**Problema 4: Roteamento de Veículos**

**Daniel Cavalcante Dourado**

**Roberto Maia Rodrigues Júnior**

Engenharia de computação – Universidade Estadual de Feira de Santana (UEFS)

Av. Transnordestina, s/n - Novo Horizonte, Feira de Santana - BA, 44036-900

Feira de Santana – BA – Brasil

[daniel10dourado@gmail.com](mailto:daniel10dourado@gmail.com)

romaijr5@gmail.com

1.**Introdução**

Além de ser um entroncamento de diversas estradas importantes da Bahia, a cidade de Feira de Santana é considerado um dos pontos de comércio mais influentes do Norte/Nordeste. Levando em consideração isso e conhecendo os elevados custos para a realização do transportes de mercadorias, a empresa EntregaFácil, que trabalha nesse ramo de transporte de mercadorias, vem buscando meios de cortar gastos e otimizar ao máximo as distancias percorridas.

Por já ter trabalhado com a equipe de alunos da matéria EXA863 2018.2 da UEFS, a empresa espera um Software que possa auxilia-los a escolher a menor rota possível entre a sede da empresa, Feira de Santana, e o destino da entrega, fornecendo dados importantes, como a distancia total, as cidades por onde a mercadoria passou e o custo total de combustível gasto pelo veiculo transportador.

Por ser uma empresa nova ainda, ela atende apenas trinta cidades da Bahia, essas cidades são passadas ao programa por meio de um arquivo de entrada, no qual, para cada cidade deverá estar presentes suas coordenadas(X,Y), além desse arquivo, existirá outro para informar a distância entre cidades vizinhas.

Independente da entrega, a origem do percurso é em Feira de Santana, o usuário deverá selecionar o destino da entrega e, caso necessário, selecionar as cidades que deverão estar no roteiro do transporte, após a entrega, o transporte deve voltar para a sede da Empresa. Os caminhos deverão ser criados com base na rota mais otimizada possível.

O quilometragem total, custo em combustível e todas as cidades incluídas no roteiro deverão estar disponível para o cliente em um arquivo de Saída, como forma garantir a prestação de serviço.

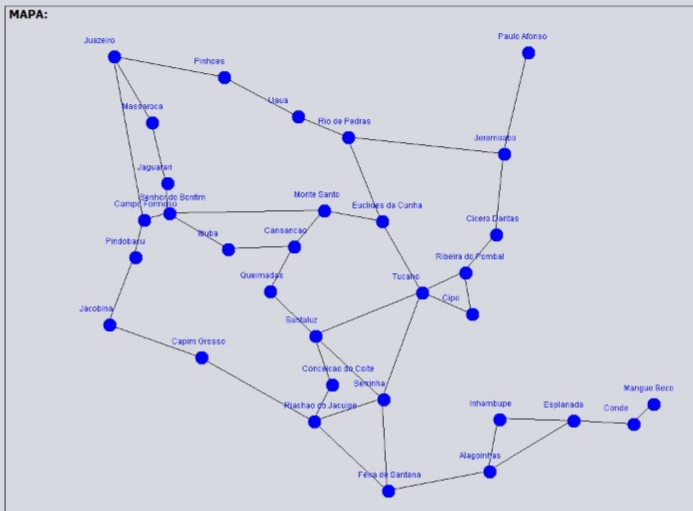
**2. Fundamentação Teórica**

Para a criação do software, em linguagem de programação Java, foi utilizado o Netbeans IDE, por ter sido considerado, entre as opções, o melhor para uso. Foi preciso aprimorar o conhecimento a respeito da programação orientada a objetos(POO), tais como descobrir a melhor estrutura de dado para o problema , utilizar o melhor algoritmo de roteirização e a criação de interfaces gráficas.

**2.1 Grafo**

Devido a necessidade de trabalhar com um conjunto de cidades, de forma que elas sejam ligadas por meio de estradas. Foi utilizado como estrutura de dados o Grafo, é uma estrutura que equivale a um conjunto de objetos em que alguns pares de objetos são, de algum modo, "relacionados", no grafo, os Vértices são como as cidades e as Arestas são como as estradas que ligam a cidades(Figura 1).

A lista tem seus benefícios, como poder inserir quantos objetos forem necessários e no momento que for conveniente, e alguns malefícios, como a necessidade de ter que acessar “nó” por “nó” até encontrar o objeto requisitado, demandando processamento, contudo, ela foi avaliada como a melhor para implementação no problema, com base na implementação e funcionalidades.

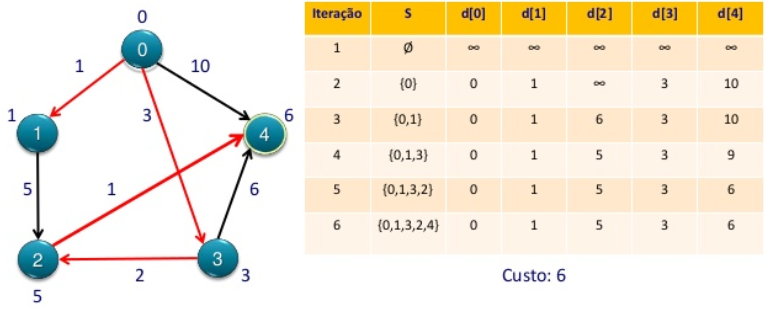


**Figura 1. Grafo com Cidades**

Cada ponto em azul representa uma cidade que por sua vez representa os vértices do Grafo, as ligações entre as cidades representam as arestas do Grafo.

**2.2 Algoritmo de Dijkstra**

Devido ao fato de que, entre cidades, há uma distância que as separa, o grafo utilizado teve suas arestas ponderadas, com pesos, caracterizando um grafo Ponderado. Com base nisso e tendo em visa a questão da otimização, foi necessário calcular o menor caminho entre pares de cidades(vértices), além disso não só a menor distancia era importante mas como o trajeto escolhido, sequencia de cidades passadas até chegar a cidade destino. Após estudar os algoritmos de roteirização, citados no problema, foi decido utilizar o algoritmo de Dijkstra, conhecido por solucionar o caminho de custo mínimo entre pares de vértices e fornecer a trajetória escolhida , com a utilização dele foi possível solucionar o problema da otimização da distância. Esse algoritmo utiliza um procedimento iterativo, determinando, na iteração 1, o vértice mais próximo de 1, na segunda iteração, o segundo vértice mais próximo de 1, e assim sucessivamente, até que em alguma iteração o vértice N seja atingido (Figura 2).



**Figura 2. Representação do Algoritmos de Dijkstra**

Nessa imagem, o vértice de partida, origem, é o de numero um, o destino é o de número seis. Há tabela representa como o algoritmo, em código, funciona, uma sucessão de iterações. Para que o menor caminho seja escolhido, não somente o peso da aresta é considerado mas como a soma do seu passo mais a distancia desse vértice à origem.

**3. Metodologia**

Devido aos componentes dos software, basicamente, interface gráfica e grafo, e ,afim de manter a organização e boas praticas de programação, foi utilizado o padrão de arquitetura MVC, padrão de projeto tem como intuito seguir um modelo pré-determinado de desenvolvimento que busca a melhor adaptação às necessidades dos desenvolvedores (Figura 3).

Para a implementação da interface de usuário gráfica do projeto foi utilizado o Form JFrame, ferramenta disponibilizada pela IDE onde é possível montar a interface GUI arrastando componentes Swing e inserindo-os no Frame principal. Essa ferramenta gera automaticamente o código da interface e não permite o usuário o altere diretamente, evitando que a estrutura da interface seja modificada, sendo o único modo de alterar os componentes através das propriedades do mesmo. O Form permite também a inserção de eventos para seus componentes, eventos que devem ser escritos pelo desenvolvedor.

**3.1 Composição**

A interface do sistema de roteamento é composta por três JPanel, o principal onde será desenhado a malha rodoviária do grafo, outro para as informações do trajeto como origem, destino e o roteiro que será seguido, e um que contém as informações relacionadas ao veículo, além dos painéis existem quatro botões na interface: calcular rota, cancelar rota, finalizar rota e ajuda (para auxílio do usuário quanto ao manuseio do software).

**3.2 Grafo**

Para desenho do grafo, foi utilizado um Form JPanel chamado de “PanelGrafo”, nele está contido o método de desenho do grafo, onde as cidades são representadas pelos vértices e suas vias são arestas. O método utilizado para desenho é o paintComponent que deve ser sobrescrito pelo programador de forma a desenhar aquilo que se deseja.

**3.3 Eventos**

Durante o desenvolvimento da interface foram implementados cinco eventos. O primeiro está relacionado ao clique do mouse no JPanel onde o grafo é desenhado, permitindo a seleção das cidades para o cálculo da rota destacando-as. O segundo evento está relacionado ao botão ajuda que lança uma mensagem na tela onde está o passo a passo para utilização do programa. O terceiro evento está relacionado ao botão calcular rota que é responsável por calcular o menor caminho entre as cidades selecionadas. O quarto evento está relacionado ao botão cancelar rota que limpa a seleção das cidades feita pelo usuário. O quinto evento está relacionado com o botão finalizar rota e é responsável por encerrar a rota calculada.

**3. Resultados e Discussões**

Devido a necessidade do fornecimento de informações para o software, foi criado dois arquivos de entrada, um contendo as cidades utilizadas e suas respectivas coordenadas e outro contendo a distancia entre as cidades vizinhas. As coordenadas foram utilizadas para desenhar as cidades na interface, o arquivo com distancias servia para criar arestas(estradas) entre os vértices(cidades) para o grafo(mapa).

Para garantir que o grafo e algoritmo de Dijkstra estavam funcionando corretamente, foram criadas, primeiramente, classes “main”, afim de explorar os erros que estavam ocorrendo e os corrigir da melhor foram, esses testes informais foram reutilizados em classes de teste, onde foi possível quantificar e formalizar o progresso do software.

**4. Conclusão**

O software desenvolvido é capaz de suprir as necessidades apresentadas, fornecendo a menor distância entre pontos específicos, gerar uma rota de trajeto, calcular o custo do transporte e exibir esses resultados em um arquivo de saída .txt.

Os comentários, padrão e javaDoc, tornaram o entendimento do software simples e acessível ao futuro usuário, sem que seja necessário grande conhecimento em linguagem JAVA.

Afim de potencializar o software, uma possível melhoria seria a opção de escolher os arquivos de entrada, os que foram usados contemplavam apenas trinta e duas cidadãs, porém o código pode suportar muito mais, abrindo a possibilidade de expansão.

**5. Bibliografia consultada**

DevMedia(2011),” Padrão MVC”, disponível em: <https://www.devmedia.com.br/padrao-mvc-java-magazine/21995>. Acessado em 07/02/2019.

SlideShare(2014), “Caminhos Mínimos- Algoritmo de Dijkstra”, disponível em: <https://pt.slideshare.net/mcastrosouza/caminhos-mnimos-algoritmo-de-dijkstra>. Acessado em 07/02/2019.