INF05010 Otimização Combinatória
Trabalho Final Parte 3
Implementação da Busca Tabu para solucionar o problema Rota mais Valiosa

Rafael Júnior Ribeiro - 265830

Maio 2021

1 Plataforma de implementação

O trabalho foi implementado e testado em um sistema operacional Linux MInt 20.1, processador Intel(R) Quad Core(TM) i5-7300HQ CPU @ 2.50GHz e 8 GB de RAM. A linguagem de programação usada foi o Júlia versão 1.4.1. Todos os arquivos do projeto podem ser visualizados em `https://github.com/rjribeiro/otimizacao-combinatoria`

2 Estruturas de dados

- **grafo:** matriz de adjacência de inteiros, com a célula (i, j) possuindo a distância entre o vértice i e o vértice j.
- **premios:** vetor de inteiros possuindo os prêmios de cada vértice.
- **solucao:** vetor de inteiros, com tamanho desconhecido, representando o solução que otimiza o problema da rota da mais valiosa. Na posição 1 do vetor sempre haverá o número 1. Não terá números repetidos
- **vertices_nao_visitados:** vetor de inteiros, representando o complemento da solução.
- **tabela_tabu:** matriz de inteiros de dimensão (2, K, K) sendo K a cardinalidade do grafo.
- **vizinho:** uma tupla de tamanho 4.
  - solução vizinha a solução atual. Possui um vértice a mais ou menos em relação a solução atual. Um vetor de inteiros

- custo da solução vizinha. Um float
- complemento da solução vizinha. Um vetor de inteiros
- Tripla de inteiros representado um atributo para inserção na tabela tabu.
  * Ação: remoção (1) ou adição (2) de vértices da solução atual.
  * Vértice: se a ação for de remoção, então será o número do vértice deletado. Se a ação for de adição, então será o número do vértice adicionado.
  * índice: se a ação for de remoção, então será sempre 1, já que o vértice removido será inserido no fim do vetor de vértices não visitados. Se a ação for de adição, então será o índice indicando em qual posição o vértice será inserido será inserido na solução.

3 Funcionamento do código

1. Inicializa grafo
2. Inicializa K como a cardinalidade do grafo
3. Inicializa premios
4. Inicializa solucao como uma solução inicial aleatória
5. Inicializa vértices_nao_visitados como os vértices do grafo que não estão solucao
6. Inicializa tabela_tabu em função de K
7. Inicializa menor_custo_global como 9999.
8. Inicializa melhor_solucao_global como um objeto null
9. Executa K * 5000 vezes o seguinte laço:
   9.1. Inicializa menor_custo_local em 9999
   9.2. Inicializa o melhor_solucao_local como um objeto null
   9.3. Escolhe aleatoriamente K * 0.7 vizinhos com um vértice a menos que solucao. A cada vizinho é verificado seu custo e pode haver 4 consequências:
      A Se o custo for menor que menor_custo_local, então faz nada.
      B Se o custo for menor que menor_custo_local e o movimento associado àquele vizinho não estiver na tabela_tabu, então atualiza o menor_custo_local e o melhor_solucao_local
      C Se o custo for menor que menor_custo_local, maior que menor_custo_global e o movimento associado àquele vizinho estiver na tabela_tabu, então faz nada

D Se o custo for menor que menor_custo local e menor que menor_custo_global, então atualiza o menor_custo_local, menor_custo_global, melhor_solucao_local e melhor_solucao_global,

9.4. Repete o procedimento anterior, porém desta vez escolhendo aleatoriamente K * 0.9 vizinhos com um vértices a mais que solucao

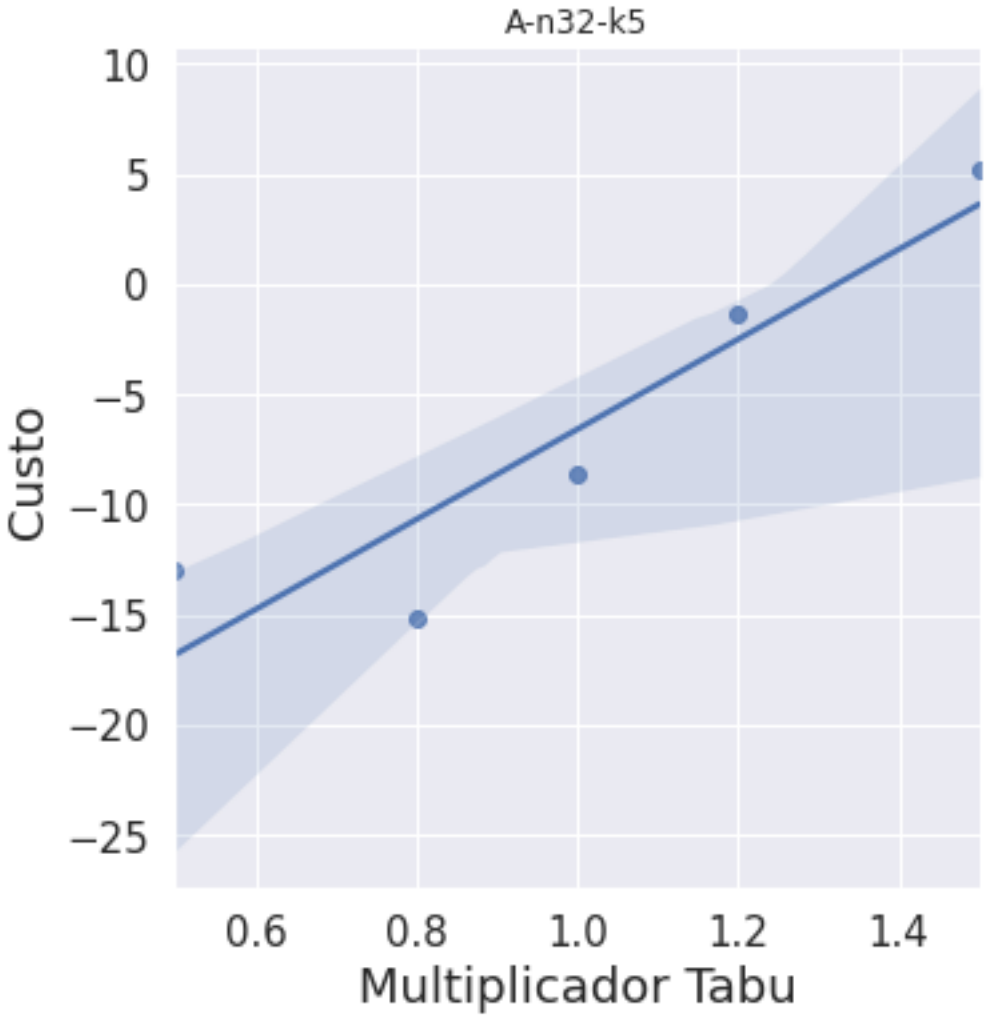
9.5. Atualiza solucao com melhor_vizinho_local e atualiza vertices_nao_visitados e tabela_tabu de acordo com essa nova solucao

10. Retorna melhor_solucao_global

4 Análise dos resultados

4.1 Otimização do parâmetro tamanho tabu

O tamanho do tabu é definido em razão do tamanho do grafo. Foi realizado uma busca em grid para verificar com qual fator multiplicante a heurística trazia o menor custo. Os fatores multiplicantes avaliados foram 0.5, 0.8, 1, 1.2 e 1.5. Para cada instância e para cada fator multiplicante foram executado 5 vezes a otimização. Os resultados presentes nos gráficos são as médias. Os gráficos estão presentes na pasta plots do repositório do projeto. Abaixo o gráfico da instância A-n32-k5:



A vísivel relação linear entre Multiplicador Tabu e Custo foi verificada em 9 das 12 instâncias. Isso consequentemente sugere uma relação linear entre tamanho do tabu e custo. Memórias mais curtas tendem a otimizar melhor que memórias com duração maior.

4.2 Testes

Instância	Valor Referência	Valor Obtido	Desvio
A-n32-k5.vrp	-9	-18	-100,00%
A-n80-k10.vrp	-24	-238	-891,70%
B-n31-k5.vrp	-118	-173	-46,60%
B-n78-k10.vrp	-235	-413	-75,70%
E-n101-k14.vrp	-543	-775	-42,70%
E-n76-k8.vrp	-564	-736	-30,50%
F-n72-k4.vrp	-113.776	-113.990	-0,20%
G-n262-k25.vrp	-7.193	-8.527	-18,50%
M-n101-k10.vrp	-931	-1.144	-22,90%
M-n200-k17.vrp	-1.789	-2.035	-13,80%
P-n101-k4.vrp	-543	-795	-46,40%
P-n76-k5.vrp	-564	-726	-28,70%

Table 1: Resultados do testes com tamanho tabu tunado

5 Referências

1. `https://www.researchgate.net/publication/331585233_Tabu_Search_Method_for_Solving_the_Traveling_salesman_Problem`
2. `https://www.youtube.com/watch?v=i1IJAVhCn6U`