ALGORITMOS HEURÍSTICOS CONSTRUTIVOS APLICADOS AO PROBLEMA DO CAIXEIRO VIAJANTE PARA A DEFINIÇÃO DE ROTAS OTIMIZADAS

Gabriel Altafini Neves da Silva[1], Francisco Assis da Silva[2], Daniela Tereza Ascencio Russi[2], Mário Augusto Pazoti[2], Robson Augusto Siscoutto[2]

[1]Discente da Faculdade de Informática da UNOESTE; [2]Docente da Faculdade de Informática da UNOESTE.

RESUMO

Definir uma rota otimizada, por exemplo, para transporte de cargas com vários pontos de entrega a serem percorridos sem planejamento prévio, pode acarretar um alto custo e tempo demasiado. Este problema pode ser abordado como o Problema do Caixeiro Viajante, que consiste em estabelecer uma única rota que passe em cada vértice de um percurso uma única vez, retornando ao vértice inicial no final do percurso de maneira que o custo seja mínimo. Este trabalho está focado em analisar os algoritmos heurísticos construtivos para resolver o Problema do Caixeiro Viajante, que constroem uma rota através de um conjunto inicial de vértices e modificam esse conjunto utilizando um critério de escolha a cada iteração. Os algoritmos heurísticos utilizados para a otimização de rotas e avaliados foram: vizinho mais próximo, inserção do mais distante, inserção do mais rápido, inserção do mais próximo. Através de um aplicativo móvel definido e implementado neste trabalho, foram obtidas as coordenadas geográficas para os vértices das rotas utilizadas nos experimentos realizados. Os resultados obtidos de cada algoritmo foram comparados entre si para a obtenção do melhor algoritmo na determinação de rota otimizada. A partir dos resultados, observou-se a vantagem do uso do algoritmo de inserção do mais distante.
Palavras-chave: problema do caixeiro viajante; algoritmos heurísticos construtivos; otimização de rotas.

CONSTRUCTIVE HEURISTIC ALGORITHMS APPLIED TO TRAVELING SALESMAN PROBLEM FOR THE DEFINITION OF OPTIMIZED ROUTES

ABSTRACT

Define a optimized route, for example, to transport cargo to various delivery points to be covered without prior planning, can lead to high cost and time. This problem can be named as the Traveling Salesman Problem, which is establish a single route that passes at each vertex of a route once, returning to the initial vertex at the end of the route so that the cost is minimal. This work is focused on analyzing the constructive heuristic algorithms to solve the Traveling Salesman Problem, building a route through an initial set of vertex, together and change this using a criterion of choice at each iteration. The heuristic algorithms used for the optimization of routes and evaluated were: nearest neighbor, furthest insertion, insertion of the fastest, closest insertion. Through a mobile application defined and implemented in this work were obtained geographic coordinates for the vertices of the routes used in the experiments. The results of each algorithm were compared to obtain the best algorithm for determining the optimal route. From the results, it was noted the advantage of using the insertion algorithm the most distant.
Keywords: traveling salesman problem; constructive heuristic algorithms; optimization of routes.

1. INTRODUÇÃO

A popularidade da Internet e a ampla utilização do comércio eletrônico fazem com que empresas e prestadoras de serviço necessitem de uma logística mais eficiente, no que diz respeito ao transporte de cargas entre grandes distâncias e com vários destinos, para serem visitados com uma única viagem. Porém, o grande problema não está somente no meio de transportar, mas sim no custo e no tempo do transporte de cargas para entrega. Considerando custo e tempo, é inevitável que o planejamento da distribuição, não apenas de transporte de produtos, mas também de pessoas sejam cada vez mais complexo.

Segundo Ballou (1998), a logística participa de pelo menos um terço total das despesas de uma empresa. Os custos estão relacionados ao deslocamento para o atendimento de ocorrências e ou solicitações de serviço pelos clientes. Uma eficiente operação logística tem impacto relevante no resultado da empresa. A eficiência é obtida através da determinação de rotas otimizadas.

Segundo Campello e Maculan (1994), o Problema do Caixeiro Viajante é um dos mais tradicionais e conhecidos problemas de programação matemática, sendo que a primeira menção ao mesmo é devida a Hassler Whitney em 1934 em um trabalho na Princeton University.

Dentre as aplicações práticas mais conhecidas do Problema do Caixeiro Viajante, destacam-se o sequenciamento das operações de máquinas em manufatura, otimização de perfurações de furos em placas de circuitos impressos e a maioria dos problemas de roteamento de veículos (NASCIMENTO et al., 2004).

O Problema do Caixeiro Viajante consiste em estabelecer uma única rota que passe em cada vértice de um grafo[1], uma única vez, retornando ao vértice inicial no final do percurso. Este roteiro denominado Hamiltoniano[2], deve ser feito de modo que a distância total percorrida seja mínima. É considerado um problema NP-difícil por Garey e Jonhson (1979), o que significa que possui ordem de complexidade exponencial, o esforço computacional para sua resolução cresce exponencialmente com o tamanho do problema.

Segundo Prestes (2006), como os tempos para otimizar rotas com vários vértices por algoritmos exatos são inviáveis, a opção nesses casos seria o uso de heurísticas. As heurísticas são algoritmos que não tem garantia de encontrar a solução ótima, ou

[1] Grafo: é um conjunto finito $G(V, E)$ onde V é não-vazio e E é um conjunto de pares não ordenados de elementos distintos de V. G é chamado de trivial quando $|V| = 1$. Pode ser visualizado através de representações geométricas, na qual seus vértices correspondem a pontos distintos do plano em posições arbitrárias, enquanto que a cada aresta (v, w) é associada uma linha arbitrária unindo os pontos correspondentes a v, w (SZWARCFITER, 1986).

[2] Roteiro Hamiltoniano: ciclo (em um grafo não-dirigido) onde cada vértice é visitado exatamente uma vez retornando ao ponto de partida (BOAVENTURA NETTO, 1996).

seja, a melhor solução para o problema. Porém, é capaz de retornar uma solução em tempo adequado para as necessidades da aplicação.

A proposta deste trabalho é avaliar a eficiência de diferentes métodos heurísticos para o Problema do Caixeiro Viajante, comparando os resultados dos custos dos caminhos obtidos. Para cada caso de teste, foram utilizados trinta grupos distintos com tamanhos diferentes (10, 20 e 30 vértices). Os resultados obtidos com os experimentos propiciaram a avaliação do melhor algoritmo heurístico entre os selecionados, considerando a distância, como quesito de comparação entre os métodos. Os vértices das rotas dos casos de teste foram determinados através de endereços ou cidades, utilizando a aplicação desenvolvida para dispositivos móveis com Sistema Operacional Android.

As demais seções deste trabalho estão organizadas da seguinte maneira: na seção 2 são apresentados os métodos heurísticos construtivos utilizados neste trabalho; na seção 3 são descritos os serviços do Google Maps API; na seção 4 são apresentadas as aplicações definidas e implementadas neste trabalho; na seção 5 são apresentados os experimentos utilizados para avaliar a eficiência dos métodos construtivos; na seção 6 são demonstrados os resultados obtidos com os experimentos, e por fim, na seção 7 são apresentadas as conclusões e considerações finais do trabalho.

2. MÉTODOS CONSTRUTIVOS

Segundo Bodin et al. (1983), as heurísticas podem ser classificadas como procedimentos de construção de rotas, procedimentos de melhoramento de rotas e procedimentos compostos (construção + melhoramento). Procedimentos de construção de rotas requerem a determinação da matriz de custos para todos os pares de vértices da rota (vértices de um grafo). Procedimentos de construção seguem uma regra que determina o vértice e a posição de sua inserção na rota em formação.

Heurísticas de construção de rotas para o Problema do Caixeiro Viajante são algoritmos que geram um circuito viável partindo de conjunto inicial de vértices, e modificando esse conjunto a cada iteração utilizando um critério de escolha. O processo busca boas soluções a um custo computacional razoável, porém, sem ser capaz de garantir a melhor solução, ou até mesmo, em vários casos, de estabelecer quão perto está da solução viável ou solução ótima (BENEVIDES, 2011).

As seguintes técnicas de construção de rotas foram utilizadas neste trabalho: vizinho mais próximo, inserção do mais distante, inserção do mais próximo e inserção do mais barato.

2.1 Vizinho mais próximo

Segundo Goldbarg e Luna (2000), esta heurística, parte-se do vértice de origem e adiciona-se a cada passo o vértice ainda não visitado (linhas 17 e 18 da Figura 1), cuja distância do último vértice visitado seja mínima (linha 12). O procedimento finaliza quando todos os vértices foram visitados. Ao final é feita a ligação entre o último vértice e o vértice de origem (linha 21). O pseudocódigo do algoritmo está representado na Figura 1.

```
1 HEURÍSTICA vizinho mais próximo;
2 INÍCIO
3      ENTRADA: Rota;
4      SAÍDA : Rotaminimizada;
5      Primeiro_ponto = Rota[1];
6      Rota_minimizada = Rota[1];
7      Ponto_verificar = Rota[1];
8      I = QuantidadePontos(Rota);
9      ENQUANTO I > QuantidadePontos(Rotaminimizada) FAÇA
10        Menor = 10000000;
11        PARA todos os Rota[K] não pertencentes a Rotaminizada FAÇA
12             SE Distância(Ponto_verificar, Rota[K]) < Menor ENTÃO
13                 Menor = Distância(Ponto_verificar, Rota[K]);
14                 Ponto_Menor = Rota[k];
15             FIM-SE;
16        FIM-PARA;
17        Rotaminimizada = Ponto_Menor;
18        Ponto_verificar = Linha_Menor;
19        I = I - 1;
20     FIM-ENQUANTO;
21     Rotaminimizada = Primeiro_ponto;
22 FIM.
```

Figura 1. Pseudocódigo do algoritmo Vizinho mais próximo.

2.2 Inserção do mais distante

A heurística de inserção do mais distante contrapõe a do vizinho mais próximo por inserir os vértices no final da solução e não gera cruzamentos de caminhos (CORDENONSI, 2008). O algoritmo com pseudocódigo representado na Figura 2, inicia com uma sub-rota que contém um vértice de origem e o vértice *k* mais distante do vértice de origem (linhas 8 a 16), para cada vértice que não esteja na rota, é escolhido o vértice *k*, tal que a distância entre o vértice *k* e todos os vértices que estão na rota seja máxima (linhas 19 a 27), o local de inserção do vértice *k*, na sub-rota é dada pelo par de vértices (*i*, *j*) que pertencem a sub-rota, ligando ao vértice *k*, tal que custo (*i*, *k*) +

custo (*k*, *j*) - custo (*i*, *j*), seja mínimo (linhas 29 a 36).

```
1 HEURÍSTICA inserção do mais distante
2 INÍCIO
3       ENTRADA: Rota;
4       SAÍDA : Rotaminimizada;
5       Ponto_verificar = Rota[1];
6       Rota_minimizada = Rota[1];
7       Maior = 0;
8       PARA Rota[k] não pertencentes a Rotaminizada FAÇA
9          SE distância(Ponto_verificar, Rota[K]) > Maior ENTÃO
10             Maior = distância(Ponto_verificar, Rota[K]);
11             Ponto_Maior = Rota[K];
12         FIM-SE;
13      FIM-PARA;
15      Rotaminimizada = Ponto_Maior;
16      Rotaminimizada = Rota[1];
17      Tamanho_Percorrer = Tamanho(Rota)- 2;
18      ENQUANTO Tamanho_Percorrer > 0 FAÇA
19         Maior = 0;
20         PARA todo Rotaminimizada[I] FAÇA
21             PARA Rota[K] não pertencente a Rotaminimizada FAÇA
22                 SE distância(Rotaminimizada[I], Rota[K]) > Maior ENTÃO
23                     Maior = distância(Rotaminimizada[I], Rota[K])
24                     Ponto_Maior = k;
25                 FIM-SE;
26             FIM-PARA;
27         FIM-PARA;
28         Tamanho = 10000000;
29         PARA todo Rotaminimizada[I] FAÇA
30             Tamanho2 = distância(Rotaminimizada[I], Ponto_Maior) + distância(Ponto_Maior,
Rotaminimizada[I+1]) - distância(Rotaminimizada[I], Rotaminimizada[I+1]);
31             SE Tamanho2 < Tamanho ENTÃO
32                 Tamanho = Tamanho2;
33                 Ponto_Inserir = Rotaminimizada[I];
34             FIM-SE;
35         FIM-PARA;
36         InserirPontoApos(Ponto_Inserir,Rotaminimizada,Ponto_Maior);
37         Tamanho_Percorrer = Tamanho_Percorrer - 1;
38      FIM-ENQUANTO;
39 FIM.
```

Figura 2. Pseudocódigo do algoritmo inserção do mais distante.

2.3 Inserção do mais próximo

Segundo Goldbarg e Luna (2000), o algoritmo da inserção do mais próximo (Figura 3) é uma heurística que possui um processo onde três níveis de decisão são envolvidos: decisão de um ciclo inicial onde está contido o vértice de origem e o vértice *k* mais próximo do vértice de origem (linhas 8 a 15), a escolha do vértice a ser inserido na solução tal que a distância entre o vértice *k* e todos os vértices que estão na rota seja mínima (linhas 18 a 26) e a posição de inserção desse novo vértice na solução, que é dada através do par de vértices *i* e *j* que pertencem à solução, ligando *k,* tal que custo

(*i*, *k*) + custo (*k*, *j*) - custo (*i*, *j*), seja mínimo (linhas 28 a 35).

```
1 HEURÍSTICA inserção do mais próximo
2 INÍCIO
3      ENTRADA: Rota;
4      SAÍDA : Rotaminimizada;
5      Ponto_verificar = Rota[1];
6      Rotaminimizada = Rota[1];
7      Menor = 100000;
8      PARA Rota[K] não pertencente a Rotaminizada FAÇA
9        SE distância(Ponto_verificar, Rota[K]) < Menor ENTÃO
10         Menor = distância(Ponto_verificar, Rota[K]);
11         Ponto_Menor = Rota[k];
12       FIM-SE;
13     FIM-PARA;
14     Rotaminimizada = Ponto_Menor;
15     Rotaminimizada = Rota[1];
16     Tamanho_Percorrer = Tamanho(Rota)- 2;
17     ENQUANTO Tamanho_Percorrer > 0 FAÇA
18        Menor = 100000000;
19        PARA todo ponto Rotaminimizada[I] FAÇA
20             PARA Rota[K] não pertencente a Rotaminimizada FAÇA
21                 SE distância(Rotaminimizada[I], Rota[K]) < Menor ENTÃO
22                     Menor = distância(Rotaminimizada[I], Rota[K])
23                     Ponto_Menor = k;
24                 FIM-SE;
25             FIM-PARA;
26        FIM-PARA;
27        Tamanho = 10000000;
28        PARA todo Rotaminimizada[I] FAÇA
29            Tamanho2 = distância(Rotaminimizada[I], Ponto_Menor) + distância(Ponto_Menor,
Rotaminimizada[I+1]) - distância(Rotaminimizada[I], Rotaminimizada[I+1]);
30            SE Tamanho2 < Tamanho ENTÃO
31               Tamanho = Tamanho2;
32               Ponto_Inserir = Rotaminimizada[I];
33            FIM-SE;
35         InserirPontoApos(Ponto_Inserir,Rotaminimizada,Ponto_Menor);
36         Tamanho_Percorrer = Tamanho_Percorrer - 1;
37      FIM-ENQUANTO;
38 FIM.
```

Figura 3. Pseudocódigo do algoritmo inserção do mais próximo.

2.4 Inserção do mais barato

O algoritmo inserção do mais barato (Figura 4) constitui em construir uma rota passo a passo, partindo de rota inicial (linhas 6, 14 e 15) e adicionar a cada passo, o vértice ainda não visitado entre a ligação dos vértices já visitados, cujo custo de inserção seja mais barato: custo (*i*, *k*) + custo (*k*, *j*) - custo (*i*, *j*), seja mínimo (linhas 17 a 32), onde *i* e *j* são os vértices já visitados e *k* o vértice a ser inserido na solução final (CORDENONSI, 2008).

```
1 HEURÍSTICA inserção do mais barato
2 INÍCIO
3       ENTRADA: Rota;
4       SAÍDA : Rotaminimizada;
5       Ponto_verificar = Rota[1];
6       Rotaminimizada = Rota[1];
7       Menor =1000000;
8       PARA Rota[K] não pertencente a Rotaminizada FAÇA
9           SE distância(Ponto_verificar, Rota[K]) < Maior ENTÃO
10             Menor = distância(Ponto_verificar, Rota[K]);
11             Ponto_Menor = Rota[K];
12          FIM-SE;
13      FIM-PARA;
14      Rotaminimizada = Ponto_Menor;
15      Rotaminimizada = Rota[1];
16      Tamanho_Percorrer = Tamanho(Rota)- 2;
17      ENQUANTO Tamanho_Percorrer > 0 FAÇA
18          Menor = 1000000;
19          Tamanho = 10000000;
20          PARA Rota[K] e não pertencente a Rotaminimizada FAÇA
21               PARA todo Rotaminimizada[I] FAÇA
22                 Tamanho2 = distância(Rotaminimizada[I], Rota[k]) + distância(Rota[k],
Rotaminimizada[I+1]) - distância(Rotaminimizada[I], Rotaminimizada[I+1]);
23                 SE Tamanho2 < Tamanho ENTÃO
24                    Tamanho = Tamanho2;
25                    Ponto_Inserir = Rotaminimizada[I];
26                    Ponto_Menor = Rota[K];
27                 FIM-SE;
28               FIM-PARA;
29           FIM-PARA;
30           InserirPontoApos(Ponto_Inserir,Rotaminimizada,Ponto_Menor);
31           Tamanho_Percorrer = Tamanho_Percorrer - 1;
32      FIM-ENQUANTO;
33 FIM.
```

Figura 4. Pseudocódigo do algoritmo inserção do mais barato.

3. RECURSOS DE MAPAS

Para obter os vértices dos casos de testes, foi utilizado um dispositivo móvel, que executa o serviço Google Geocoding API, realizando o processo de geocodificação (conversão de endereços em coordenadas geográficas) por meio de uma solicitação HTTP[3]. Um exemplo de solicitação HTTP para o serviço do Google Geocoding API é mostrado a seguir.

http://maps.googleapis.com/maps/api/geocode/xml?address=Quincas+Vieira+1254,+Presidente+Prudente,+SP&sensor=false

Após todos os vértices obtidos, estes são enviados para um WebService, que é responsável pela geração da matriz de custo entre todos os pares de vértices da rota. Para obter os valores da matriz de custo é utilizado o Google Directions API, um serviço que calcula rotas entre locais usando uma solicitação HTTP, que a partir da informação

[3] HTTP: *Hypertext Transfer Protocol* ,em português Protocolo de Transferência de Hipertexto, é um protocolo de comunicação, utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos (TANENBAUM, 2003).

do vértice de origem e vértice de destino, retorna a distância total e o tempo do percurso. Um exemplo de solicitação HTTP ao serviço Google Directions, com origem na cidade de Presidente Prudente e destino Curitiba é mostrado a seguir.

http://maps.google.com/maps/api/directions/xml?origin=Presidente+Prudente,SP&destination=Curitiba,PR&sensor=false

Para a exibição gráfica da rota solicitada, foi utilizado o Google Static Maps API, que retorna uma imagem (GIF, PNG ou JPEG) em resposta a uma solicitação HTTP. Para cada solicitação, pode-se especificar: a localização do mapa, o tamanho da imagem, o nível de zoom (fator de aproximação ou distanciação no mapa), o tipo mapa e as posições de marcadores nele. Um exemplo de solicitação HTTP ao serviço Google Static Maps, contendo um marcador na rua Quincas Veira, cidade de Presidente Prudente, estado de São Paulo é mostrador a seguir.

http://maps.googleapis.com/maps/api/staticmap?center=Presidente+Prudente,SP&zoom=13&size=600x300&maptype=roadmap&markers=color:blue%7Clabel:S%7CRua+QuincasViera+1254,SP&sensor=false

Os serviços Google Static Maps API, Google Geocoding API e Google Directions API, provêm do recurso Google Maps API, que é um recurso gratuito, para qualquer site ou aplicação.

4. APLICAÇÕES

Neste trabalho foram desenvolvidas três aplicações para auxiliar nos experimentos com os algoritmos heurísticos construtivos: uma aplicação para dispositivos móveis que utilizam o sistema operacional Android 2.2, um WebService e uma aplicação desenvolvida para Desktop.

A aplicação para dispositivos móveis tem como função possibilitar a obtenção dos vértices (endereços ou cidades) para compor a rota a ser minimizada. Os vértices escolhidos são convertidos em coordenadas geográficas, por meio de uma solicitação HTTP para o serviço Google Geoconding. Como resposta a solicitação é retornado um arquivo XML[4] contendo as coordenas geográficas do vértice enviado na solicitação. Ao completar a obtenção da rota, é feita a solicitação de sua otimização ao WebService, através de um arquivo XML contendo as coordenadas geográficas dos vértices da rota a ser otimizada. Ao receber essa solicitação,

[4] XML: Uma linguagem de marcação utilizada na criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais (GOLDBERG, 2009).

o WebService grava em um banco de dados os vértices dessa solicitação. Após realizar esse processo, é obtida a matriz de custo (distância ou tempo entre todos os pares vértices da rota) utilizada para aplicar os algoritmos heurísticos construtivos. O custo entre cada par de vértices obtido através de uma requisição HTTP para o serviço Google Directions, que retorna um arquivo XML, com o tempo e a distância entre os pares de vértices. Cada algoritmo tem seu custo total (distância ou tempo do percurso) e rota otimizada armazenados no banco de dados. O percurso do algoritmo que obteve melhor resultado é armazenado no banco de dados em pares (vértice *i* para vértice *j*, vértice *j* para vértice *k*, vértice *k* para vértice *i*).

A exibição da rota otimizada é solicitada à rota atual, requisitando o próximo vértice a ser visitado, e enviando o primeiro par de vértices da rota otimizada, o qual pode retroceder os pares de vértices ou avançar de acordo com as solicitações realizadas. Também é possível obter rotas anteriores e os seus respectivos pares de vértices.

A Figura 5 apresenta as funcionalidades e troca de informações realizadas entre o WebService, o Google Maps API, o banco de dados e a aplicação para dispositivos móveis.

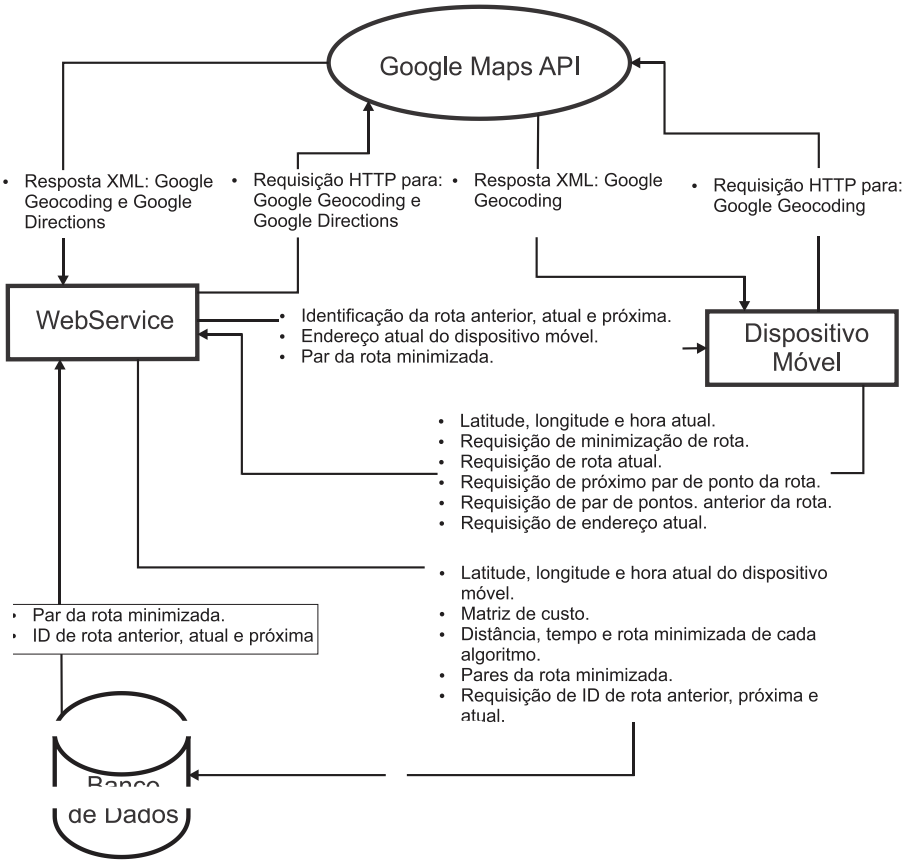

Figura 5. Representação das funcionalidades da aplicação para dispositivo móvel e o WebService.

A aplicação para Desktop utiliza as informações disponíveis no banco de dados para realizar uma análise dos resultados obtidos com execução dos algoritmos heurísticos construtivos e os vértices que compõem a rota otimizada. Para exibir a rota que foi otimizada, é feita uma requisição HTTP ao Google Statics Maps, contendo as coordenadas geográficas de cada vértice da rota, sendo possível exibir no máximo 29 vértices, pois a requisição HTTP é limitada a um número máximo de caracteres (2056). Como resposta à requisição, é retornada uma imagem do mapa com os marcadores de cada vértice pertencentes à rota. Também são exibidos o custo total do percurso (tempo ou distância) e rota minimizada, obtidos pelos algoritmos heurísticos construtivos.

Outra importante funcionalidade da aplicação Desktop é a exibição dos vértices percorridos pelos dispositivos móveis em forma gráfica e seus respectivos endereços, cujas coordenadas geográficas dos vértices estão armazenadas no banco de dados. Há a limitação de 29 vértices para exibição em um mapa retornado pelo Google Statics Maps através de uma requisição HTTP. Para converter coordenadas geográficas em endereço é utilizada a geocodificação reversa disponível no serviço Google Geocoding, efetuada por meio de uma requisição HTTP e como resposta a essa solicitação retorna-se um arquivo XML contendo o endereço desejado. Todas as funcionalidades e comunicações descritas são representadas na Figura 6.

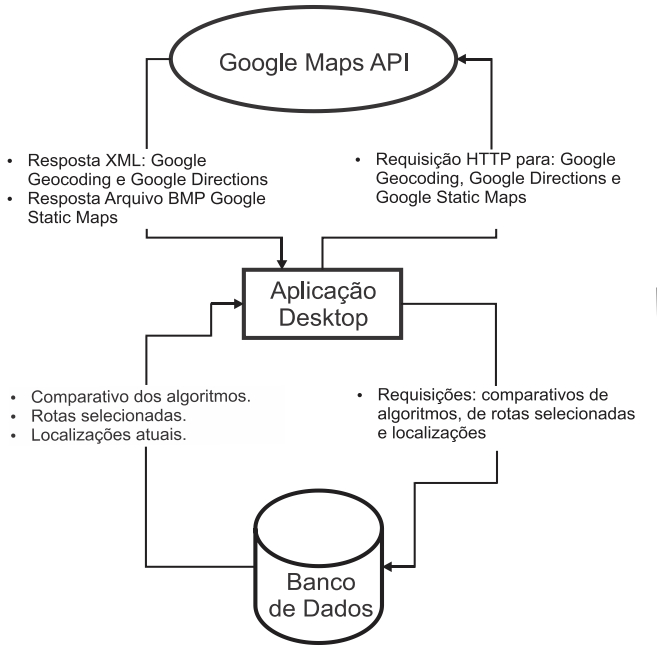

Figura 6. Representação das funcionalidades da aplicação para Desktop.

5. EXPERIMENTOS

Para testar a eficiência e o desempenho dos algoritmos apresentados, foram consideradas três classes de testes distintas, e para cada classe, 10 grupos. As classes foram divididas conforme o número de vértices: grafos de 10, 20 e 30 vértices. Para cada uma destas classes, foram realizados 10 casos de testes com vértices obtidos por meio de endereços ou cidades, através do aplicativo para dispositivos móveis implementado, registrando para cada grafo e algoritmo, o custo da rota e a rota otimizada. Os testes foram realizados, utilizando um microcomputador Intel Core i7 – 2670QM CPU 2.20GHz com 8 GB de memória.

Os vértices representados por coordenadas geográficas, exibidos na Tabela 1, foram obtidos através de endereços da cidade de Curitiba e representa um dos 10 casos de testes de 30 vértices.

Tabela 1. Coordenadas dos vértices de um dos casos de testes.

Vértice	Latitude	Longitude	Vértice	Latitude	Longitude
1	-25.4109410	-49.3601891	16	-25.4436014	-49.2779636
2	-25.419061	-49.3616770	17	-25.4479813	-49.2897537
3	-25.4399511	-49.3490341	18	-25.4597363	-49.2999539
4	-25.4522361	-49.3572542	19	-25.4949503	-49.3461243
5	-25.440341	-49.3748143	20	-25.4980713	-49.3496393
6	-25.4344611	-49.3569041	21	-25.4748872	-49.3634644
7	-25.4514614	-49.2673636	22	-25.4315113	-49.2926347
8	-25.4721215	-49.2782938	23	-25.5468917	-49.2603099
9	-25.4663114	-49.2841637	24	-25.5531318	-49.2575139
10	-25.4454613	-49.3065548	25	-25.4030811	-49.3168637
11	-25.3657112	-49.2558132	26	-25.4177212	-49.2973237
12	-25.3584212	-49.2609432	27	-25.4112813	-49.2806335
13	-25.5401706	-49.2879571	28	-25.4068413	-49.2557433
14	-25.4432214	-49.2809737	29	-25.5562216	-49.3018542
15	-25.4530814	-49.2680186	30	-25.5314917	-49.2685949

As coordenadas geográficas exibidas na Tabela 1, foram utilizadas no Google Earth, para exibição gráfica do caso de teste e para demonstração das rotas e resultados obtidos através da aplicação dos algoritmos construtivos. A imagem obtida através do Google Earth é exibida na Figura 7.

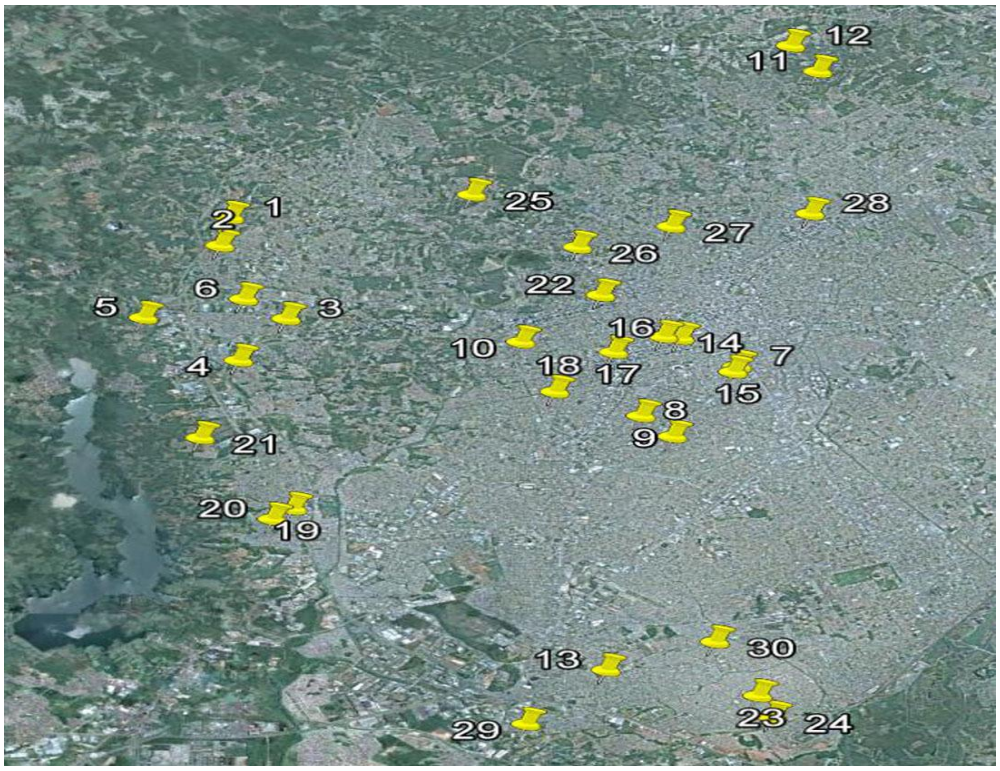

Figura 7. Exemplo de um dos casos de testes com os marcadores dos vértices enumerados referente à Tabela 1.

6. RESULTADOS

A Figura 8 apresenta os resultados da aplicação das heurísticas construtivas em um dos casos de testes. O custo total do percurso dado pelo algoritmo de inserção mais distante (IMD) foi de 112.122 metros, o algoritmo de inserção mais próximo (IMP) foi de 122.770 metros, o algoritmo de inserção do mais barato (IMB) foi de 117.978 metros e o algoritmo do vizinho mais próximo (VMP) foi de 140.180 metros.

A Tabela 2 representa o percurso e a distância total obtida por cada heurística construtiva aplicada no caso do teste de 30 vértices que é representado na Figura 7. A coluna “Heurística” apresenta cada método aplicado. A coluna “Metros” apresenta a distância do percurso, sendo que quanto menor for esse valor, melhor o desempenho do método em comparação com os demais. A coluna “Percurso” contém a sequência de vértices correspondentes ao percurso da rota otimizada.

Tabela 2. Resultados obtidos com o caso de teste exibido na Figura 7.

Heurística	Metros	Percurso
VMP	140180	1,3,10,18,9,8,15,7,14,16,17,22,26,27,28,11,12,25,6,4,21,20,19,29,13,24,23,30,5,2,1
IMB	117978	1,25,26,27,12,11,28,22,14,16,7,15,17,13,29,24,23,30,8,9,18,10,3,6,4,21,20,19,5,2,1
IMP	122770	1,2,6,3,4,5,21,19,20,18,10,17,14,16,7,15,9,8,22,26,27,28,11,12,25,29,13,30,23,24,1
IMD	112122	1,2,5,6,3,4,21,19,20,29,24,23,30,13,8,9,7,15,14,16,28,11,12,27,17,18,10,22,26,25,1

Também foi possível verificar por meio da Figura 8 que conforme citado por Cordenonsi (2008), que o algoritmo da inserção do mais distante (IMD) não gerou cruzamentos na rota, enquanto que os outros três algoritmos apresentaram cruzamentos em suas rotas.

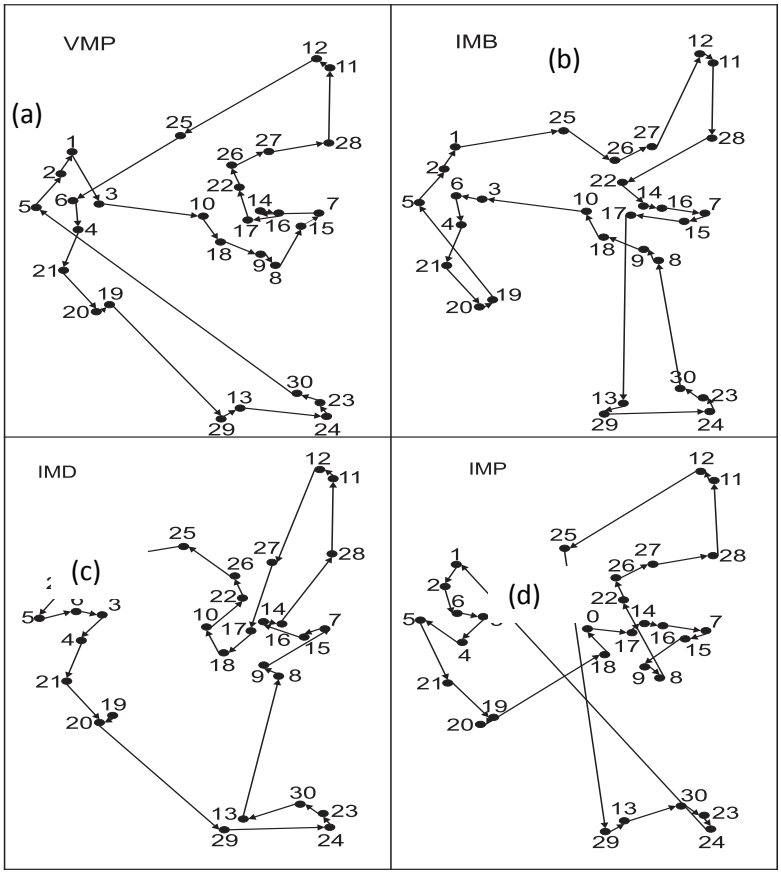

Figura 8. Resultado da aplicação dos algoritmos heurísticos construtivos. Em (a) tem-se o algoritmo do vizinho mias próximo (VMP), em (b) o algoritmo inserção do mais barato (IMB), em (c) o algoritmo de inserção do mais distante (IMD) e em (d) o algoritmo de inserção do mais próximo (IMP).

As Tabelas 3, 4 e 5 apresentam os casos de testes de grafos de 10, 20 e 30 vértices. A coluna “Rota” apresenta o identificador de cada caso de teste, a coluna “Método” apresenta cada método aplicado e a coluna “Metros” apresenta a distância total do percurso. Nesta última coluna o melhor desempenho é representado por um valor menor.

A Tabela 3 apresenta os resultados dos casos de testes dos grafos de 10 vértices. A heurística inserção do mais distante obteve os melhores resultados para oito dos casos de testes, a heurística inserção do mais barato obteve dois melhores resultados juntamente com heurística inserção do mais próximo, que também obteve dois melhores resultados.

Tabela 3. Resultados dos grafos de 10 vértices, onde as linhas em destaque representam os melhores resultados.

Rota	Método	Metros	Rota	Método	Metros
277	VMP	43212	282	VMP	38064
277	IMP	37587	**282**	**IMP**	**36363**
277	IMB	37587	**282**	**IMB**	**36363**
277	**IMD**	**37436**	282	IMD	37898
278	VMP	87248	283	VMP	39943
278	**IMP**	**70351**	283	IMP	35936
278	**IMB**	**70351**	283	IMB	35936
278	IMD	70396	**283**	**IMD**	**35234**
279	VMP	24756	284	VMP	50036
279	IMP	25408	284	IMP	45358
279	IMB	25408	284	IMB	45358
279	**IMD**	**24322**	**284**	**IMD**	**44595**
280	VMP	35378	285	VMP	25712
280	IMP	35140	285	IMP	24513
280	IMB	35140	285	IMB	24513
280	**IMD**	**33855**	**285**	**IMD**	**24395**
281	VMP	74646	286	VMP	61239
281	IMP	72159	286	IMP	61379
281	IMB	72159	286	IMB	61379
281	**IMD**	**70298**	**286**	**IMD**	**57005**

A Tabela 4 apresenta os casos de testes de grafos de 20 vértices. A heurística inserção do mais distante obteve os melhores resultados para oito dos casos de testes, a heurística inserção do mais barato obteve dois melhores resultados e a heurística inserção do mais próximo obteve um melhor resultado.

Tabela 4. Resultados dos grafos de 20 vértices, onde as linhas em destaque representam os melhores resultados.

Rota	Método	Metros	Rota	Método	Metros
287	VMP	51721	292	VMP	105894
287	IMP	51929	292	IMP	94291
287	IMB	48724	292	IMB	94291
287	**IMD**	**45802**	**292**	**IMD**	**93090**
288	VMP	52803	293	VMP	117888
288	IMP	49667	293	IMP	96451
288	IMB	49667	293	IMB	96451
288	**IMD**	**43631**	**293**	**IMD**	**90964**
289	VMP	105667	294	VMP	60011
289	IMP	96620	294	IMP	55036
289	IMB	94802	294	IMB	55036
289	**IMD**	**91907**	**294**	**IMD**	**48811**
290	VMP	82499	295	VMP	63919
290	**IMP**	**67145**	295	IMP	56646
290	**IMB**	**67145**	295	IMB	52588
290	IMD	72559	**295**	**IMD**	**49749**
291	VMP	47946	296	VMP	7569414
291	IMP	39119	296	IMP	6573453
291	**IMB**	**33713**	296	IMB	6573453
291	IMD	36589	**296**	**IMD**	**6464930**

A Tabela 5 apresenta os casos de testes de grafos de 30 vértices. A heurística inserção do mais distante obteve os melhores resultados para oito dos casos de testes, a heurística inserção do mais barato obteve um melhor resultado e a heurística inserção do mais próximo obteve dois melhores resultados.

Tabela 5. Resultados dos grafos de 30 vértices, onde as linhas em destaque representam os melhores resultados.

Rota	Método	Metros	Rota	Método	Metros
297	VMP	1140202	302	VMP	7254103
297	IMP	1133363	302	IMP	6832676
297	IMB	1133363	302	IMB	6657871
297	**IMD**	**1023082**	**302**	**IMD**	**6607707**
298	VMP	8368396	303	VMP	8318090
298	IMP	8196330	**303**	**IMP**	**7790783**
298	IMB	8250163	303	IMB	8199030
298	**IMD**	**7823341**	303	IMD	7811979
299	VMP	8382692	304	VMP	140180
299	IMP	8217722	304	IMP	122770
299	IMB	8219744	304	IMB	117978
299	**IMD**	**7848335**	**304**	**IMD**	**112122**
300	VMP	8033534	305	VMP	119057
300	IMP	7396782	**305**	**IMP**	**112160**
300	IMB	7396782	**305**	**IMB**	**112160**
300	**IMD**	**7358841**	305	IMD	114548
301	VMP	8239387	306	VMP	1060014
301	IMP	7361112	306	IMP	1030514
301	IMB	7295477	306	IMB	1060014
301	**IMD**	**6939587**	**306**	**IMD**	**1000013**

7. CONCLUSÃO

O objetivo deste trabalho foi análise e aplicação das heurísticas construtivas para a construção de rotas, comparando os resultados obtidos dos algoritmos propostos para chegar à conclusão do melhor algoritmo. Este trabalho mostra como é possível a minimização de custos, pois há uma redução significativa na distância ou tempo entre cada um dos algoritmos. Em relação à comparação de desempenho entre as heurísticas, a inserção do mais distante se destacou com as classes de testes apresentadas. Portanto para grafos de 10, 20 e 30 vértices, pode-se concluir que, o melhor método heurístico construtivo é a inserção do mais distante.

REFERÊNCIAS

BALLOU, R **Business logistics management:** planning, organizing and controlling the supply chain. 4. ed. Londres: Prentice Hallm, 1998. 696 p.

BENEVIDES, P.F. **Aplicação de heurísticas e meta-heurísticas para o problema do caixeiro viajante em um problema real de roteirização de veículos**. 2011. Dissertação (Mestrado) – Universidade Federal do Paraná, 2011.

BOAVENTURA NETTO, P. O. **Grafos:** teoria, modelos, algoritmos. São Paulo: Edgard Blücher, 1996.

BODIN, L. et al. Routing and scheduling of vehicles and crews – the state of the art. **Computers & Operations Research**, v. 10, n. 2, 1983.

CAMPELLO, C. R.; MACULAN, N. **Algoritmos e heurísticas**. Niterói, RJ: Editora da Universidade Federal Fluminense, 1994.

CORDENONSI, A. Z. **Ambientes, objetos e dialogicidade:** uma estratégia de ensino superior em heurísticas e meta-heurísticas. 2008. Tese (Doutorado) - Universidade Federal do Rio Grande do Sul, 2008.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability:** a guide to the theory of NP-completeness. New York: W. H. Freeman, 1979.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear:** modelos e algoritmos. Rio de Janeiro: Campus, 2000.

GOLDBERG, K. H. **Guia prático visual XML**. Rio de Janeiro: Alta Books, 2009.

NASCIMENTO, D. B. et al. Análise comparativa de algoritmos heurísticos para resolução do problema do caixeiro-viajante em grafos não clusterizados. In: Encontro Nacional de Engenharia de Produção, 24.. **Anais...** Florianópolis, SC, Brasil, 03 a 05 de nov de 2004.

PRESTES, A. N. **Uma Análise experimental de abordagens heurísticas aplicadas ao problema do caixeiro viajante.** Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte, 2006.

SZWARCFITER, J. L. **Grafos e algoritmos computacionais.** 2. ed. Rio de Janeiro: Campus, 1986.

TANENBAUM, A. S. **Redes de computadores**. 4. ed. Rio de Janeiro: Campus, 2003.