UENF

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Curso: Ciência de Computação **Data:** 20./.10./2022
Trabalho: Trabalho2 **Período:** 7º **Disciplina:** Top. Esp.: Heurísticas e Complexidade
Professor: Fermín Alfredo Tang **Turno:** Diurno
Nome do Aluno: .....Matrícula: .....

1.- Com base nos problemas estudados no Trabalho1, escolher um problema da categoria NP da sua preferência e realize as seguintes tarefas:

i) Propor e implementar uma heurística construtiva para achar uma solução inicial para o problema escolhido. Descrever a representação da sua solução e seu método construtivo e implemente.

ii) Propor e implementar uma heurística de busca local para achar uma solução de melhor qualidade que a encontrada no item i). Descrever o mecanismo de busca local adotado e implemente.

iii) Teste as suas duas heurísticas para instâncias pequenas do problema.

iv) Teste as suas duas heurísticas para instâncias mais difíceis. Por exemplo, procure conjuntos de dados de teste (*test data sets*) para o problema escolhido, em repositórios semelhantes a: http://people.brunel.ac.uk/~mastjjb/jeb/info.html e escolha um conjunto de pelo menos três problemas para resolver. Mostre os resultados em uma tabela. Se possível compare a sua solução com a solução ótima, caso disponível.

i)

Foi feito um algoritmo guloso para o TSP

ii)

Como busca local foi feito uma permutação de uma rota inicial para estabelecer a vizinhança e após a troca era assim feita logo após achar uma permutação melhor. Para a solução inicial foi utilizada a mesma solução retirada do algoritmo guloso, porém até uma solução aleatória serviria.

iii)

Como o TSP é um problema de otimização, achar uma solução viável é bem rápido, o problema é garantir que essa solução seja otima.

Para n=10

```
PS C:\Users\mathe\OneDrive\Área de Trabalho\TSP> & C:/Users/mathe/AppData/Local/Microsoft/WindowsApps/pyt
hon3.10.exe "c:/Users/mathe/OneDrive/Área de Trabalho/TSP/greedyTSP.py"
Algoritmo Guloso
Custo do caminho: 232
[10, 6, 5, 9, 7, 4, 3, 8, 2, 1]
Tempo de Execução da Busca Gulosa: 0.0 s
Tempo de Execução da Busca Local 0.0
Rota após busca local [2, 10, 6, 5, 9, 7, 4, 3, 8, 1]
Custo após a busca por permutação 208
PS C:\Users\mathe\OneDrive\Área de Trabalho\TSP>
```

iv)

Para n=50

```
PS C:\Users\mathe\OneDrive\Área de Trabalho\TSP> & C:/Users/mathe/AppData/Local/Microsoft/WindowsApps/pyt
hon3.10.exe "c:/Users/mathe/OneDrive/Área de Trabalho/TSP/greedyTSP.py"
Algoritmo Guloso
Custo do caminho: 532
[3, 11, 7, 23, 17, 36, 10, 16, 21, 32, 44, 25, 19, 9, 5, 8, 42, 24, 39, 2, 13, 37, 6, 35, 28, 46, 40, 15,
 38, 18, 27, 50, 30, 41, 22, 29, 33, 26, 12, 4, 45, 43, 20, 14, 48, 49, 34, 31, 47, 11]
Tempo de Execução da Busca Gulosa: 0.0 s
Tempo de Execução da Busca Local 0.007001638412475586
Rota após busca local [3, 11, 7, 23, 17, 36, 10, 16, 21, 32, 44, 25, 19, 9, 5, 8, 42, 24, 39, 2, 13, 37,
6, 35, 28, 46, 40, 15, 38, 18, 27, 50, 30, 41, 22, 29, 33, 26, 12, 4, 45, 43, 20, 14, 48, 49, 34, 31, 47,
 11]
Custo após a busca por permutação 532
PS C:\Users\mathe\OneDrive\Área de Trabalho\TSP>
```

Com testes também foi notado que é mais comum a mudança de rota em grafos menores, acredito que seja porque a solução gulosa faz um trabalho melhor quando há mais opções de casas para escolher.

Para a complexidade do algoritmo guloso, temos $O = n^2$ por estar percorrendo uma matriz e a busca por permutação é $O = log(n)$

O código comentado está em anexo a atividade