SISTEMA DE ROTEIRIZAÇÃO URBANA

EVANDRO HARRISON HOFFMANN[1], KELVIN WILLIAM ZIMMERMANN AEBI[2], SILVIO BORTOLETO[3]

UnicenP – Centro Universitário Positivo
Bacharelado em Sistemas de Informação
[1]ehh@unicenp.edu.br, [2]kelvin@unicenp.edu.br, [3]silvio.bortoleto@unicenp.edu.br

ABSTRACT

With the high growth of the cities, the transport ways become each time more complex, due to the great varieties of ways to get to a determinate local. An urban routing system is a tool each time more essential on the individuals and organizations days. There are applications that calculate the shortest route between two or more points (addresses), can be consulted several transport ways, those are walking, by particular vehicle or by subway. This article has as objective demonstrate the built of a public application that calculates a route between two distinct points inside a city, aiming not only through particular vehicles, thus as public transportation, and routes on foot associated with collective transportation, turning possible the comparison between them. Presenting some shortest path algorithms with the utilization of a developed tool for Floyd algorithm tests execution (algorithm that calculates the shortest path).
Key-words: Urban Routing System, Shortest Path, Floyd-Warshall, Public Routing System.

RESUMO

Com o elevado crescimento das cidades, os meios de transporte tornam-se cada vez mais complexos, devido a grande variedade de maneiras de se chegar a um determinado local. Um sistema de roteirização urbana é uma ferramenta cada vez mais essencial no dia-a-dia dos indivíduos e das organizações. Existem aplicativos que calculam a rota mais curta entre dois ou mais pontos (endereços), sendo que podem ser consultados diversos meios de transporte, tais como caminhando, de automóvel ou transportes coletivos. Este artigo objetiva demonstrar a construção de um aplicativo que será disponibilizado ao público e que calcule uma rota entre dois pontos distintos dentro de uma cidade, visando não somente através de veículos particulares, mas para atender o transporte público e rotas a pé associadas com transporte coletivo, tornando assim possível a comparação entre elas. É apresentado alguns algoritmos de caminhos mínimos com a utilização de uma ferramenta desenvolvida para execução de testes com o algoritmo de Floyd.
Palavras-chave: Sistema de Roteirização Urbana, Menor Caminho, Floyd-Warshall, Sistema de Roteirização Pública.

1. INTRODUÇÃO

Neste documento apresenta-se um estudo de alguns dos principais problemas de roteamento em grafos bem como o desenvolvimento e a implementação em um Software gráfico e prático, utilizando algoritmos que os solucionem em tempo hábil a roteirização urbana.

Descreve-se também o desenvolvimento de um aplicativo que utiliza o algoritmo de Floyd-Warshall para execução de testes possibilitando a utilização do mesmo com mapas reais.

2. OBJETIVOS

Este artigo tem como objetivo demonstrar a construção de um aplicativo de roteirização urbana para uso público, possibilitando aos usuários o utilizarem através da Internet e através de outros meios eletrônicos, tais como terminais de acesso público, palms, celulares, entre outros.

Neste aplicativo serão calculadas rotas para veículos particulares, transporte coletivo e também rotas a pé.

2.1. Rotas a pé

Compreende a distância percorrida do ponto de origem ao ponto de destino sem considerar os sentidos de ruas e possíveis obstáculos aos veículos, tais como caneletas, canteiros, entre outros.

2.2. Rotas com veículos particulares

Estão compreendidas as rotas que consideram sentidos de ruas e possíveis obstáculos a veículos.

2.3. Rotas com transporte coletivo

Compreende as rotas que levam em consideração os pontos de ônibus da região em que foi implementado, assim uma rota com transporte público compreende a ida a pé do local de origem até o ponto de ônibus mais próximo (considerando que este ônibus chegue ao destino especificado) e a rota que o ônibus irá fazer até chegar ao ponto de ônibus mais próximo do destino e o percurso a pé deste ponto de ônibus até o local desejado.

3. ROTEIRIZAÇÃO URBANA

A roteirização urbana com o foco em grandes cidades é de grande importância, pois os indivíduos necessitam de uma fonte confiável e rápida de acessar esse tipo de informação.

Segundo CUNHA, o termo *roteirização*, embora não encontrado nos dicionários de língua portuguesa, tem como equivalente ao inglês *"routing"* (ou *"routeing"*) para designar o processo para a determinação de um ou mais roteiros ou seqüências de paradas a serem cumpridos, com o objetivo de visitar um conjunto de pontos distintos geograficamente e pré-determinados.

A roteirização segundo ASSAD (1988) citado por CUNHA "consiste em uma das histórias de grande sucesso da Pesquisa Operacional nas últimas décadas, que pode ser medido pelo expressivo número de artigos que vêm sendo publicados ao longo dos anos na literatura especializada, incluindo os anais de congressos da ANPET (Associação Nacional de Pesquisa e Ensino em Transportes)".

Fazem parte da roteirização urbana os meios de transporte: a pé, que considera o caminho direto, do ponto de origem ao ponto de destino, sem restrições; de carro, que tem como restrição os sentidos das ruas e caneletas; e ônibus, que considera as linhas de ônibus disponíveis à população.

3.1. Caixeiro Viajante

Problemas de roteirização são muitas vezes definidos como problema de um caixeiro viajante, que foi o primeiro problema de roteirização a ser estudado (no inglês *"traveling salesman problem"* ou TSP), que consiste em encontrar o roteiro ou sequência de cidades a serem visitadas, minimizando a distância total percorrida e assegurando que cada cidade seja visitada exatamente uma vez. Desde então, novas restrições vêm sendo incorporadas ao problema do caixeiro viajante, de modo a melhor representar os diferentes impasses que envolvem roteiros de pessoas e veículos, entre as quais: restrições de horário de atendimento (conhecidas na literatura como janelas de tempo ou janelas horárias); capacidades dos veículos; duração máxima dos roteiros dos veículos (tempo ou distância); restrições de tipos de veículos que podem atender determinados clientes.

Segundo as pesquisas de CUNHA, os problemas de roteirização sob a ótica de otimização que incluem o caso particular do caixeiro viajante, pertencem à categoria conhecida como *NP*-difícil (do inglês "*NP-hard*"), o que significa que possuem ordem de complexidade exponencial. Em outras palavras, o esforço computacional para a sua resolução cresce exponencialmente com o tamanho do problema (dado pelo número de pontos a serem atendidos).

Este problema também pode abranger indivíduos, que precisam localizar um determinado local, partindo do ponto em que está e chegar da forma mais rápida e de menos custo.

4. MENOR CAMINHO

O problema do menor caminho é um subproblema do problema do Caixeiro Viajante, pois trata de calcular o menor percurso entre dois ou mais vértices (nós) de um grafo. Neste caso, um grafo pode representar uma malha rodoviária, distâncias geográficas e demais correlatas.

O menor caminho pode não somente representar caminhos de distâncias mínimas, que é o mais comumente entendido, mas também podem representar caminhos de tempos mínimos, custos mínimos, entre outras formas de medição de percurso. Isto pode ser afirmado uma vez que na implementação de algoritmos de caminhos mínimos é possível utilizar qualquer parâmetro (assim como as distâncias), uma vez que estes dados estejam disponíveis.

Segundo SAMPAIO, existe uma motivação muito mais importante para se começar a estudá-los do que para simplesmente problemas de motoristas, a determinação dos

menores caminhos aparece constante e consistentemente como um subproblema da complexidade em grafos.

Existem diversos algoritmos já implementados que calculam os caminhos mínimos entre dois ou mais pontos distintos, entre eles estão o algoritmo de Dijkstra e o algoritmo de Floyd.

4.1 Algoritmo de Djikstra

O algoritmo de Dijkstra, segundo SAMPAIO, tem como objetivo obter o menor caminho entre um dado vértice fixo e todos os demais vértices do grafo (como por exemplo, saber a distância mínima entre uma loja e todos os seus clientes). Consiste basicamente em fazer uma visita por todos os nós do grafo, iniciando no nó fixo dado e encontrando sucessivamente o nó mais próximo, o segundo mais próximo, o terceiro mais próximo e assim por diante, um por vez, até que todos os nós do grafo tenham sido visitados.

4.2 Algoritmo de Floyd-Warshall

O Floyd-Warshall calcula os caminhos mínimos de todas as origens para todos os destinos em um grafo valorado. O algoritmo admite grafos com laços e são também calculados os valores de caminhos começando e terminando no mesmo vértice (usando laços ou não). Segundo SAMPAIO, "a idéia geral desse algoritmo é atualizar a matriz de menores distâncias n vezes (onde n é o número de nós do grafo) procurando na K-ésima interação por melhores distâncias entre pares de nós que passem pelo vértice K".

Abaixo está demonstrado o algoritmo de FLOYD-WARSHALL (Quadro 1).

```
Algoritmo FLOYD-WARSHALL
    C[n,n] ← C {é a matriz de custos}
    Gerar K, a partir de C (trocando 0 por ∞)
    Para Q de 1 até N faça
        Para S de 1 até N faça
            Para T de 1 até N faça
                K Se K[S][Q] + K[Q][T] < K[S][T] então
                    K[S][T] ← K[S][Q] + K[Q][T]
                Fim {Se}
            Fim {Para}
        Fim {Para}
    Fim {Para}
Fim {Algoritmo}
```

Quadro 1 - Algorítmo de Floyd-Warshall

A entrada para o algoritmo é a matriz K_0, obtida a partir da matriz de custos do grafo, efetuando-se a mudança de trocar ZERO por INFINITO. Um exemplo de implementação pode ser observado no Quadro 3.

A cada passo do algoritmo, uma nova matriz K_n (n = 1,2,...) é calculada. A última matriz é a resposta dos caminhos mínimos. Então no exemplo do Quadro 3 a resposta final é a matriz K_f, indicando que os caminhos mínimos estão indicados por linha – coluna, ou seja, referenciando as linhas e colunas como 1, 2, 3, 4 e 5, assim o valor do caminho mínimo de 1-5 é 2, 2-1 é 3 e assim por diante. Esses resultados são de sentido único, ou

seja, não passam pelo mesmo caminho para voltar, como pode ser observado no exemplo que custo do caminho de 1-2 é diferente do custo do caminho 2-1. Os valores ZERO (0) significam que não há rota possível entre os dois pontos, como pode ser visualizado na Figura 1.

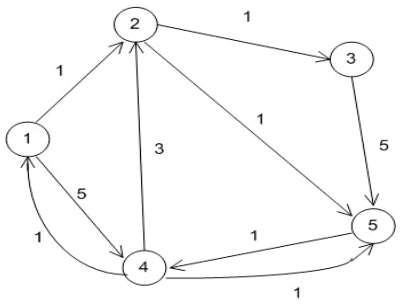


Figura 1 - Representação Gráfica do Exemplo do algorítmo de Floyd-Warshall.

Foi utilizado para implementação deste artigo o algoritmo de Floyd-Warshall modificado (Introdução à Teoria dos Grafos). Sua modificação dá-se a uma otimização do algoritmo original.

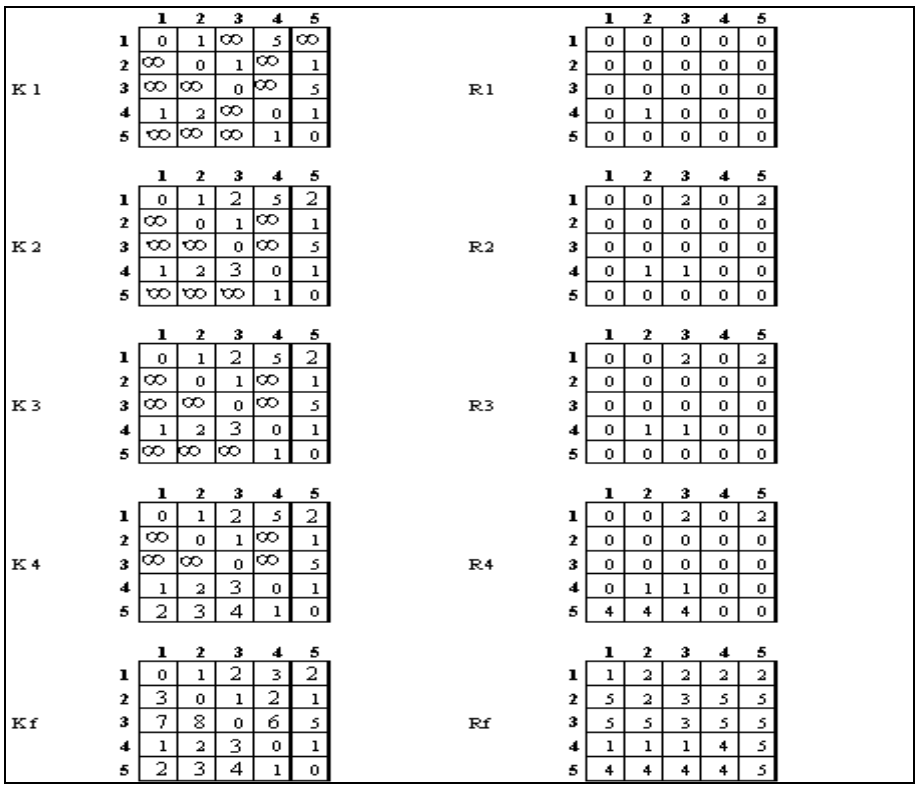
4.3. Algoritmo de Floyd-Warshall Modificado

Este algorítmo cria 2 matrizes, a K_n, que é a matriz de menores custos de ligação entre dois pontos (conhecida no algoritmo comum de Floyd-Warshall), e a matriz R_n, que indica as rotas, ou seja, mostra a rota ponto a ponto. O algoritmo pode ser observado no Quadro 3. Fonte: Rabuske (1992).

```
Algoritmo FLOYD-MOD
    C[n,n] ← matriz de custos
    R[n,n] ← 0
    Gerar K, a partir de C (trocando 0 por ∞ e ponha 0 em nós que acessam a si mesmos,
exemplo 1-1, 2-2, etc.)
    Para Q de 1 até N faça
        Para S de 1 até N faça
            Para T de 1 até N faça
                Se K[S][Q] + K[Q][T] < K[S][T] então
                    K[S][T] ← K[S][Q] + K[Q][T]
                    Se R[S][Q] = 0 então
                        R[S][T] ← Q
                    Senão
                        R[S][T] ← R[S][Q]
                    Fim-Se
                Fim-Se
            Fim {Para}
        Fim {Para}
    Fim {Para}
Fim {Algoritmo}
```

Quadro 2 – Algoritmo de Floyd-Warshall Modificado (em vermelho)

Observe no Quadro 3 um exemplo, com o grafo da Figura 1, utilizando o algoritmo de Floyd-Warshall Modificado.



K1	1	2	3	4	5
1	0	1	∞	5	∞
2	∞	0	1	∞	1
3	∞	∞	0	∞	5
4	1	2	∞	0	1
5	∞	∞	∞	1	0

R1	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	0	0	0
5	0	0	0	0	0

K2	1	2	3	4	5
1	0	1	2	5	2
2	∞	0	1	∞	1
3	∞	∞	0	∞	5
4	1	2	3	0	1
5	∞	∞	∞	1	0

R2	1	2	3	4	5
1	0	0	2	0	2
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	1	0	0
5	0	0	0	0	0

K3	1	2	3	4	5
1	0	1	2	5	2
2	∞	0	1	∞	1
3	∞	∞	0	∞	5
4	1	2	3	0	1
5	∞	∞	∞	1	0

R3	1	2	3	4	5
1	0	0	2	0	2
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	1	0	0
5	0	0	0	0	0

K4	1	2	3	4	5
1	0	1	2	5	2
2	∞	0	1	∞	1
3	∞	∞	0	∞	5
4	1	2	3	0	1
5	2	3	4	1	0

R4	1	2	3	4	5
1	0	0	2	0	2
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	1	0	0
5	4	4	4	0	0

Kf	1	2	3	4	5
1	0	1	2	3	2
2	3	0	1	2	1
3	7	8	0	6	5
4	1	2	3	0	1
5	2	3	4	1	0

Rf	1	2	3	4	5
1	1	2	2	2	2
2	5	2	3	5	5
3	5	5	3	5	5
4	1	1	1	4	5
5	4	4	4	4	5

Quadro 3 – Exemplo de implementação do algoritmo de Floyd-Warshall Modificado

Como pode ser observado no exemplo do Quadro 4, a seqüência de pontos de caminho mínimo é dada pela matriz Rf, que indica que, por exemplo, o caminho do nó 2-1 é feito pela seqüência 2-5-4-1. Esta seqüência de pontos foi encontrada da seguinte forma:

- R[2,1]=5, então é preciso ir de 2 para 5, pagando 5
- R[5,1]=4, então é preciso ir de 5 para 4, pagando 4
- R[4,1]=1, então chega-se em 1, pagando 1.

Como observado, foi-se de 2-5-4-1, pagando respectivamente 5+4+1=10, então o custo mínimo de 2-1 é 10.

Como teste, foi implementado o algoritmo de Floyd-Warshall em Java. No programa foram inseridos os pontos (nós) para calcular o menor caminho. Observe na figura 2.

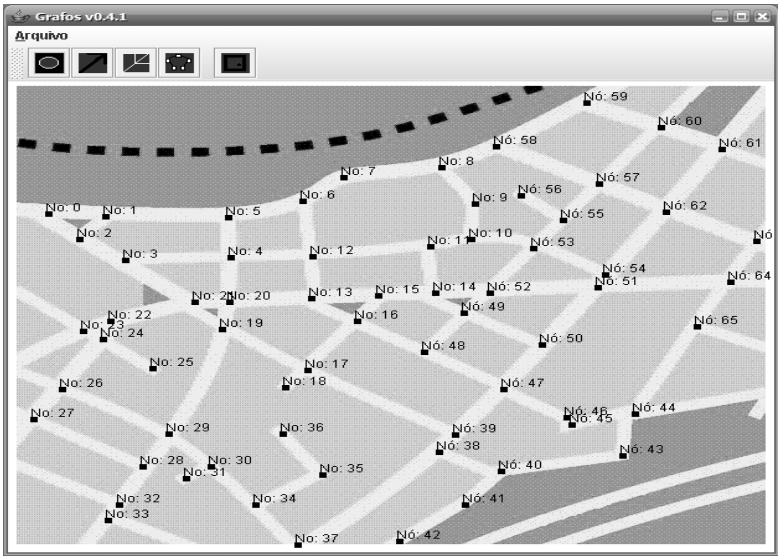


Figura 2 - Pontos (nós) definidos no mapa.

Os pontos são colocados de tal forma que sempre forme uma reta entre um ponto e seu vizinho, ou seja, as retas devem estar dentro das ruas.

Após a inclusão dos pontos, são definidas as rotas entre os mesmos (Figura 3).

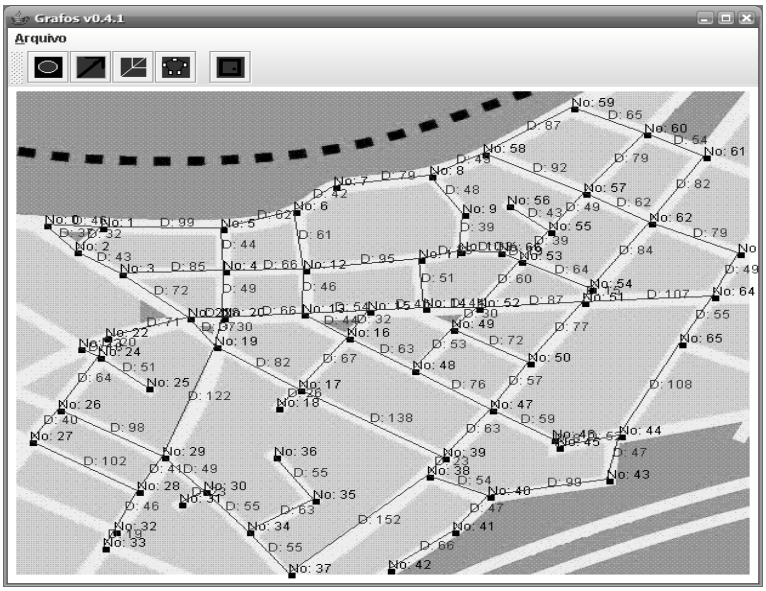


Figura 3 – Caminhos (arestas) com as distâncias entre os pontos.

Ao definir a rota entre dois pontos, o programa automaticamente calcula a distância em pixels entre ambos os pontos e marca, a distância calculada no meio da linha que conecta os dois pontos.

Após feito isso, é possível verificar a rota de menor distância entre dois pontos selecionados. Neste exemplo é verificada a rota entre o nó 0 e o nó 65 (Figura 4).

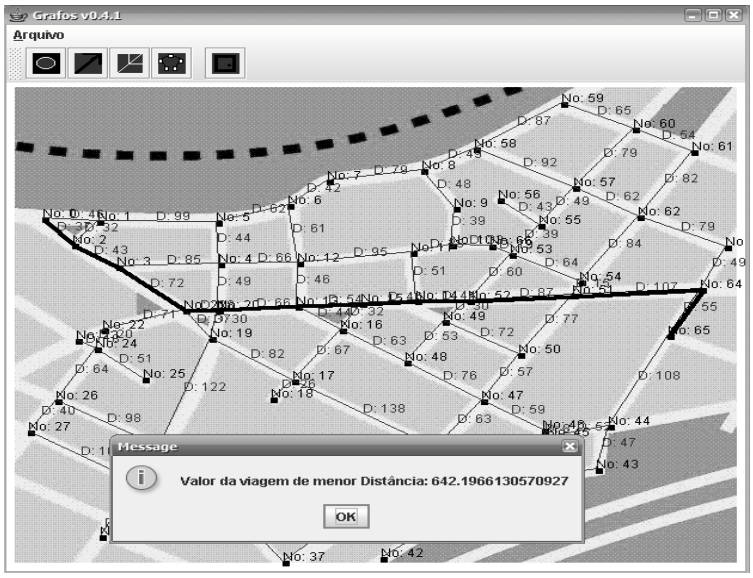


Figura 4 – Rota de menor caminho traçada entre os NÓs 0 e 65.

O programa traça a rota de menor caminho, destacando-a, e mostra o valor percorrido do nó 0 ao nó 65, que foi de 642.19 pixels.

5. IMPLEMENTAÇÃO

Utilizando o software acima descrito, porém modificado para um mapa real, calcularam-se as rotas a pé, com veículo particular e com transporte coletivo. O mapa utilizado foi o demonstrado na Figura 5, onde pode ser observado que as ruas com setas indicam mão única, e as ruas sem setas indicam mão dupla.

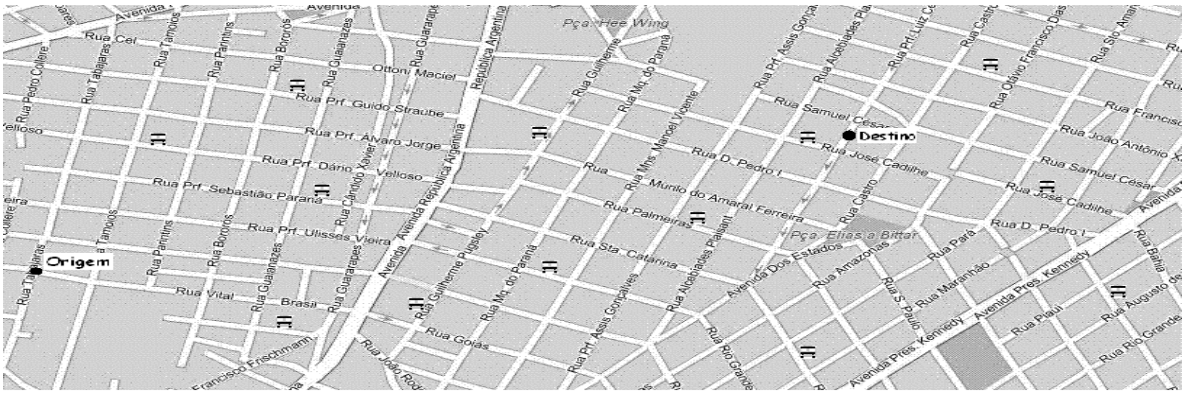


Figura 5 – Mapa utilizado para demonstração de implementação.

Através deste mapa, foi calculada a rota saindo da Rua Tabajaras (Ponto de Origem no mapa) chegando à Rua Prf. Luiz Celso (Ponto de Destino no mapa), fazendo a comparação entre os três tipos de rotas comentados anteriormente.

Primeiramente é demonstrada a rota a pé, que percorre um total de 2199 metros na sua rota de menor distância. Sua rota é demonstrada na Figura 6.

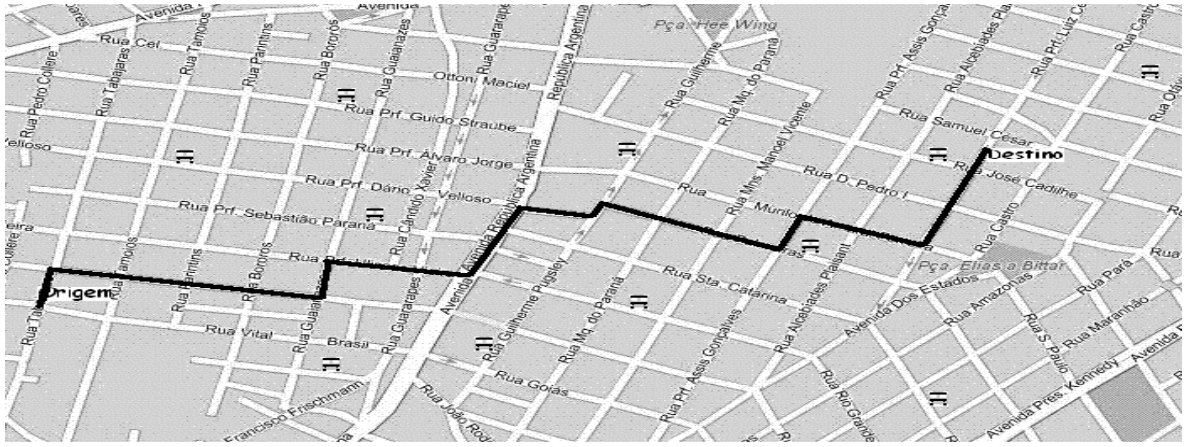


Figura 6 – Rota a pé.

Logo após, é apresentada a rota utilizando um veículo particular, que percorreu em seu percurso total 2340 metros, como pode ser observada na Figura 7.

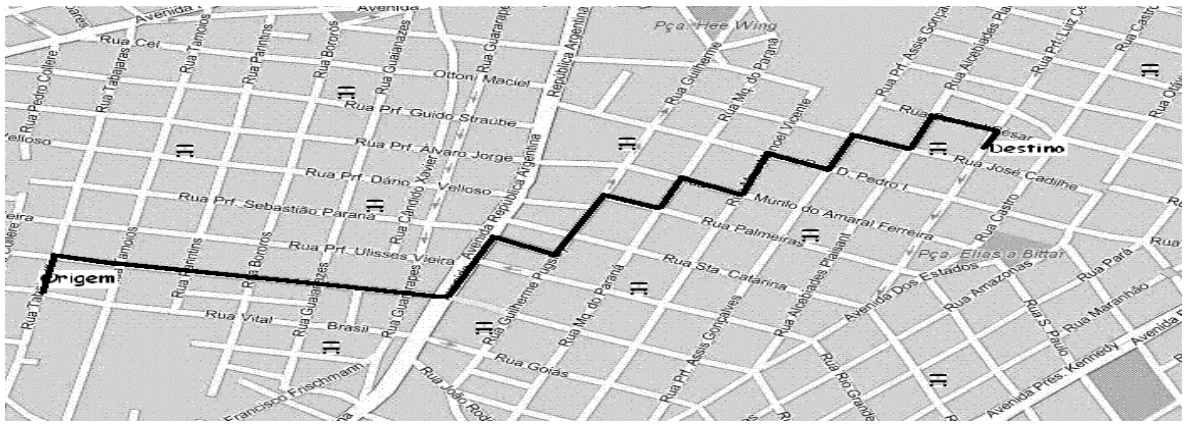


Figura 6 – Rota com veículo particular.

E finalmente é demonstrada a rota utilizando o transporte público, que percorreu um total de 2535 metros, sendo que foram percorridos 600 metros do local de origem até o ponto de ônibus mais próximo, 1809 metros com o ônibus e 126 metros do ponto de ônibus desembarcado até o local de destino, como observado na Figura 7.

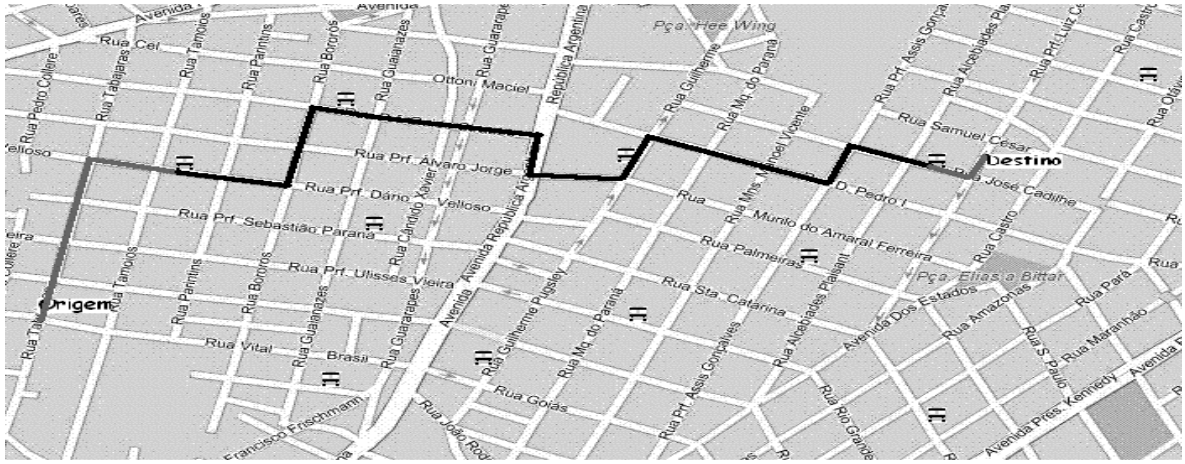


Figura 6 – Rota com transporte público (sendo a linha destacada mais clara a rota a pé e a linha destacada mais escura a rota de ônibus).

6. CONCLUSÃO

Com a utilização do algoritmo de Floyd na busca da rota de menor caminho (não somente utilizando a distância, mas também qualquer outro parâmetro que pode ser inserido no contexto, como por exemplo, o tempo), este aplicativo é de grande utilidade no dia-a-dia dos indivíduos, pois os auxiliará a se locomoverem dentro do perímetro urbano em que o aplicativo foi implementado.

A necessidade de auxílio na decisão ou na descoberta da melhor rota independentemente do meio utilizado é cada vez maior e então os indivíduos precisam de uma ferramenta para executar este tipo de tarefa.

Como pôde ser observado na implementação do aplicativo de roteirização urbana, uma rota a pé pode ser mais próxima do que uma rota com um veículo motorizado, mas isto não significa que é a mais vantajosa, pois o tempo de viajem do ponto de origem ao ponto de destino depende da velocidade em que o corpo está se locomovendo.

Em um perímetro urbano existem diversos fatores que podem influenciar na escolha da melhor rota, estes são tempo, tráfego, distância, entre outros e estes fatores também podem ser implementados no aplicativo demonstrado, assim tornando-o mais flexível.

Sendo que este aplicativo pode ser utilizado com outros fatores de influência em uma rota, então também é possível combinar os mesmos, assim adquirindo uma rota mais confiável, que não dependa de um só fator, mas de vários.

Com esta combinação "multi-fatores" seria possível saber com uma precisão muito maior qual é o meio de transporte mais vantajoso para uma determinada rota.

7. REFERÊNCIAS

[1] ASSOCIAÇÃO NACIONAL DE PESQUISA E ENSINO EM TRANSPORTE. **ANPET.** Disponível em: <http://www.anpet.org.br/anpet/index.php>. Acesso em 01/04/2006.

[2] CUNHA, C. B. **Aspectos Práticos Da Aplicação De Modelos De Roteirização De Veículos A Problemas Reais.** Disponível em: <http://www.ptr.usp.br/docentes/cbcunha/files/roteirizacao_aspectos_praticos_CBC.pdf >. Acesso em 15/03/2006.

[3] DEITEL, H.M.; DEITEL, P.J. **Java Como Programar.** São Paulo: Prentice Hall, 2000. 3ª. Edição.

[4] ESTudo e implementação de algoritmos de roteamento sobre grafos em um sistema de informações geográficas. Disponível em: <http://arxiv.org/ftp/cs/papers/0505/0505031.pdf>. Acesso em 10/10/2005.

[5] ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE. **ArcGis**: GIS and mapping software. Disponível em: <http://www.esri.com/software/arcgis/index.html>. Acesso em: 10/02/2006.

[6] FURTADO, A. L. **Teoria dos grafos: algoritmos**. Rio de Janeiro: Livros Técnicos e Científicos, c1973.

[7] GEOSERVER. **Open Gateway for Spatial Data.** Disponível em: <http://docs.codehaus.org/display/GEOS/Home>. Acesso em: 06/06/2006.

[8] GERSTING, J.L**. Fundamentos matemáticos para a ciência da computação**. Rio de Janeiro: LTC, 2001.

[9] GOODRICH, M.T. **Estruturas de Dados e algoritmos em Java.** Porto Alegre: BookMan, 2002. 2ª. Edição.

[10] LOY, M. **Java Swing.** Beijing (China): O'Reilly, 2003. 2ª. Edição.

[11] MAPSOLUTE GMBH. **Map24**. Disponível em: <http://www.br.map24.com>. Acesso em: 20/10/2005.

[12] NETTO, P. O. B. **Grafos: teoria, modelos, algoritmos**. 2.ed. rev. e ampl, São Paulo: Edgard Blücher, 2001.

[13] NIEMEYER, P. **Aprendendo Java 2 SDK.** Rio de Janeiro: Campus, 2000.

[14] ORMSBY, T.(CO-AUTOR). **Getting to know ArcGIS desktop: basics of ArcView, ArcEditor, and ArcInfo.** Redlands, CA: ESRI Press, 2004.

[15] PROblema do caixeiro viajante. Disponível em: <http://www.mat.ufrgs.br/~portosil/caixeiro.html>. Acesso em 25/10/2005.

[16] PUBLICAR DO BRASIL LISTAS TELEFÔNICAS LTDA. **Lista on-line**. Disponível em: <http://www.listaonline.com.br>. Acesso em: 12/03/2006.

[17] RABUSKE, M. A. **Introdução à teoria dos grafos**. Florianópolis: UFSC, 1992.

[18] SAMPAIO, R. M.: YANASSE, H. H. **Estudo e Implementação de Algoritmos de Roteamento sobre Grafos em um Sistema de Informações Geográficas.** Disponível em: <http://www.ptr.usp.br/docentes/cbcunha/files/roteirizacao_aspectos_praticos_CBC.pdf >. Acesso em 15/03/2006.

[19] SCIELO BRAZIL. **Integração de modelos de localização a sistemas de informações geográficas.** Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2001000200006>. Acesso em: 15/02/2006

[20] SEDGEWICK, R. **Algorithms**. Reading, Mass: Addison-Wesley, 1988.

[21] YUSOF, Y. B. **Floydfloyd--Warshall Shortest Path Warshall Shortest Path Model For Mesh Networkmodel For Mesh Network.** Disponível em: <http://institut.fs.utm.my/~ss/ssu4904/yuhani.pdf>. Acesso em 05/05/2005.

[22] WIKIPEDIA, **Floyd-Warshall algorithm**. Disponível em: <http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm>. Acesso em 10/05/2006.