Implementação Algorítmica

Trabalho Final (versão 2)

Quarta-feira, 22 de junho de 2022

O **Problema do Caixeiro Viajante - Traveling Salesman Problem (TSP)** consiste na determinação do percurso pelo caixeiro, passando por todas as cidades em seu território exatamente uma vez e retornando à origem, de forma a cobrir a menor distância, dado que o custo entre os pares de cidade é conhecido.

O problema do caixeiro viajante é o problema NP-completo mais notório. Isto em função tanto de sua utilidade geral quanto da facilidade com que pode ser explicado ao público em geral. Imagine um caixeiro viajante planejando uma viagem de carro para visitar um conjunto de cidades. Qual é o percurso mais curto que lhe permitirá fazer isso e voltar para casa, minimizando assim sua rota total? O problema do caixeiro viajante surge em muitos problemas de transporte e rota. Outras aplicações importantes envolvem a otimização de caminhos de ferramentas para equipamentos industriais.

O objetivo deste trabalho é implementar duas soluções do TSP, utilizando as heurísticas *Hill-Climbing* Iterativo e *Simulated Annealing*.

A entrada para execução dos testes será um arquivo contendo um grafo completo, cujos vértices são dados por um conjunto de coordenadas cartesianas. O peso nas arestas corresponde à distância euclidiana. Assim, dados dois vértices u e v com coordenadas (x_u, y_u) e (x_v, y_v). A distância euclidiana é calculada pela fórmula:

$$d(u, v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}.$$

Os arquivos de entrada serão fornecidos no formato de uma lista de vértices.

Os dados dos arquivos estão no formato:

identificador_do_nó coordenada_x coordenada_y

Os programas serão executados por linha de comando no terminal, e, além do arquivo, deverão ser lidos os parâmetros necessários à execução de cada metaheurística.

As melhores soluções conhecidas para os arquivos dados são:

- att48 : 10628
- berlin52 : 7542
- bier127 : 118282
- eil76 : 538
- kroA100 : 21282

- kroE100 : 22068
- pr76 : 108159
- rat99 : 1211
- st70 : 675

Os casos de estudo para o parâmetros do *Simulated Annealing* serão:

Casos	T_{max}	k	K_T	T_{min}
A	10	0.95	20	5
B	100	0.9	25	10
C	*	*	*	*

Os parâmetros do caso C serão fixados por vocês de forma a melhorar a qualidade da solução (se possível). T_{max} é a temperatura inicial, k é a razão de resfriamento, K_T é a quantidade de iterações e T_{min} é a temperatura final.

A vizinhança da solução corrente v_c será obtida pelo mapeamento 2-troca em que duas arestas não adjacentes são trocadas. A geração de um vizinho de uma solução corrente $SolucaoCorrente$, trocando as arestas $i?1, i$ e $k, k+1$ é dada pelo seguinte código:

Procedimento GeraVizinho($SolucaoCorrente, Vizinho, i, k$)

```
1  início
2  |  para j de 0 até i − 1 faça
3  |  |  Vizinho[j] ← SolucaoCorrente[j]
4  |  fim
5  |  para j de i até k faça
6  |  |  Vizinho[j] ← SolucaoCorrente[k − j + i]
7  |  fim
8  |  para j de k + 1 até n − 1 faça
9  |  |  Vizinho[j] ← SolucaoCorrente[j]
10 |  fim
11 fim
```

O procedimento retorna o $Vizinho$ da $SolucaoCorrente$, que são vetores de n posições indexados de 0 a $n-1$.

Para gerar todos os vizinhos de uma $SolucaoCorrente$, o procedimento $GeraVizinho$ deverá ser executado por meio do seguinte trecho de código:

```
1  início
2  |  ⋮
3  |  para k de 2 até n − 2 faça
4  |  |  GeraVizinho(SolucaoCorrente, Vizinho, 1, k)
5  |  |  ⋮
6  |  fim
7  |  para i de 2 até n − 2 faça
8  |  |  para k de i + 1 até n − 1 faça
9  |  |  |  GeraVizinho(SolucaoCorrente, Vizinho, i, k)
10 |  |  |  ⋮
11 |  |  fim
12 |  fim
13 |  ⋮
14 fim
```

Para a implementação do *Hill Climbing* Iterativo utilize como referência o seguinte algoritmo:

```
1  início
2  |  t ← 0
3  |  inicializa melhor
4  |  repita
5  |  |  local ← F
6  |  |  selecionar um ponto corrente v_c de forma aleatória
7  |  |  aval(v_c)
8  |  |  repita
9  |  |  |  selecione todos os novos pontos em N(v_c)
10 |  |  |  selecione o ponto v_n de N(v_c) com o melhor valor de avaliação
11 |  |  |  se aval(v_n) é melhor do que aval(v_c) então
12 |  |  |  |  v_c ← v_n
13 |  |  |  fim
14 |  |  |  senão
15 |  |  |  |  local ← V
16 |  |  |  fim
17 |  |  até local
18 |  |  t ← t + 1
19 |  |  se v_c é melhor do que melhor então
20 |  |  |  melhor ← v_c
21 |  |  fim
22 |  até t = MAX
23 fim
```

Para a implementação do *Simulate Annealing* utilize como referência o seguinte algoritmo:

```
1  início
2  |  t ← 0
3  |  T ← T_max
4  |  seleciona um ponto corrente v_c de forma aleatória
5  |  aval(v_c)
6  |  repita
7  |  |  t ← 0
8  |  |  repita
9  |  |  |  seleciona um novo ponto v_n em N(v_c)
10 |  |  |  se aval(v_n) < aval(v_c) então
11 |  |  |  |  v_c ← v_n
12 |  |  |  fim
13 |  |  |  senão
14 |  |  |  |  se random[0,1) < e^((aval(v_c)−aval(v_n))/T) então
15 |  |  |  |  |  v_c ← v_n
16 |  |  |  |  fim
17 |  |  |  fim
18 |  |  |  t ← t + 1
19 |  |  até t = K_T
20 |  |  T ← k * T
21 |  até T < T_min
22 fim
```

Deverão ser entregues os seguintes documentos:

- Os códigos fontes de cada implementação.
- Um arquivo *readme* com as instruções de compilação e execução.
- Um relatório sucinto contendo:
  - a descrição de uma aplicação do TSP, e
  - os resultados encontrados comparados com os melhores resultados existentes para as instâncias fornecidas.

Observações:

- O trabalho será em duplas.
- As implementações poderão ser **somente** em Python, C/C++ ou Java.
- A avaliação será da seguinte forma:
  - As implementações valerão 5.0 pontos.
  - O relatório valerá 2.0 pontos.
  - A parte oral valerá 3.0 pontos, sendo 2.0 pontos do avaliado e 1.0 da avaliação do parceiro de dupla.

- A parte oral, para ambos os componentes, poderá ser de uma das seguintes formas:
  - relato escrito no dia 30/06/2022, ou
  - entrevista em data a ser fixada, ou
  - um vídeo de 5 minutos de cada integrante da dupla, entregue até 30/06/2022, via AVA.
- Na avaliação das implementações serão avaliados, entre outros, se o código é legível, se o programa compila e se ele é executado com os arquivos dados.
- A parte oral consiste da explicação de uma das implementações por cada membro do grupo, de forma que as duas implementações sejam relatadas.

Datas

1. Definição das duplas: até 15/06
2. Definição da forma de apresentação da parte oral: até 22/06
3. Entrega do trabalho: até 30/06