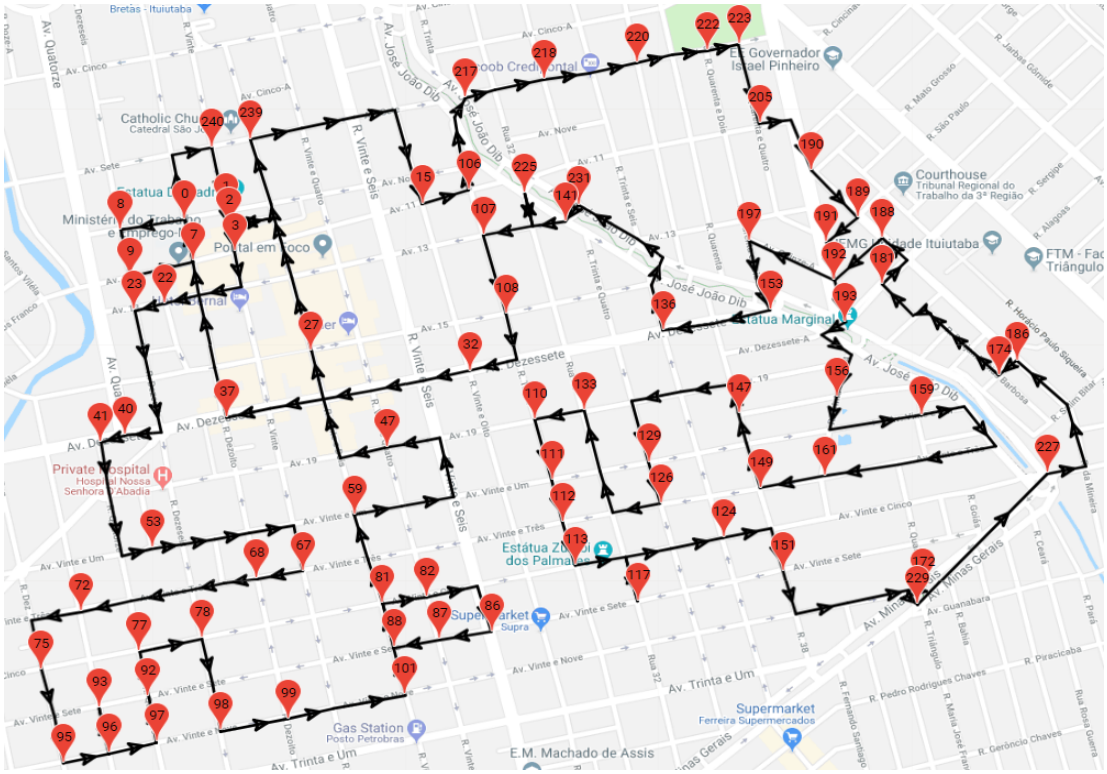
Fonte: API Google Maps

Figura 36. Resultado do roteamento da Instância 16



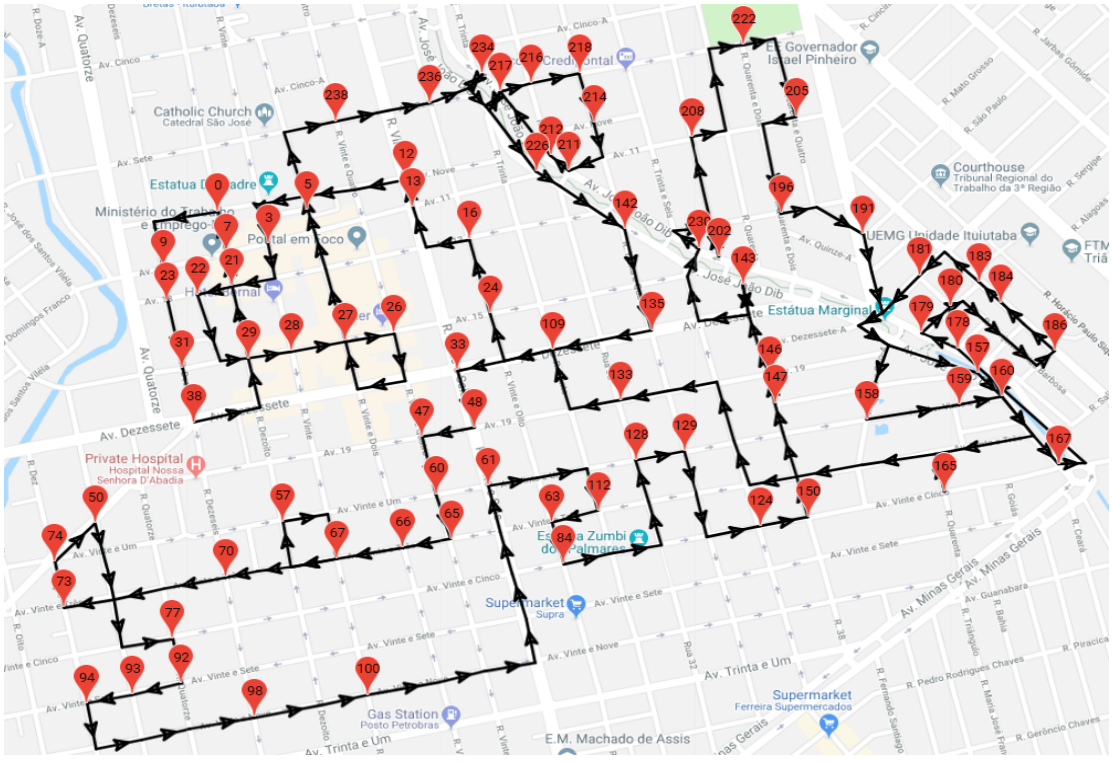
Fonte: API Google Maps

Figura 37. Resultado do roteamento da Instância 17



Fonte: API Google Maps

Figura 38. Resultado do roteamento da Instância 18

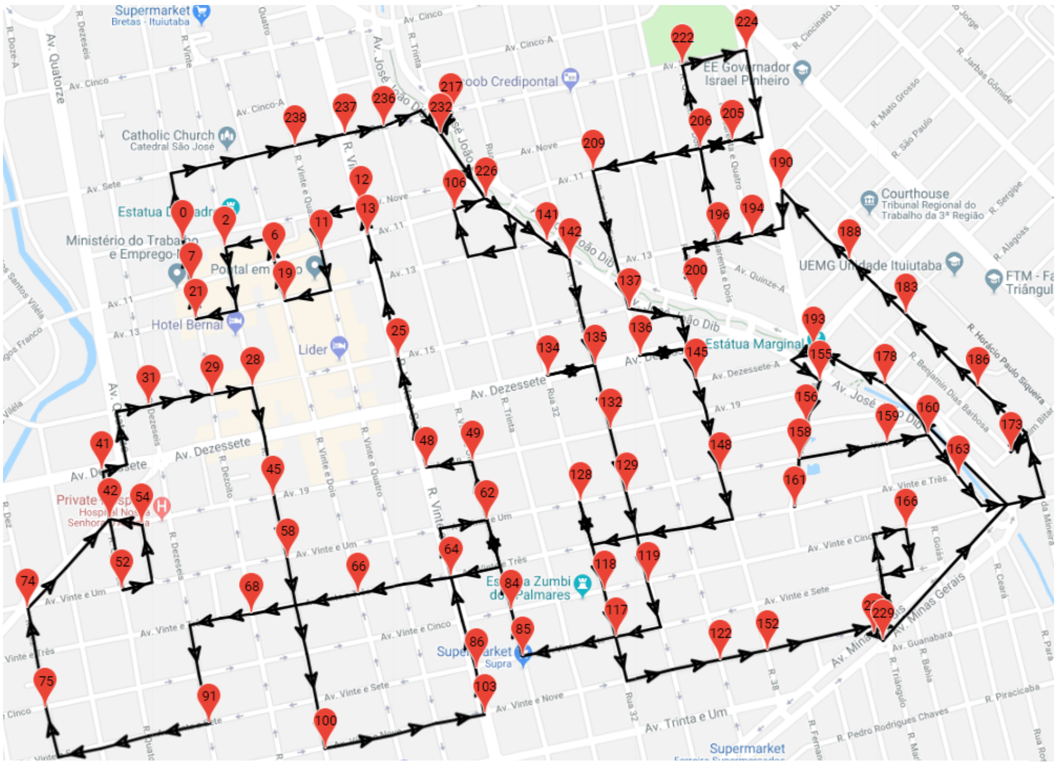


Fonte: API Google Maps

Figura 39. Resultado do roteamento da Instância 19

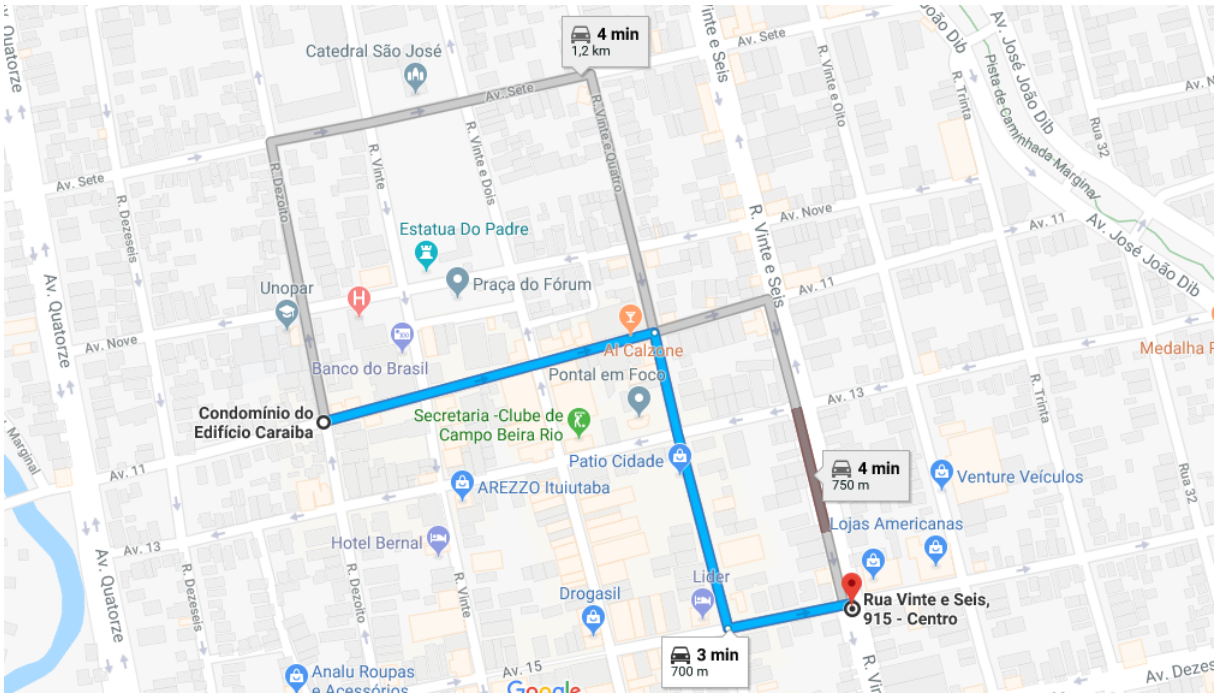
Fonte: API Google Maps

Figura 40. Resultado do roteamento da Instância 20



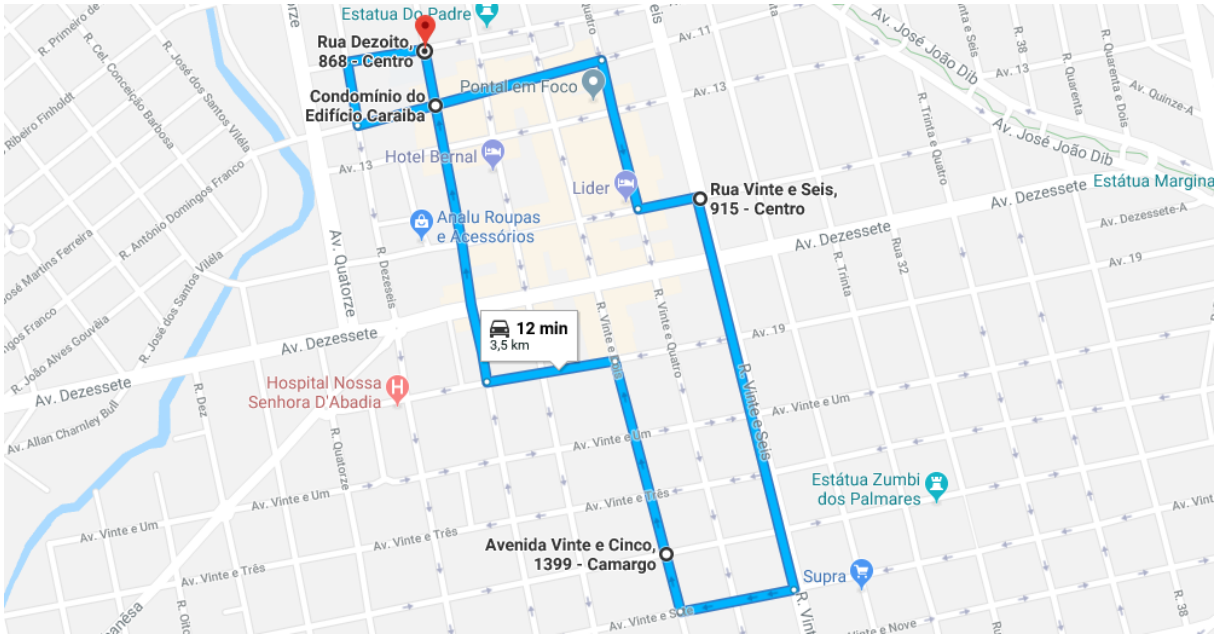
Fonte: API Google Maps

Figura 41. Exemplo de rota gerado pelo aplicativo do Google Maps



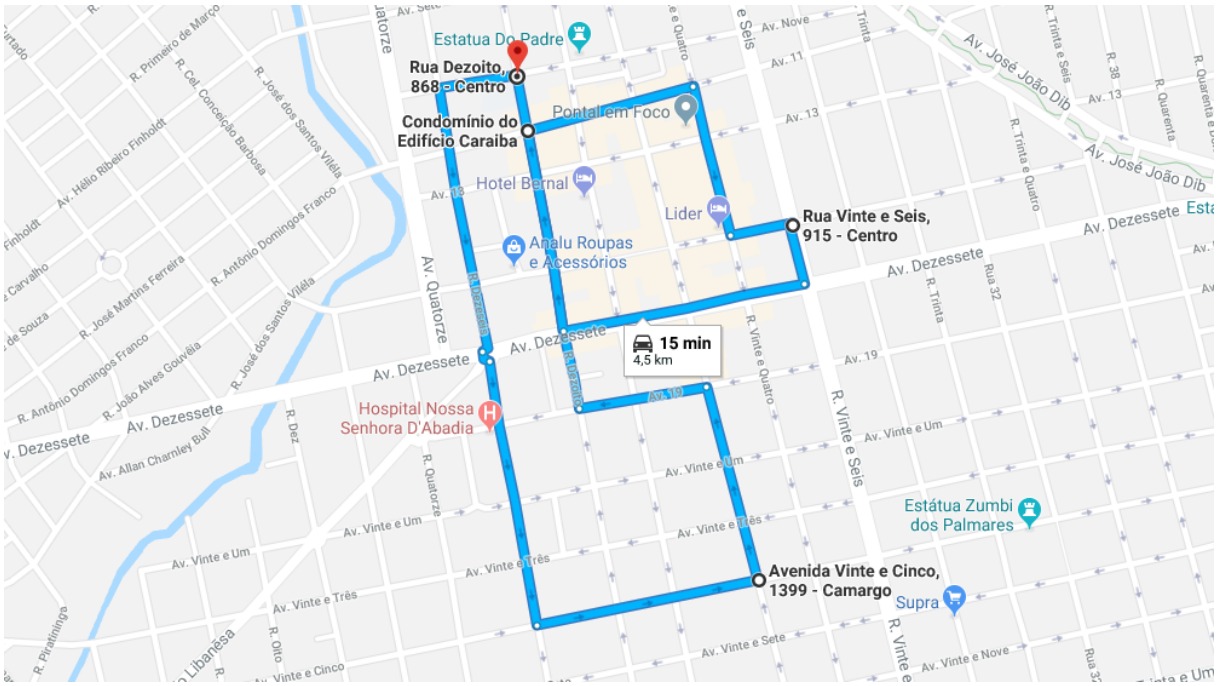
Fonte: Google Maps

Figura 42. Rota 0 – 7 – 25 – 81 – 0



Fonte: Google Maps

Figura 43. Rota 0 – 81 – 25 – 7 – 0



Fonte: Google Maps

Durante a pesquisa encontramos bastante dificuldades em encontrar artigos ou softwares que resolvam o problema proposto nesse trabalho. O aplicativo que mais se aproxima de uma solução do nosso problema é o aplicativo do Google Maps. Assim, os resultados obtidos nesse trabalho serão comparados com o roteador implementado no aplicativo da Google Maps disponível no site https://www.google.com.br/maps. Esse aplicativo gera uma rota entre dois pontos dados no mapa. Por exemplo, considerando os clientes 7 e 25 de nossa malha e introduzindo as coordenadas desses clientes no site do google maps se obtém o roteamento mostrado na Figura 41. O aplicativo do Google Maps mostra duas possíveis rotas, uma em azul e outra em cinza, partindo das coordenadas do vértice 7 e chegando nas coordenadas do vértice 25 de nossa malha. O aplicativo mostra a distância percorrida em cada rota e o tempo de percurso.

Um problema com o roteador do Google Maps é que ele roteia apenas dois vértices um de partida e um de chegada, contudo, permite a inserção de vários vértices e faz o roteamento desses vértices dois a dois. Para explicar melhor isso, considere o percurso 0 – 7 – 25 – 81 – 0 introduzindo no aplicativo do Google Maps. O resultado obtido é mostrado na Figura 29 cujo percurso é de 3500 metros. Alterando-se a ordem do vértice 7 com o vértice 81 (0 – 81 – 25 – 7 – 0), obtém-se a rota da Figura 43, cujo percurso é de 4500 metros, que é bem maior que o percurso original. Assim, pode-se ver que a rota gerada pelo aplicativo do Google Maps aparece

na ordem em que os vértices são postos e o roteamento ocorre entre cada vértice dado, portanto apenas uma comparação parcial é possível de se fazer. Para fins de comparação, vamos introduzir os vértices das instâncias estudadas nesse trabalho no aplicativo do Google Maps na ordem gerada por nosso roteador para comparar o tamanho das rotas obtidas. A comparação só pode ser feita nas instancias 01 a 10. Nos problemas apresentados pelas instâncias 11 a 20 o aplicativo do Google Maps não suportou a quantidade de arestas. Os resultados do Google Maps pode ser visualizado nas Figuras 8b a 17b. A Tabela 2 mostra um comparativo dos resultados obtidos no aplicativo do Google Maps com os resultados obtidos pelo nosso algoritmo.

Tabela 2. Comparativo dos resultados do Google Maps com nosso algoritmo

Instâncias	Google Maps	Nossa Heurística
Instância 01	7800 metros.	7538.34 metros.
Instância 02	7700 metros.	6742.00 metros.
Instância 03	9500 metros.	7995.59 metros.
Instância 04	7800 metros.	7298.71 metros.
Instância 05	8400 metros.	8681.01 metros.
Instância 06	11900 metros.	10522.4 metros.
Instância 07	11400 metros.	10487.5 metros.
Instância 08	10200 metros.	10203.2 metros.
Instância 09	11900 metros.	10730.1 metros.
Instância 10	10000 metros.	9141.85 metros.

5 ANÁLISE DOS RESULTADOS

Com a execução do algoritmo implementado em linguagem C^{++}, em um computador, foi possível obter os resultados do roteamento das instâncias 01 a 20, conforme mostrado pelas figuras 12 à 40. Esses resultados mostram rotas bastante abertas, resultado esperado, qualitativamente, devido à utilização das heurísticas de busca local 2-OPT e 3-OPT, cuja função primordial é descruzar as rotas. Contudo devido a orientação das ruas, em alguns casos há necessidade de cruzamento dessas rotas.

Para uma comparação mais qualitativa, a Tabela 2 mostra que os resultados obtidos pelo algoritmo implementado e pelo aplicativo do Google Maps são bastante consistentes um com o outro e que a metodologia utilizada para o roteamento nesse trabalho se mostrou melhor que a empregada no aplicativo do Google Maps, no sentido de se obter rotas mais curtas e permitir um roteamento completo.

A dificuldade de se encontrar softwares, aplicativos, artigos, teses ou dissertações contendo banco de dados de nós com resultados roteados impediu uma comparação mais aprofundada do algoritmo apresentado nesse trabalho com outros trabalhos da literatura. Na grande maioria dos trabalhos científicos usa-se a distância linear entre os nós como meio de comparação, diferentemente do caso proposto nesse trabalho, cuja distância entre nós considera o percurso e a orientação através das ruas e avenidas da cidade.

O principal resultado desse trabalho, foi a proposição de um método heurístico para calcular a rota de um carteiro de maneira a otimizar seu tamanho considerando a orientação das ruas e avenidas da cidade. Estudo este realizado devido a otimização de rotas ser de alta relevância no contexto acadêmico ou empresarial, pois permite economia de tempo e dinheiro na entrega das correspondências. Dentre todos os custos logísticos, o custo de transporte representa o maior percentual e por esta razão as empresas buscam sua redução, principalmente, por meio de métodos heurísticos. A aplicabilidade do método desenvolvido é ampla no contexto em que foi desenvolvido, permitindo sua aplicação a qualquer tipo de mapa.

6 CONCLUSÃO

O algoritmo apresentado neste trabalho, foi elaborado com intuído de calcular rotas para a entrega de cartas, minimizando o custo e a trajetória percorrida.

Os resultados obtidos foram satisfatórios, pois através da comparação com o aplicativo Google Maps, notou-se que o algoritmo proposto obteve melhores soluções. As comparações feitas, em sua maioria, mostraram que este algoritmo calcula rotas visualmente mais abertas, com distâncias menores, obtendo um custo menor.

O presente programa pode atingir mais de 80 clientes, já o aplicativo do Google Maps suporta no máximo cerca de 20 clientes e o roteamento é realizado entre dois pontos. Já no algoritmo exposto, a rota é estabelecida por todos os nós, de maneira ampla.

A perspectiva futura deste trabalho é a realização de estudo mais aprofundado em relação os efeitos estatísticos da variação dos parâmetros Tabu e dos controles estabelecidos no algoritmo.

REFERENCIAL BIBLIOGRÁFICO

ARENALES, M. et al. *Pesquisa Operacional para cursos de engenharia*. 6. ed. Rio de Janeiro: Editora Elsevier, 2007.

BELFIORE, P.; FÁVERO, L.P. *Pesquisa Operacional para cursos de Engenharia.* Rio de Janeiro: Elsevier, 2013.

BRANDÃO, MILENA ALMEIDA LEITE; DORICIO, JOSÉ LAERCIO; SARAMAGO, SEZIMÁRIA DE FÁTIMA PEREIRA. *Evolução Diferencial Melhorada Utilizando Processamento Paralelo Aplicado à Solução de Grandes Sistemas Lineares.* In: III CMACSE Congresso de Matemática Aplicada e Computacional Sudeste, 2015. v. 3.

BRANDÃO, M. A. L.; OLIVEIRA, G. T. S.; SARAMAGO, S. F. P.; DORICIO, J. L. . *Using metaheuristics for optimum design of 3R orthogonal manipulators considering their topology.* Journal of the Brazilian Society of Mechanical Sciences and Engineering, p. 1-18, 2014.

BUENO, FABRÍCIO. *Métodos Heurísticos: Teoria e Implementação.* Araranguá, 2009. Disponível em: <https://wiki.ifsc.edu.br/mediawiki/images/b/b7/Tutorial_m%C3%A9todos_heur%C3%ADsticos.pdf> . Acesso em:03 de fevereiro de 2018.

CROES, G. A. *A method for solving traveling salesman problems.* Operations Res. 6. ed. (1958) , pp., 791-812.

DORICIO, José Laércio; HALLAL, Renato; MARQUES, Antônio Carlos Henriques. *An Adaptive Algorithm to Solve the Periodic Vehicle Routing Problem.* Far East Journal of Applied Mathematics, v. 53, p. 1-31, 2011.

FARIA, Ana Cristina e COSTA, Maria de Fátima Garmeiro. *Gestão de custos logísticos: custeio baseado em atividades (ABC), balanced scorecard (BSC) e valor econômico agregado (EVA).* São Paulo: Atlas, 2008.

GLOVER, Fred. *"Future Paths for Integer Programming and Links to Artificial Intelligence".* Computers and Operations Research, 1986. 13 (5): 533–549. doi:10.1016/0305-0548(86)90048-1.
https://doi.org/10.1016/0305-0548(86)90048-1

GLOVER, Fred. *"Tabu Search – Part 1".* ORSA Journal on Computing, 1989. 1 (2): 190–206. doi:10.1287/ijoc.1.3.190.
https://doi.org/10.1287/ijoc.1.3.190

GLOVER, Fred. *"Tabu Search – Part 2".* ORSA Journal on Computing, 1990. 2 (1): 4–32. doi:10.1287/ijoc.2.1.4.
https://doi.org/10.1287/ijoc.2.1.4

GOMES, André. *Uma Introdução a Busca Tabu.* São Paulo, 2009. Disponível em: < https://www.ime.usp.br/~gold/cursos/2009/mac5758/AndreBuscaTabu.pdf>. Acesso em:15 de janeiro de 2018.

HILLIER, F. S.; LIEBERMAN, G. J. *Introdução à Pesquisa Operacional.* 8.ed. Porto Alegre: AMGH, 2010.

LAMBERT, R.; COOPER, M.; PAGH, C. *Supply chain management: implementation issues and research opportunities.* The International Journal of Logistics Management, v. 9, n. 2, 1998. https://doi.org/10.1108/09574099810805807

MICHALEWICZ, Z.; FOGEL.D. *How to Solve It: Modern Heuristic*. 2. ed. Springer, 2004. https://doi.org/10.1007/978-3-662-07807-5

NETTO, P. O. B. *Grafos: Teoria, Modelos, Algoritmos.* 4.ed. São Paulo: Edgard Blucher, 2006.

TAHA, H. A. *Pesquisa Operacional*. 8. ed. São Paulo: Pearson Prentice Hall, 2008.

ANEXOS

ANEXO A

Tabela contendo a localização dos vértices utilizados no grafo G = (V, A) que representa a malha do roteamento e as arestas através da lista de adjacências. O vértice zero representa o centro de distribuição. A latitude e a longitude estão apresentadas em graus.

Vértice	Latitude	Longitude	Lista de Adjacências
0	-18.971081342969114	-49.46526167130315	0 8 241
1	-18.970913932639494	-49.46436581349218	1 0 2
2	-18.971213241898248	-49.46429607605779	2 3 5
3	-18.971781420707423	-49.46417805886114	3 6 20
4	-18.970736376045544	-49.46339485382879	4 1 239
5	-18.971035685623153	-49.4633251163944	5 2 4
6	-18.971537915740683	-49.4631802771076	6 11 5
7	-18.971999560271378	-49.46507928109014	7 3 0
8	-18.971294410418157	-49.46665105557287	8 9
9	-18.972329305581855	-49.466447207687736	9 7 23
10	-18.97057911147581	-49.46230587696874	10 4 11
11	-18.971279191323667	-49.46211275791967	11 13 18
12	-18.970366043112367	-49.461211535690666	12 10 13 237
13	-18.970984955224107	-49.46105060314977	13 15 12 17
14	-18.970173266739305	-49.46027812695348	14 12 15 236
15	-18.970731302997226	-49.46009573674047	15 14 16 106
16	-18.97169517940574	-49.45982751583898	16 17 15 24
17	-18.971867661964513	-49.46077165341222	17 18 13 25

Vértice	Latitude	Longitude	Lista de Adjacências
18	-18.972121312461987	-49.46190891003454	18 19 26
19	-18.972354670578707	-49.46293887829626	19 20 6
20	-18.972547444428688	-49.46403321957433	20 21 28
21	-18.972740218055684	-49.464955899475456	21 22 7
22	-18.972922845496615	-49.465696189163566	22 23 30
23	-18.973075034877848	-49.4663506481632	23 31
24	-18.973359121350974	-49.45937690472448	24 16 32 108
25	-18.97357218588802	-49.46036395764196	25 24 17 33
26	-18.97378525015263	-49.461458298920036	26 25 34
27	-18.973967876448356	-49.462509724853874	27 26 19
28	-18.974191086093555	-49.46367916798437	28 27 36
29	-18.974343274316595	-49.46460184788549	29 28 21
30	-18.974394003693398	-49.465213391540885	30 29
31	-18.974576629321973	-49.46602878308141	31 30 38
32	-18.974404149566883	-49.45906576847875	32 33 24 49 109
33	-18.974617212767814	-49.46008500790441	33 34 32 25 48
34	-18.97484042154324	-49.461222264526725	34 35 33 47
35	-18.975023046682573	-49.462252232788444	35 36 34 27
36	-18.97523610909217	-49.46340021824682	36 37 35 45
37	-18.975439025419448	-49.464344355820006	37 38 36 29
38	-18.975763691029133	-49.46577129101598	38 40 37 43
39	-18.974708525484814	-49.46677980160558	39 31 40
40	-18.97582456576052	-49.46654376721227	40 38 41 39

Vértice	Latitude	Longitude	Lista de Adjacências
41	-18.9759767524917	-49.46708020901525	41 40 42
42	-18.97697103571475	-49.46691927647436	42 38 41 52
43	-18.97698118143133	-49.46556744313085	43 54 55
44	-18.9766768096648	-49.46412977909887	44 43 37
45	-18.97649418633798	-49.463196370361686	45 44 58
46	-18.97632170856776	-49.46188745236242	46 45 35
47	-18.97608835600622	-49.4608574841007	47 46 60
48	-18.975885440469664	-49.459720227478385	48 47 33 61
49	-18.97575354523841	-49.45867953038061	49 48 32 62
50	-18.97801604127849	-49.46790632939184	50 42 51
51	-18.9786755074357	-49.467734668014884	51 52 72 50
52	-18.97847259504998	-49.46662959790075	52 53 42 71
53	-18.97831026496346	-49.465953681228996	53 55 54
54	-18.9770927842731	-49.466189715622306	54 42 53
55	-18.978218954220274	-49.465256306885124	55 56 70
56	-18.978107352132884	-49.46461257672155	56 57 69
57	-18.977975458660488	-49.46386155819738	57 58 44
58	-18.977813128089583	-49.46288523411596	58 59 67
59	-18.97756963193668	-49.46157631611669	59 60 46
60	-18.977376863896584	-49.46054634785497	60 61 65
61	-18.977184095633465	-49.45941982006872	61 62 48 64
62	-18.97700147286267	-49.45834693646276	62 49 63 111
63	-18.977995749970724	-49.45804652905309	63 64 62 84

Vértice	Latitude	Longitude	Lista de Adjacências
64	-18.978208808579062	-49.45916232800329	64 65 61 83
65	-18.978360993132494	-49.460224482773185	65 66 82
66	-18.97855376003393	-49.46127590870702	66 67 59
67	-18.978807400353904	-49.462692115066886	67 68 80
68	-18.978979875550902	-49.463689896820426	68 69 57
69	-18.979091477053824	-49.46441945767248	69 70 56
70	-18.9792233696427	-49.46509537434423	70 71 78
71	-18.979477008943217	-49.466436478851676	71 72 52 77
72	-18.979710356758634	-49.467498633621574	72 73 76 51
73	-18.979892976560087	-49.46858224606359	73 75 74
74	-18.97886827397324	-49.46877536511266	74 51 50 73
75	-18.98097854569216	-49.4683676693424	75 76 94 73
76	-18.980765490626066	-49.46733770108068	76 77 93 72
77	-18.98058287178111	-49.466243359802604	77 78 71 92
78	-18.98029879762436	-49.46488079762304	78 79 91
79	-18.980014722983178	-49.463475320099235	79 80 68
80	-18.97983210331518	-49.46243462300146	80 81 89
81	-18.979527736756985	-49.460964772461296	81 82 66
82	-18.979385698839515	-49.459999177215934	82 83 87
83	-18.979152350569358	-49.45890483593786	83 84 64 86
84	-18.978919001972354	-49.45778903698766	84 63 85 113
85	-18.979881128980068	-49.45755120269837	85 86 84 104
86	-18.98014491280306	-49.45860262863221	86 87 83 103

Vértice	Latitude	Longitude	Lista de Adjacências
87	-18.980317386615322	-49.45972915641846	87 88 102
88	-18.980500005751328	-49.460726938172	88 89 81
89	-18.98082466149879	-49.4622397040564	89 90 100
90	-18.980997134607286	-49.46326967231812	90 91 79
91	-18.981301498480747	-49.464664421005864	91 92 98
92	-18.981565280054564	-49.46605916969361	92 93 77 97
93	-18.981818915789326	-49.467121324463506	93 94 96 76
94	-18.982011678689418	-49.468097648544926	94 95 75
95	-18.983026216592194	-49.467893800659795	95 96 94
96	-18.982813164145202	-49.466906747742314	96 97 93
97	-18.982569675300656	-49.465876779480595	97 98 92
98	-18.982326186100185	-49.46448203079285	98 99
99	-18.98205226032419	-49.46302290908875	99 100 90
100	-18.98182906121067	-49.46203585617127	100 101
101	-18.981534843740445	-49.46048017494263	101 102 88
102	-18.98129135302725	-49.45951457969727	102 103
103	-18.981108734758784	-49.458409509583134	103 104 86
104	-18.980834806980706	-49.45724006645264	104 85 115
105	-18.96990396383766	-49.45922566795349	105 14 106 232
106	-18.970431562654845	-49.459096921920775	106 107 105 226
107	-18.971425878963068	-49.45877505683899	107 16 108 106
108	-18.973150699305204	-49.45828153038025	108 109 139 107
109	-18.97416529116768	-49.45804549598694	109 32 110 134 108

Vértice	Latitude	Longitude	Lista de Adjacências
110	-18.975484251353915	-49.45765925788879	110 49 111 109
111	-18.97680320110004	-49.45729447746277	111 112 110
112	-18.977736605380716	-49.45705844306946	112 63 113 111
113	-18.978690295657824	-49.45677949333191	113 118 114 112
114	-18.979704853789286	-49.45650054359436	114 85 115 113
115	-18.980617950824772	-49.45626450920105	115 116 114
116	-18.980415040804967	-49.45517016792297	116 117 121
117	-18.97948165153048	-49.4554276599884	117 114 116 118
118	-18.978507674538005	-49.45568515205383	118 117 127 119
119	-18.97832505321799	-49.45469809913635	119 120 124
120	-18.979278740126624	-49.45439769172668	120 121 117
121	-18.980171548454944	-49.454118741989134	121 122
122	-18.97996863789139	-49.4530673160553	122 123 152
123	-18.979035246115767	-49.453324808120726	123 124 120
124	-18.978081557813272	-49.45356084251404	124 125 150
125	-18.9771684468761	-49.45377541923523	125 130 126
126	-18.977330778075125	-49.45493413352966	126 119 127
127	-18.977533691851487	-49.45596410179138	127 118 112 128
128	-18.976620577911756	-49.45624305152893	128 127 111 133 129
129	-18.976417663023362	-49.45519162559509	129 130 126
130	-18.976174164831075	-49.454118741989134	130 131 148
131	-18.97487550179599	-49.45448352241516	131 136 132
132	-18.97507841856239	-49.45555640602112	132 133 129

Vértice	Latitude	Longitude	Lista de Adjacências
133	-18.975281335081696	-49.456586374282836	133 128 110 134
134	-18.973942081487927	-49.45697261238098	134 133 109 139 135
135	-18.973718871509206	-49.45589972877502	135 134 136 132
136	-18.97351595308729	-49.454869760513304	136 137 135 144
137	-18.972521649248392	-49.455127252578734	137 143 230
138	-18.972704276928845	-49.45620013618469	138 137 135
139	-18.972907196339058	-49.45723010444641	139 134 140 138
140	-18.971222957749273	-49.45770217323303	140 139 107 225
141	-18.97112149704973	-49.4569940700531	141 140 142 231
142	-18.97146646317619	-49.45647908592224	142 138 137
143	-18.972704276928845	-49.45392562294006	143 144 153
144	-18.97337391004493	-49.45379687690735	144 136 153 153 145 143
145	-18.974023248678787	-49.453603757858275	145 146 144 154
146	-18.974713168204932	-49.453367723464964	146 131 147 145
147	-18.97534220998931	-49.45323897743225	147 156 148 146
148	-18.975991540953363	-49.4530673160553	148 158 149 147
149	-18.976985824088146	-49.45276690864563	149 125 150 148
150	-18.97787864470414	-49.45253087425232	150 164 151 149
151	-18.978812042959827	-49.45227338218689	151 123 152 150
152	-18.979765727080693	-49.45199443244934	152 171 151
153	-18.973069531689358	-49.45255233192444	153 144 154 198
154	-18.973617412329137	-49.45143653297424	154 145 155
155	-18.974063832259397	-49.45079280281067	155 146 157 179 156

Vértice	Latitude	Longitude	Lista de Adjacências
156	-18.974976960210068	-49.451114667892455	156 157 147 155 158
157	-18.974672584782475	-49.448883069992064	157 156 160
158	-18.975707458967946	-49.45126487159729	158 159 156 161
159	-18.975382793248695	-49.44926930809021	159 160
160	-18.97520016850363	-49.448346628189086	160 163
161	-18.976722035263297	-49.451372159957884	161 149 158 164
162	-18.976478537515824	-49.44978429222107	162 161 165
163	-18.976072707145935	-49.44765998268127	163 162 167
164	-18.977614857292835	-49.45130778694153	164 165 161
165	-18.977310486683887	-49.449591173171996	165 166 162 168
166	-18.97718873828463	-49.448840154647826	166 167 169
167	-18.976701743797925	-49.447123540878295	167 227 170
168	-18.97818301427515	-49.44939805412292	168 151 165 172
169	-18.97804097521122	-49.44868995094299	169 168 166
170	-18.977980101289603	-49.44800330543518	170 169 172
171	-18.979522233781605	-49.450878633499144	171 172
172	-18.979177284332014	-49.44924785041809	172 228 168
173	-18.975565417793614	-49.4464583530426	173 187 174
174	-18.974510250993763	-49.447595609664916	174 173 186 175
175	-18.97392178968403	-49.44826079750061	175 174 185 176
176	-18.97351595308729	-49.44871140861511	176 175 184 177
177	-18.973069531689358	-49.44918347740173	177 176 183 180
178	-18.974124707611775	-49.44931222343445	178 179

Vértice	Latitude	Longitude	Lista de Adjacências
179	-18.9737797469876	-49.450084699630736	179 193 180
180	-18.9731912830983	-49.44946242713928	180 177 179 181
181	-18.97250135727155	-49.450127614974974	181 182 180 193
182	-18.97197376500628	-49.449591173171996	182 183 181 188
183	-18.9726425291374	-49.448834790229796	183 177 184 182
184	-18.973109243584467	-49.448351992607115	184 176 185 183
185	-18.973575956724506	-49.447879923820494	185 175 186 184
186	-18.974174565142935	-49.44720400714874	186 174 187 185
187	-18.97515871329275	-49.446109665870665	187 235 173 186
188	-18.971475737301628	-49.450122250556944	188 189 182 192
189	-18.971029310438688	-49.45065869235992	189 190 188 191
190	-18.96999440720301	-49.45165647411346	190 195 189 204
191	-18.97158734383054	-49.451334609031676	191 192 195 189 194
192	-18.972409171423728	-49.45116294765472	192 193 191 188 197
193	-18.973332454379726	-49.45092691326141	193 198 154 181 192
194	-18.9709785800377	-49.452332390785216	194 195 196 191 205
195	-18.970806096558814	-49.451624287605284	195 191 190 194
196	-18.971120625121564	-49.453104866981505	196 194 199 197 206
197	-18.971546759646845	-49.45300830745697	197 196 198 192
198	-18.972510631339397	-49.45280445957184	198 200 153 197
199	-18.97126267008438	-49.453855885505675	199 196 201 200 207
200	-18.972297565445185	-49.453609122276305	200 202 199
201	-18.971414860981632	-49.454585446357726	201 199 203 202 208

Vértice	Latitude	Longitude	Lista de Adjacências
202	-18.97207435326385	-49.45444597148895	202 230 201
203	-18.97149602940334	-49.45543302440643	203 137 231 201 209
204	-18.968878327884827	-49.452096356391905	204 190 205 224
205	-18.96896964374611	-49.45278300189972	205 204 206 194 223
206	-18.969131982931508	-49.45350183391571	206 205 207 196 222
207	-18.969274029589066	-49.45428503894806	207 206 208 199 221
208	-18.969416076125555	-49.45503605747223	208 207 209 201 220
209	-18.969598707208895	-49.45593727970123	209 208 210 215 203
210	-18.96986250730934	-49.45694579029083	210 209 211 214 231
211	-18.97015674539039	-49.45768607997894	211 212 210
212	-18.969821922705677	-49.458083046913146	212 213 217
213	-18.969314614326123	-49.45823325061798	213 216 214
214	-18.969111690541983	-49.4571389093399	214 213 215 210 218
215	-18.968888474094115	-49.45616258525848	215 214 209 219
216	-18.968208676707004	-49.45851220035553	216 218
217	-18.9683913091132	-49.45916665935516	217 216 234 232
218	-18.967985459049938	-49.45746077442169	218 219 214
219	-18.967843411294353	-49.45646299266815	219 220 215
220	-18.967630339434045	-49.45544375324249	220 221 208
221	-18.967447706194164	-49.45464981937408	221 222 207
222	-18.96736653580111	-49.45390952968597	222 223 206
223	-18.967214341207598	-49.45323361301422	223 224 205
224	-18.96704185383364	-49.45246113681793	224 204

Vértice	Latitude	Longitude	Lista de Adjacências
225	-18.970500056970447	-49.45787883916904	225 140 141 211
226	-18.970180454171558	-49.45838845888187	226 225
227	-18.976669352133992	-49.44653939170291	227 178 235
228	-18.979395735593766	-49.449558401676995	228 229
229	-18.97954791906296	-49.44936528262792	229 227
230	-18.971897991224406	-49.45487990436254	230 203
231	-18.970751486368624	-49.45671453532873	231 141 211 210
232	-18.968823717063042	-49.45940747317968	232 226 217
233	-18.968316405645528	-49.45985808429418	233 234 232
234	-18.968001871791323	-49.459557676884515	234 233
235	-18.976508102853757	-49.44571365295832	235 227 187
236	-18.968665257347336	-49.46070088291168	236 14 233
237	-18.968827596829176	-49.461580647468566	237 236
238	-18.969060959557346	-49.46271790409088	238 10 237
239	-18.969304468142777	-49.463833703041075	239 238
240	-18.96949724552063	-49.46467055225372	240 1 239
241	-18.969659584192215	-49.46553958797455	241 240
242	-18.969892945755614	-49.46690215015411	242 8 241