Aluno: Lucas Torquato de Melo
UFLA - Ciência da Computação

Introdução

O problema a ser resolvido é sobre uma empresa do ramo de transportes urbanos que iniciou investimentos para otimizar as rotas de um ônibus. O ônibus tem que passar por n pontos em que o motorista deve realizar uma parada. Ao fim do trajeto, o motorista deve voltar ao ponto inicial. Cada ponto deve ser visitado exatamente uma vez no percurso.

Este problema necessita que minimize o custo total do percurso, porém o mais importante é que a distância máxima entre dois pontos quaisquer do percurso seja minimizado.

Formulação

Para a formulação deste código foi utilizado os conhecimentos adquiridos em sala de aula. Principalmente a habilidade de pegar problemas do mundo real e transformar em problemas de grafos, que podem ser resolvidos com algoritmos conhecidos.

Descrição da Solução

O código apresenta uma solução para o problema do Caixeiro Viajante (TSP), onde é necessário encontrar o caminho mais curto que percorre todos os pontos (nós) e retorna ao ponto de origem. Primeiramente, ele lê a entrada, que contém os nós e suas coordenadas em 2D, a partir de um arquivo com o formato TSPLIB. A seguir, o algoritmo de Kruskal é utilizado para gerar uma Árvore Geradora Mínima (AGM), que conecta todos os nós com as arestas de menor distância, evitando ciclos.

Com a AGM pronta, é realizada uma busca em profundidade (DFS) para gerar um caminho inicial que percorre todos os nós. Para otimizar esse caminho, é aplicada a otimização 2-opt, que realiza trocas de segmentos do caminho, visando reduzir a distância total percorrida. Após a otimização, o código calcula o custo total do caminho, somando as distâncias entre os nós consecutivos, e também determina a distância máxima entre dois nós adjacentes.

Finalmente, o caminho otimizado é salvo em um arquivo de saída, e o código exibe a distância máxima e o custo total do percurso. Esse código oferece uma solução aproximada para o problema do TSP, utilizando técnicas clássicas como Kruskal e DFS, combinadas com a otimização 2-opt para melhorar o resultado.

Algoritmos Utilizados

Kruskal (Eficiência e Árvore Geradora Mínima)

–parâmetros (nós do grafo, semente aleatoria)

Kruskal foi escolhido para AGM pois ele é eficiente para encontrar o menor conjunto de arestas que conecta todos os nós de um grafo sem formar ciclos.

Inicialmente, o Kruskal cria um conjunto de arestas que ligam todos os nós do problema, onde cada aresta tem um peso igual à distância entre dois nós.

O algoritmo ordena essas arestas por distância, garantindo que as conexões de menor distância sejam usadas primeiro.

Com o uso da estrutura de dados **UnionFind**, o Kruskal vai adicionando as arestas de forma incremental, garantindo que não se formem ciclos.

Quando o número de arestas na árvore chega a **n-1** (onde **n** é o número de nós), a árvore geradora mínima está formada, conectando todos os nós com o menor custo possível

2-opt (Melhorar o percurso)

–parâmetros (caminho, nós do grafo, maximo de iterações)

O caminho gerado pela AGM pode ser longe da solução ótima do TSP. O 2-opt tenta melhorar isso trocando partes do caminho.

A ideia básica do 2-opt é pegar dois segmentos do caminho e trocar suas conexões (invertendo a ordem dos nós em uma parte do percurso) para ver se o novo caminho gera uma distância menor.

Esse processo de troca (inversão) é repetido até que nenhuma melhoria seja mais encontrada ou o número máximo de iterações seja alcançado.

Resultados obtidos com

Estrutura de dados escolhida: UnionFind (DSU)

ARQ - arquivo
SI - solução inicial
SF - solução final
SO - solução ótima

p.s. Os resultados foram prejudicados por conta das sementes geradoras randomicas, porém elas ajudaram o algoritmo a encontrar as soluções quase ou senão ótimas muitas vezes. Porém na tabela foi utilizado a média feita por 5 sementes diferentes.

ARQ	SI	Desvio	SF	Desvio	SO
01	-	-	-	-	3986
02	2500	33,8%	1655	22,1%	1289
03	1749	10,7%	1562	5,5%	1476
04	2709	56,7%	1173	3,41%	1133
05	807	17,7%	664	17,7%	546
06	876	27,0%	639	32,5%	431
07	537	26,4%	395	44,5%	219
08	1214	61,7%	464	42,6%	266
09	127	41,7%	74	29,7%	52
10	586	38,7%	359	33.9%	237

Complexidade big O notation (O n^2 log n)

Testes feitos em um Notebook com gtx 1050 e Intel I5 7a geração

Conclusão

Este trabalho teve como objetivo a aplicação de algoritmos clássicos para resolver o problema do Caixeiro Viajante (TSP), com ênfase na minimização da distância máxima entre os pontos do percurso. A solução inicial foi construída utilizando o algoritmo de Kruskal para gerar uma Árvore Geradora Mínima (AGM), e uma busca em profundidade (DFS) para percorrer os nós. No entanto, sem a aplicação de uma técnica de otimização, o caminho gerado não estava suficientemente refinado para atender de forma ideal ao problema. Foi então introduzido o algoritmo de otimização 2-opt, que proporcionou uma melhoria significativa no percurso, reduzindo a distância total e a distância máxima entre pontos.

A escolha de uma boa estrutura de dados foi crucial para o funcionamento do código, embora os resultados estejam dentro de uma margem razoável, foi possível perceber que o desempenho inicial, sem a aplicação do 2-opt, deixou a desejar em termos de custo total e distância máxima. A adição da otimização 2-opt trouxe um avanço considerável na qualidade das soluções. Ainda assim, com mais estudos e refinamentos, é possível melhorar ainda mais a eficiência do algoritmo, ajustando melhor os parâmetros e implementando técnicas de busca mais sofisticadas, como algoritmos de otimização estocástica ou heurísticas mais avançadas.