Aluno: Rodrigo Marins Esteves

Problema do Caixeiro Viajante Assimétrico

Descrição: Um caixeiro pretende passar por n cidades percorrendo o menor caminho possível. Para isso, presumem-se as seguintes afirmativas:

1. Serão selecionadas uma cidade inicial e uma cidade final, onde a viagem respectivamente se iniciará e se encerrará.
2. O caixeiro irá passar apenas uma vez por cada cidade.
3. Todos as cidades possuem conexões entre si, onde tais conexões possuem suas respectivas distancias.
4. As distancias entre duas cidades A e B podem ser diferente se considerarmos de qual delas iremos partir e em qual iremos chegar, ou seja, a distância de A para B pode ser diferente da distancia de B para A.

Modelagem: O problema será modelado em grafos, onde cada vértice representará uma cidade e cada aresta será um caminho entre tais cidades. Cada aresta, portanto, terá seu peso, que por sua vez representará a distancia entre as cidades.

Bibliografia: http://www.mat.ufrgs.br/~portosil/caixeiro.html

Repositórios:

- https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/
- https://projects.coin-or.org/metslib/browser/Examples/trunk/atsp/data/ry48p.atsp

Heurística Construtiva:

- Heurística do Vizinho mais Próximo (Nearest Neighbor Heuristic)

  Seja G(V,E) um grafo completo, onde V são os vértices e E as arestas.

  1. Selecione um v ∈ V como nó inicial.
  2. Encontre o menor e ∈ E que leve a um u ∈ V que não tenha sido visitado.
  3. Selecione u como o vértice atual.
  4. Marque u como visitado.
  5. Se todos os nós de V foram visitados, termine.
  6. Volte ao passo 2.

Primeira implementação

O primeiro trabalho consistiu na implementação de duas heurísticas construtivas para resolver o ATSP. Para este trabalho, foram implementadas a Heurística do Vizinho mais Próximo e a Heurística do Vizinho mais Próximo com transição dupla (construindo o caminho através dos dois nós mais extremos que já estão na solução). Foi também feita a randomização do primeiro algoritmo citado. Apesar do segundo ter sido mais eficiente, como será mostrado, achei mais prudente utilizar o primeiro para a randomização pois ele cabe melhor nas comparações que mostrarei adiante. Os algoritmos foram construídos em C e para a randomização foi utilizada uma implementação do algoritmo Mersenne Twister encontrada na internet.

- Heurística do Vizinho mais Próximo x Heurística do Vizinho mais Próximo Bidirecional

Como podemos ver na imagem abaixo, a heurística bidirecional apresentou uma pequena vantagem em relação à solução nos 3 arquivos testados.

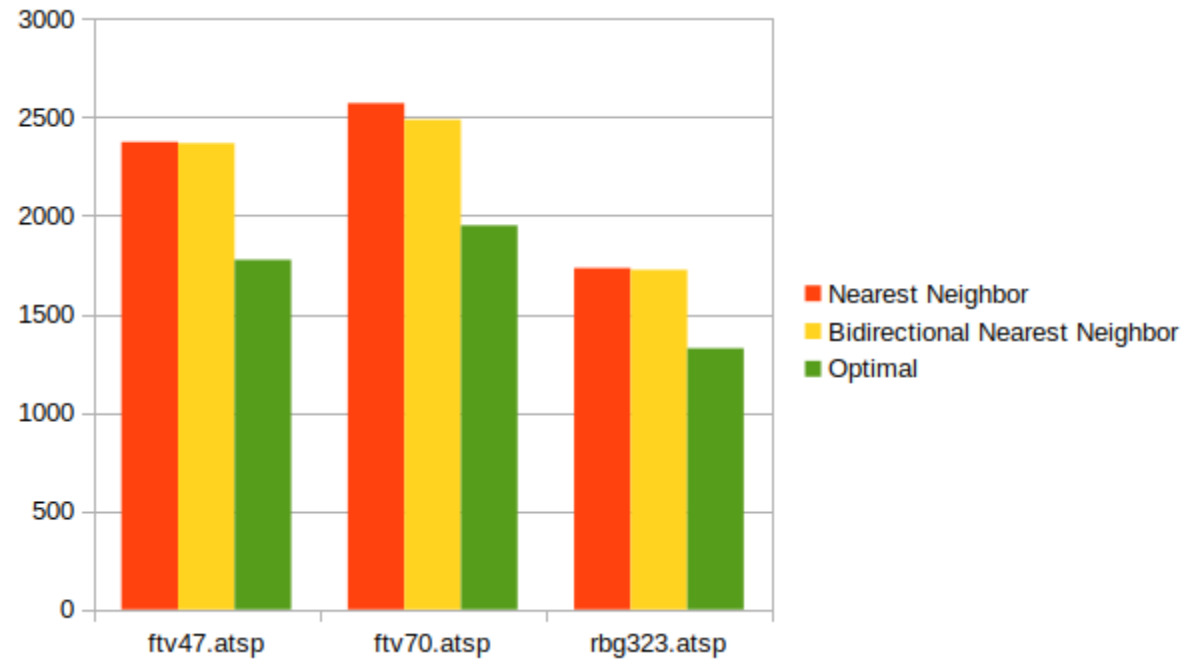

Fig1: (Heuristica/Arquivo x Tamanho da solução).

- Randomização da Heurística da Vizinho mais Próximo

Para testar o Algoritmo Randomizado, foram feitas 100 execuções em um grafo contendo 323 nós, todos conectados uns aos outros. Como podemos ver na imagem abaixo, a randomização pode trazer boas ou péssimas soluções. Ao aproximar o alpha de 0, temos soluções tendendo às soluções de um Algoritmo Guloso. Ao aproximar o alpha de 1, temos soluções que tendem às soluções de um algoritmo completamente randômico. Em alguns casos, com alpha = 0.1, foram encontradas algumas soluções melhores que a solução do Algoritmo Guloso, como pode ser visto na figura 3.

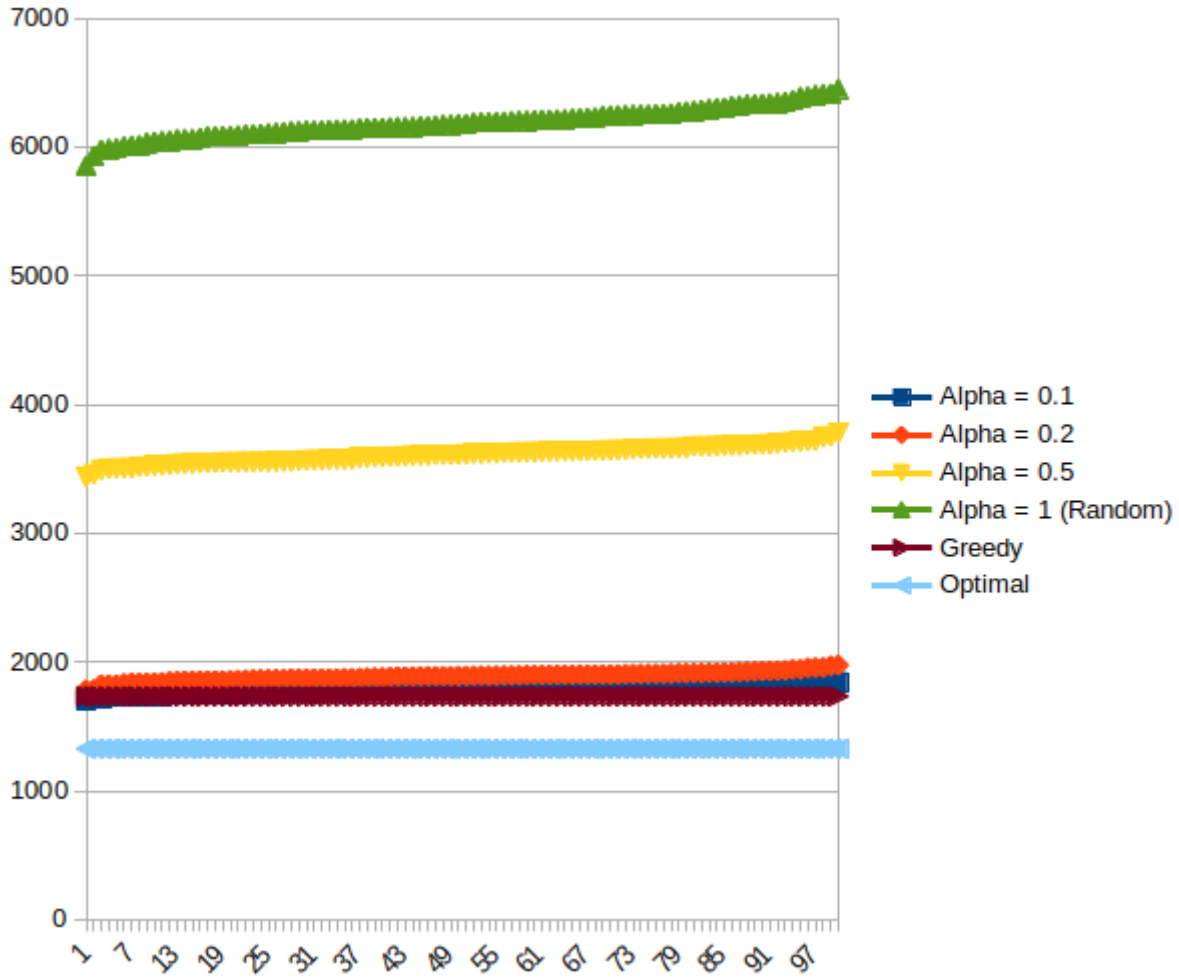

Figura 2: (Execuções x Valor das soluções).

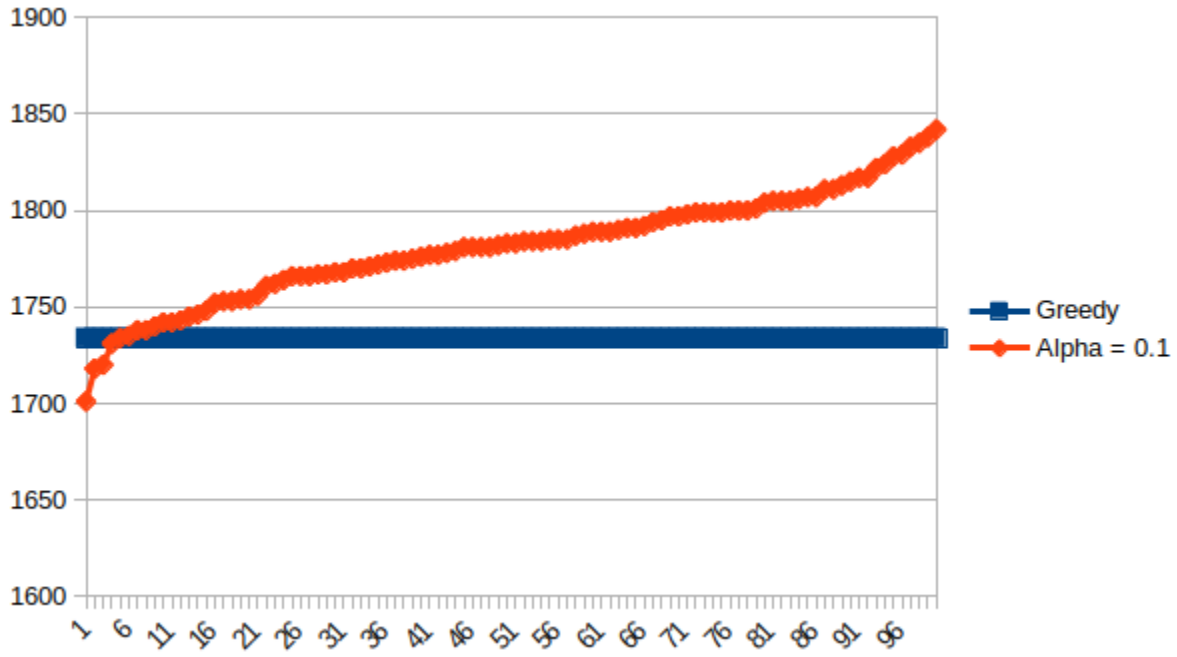

Figura 3: (Execuções x Valor das soluções). Comparação entre o Algoritmo Guloso e o Randomizado com aplha = 0.1. Pode se notar que em alguns casos o Algoritmo Randomizado encontra soluções melhores.