Trabalho de Projeto e Análise de Algoritmos

Problema do Caixeiro Viajante

**Rafael Francisco Ferreira, Luís Fernando S. Gaspar, Ivanildo da Silva Barros**

**Ciência da Computação – Universidade Estadual do Paraná (UNESPAR)**

**Apucarana – PR – Brasil**

rafaelfrancisco\_97@hotmail.com , gasparfluis@gmail.com , iva-nildo@hotmail.com

***Resumo:*** *Este artigo apresenta duas possíveis soluções para o problema do Caixeiro Viajante****,*** *que tenta determinar a menor rota para percorrer uma série de cidades, visitando cada uma delas uma única vez e retornando à cidade de origem. O primeiro algorítimo utiliza o método da força bruta, já o segundo utiliza uma heurística baseada no algoritmo do vizinho mais próximo. A proposta desta heurística é tentar uma resolução para o problema em tempo* ***polinomial****. No decorrer do artigo discutiremos cada um deles.*

**1. Introdução**

O problema do caixeiro viajante (PCV) consiste na complexidade de encontrar o *ciclo hamiltoniano* de menor custo. O PCV representa um problema de grafos composto por n vértices (cidades), e um custo em cada aresta, ou cada par de vértices, cujo objetivo é encontrar a possibilidade que passe por todas as cidades com o menor custo, dentre as (n-1)! possibilidades existentes. Por se tratar de um problema de otimização combinatória, pesquisadores de diversas áreas são atraídos por ele.

Este artigo apresenta duas possíveis soluções para o problema do Caixeiro Viajante.

O primeiro algorítimo é a primeira solução obvia, a qual é implementada utilizando o método de força bruta, que compara todas as possibilidades até encontrar a melhor escolha. Este algorítimo vai retornar a melhor solução, porém quando considera-se um conjunto relativamente grande de dados a serem analisados, o tempo gasto para encontrar a resposta será exponencial, logo o tempo gasto pelo algorítimo será n! (fatorial), com “n” sendo o número de cidades.

O segundo algorítimo é baseado na heurística do vizinho mais próximo, que procura sempre o vizinho de menor custo até ter visitado todos os vértices do grafo.

Este algorítimo vai retornar a possível melhor escolha, porém, por ser uma heurística, pode, às vezes, deixar de lado alguns caminhos mais curtos.

Ambos os algoritmos serão discutidos no decorrer deste artigo.

**2. Algorítimo de Força Bruta**

Um algorítimo que implementa uma solução para um problema utilizando o método de força bruta é aquele que compara todas as possíveis soluções até encontrar a melhor solução, ou a solução mais correta.

Para o problema do Caixeiro Viajante, o algorítimo permuta todas as possibilidades de caminho entre as n cidades do grafo, depois soma os custos de cada caminho e compara todos os resultados até encontrar o de menor custo.

**1 – Algorítimo Força Bruta para o problema do Caixeiro Viajante**

**c = 0, custo = 9999999, conta = 0, N = número de cidades;**

**Grafo[N][N];**

**cidades[N] = {0,…,N-1};**

**Troca (x, y)**

**temp;**

**temp = x;**

**x = y;**

**y = temp;**

**Copia\_array(cidades, n)**

**i, soma = 0;**

**Para i = 0; i <= n; i++**

**soma += grafo[cidades[i % N]][cidades[(i + 1)%N]];**

**Se custo > soma**

**custo = soma;**

**Permuta(cidades, i, n)**

**j, k;**

**Se i == n**

**Copia\_array(cidades, n);**

**Senão**

**Para j = i; j <= n; j++**

**Troca((cidades + i), (cidades + j));**

**Permuta(cidades, i + 1, n);**

**Troca((cidades + i), (cidades + j));**

A solução deste problema por este algorítimo tem consumo T(n) = **Θ(n!)**, onde n é a quantidade de cidades, mostrando-se eficiente em termos de tempo apenas para casos de poucas variáveis. Este algorítimo é ineficiente principalmente por ter que refazer a solução de vários subproblemas já calculados anteriormente.

**3. Heurística baseada no Algorítimo do Vizinho mais Próximo**

Este algorítimo consiste em partir sempre para o caminho de menor custo dentre as opções de caminho que houverem a partir da posição atual, enquanto todos os n vértices não forem visitados.

No caso do Problema do Caixeiro Viajante, partindo da cidade inicial, o algorítimo irá sempre procurar o vizinho de menor custo e partir para ele. Feito isso, irá marcá-lo como visitado e checar se todas as n cidades já foram visitadas. Caso negativo, o algorítimo parte para o vizinho mais próximo da cidade onde está, marca-o como visitado e checa novamente se todos foram visitados. Estas etapas se repetem por meio de recursão até que todas as n cidades tenham sido visitadas, sem passar mais de uma vez por nenhuma delas.

**N = número de cidades;**

**custo, v = 0, menor = 9999999;**

**cidades[N] = {0,…,N-1};**

**inicial = cidades[0];**

**visitados = "";**

**grafo[N][N];**

**menorVizinho(int cidadeAtual)**

**int proximaCidade = cidadeAtual;**

**Para int i=0; i<N; i++**

**Se grafo[cidadeAtual][i] != 0**

**E visitados.nãoContem(i)**

**E grafo[cidadeAtual][i] < menor**

**menor = grafo[cidadeAtual][i];**

**visitados += cidadeAtual;**

**proximaCidade = i;**

**custo += menor;**

**menor = 9999999;**

**v++;**

**Se v < N**

**menorVizinho(proximaCidade);**

**Senão**

**custo += grafo[cidadeAtual][0];**

**main()**

**menorVizinho(inicial);**

**custo -= 9999999;**

**1 – Heurística baseada no Alg. do Vizinho mais Próximo para o PCV**

Este algorítimo executa rápido e é de fácil implementação, porém, por ser uma heurística de baixo custo, com T(n) = **Θ(n²)**, nem sempre encontra o melhor resultado, podendo às vezes deixar caminhos mais curtos de lado devido ao seu critério de decisão.

O tempo de O(n²) se dá pois, cada vez que visitar uma das n cidades e entrar na função *menorVizinho*, precisará comparar os n vizinhos da cidade atual para descobrir o de menor custo, ou seja, n cidades x n vizinhos.

**4 Discussão sobre os limites e apresentação da base de entrada**

Os limites que serão vistos e discutidos aqui foram baseados e executados no seguinte ambiente:

Intel Celeron 1007U @ 1.50GHz 1.50GHz (clock real de 1496.97MHz) Dual Core, cache L1 32KB (X2), cache L2 256KB (X2), cache L3 2MB, 2GB de memória RAM (DDR3).

Sistema Operacional: Microsoft Windows 7 Ultimate (64 bits).

Os códigos-fonte de ambos os algorítimos foram compilados com o gcc 4.9.2.

Os tempos foram adquiridos do próprio console de execução, tirando-se de uma média de ao menos 5 execuções para cada valor de entrada.

**4.1 Limite por Alg. Força Bruta**

O algorítimo por força bruta apresentou comportamento exponencial para as seguintes entradas:

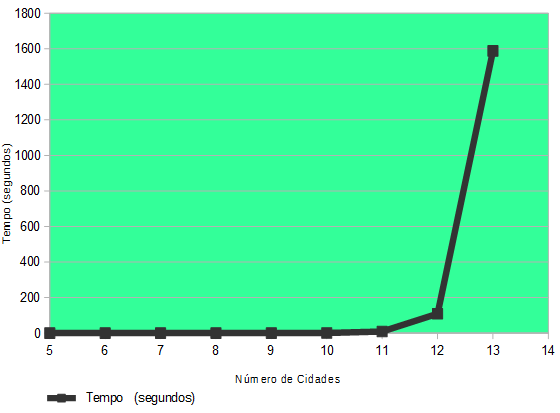
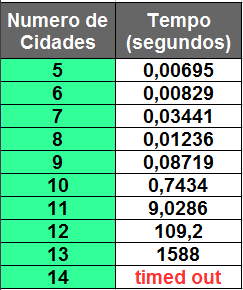


Tabela 1 e Gráfico 1: resultados do algorítimo por força bruta

O algorítimo não pôde ser executado para a quantidade de 14 cidades, pois após mais de 180 minutos, ou **3 horas** (!) de execução, ainda não havia finalizado, considera-se então o resultado de *timed out* para esta quantidade de cidades.

Portanto, o limite para este algorítimo fica definido em 13 cidades. Sem dúvidas, este algorítimo possui um crescimento exponencial.

**4.2 Limite por Heurística baseada no Alg. do Vizinho mais Próximo**

O algorítimo por vizinho mais próximo apresentou os seguintes resultados:

****

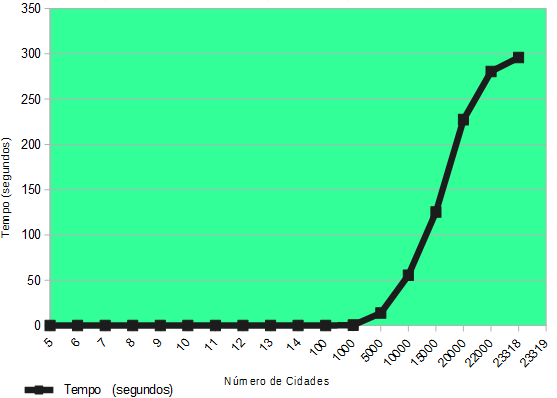
****

Tabela 2 e Gráfico 2: resultados do algorítimo do Vizinho mais Próximo

O algorítimo foi executado para uma base de entrada extremamente superior aos testes efetuados para o algoritmo por força bruta. De fato, nota-se que o crescimento da base de entrada foi muito maior, e que o tempo de execução foi relativamente baixo.

Para este, uma limitação de hardware foi atingida durante os testes, onde, com o número de entradas **22319** o compilador não conseguia mais executar o programa por conta do seu alto uso de memória RAM, chegando a picos próximos a 1.3Gb (!), dos 1.89Gb disponíveis para uso. Este número é definido na tabela como **overflow**.

Portanto, não foi encontrado um limite para a heurística baseada no algorítimo do Vizinho mais Próximo, cabendo este ser limitado por razões técnicas, tais quais como arquitetura, memória e etc.

**5 Benchmark**

O benchmark foi realizado comparando os dois algorítimos com diferentes números de entradas.

A geração da base de testes foi criada utilizando *rand - random number generator*, que está disponível na biblioteca stdlib.h das linguagens C; C++.

****

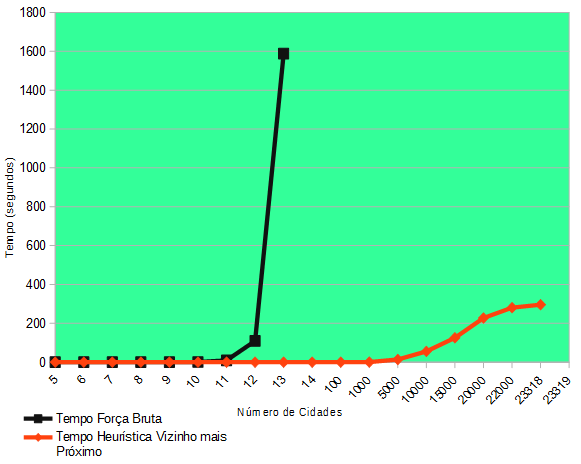
Tabela 3: resultados comparativos entre os dois algoritmos para uma base gerada aleatoriamente

Gráfico 3: Benchmark entre os dois algorítimos.

Através do gráfico nota-se que a heurística proposta tem um crescimento extremamente menor que o algorítimo de força bruta, que cresce exponencialmente.

**6 Compensação entre tempo de execução e a qualidade dos resultados**

Analisandoos algoritmos apresentados, nota-se que uma diminuição na complexidade de tempo proporcionou um ganho no tempo de execução, porém uma queda na qualidade do resultado.

No caso do algorítimo por Força Bruta pode-se notar que o elevado esforço temporal necessário para o cálculo exaustivo até se encontrar a solução, resulta na solução ótima para o problema. Nota-se também que ele está longe de entregar a solução em tempo ótimo, e que possui um limite reduzido, pois com entradas reduzidas ele sempre fornece a solução ótima, mas não é capaz de suportar uma entrada de dados muito grande

No algorítimo que utiliza a técnica do Vizinho mais Próximo, nota-se que o esforço temporal necessário para encontrar uma solução já não é tão elevado. A complexidade de tempo deste é menor que a do algorítimo de força bruta, porém a solução nem sempre será a solução ótima e sim uma aproximação da mesma.

Portanto, conclui-se que a diminuição da complexidade de tempo proporcionou uma diminuição do tempo de execução, entretanto, diminuiu a qualidade do resultado final.

**7 Curva n X Taxa de aproximação**

Abaixo são apresentados os dados obtidos em testes realizados com uma matriz de distâncias gerada aleatoriamente para cada quantidade de cidades, sendo testada em ambos os algorítimos.

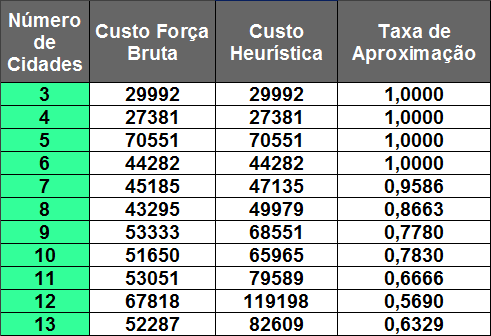


Tabela 4: Taxa de aproximação (Força Bruta/Heurística)

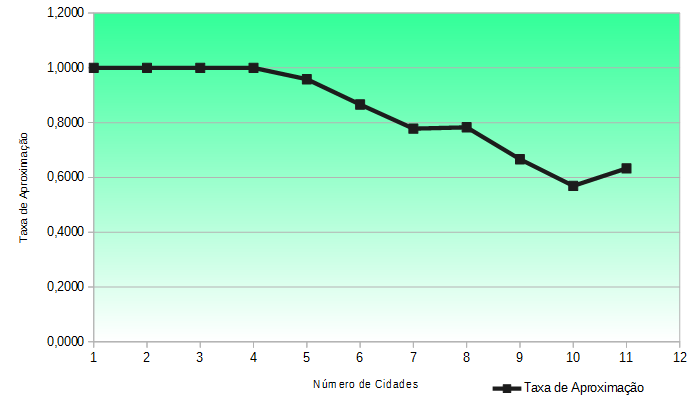


Gráfico 4: Taxa de aproximação/Número de cidades

Observa-se que a taxa de aproximação média para os valores testados é de 0,8413 (84,13%). Com isso se pode notar que o desempenho do algorítimo heurístico para entradas pequenas é razoavelmente bom, porém quando o número de entradas aumenta a taxa de aproximação ao resultado ótimo tende a diminuir consideravelmente, tornando-o um algorítimo ineficaz.

**8 Conclusão**

Escolher um algorítimo de aproximação (heurística) para o problema do Caixeiro Viajante envolve muitas decisões. O objetivo é uma solução com uma precisão muito alta ou que execute em um tempo razoavelmente baixo?

Este artigo apresentou dois possíveis algorítimos para solucionar o problema do caixeiro viajante, o primeiro algorítimo através da força bruta, e o segundo uma heurística baseada na técnica do Vizinho mais Próximo.

No caso deste problema, quanto maior o custo, melhor o resultado e maior o tempo para atingi-lo, quanto menor o custo, pior o resultado, porém o tempo para atingi-lo diminui consideravelmente.

Podemos notar que através da heurística proposta os limites de entrada aumentaram consideravelmente, e também se diminuiu muito o tempo para se atingir uma aproximação da melhor resposta, mas nem sempre encontrando uma boa aproximação.

Portanto, conclui-se que o algorítimo heurístico apresentado atingiu sua proposta de execução em tempo polinomial, e que o mesmo é útil quando se procura minimizar o tempo de execução, porém não é útil e nem eficaz quando se busca um resultado ótimo em uma grande base de entradas.

**9 Referências**

Cormen, Thomas H. ; Leiserson, E. Charles; Rivest, L. Ronald; Stein,

Clifford R.; (2002) “Algoritmos – Teoria e Prática”, Editora Campos.

Nilsson, Christian. ; “Heuristics for the Traveling Salesman Problem”,

Linköping University (chrni794@student.liu.se).

Disponível em: <https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf> Acessado em: 11/07/2017 – 15:28.