Departamento de Ciência da Computação
D C C

Heurísticas para o problema do caixeiro viajante

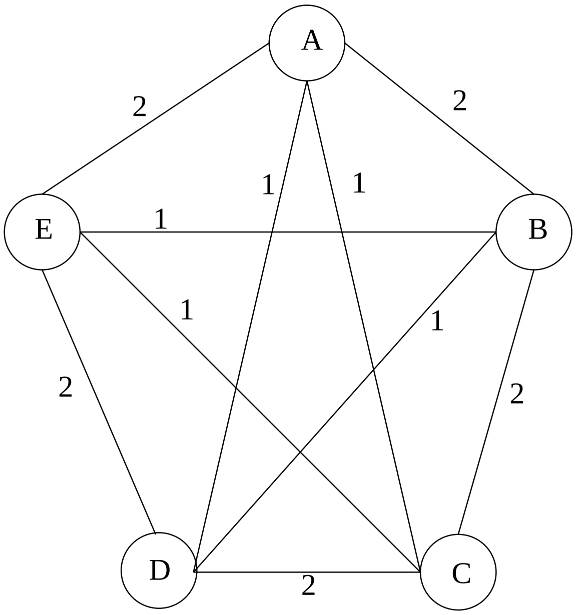
André Luiz de Senna
Wagner Meira Jr.

Março - 1998

U F M G
Universidade Federal de Minas Gerais

1 Introdução

O problema do caixeiro viajante (PCV) tradicional consiste em encontrar o circuito hamiltoniano de menor custo em um grafo completo não direcionado. Um circuito hamiltoniano é um caminho fechado (começando e terminando no mesmo vértice) que passa por todos os vértices do grafo sem repetição. Exemplificando, seja o grafo:



Definimos $custo(i, j)$ como o valor do peso da aresta entre os vértices i e j. Circuitos hamiltonianos possíveis seriam: A-B-C-D-E-A, C-B-D-A-E-C, A-E-C-B-D-A, etc. Um circuito hamiltoniano de custo mínimo seria: A-C-E-B-D-A com um custo total de 5.

O PCV pertence à classe $\mathcal{NP}$-completo, portanto não se conhece algoritmos ótimos para resolvê-lo. Diversas heurísticas, no entanto, foram propostas na tentativa de se conseguir respostas boas, ou seja, circuitos hamiltonianos cujo custo não necessariamente é o menor possível mas é bastante reduzido. Uma descrição detalhada das diversas variantes do PCV assim como dos diversos algoritmos propostos pode ser encontrada em [1].

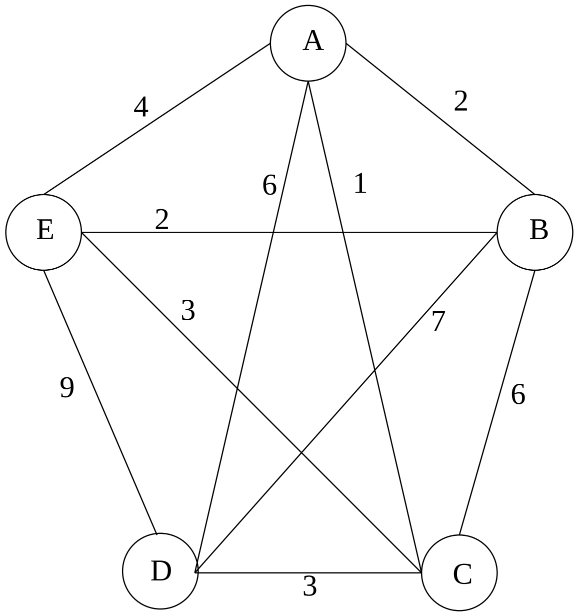
Neste trabalho são comparadas três heurísticas detalhadas em [1]: *inserção da cidade mais próxima, inserção da cidade mais distante* e a heurística do *vizinho mais próximo*. O principal atrativo destas heurísticas são a simplicidade de implementação e a pequena complexidade teórica.

2 As heurísticas implementadas

Inserção da cidade mais próxima

Selecione um vértice i qualquer para começar. Seja j um vértice tal que $custo(i, j) \leq custo(i, k) \;\forall\; k$ (j é o vértice mais "próximo" de i). Forme o sub-circuito inicial $i - j - i$.

Iterativamente, até todos os vértices entrarem no sub-circuito atual (SCA), escolha um vértice $v \notin SCA$ tal que $\exists x \in SCA \mid custo(x, v) \leq custo(y, z) \ \ \forall \ \ y \in SCA, z \notin SCA$, ou seja, v é o vértice mais "próximo" do sub-circuito atual. v é então colocado no sub-circuito atual, na posição que minimiza o custo total do novo sub-circuito. Exemplificando, considere o grafo:



Iniciando o algoritmo com o vértice B teríamos:

1. Os vértices mais próximos de B são A e E
2. Escolhemos arbitrariamente E
3. Sub-circuito inicial: B-E-B com custo 4
4. O vértice mais próximo de B ou E é A (A-B tem custo 2)
5. A pode ser inserido de duas formas diferentes: B-A-E-B (custo 8) ou B-E-A-B (custo 8)
6. Escolhemos arbitrariamente B-A-E-B com custo 8
7. O vértice mais próximo de B-A-E-B é C (A-C tem custo 1)
8. C pode ser inserido de três formas diferentes: B-C-A-E-B (custo 13), B-A-C-E-B (custo 8) ou B-A-E-C-B (custo 15)
9. Escolhemos o menor B-A-C-E-B
10. O único vértice restante é D

11. D pode ser inserido de quatro maneiras diferentes: B-D-A-C-E-B (custo 19), B-A-D-C-E-B (custo 16), B-A-C-D-E-B (custo 17) ou B-A-C-E-D-B (custo 22)

12. Escolhemos o menor que é a resposta do algoritmo: B-A-D-C-E-B (custo 16)

Inserção da cidade mais próxima

Este método repete os mesmos pessos do método anterior com exceção da escolha do próximo vértice a entrar no sub-circuito atual (SCA): é escolhido o vértice v (que ainda não entrou no circuito) tal que $\exists x \in SCA \mid custo(x, v) \geq custo(y, z) \;\; \forall \;\; y \in SCA, z \notin SCA$, ou seja, v é o vértice mais "distante" do sub-circuito atual.

Vizinho mais próximo

1. Comece com um sub-caminho arbitrário formado por um único vértice V_1

2. Se o sub-caminho atual (SCA) é $V_1 - V_2 - ... - V_k$, $k <$ número de vértices, então V_{k+1} é o vértice v (que ainda não entrou no caminho) tal que $custo(V_k, v) \leq custo(V_k, x) \;\; \forall \;\; x \notin SCA$, ou seja, v é o vértice mais "próximo" de V_k.

3. Coloque V_{k+1} no final do caminho atual

4. Pare quando todas as cidades estiverem no caminho atual e feche o ciclo acrescentando o custo do último vértice para o primeiro

3 Estratégia dos testes

Os algoritmos serão comparados segundo dois critérios: qualidade das respostas encontradas e tempo de execução.

Para a realização dos testes foi implementado um gerador de problemas que cria grafos completos com pesos aleatórios uniformemente distribuídos dentro de um intervalo. Os parâmetros do programa gerador são o número de vértices do grafo e o intervalo em que os pesos devem ser gerados.

O objetivo dos testes é avaliar o comportamento dos algoritmos com a variação de dois parâmetros diferentes:

1. Número de vértices do grafo

2. Tamanho do intervalo em que os pesos estão distribuídos.

Variando o número de vértices poderemos avaliar a complexidade de tempo dos algoritmos pois tal parâmetro é a dimensão do problema. A variação no tamanho do intervalo em que os pesos estão distribuídos nos permitirá avaliar a qualidade das soluções para problemas mais ou menos difíceis (teoricamente, quanto menor o tamanho do intervalo mais difícil é o problema).

O resultado de cada avaliação foi feito com base na média aritmética de dez execuções.

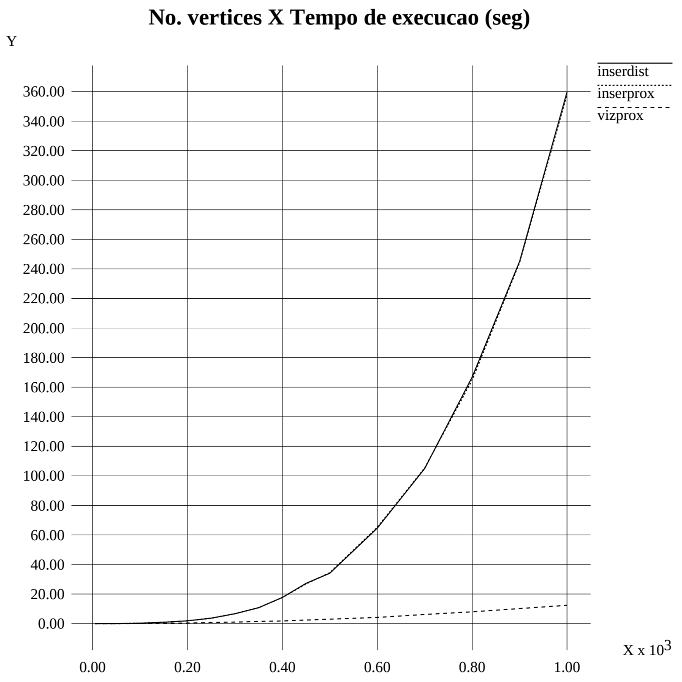


Figura 1: tempo de execução para pesos no intervalo 0 - 5

4 Ambiente de testes

Todos os testes foram feitos em uma Sun Sparc Ultra 1 com 1Gb de RAM dedicada (apenas com os processos normais do Unix).

5 Resultados

Os resultados são apresentados nas figuras 1, 2, 3, 4, 5 e 6. Legenda:

inserdist - Inserção da cidade mais distante

inserprox - Inserção da cidade mais próxima

vizprox - Vizinho mais próximo

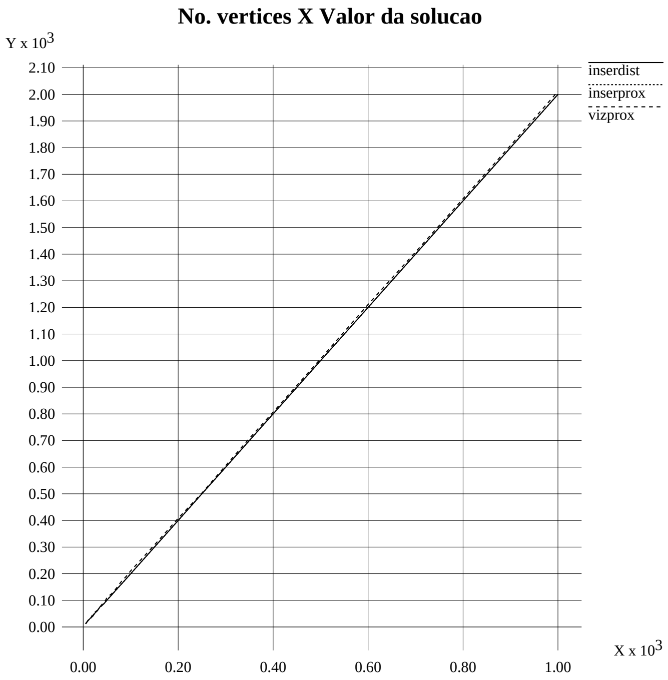


Figura 2: qualidade das respostas para pesos no intervalo 0 - 5

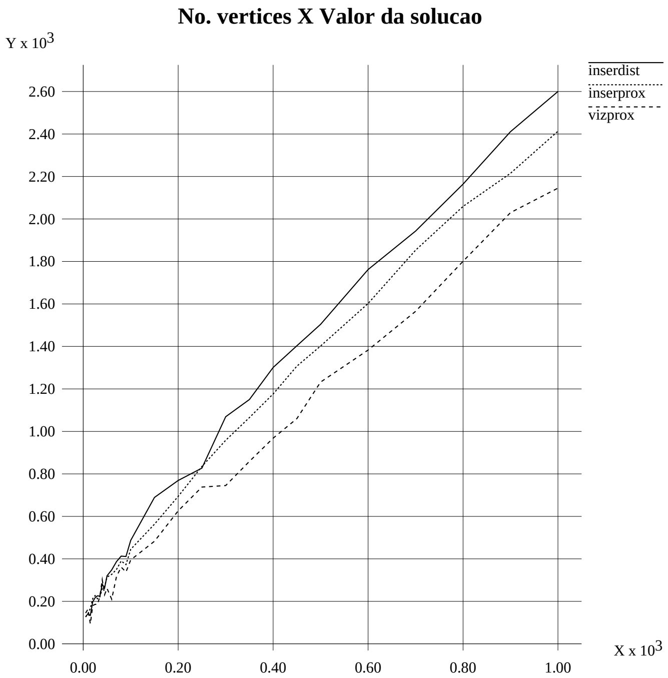


Figura 3: qualidade das respostas para pesos no intervalo 0 - 50

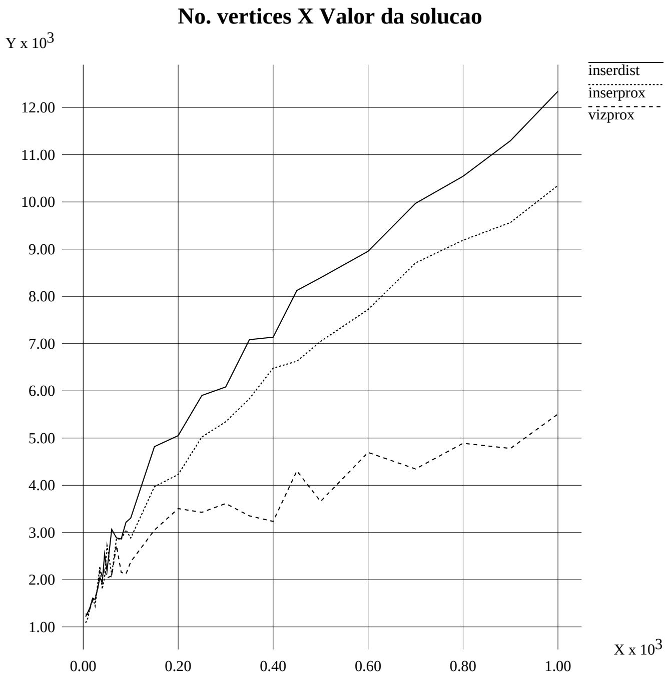


Figura 4: qualidade das respostas para pesos no intervalo 0 - 500

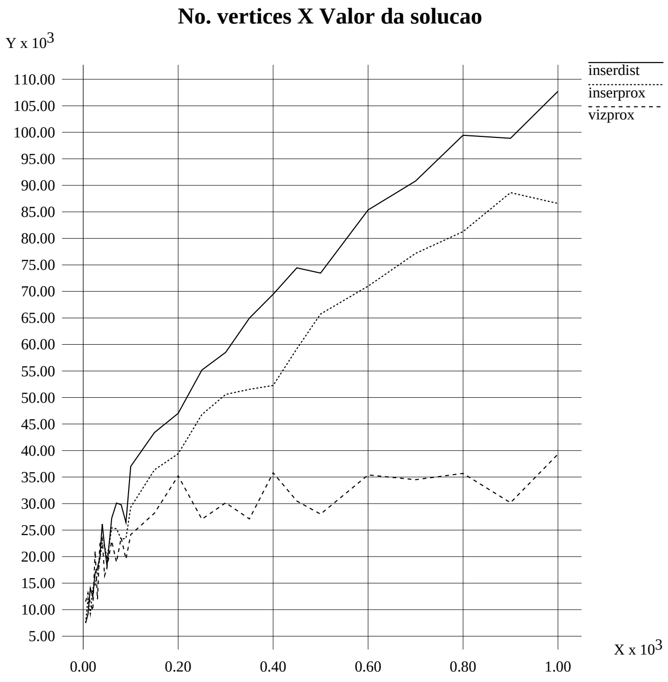


Figura 5: qualidade das respostas para pesos no intervalo 0 - 5000

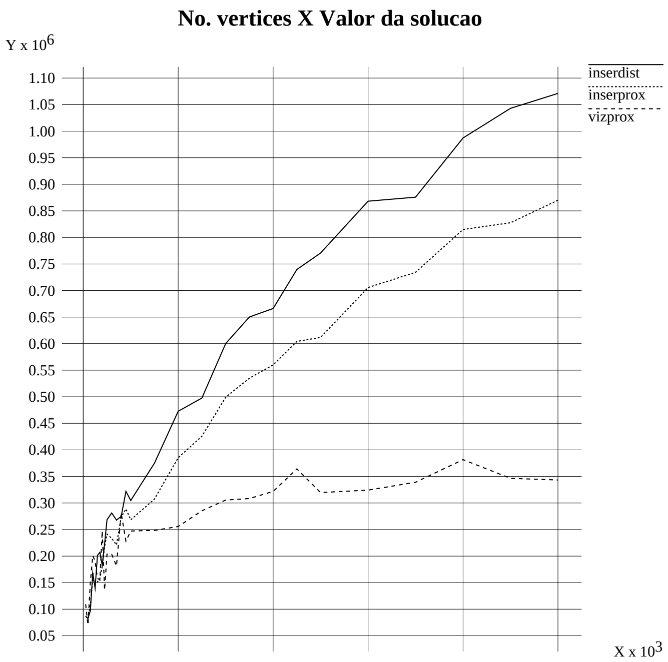


Figura 6: qualidade das respostas para pesos no intervalo 0 - 50000

6 Análise dos resultados

Observando o gráfico apresentado na figura 1 podemos concluir que a complexidade (em função do número de vértices do grafo) dos algoritmos de inserção de cidades é maior que do vizinho mais próximo. Este resultado já era esperado visto que a complexidade teórica de ambos os algoritmos de inserção é $O(n^2)$ enquanto a do vizinho mais próximo é $O(n)$. O tempo de execução para os demais intervalos de distribuição dos valores dos pesos das arestas não se alterou, razão pela qual os gráficos das execuções com tais intervalos não são apresentados.

A análise do gráfico apresentado na figura 2 nos permite concluir que, para este intervalo, independente do tamanho do problema, a qualidade das respostas das três heurísticas é muito semelhante. Para instâncias maiores do problema, a heurística do vizinho mais próximo se mostrou ligeiramente pior: 0.6 % pior no máximo e 0.4 % pior na média para grafos com mais de 100 vértices.

No entanto, na medida que o tamanho do intervalo aumenta, podemos observar pelos gráficos das figuras 3, 4, 5 e 6, que este comportamento se inverte. As soluções encontradas pela heurística do vizinho mais próximo se tornam gradativamente melhores que as das heurísticas de inserção. Mesmo entre essas últimas ocorre uma diferenciação cada vez maior na medida que o intervalo onde os valores dos pesos estão distribuídos aumenta.

Vértices	inserdist	inserprox	vizprox	inserdist/inserprox	inserprox/vizprox	inserdist/vizprox
100	304664	268297	247208	1.13555	1.08531	1.23242
150	374933	307375	248341	1.21979	1.23771	1.50975
200	472603	385201	255637	1.22690	1.50683	1.84873
250	497636	425716	285346	1.16894	1.49193	1.74397
300	600141	499216	305497	1.20217	1.63411	1.96447
350	650178	534917	308331	1.21547	1.73488	2.10870
400	666193	559986	321769	1.18966	1.74034	2.07041
450	739520	604257	364045	1.22385	1.65984	2.03140
500	770606	611722	319652	1.25973	1.91371	2.41077
600	868296	705717	324252	1.23037	2.17645	2.67784
700	875775	734275	339014	1.19271	2.16591	2.58330
800	987147	814965	381635	1.21128	2.13546	2.58663
900	1043110	827604	346409	1.26040	2.38909	3.01121
1000	1071117	870117	343344	1.23100	2.53424	3.11966
			Mínimo	1.13555	1.08531	1.23242
			Máximo	1.26040	2.53424	3.11966
			Média	1.21199	1.81469	2.20709

Tabela 1: Diferença das respostas das heurísticas para grafos com pesos no intervalo 0-50000

No caso extremo onde o intervalo dos pesos foi 0-50000 a diferença observada é mostrada na tabela 1. A resposta encontrada pela heurística de inserção da cidade mais distante chega a ser 3.12 vezes pior que a heurística vizinho mais próximo.

Para instâncias menores do problemas (abaixo de 100 vértices), as respostas das três heurísticas não apresentam tendência favorável a nenhuma delas, não importando o tamanho do intervalo.

7 Conclusões

Para pequenas instâncias do problema as três heurísticas têm comportamento muito semelhante, tanto em relação à qualidade das soluções quanto em relação ao tempo de execução.

Para grafos com os pesos distribuídos em um pequeno intervalo, as heurísticas de inserção são ligeiramente melhores que a do vizinho mais próximo mas esta situação se inverte quando o intervalo onde os valores dos pesos estão distribuídos aumenta. Neste caso, a pior heurística é a inserção da cidade mais distante.

Independente do tamanho do intervalo de distribuição dos pesos, o tempo de execução das heurísticas de inserção cresce de forma quadrática enquanto o da heurística do vizinho mais próximo cresce linearmente.

Uma proposta para extensão deste trabalho é estudar o comportamento das heurísticas para problemas existentes em bibliotecas eletrônicas como a netlib (http://www.netlib.com) e outros tipos particulares de grafos (grafos simétricos, euclideanos, esparços, etc.). Outra proposta é estudar outras distribuições para os pesos das arestas (distribuições normais, exponenciais, etc.).

Referências

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons, 1985.