TSP-SA

1. Carregar instância

2. Calcular matriz de distância euclidiana. Pode ser Matriz Diagonal por se tratar de TSP Simétrico.

3. Gerar solução inicial aleatória. Permutação de N inteiros com N sendo a quantidade de cidades do TSP.

4. Calcular custo do caminho da solução aleatória.

5. Aplicar rotina de geração de vizinhos.

Dica de perturbação (gerar vizinho) a cada iteração. Uma perturbação consiste em trocar de posição (*swap*) um par de cidades escolhido aleatoriamente. Estipular um máximo de 5 perturbações a cada iteração. Perguntar quantas vão ocorrer de maneira aleatória com mínimo de uma. Realizar as perturbações. Ao final do processo você pode ter uma nova solução composta por máximo de 5 pares trocados.

6. Entrar na lógica de otimização do SA.

. Do processo de otimização, trazer gráfico de convergência do SA. O gráfico de convergência consiste no gráfico com o valor do custo atual (eixo-y) a cada iteração do alg (eixo-x).

. Trazer média e desvio-padrão dos resultados obtidos considerando 10 RUNS do SA para cada instância. Montar tabela de média e desvio-padrão para cada instância: avg +/- std

. Trazer box-plot do resultado das 10 RUNS para cada instância com 3 equações de queda de energia. <https://www.delftstack.com/pt/howto/matplotlib/matplotlib-boxplot-python/> <https://www.voitto.com.br/blog/artigo/boxplot>

<http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html>
<http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html>