Trabalho de Algoritmos em Grafos

Otimização de Transportes Urbanos

Professor: Mayron Cesar de Oliveira Moreira

Alunos: Carlos Eduardo Borges de Sousa
Raul Souza Lima

Data: 28/01/2025

Introdução

O desafio de aprimorar os transportes urbanos é crucial na logística e na mobilidade urbana. A eficiência das rotas de ônibus impacta diretamente o tempo de deslocamento dos passageiros, os custos operacionais e a sustentabilidade do sistema de transporte. Neste trabalho, modelamos o problema como um grafo ponderado não direcionado, onde os pontos de parada dos ônibus são representados como vértices, e as conexões possíveis entre eles como arestas, ponderadas com a distância entre os pontos.

A solução implementada combina Árvore Geradora Mínima (MST) e Algoritmos Gulosos para minimizar os custos de deslocamento, garantindo conectividade entre todas as paradas com a menor distância possível. Além disso, utilizamos o Algoritmo de Fleury para encontrar um ciclo Euleriano, na intenção de reduzir o número de repetições no percurso.

Utilizamos algumas estratégias no nosso projeto, como:

- Nosso código foi feito de forma modularizada, no intuito de facilitar a manutenção e elaborar mais testes durante a execução.
- A leitura dos dados foi feita a partir de um arquivo.
- Adaptamos alguns algoritmos de acordo com o problema.
- Comparamos os resultados obtidos com a melhor solução possível.
- Os resultados foram impressos em um arquivo de saída.

Este trabalho tem como objetivo desenvolver uma solução eficiente para o problema de otimização das rotas de ônibus, comparando diferentes abordagens e analisando os resultados obtidos. Dessa forma, a maior dificuldade é chegar o mais próximo possível da solução ótima.

Formulação

O problema é voltado para otimização de rotas de ônibus, podendo ser relacionado como uma variante do famoso Problema do Caixeiro Viajante(TSP), no qual o objetivo é minimizar a distância total percorrida e que todos os vértices sejam visitados ao menos uma vez.

Como as entradas são maiores (acima de 1000), alguns algoritmos não poderiam ser usados devido a pouca eficiência para grafos grandes. Dessa forma, optamos por utilizar algoritmos como Árvore Geradora Mínima(MST) e Fleury, assim como alguns mais heurísticos como Guloso e Christofides, Como um recurso utilizamos leitura e gravação em arquivo.

O problema impõe algumas restrições como: o ônibus deve passar por cada parada pelo menos uma vez; o caminho deve ser o menor possível; todas as paradas tem que estar conectadas; e o percurso final deve formar um caminho Euleriano, no qual o ônibus saia de uma determinada parada e ao fim do caminho ele volte nessa mesma parada.

A leitura foi feita no padrão EUC_2D onde cada linha possui uma coordenada de um plano de 2 dimensões que representam uma cidade, a distância é calculada com a fórmula de distância euclidiana para espaços bidimensionais.

Matematicamente, representamos o problema como um grafo G = (V, E, w), onde:

- V é o conjunto de vértices (paradas de ônibus).
- E é o conjunto de arestas (conexões entre as paradas).
- w é a distância entre os vértices u e v.

Descrição da Solução

O primeiro passo foi implementar uma função para leitura dos dados a partir de um arquivo de entrada, juntamente realizamos o cálculo da distância euclidiana entre dois pontos. Com isso, decidimos utilizar uma matriz de adjacência para armazenar esses valores, gerando assim a matriz que será usada ao longo do programa.

Após, procuramos encontrar uma Árvore Geradora Mínima (MST) do grafo original, utilizando o Algoritmo de Kruskal. A MST vai garantir que todas as paradas estejam conectadas com a menor soma total das distâncias. No entanto, essa estrutura inicial pode conter vértices de grau ímpar, o que impede a formação de um ciclo Euleriano.

Para que tenha um ciclo Euleriano, é necessário que os vértices tenham grau par, assim adicionamos arestas extras ao grafo da MST, conectando os vértices ímpares. Utilizamos um emparelhamento mínimo ponderado para essa etapa, garantindo que os novos caminhos introduzidos tenham o menor impacto possível no custo final.

Com o grafo ajustado, aplicamos o Algoritmo de Fleury para gerar um ciclo Euleriano, garantindo que cada aresta seja percorrida exatamente uma vez. Esse processo não garante a solução ótima, e visando melhorar esse resultado, é rodado o algoritmo Guloso na matriz de adjacência.

O algoritmo guloso busca a melhor solução em cada passo na esperança de que, no cenário maior, o objetivo seja alcançado. Apesar de não garantir a solução ótima, este algoritmo tem uma complexidade de $O(n^2)$ e em boas partes das entradas, entrega resultados mais próximos do ótimo que o algoritmo anterior e com tempo de execução menor.

Entretanto o algoritmo baseado na MST também possui complexidade de $O(n^2)$ que é a complexidade do emparelhamento perfeito mínimo, por isso optamos por fazer as duas implementações, buscando sempre a mais próxima da melhor solução.

Resultados obtidos com análise

MÉTODO	INSTÂNCIA	SI	SF	DESVIO PERCENTUAL	TEMPO EXECUÇÃO
GULOSO	02.ins	60214	50801	18.52%	0.093s
	03.ins	74032	62128	19.16%	0.141s
	07.ins	68964	56638	21.76%	0.105s
	10.ins	72030	57201	25.92%	0.165s
MST	02.ins	63513	50801	25.02%	2.968s
	03.ins	77781	62128	25.19%	5.216s
	07.ins	71758	56638	26.69%	4.059s
	10.ins	73527	57201	28.54%	6.102s

Comparação entre os métodos:

O Guloso obteve melhores soluções, pois minimiza a distância total percorrida de maneira mais eficiente.
O MST com os outros algoritmos demorou muito mais para rodar, chegando a levar mais de 6 segundos na instância 10.ins, enquanto o Guloso rodou todas as instâncias em menos de 0.2 segundos. Essa diferença sempre vai acontecer, porém a tendência é que com cidades maiores ela diminua pois as suas complexidades máximas são as mesmas.

Possíveis Melhorias e Alternativas:

O algoritmo Guloso poderia ser refinado de maneira a considerar não apenas a distância para o próximo vértice, mas também a distância para os próximos além desse, outra maneira de otimizar seria rodar o mesmo com vários pontos de partida e selecionar a que gera a menor solução.
Outras abordagens, como Programação Dinâmica ou metaheurísticas, poderiam ser outras opções para tentar uma solução mais próxima da ótima.

Conclusão

Neste trabalho, exploramos duas abordagens para otimizar as rotas de transporte urbano: um algoritmo guloso e um baseado em Árvore Geradora Mínima (MST). Os testes mostraram que o método guloso foi mais eficiente tanto em tempo de execução quanto na qualidade das soluções, enquanto o MST, apesar de garantir uma solução mais estável, teve maior custo computacional e rotas menos otimizadas.

No geral, o algoritmo guloso se mostrou mais viável para o problema, oferecendo respostas rápidas e soluções satisfatórias. Como melhoria futura, seria interessante explorar heurísticas mais avançadas para equilibrar melhor qualidade e desempenho. Este trabalho reforçou a importância de escolher a abordagem certa para cada cenário, especialmente em aplicações de grande escala.

Bibliografia

https://en.wikipedia.org/wiki/Travelling_salesman_problem
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

A maior parte das nossas consultas foram nos materiais fornecidos no próprio campus virtual.