Relatório trabalho - Algoritmo Aproximativo para TSP

UFPEL Algoritmos e Estruturas de Dados 3

João Vitor Laimer e Vinícius Hallmann

Neste trabalho nos foi apresentado o problema do caixeiro viajante em que nós precisamos solucioná-lo usando um algoritmo aproximativo. Primeiro nós resolvemos as instâncias do problema por algoritmo exato, mas pela alta complexidade do problema e de tempo de execução não foi possível resolver todas as instâncias

O algoritmo que nós decidimos utilizar foi o *Christofides Algorithm*. Ele é um algoritmo que possui vários passos até a sua conclusão e garante para o problema TSP uma resposta de no máximo 3/2 pior do que a solução ótima e sua complexidade chega a O(n^3). Uma boa referência sobre o funcionamento do algoritmo pode ser visto aqui: Acesso

Para nossa solução o código está estruturado da seguinte forma:

1. Gerar o grafo da instância
2. Gerar a MST
3. Procurar pelas menores tuplas do grafo (Minimum-Cost Perfect Matching)
4. Juntar a MST com essas tuplas
5. Verificar se o grafo é Euleriano e depois procurar pelo Circuito Euleriano
6. Transformar o grafo em um Ciclo Hamiltoniano tirando todos vértices repetidos

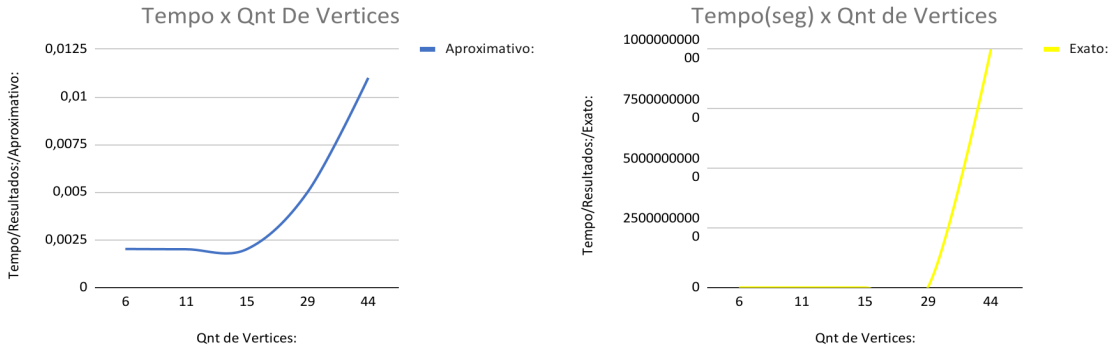
Aqui nós temos os dados coletados durante o desenvolvimento deste trabalho:

Tabela com os resultados em relação ao tempo de execução de cada TSP:

		Tempo(segundos)/Resultados:	
Arquivo:	**Qnt de Vertices:**	Aproximativo:	Aproximativo:
tsp2_1248	6	0,002024	1272
tsp1_253	11	0,002009	263
tsp3_1194	15	0,002012	1424
tsp5_27603	29	0,005000	35036
tsp4_7013	44	0,011001	7664
		Tempo(segundos)/Resultados:	
Arquivo:	**Qnt de Vertices:**	Exato:	Exato:
tsp2_1248	6	0,001	1248
tsp1_253	11	3,168	253
tsp3_1194	15	87113,011	1194
tsp5_27603	29	99999999	27603
tsp4_7013	44	99999999999	7013

Obs.: O resultado dos testes aproximativos foi o pior tempo encontrado entre 20 testes.

Gráfico: Tempo X quantidade de vértices



A análise tanto do gráfico à esquerda quanto da tabela de resultados aproximativos revela que não houve um aumento expressivo no tempo conforme a quantidade de vértices aumentou. Contudo, o gráfico à direita apresenta que a partir de 15 vértices em um algoritmo exato, a resolução do problema torna-se extremamente complexa, causada por uma explosão no tempo de resolução de forma fatorial.

Os resultados obtidos sugerem que os algoritmos aproximativos demonstram eficiência, enquanto a resolução exata para instâncias mais extensas pode se tornar impraticável dentro dos limites de tempo estabelecidos. A escolha do algoritmo para solucionar um problema dependerá da busca por um equilíbrio entre a precisão e tempo de execução desejado.

Para execução do código é possível obter todos os códigos-fonte nesse repositório do Github: Acesso e também no arquivo Zip enviado no E-Aula.

Primeiro é preciso instalar a biblioteca scipy, para fazer isso você pode instalar com o comando " pip install scipy " no terminal.

O arquivo principal é o christofides.py e nele você pode rodar individualmente os 5 arquivos do problema TSP através da variavel `path = ["arquivos tsp\\tsp1_253.txt",`
`"arquivos tsp\\tsp2_1248.txt",`
`"arquivos tsp\\tsp3_1194.txt",`
`"arquivos tsp\\tsp4_7013.txt",`
`"arquivos tsp\\tsp5_27603.txt"]`

`e para alternar entre os problemas basta mudar qual path será utilizado:`
`Exemplo:`

```
matrix = read_file(path[4])
matrix = read_file(path[0])
```