Trabalho Prático II
Algoritmos II

Igor Lacerda Faria da Silva

1 Introdução

O Trabalho Prático 2 da disciplina de Algoritmos II possui como proposta a análise de diferentes algoritmos para resolver o Problema do Caixeiro Viajante (PCV), com algumas restrições. Em suma, foi implementado um gerador de instâncias do problema, que são submetidas aos três algoritmos desenvolvidos (*Twice Around The Tree*, Christofides e *Branch And Bound*) e suas métricas de execução coletadas e analisadas.

As instâncias do PCV seguem a restrição de possuir como função de custo uma métrica:

- $c(u, v) \geq 0$
- $c(u, v) = 0 \Leftrightarrow u = v$
- $c(u, v) = c(v, u)$ (Simetria)
- $c(u, v) \leq c(u, w) + c(w, v)$ (Desigualdade Triangular)

As métricas usadas são a distância euclidiana, definida para $P = (p_x, p_y) \wedge Q = (q_x, q_y)$ como:

$$d_{\text{euclidiana}} = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

E a distância de Manhattan, definida como:

$$d_{\text{Manhattan}} = |p_x - q_x| + |p_y - q_y|$$

2 Implementação

O trabalho foi implementado na linguagem Python, versão 3.10.8, no sistema operacional Linux. O programa segue o paradigma de programação procedural, visto que não há uma distinção muito clara de quais seriam as classes em uma abordagem de programação orientada a objetos. Dito isso, foi implementada uma única classe. O programa foi testado usando a biblioteca `pytest`.

2.1 Arquivos

O programa está dividido em 3 módulos: `calculate`, `generators`, `algorithms`, sendo os dois primeiros de auxílio ao terceiro, que implementa de fato os algoritmos. Além disso, existe um módulo de testes que engloba os testes dos outros módulos.

2.1.1 Calculate

O módulo `calculate` é o mais simples, tendo como propósito apenas o cálculo de algumas medições úteis. Possui duas funções: uma para calcular a distância de um conjunto de pontos usando uma determinada métrica (que pode ser ou a distância euclidiana ou a distância de Manhattan). A outra função recebe um ciclo e um grafo e retorna o custo de se percorrer esse caminho.

A função implementada pela biblioteca `networkx` para o algoritmo de Christofides retorna uma lista de vértices, que inclui o vértice de partida duas vezes (uma vez para fechar o ciclo). Para manter a compatibilidade com essa decisão da biblioteca, a função `calculate_cost()` assume que o caminho possui o vértice inicial igual ao vértice final.

2.1.2 Generators

O módulo `generators` possui duas funções principais e uma função auxiliar. Seu propósito é, naturalmente, gerar instâncias do PCV. A função `generate_points()` gera um conjunto de n pontos no plano cartesiano, entre um piso e teto passados como parâmetros. As coordenadas dos pontos são números inteiros, pela especificação. A função `generate_instances()` cria um grafo do PCV usando as funções `generate_points()` e `calculate_distance()`.

2.1.3 Algorithms

O módulo principal do programa é baseado nos três algoritmos para solução do PCV: algoritmo de Christofides, *Branch And Bound*, *Twice Around The Tree*. Também existe uma camada de abstração que facilita a execução em escala.

- *Twice Around The Tree*

  O algoritmo de implementação mais fácil foi o *Twice Around The Tree*, em particular pois foi permitido o uso da biblioteca `networkx` para fazer o cálculo da árvore geradora e o caminhamento pré-ordem dos vértices. Esse algoritmo é aproximativo e explora as árvores geradoras mínimas (AGM) como ponto de partida: é computacionalmente simples calcular a AGM e é esperado que o circuito hamiltoniano mínimo compartilhe *pelo menos* algumas arestas. Com a AGM em mãos, o grafo é percorrido usando uma DFS para construir o caminho de menor custo a partir desta, excluindo repetições.
- Christofides
- *Branch And Bound*