12ª Jornada Científica e Tecnológica
INSTITUTO FEDERAL Sul de Minas Gerais
ISSN 2319-0124

ANÁLISE DE COMPLEXIDADE DO USO DE PROGRAMAÇÃO DINÂMICA NO PROBLEMA DO CAIXEIRO VIAJANTE

Andreza C. B. SERRA[1]; Jairo de S. JÚNIOR[2]; Diego SAQUI[3]

RESUMO

O presente artigo tem como principal objetivo uma avaliação, com base em relato de experiência, da eficácia do uso de Programação Dinâmica em algoritmos solucionadores do Problema do Caixeiro Viajante na redução de seu custo computacional, nesse sentido buscou-se comparar os desempenhos de um algoritmo solucionador com e sem o uso de Programação Dinâmica. Como métrica para comparação foi utilizada a técnica de análise de algoritmos, por meio do cálculo da complexidade computacional de cada uma das soluções e testes de tempo de execução dos mesmos. O resultado obtido foi que o uso da programação dinâmica diminui a complexidade algorítmica do Problema de **O**(n!) para **O**(n^2 * 2n). Em linhas gerais, descobriu-se que é possível passar de um tempo que cresce em fatorial para um que cresce em exponencial, o que representa um bom avanço, apesar de não resolver totalmente o problema de custo computacional para uma quantidade muito grande de nós.

Palavras-chave:
análise de algoritmos; complexidade computacional; otimização combinatória; problemas NP-Completos.

1. INTRODUÇÃO

O Problema do Caixeiro Viajante (PCV) é um problema de otimização combinatória apresentado por pesquisas de Dantzig, Fulkerson e Johnson em 1954 que consiste na tarefa de encontrar uma rota em um dado conjunto de cidades com o menor comprimento possível. NILSSON (1982) definiu o problema como: encontrar um caminho com a menor distância possível que parta de uma cidade, visite todas as outras cidades única e exclusivamente uma vez, e retorne à cidade de onde partiu. O PCV já se mostra relevante sob o ponto de vista teórico-matemático, mas também possui aplicações práticas, como por exemplo: Fabricação de placas de circuitos eletrônicos; Raio-x cristalográfico; Sequenciamento de tarefas; e Roteamento de veículos;

Como consequência da grande quantidade de aplicações, busca-se ajuda em algoritmos que otimizem esses processos (que resolvam o PCV com eficiência). No entanto, o maior desafio encontrado nessa linha é que o PCV, quando aplicado na prática, possui um número de nós (cidades) que variam de centenas até milhares, e o problema se torna exponencialmente mais difícil de calcular a cada cidade adicionada à lista.

A busca por algoritmos solucionadores do PCV só começou a ter êxito com a evolução do poder de *hardware* e o desenvolvimento da teoria matemática. Mesmo assim, até hoje os algoritmos

[1]Discente, IFSULDEMINAS – *Campus* Muzambinho. E-mail: acbserra@gmail.com.
[2]Discente, IFSULDEMINAS - Campus Muzambinho. E-mail: jairosousajunior@gmail.com.
[3]Orientador, IFSULDEMINAS – *Campus* Muzambinho. E-mail: diego.saqui@muz.ifsuldeminas.edu.br.

que são capazes de resolver o problema satisfatoriamente não se mostram estáveis em tempo de execução (CONTE, 2002; SILVEIRA, 2000). Nesse sentido, diversas metodologias de programação podem minimizar o custo dessa solução, como a Programação Dinâmica (PD), que tem como objetivo principal melhorar a complexidade de um algoritmo utilizando da técnica de memorização de estados já calculados.

Bellman (1962) publicou um dos primeiros trabalhos envolvendo a abordagem do PCV com o uso da técnica de Programação Dinâmica. Considerando o problema para **k** cidades, a técnica de PD sugere a utilização de conjuntos de memorização das distâncias entre cada uma das **k** cidades que se deseja visitar e a cidade de origem. Com essas distâncias já memorizadas, a quantidade de cálculos para encontrar o melhor caminho diminui e, consequentemente também diminui seu custo computacional.

Considerando esse contexto, este trabalho busca demonstrar até que ponto o uso de PD pode melhorar o tempo de execução de algoritmos para o PCV. Para isso, serão analisadas as complexidades e os desempenhos de tempo na execução de dois algoritmos: um com PD e outro sem.

2. MATERIAL E MÉTODOS

Para o desenvolvimento dos algoritmos foi utilizada a linguagem de programação *C++* em conjunto com o compilador GNU *Compiler Collection* (GCC) para compilar e testar o funcionamento dos mesmos.

Na abordagem com PD foi utilizada uma matriz de memorização de tamanho N x 2^n, onde N é a quantidade de vértices, sua função é guardar os resultados das operações já realizadas para utilizá-las posteriormente e reduzir a quantidade de operações calculadas pelo algoritmo. Para a manipulação do conjunto das cidades foi utilizado o conceito de máscara de bits (*bitmask*), uma técnica de programação que consiste em representar os valores de um conjunto de forma binária. Já na abordagem sem PD optou-se pela técnica trivial de força bruta, onde o algoritmo gera todas as combinações possíveis de nós para depois compará-las e decidir qual o melhor caminho.

Para comparar o desempenho das duas abordagens, foi utilizado o cálculo da complexidade de cada algoritmo linha a linha (notação **O**). Além disso, foram feitos testes de tempo de execução com números crescentes de nós no Problema.

3. RELATO DA EXPERIÊNCIA

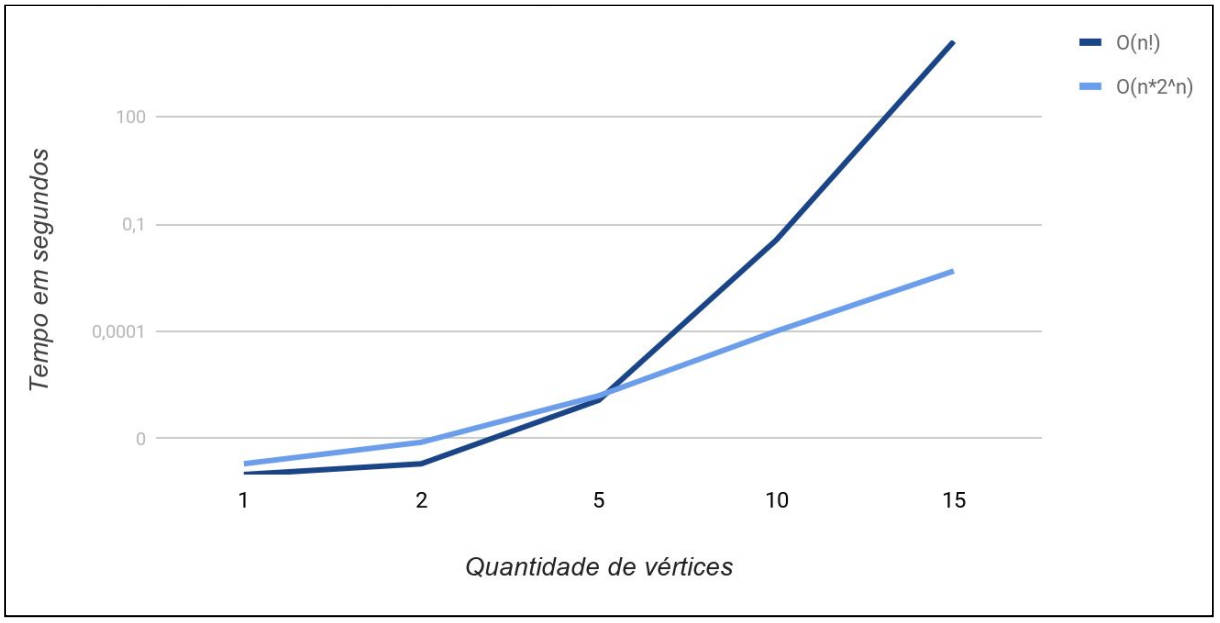
Para o algoritmo com PD, o momento de maior complexidade de tempo ocorre na criação da *bitmask*, ao verificar se os valores [1, n] já estão contidos na máscara. Se um valor ainda não estiver contido ele deve ser inserido na máscara por meio do comando *"shift"*, que "empurra" o bit **k** vezes à esquerda, para marcar aquela cidade como visitada.

No algoritmo sem PD, o fator de maior complexidade está no trecho de código "*do while*", onde a função de permutação busca e localiza todas as combinações existentes no laço de repetição, o que resulta num tempo de execução que cresce fatorialmente.

Como resultado (utilizando a notação *big* ***O***) foram obtidas as complexidades **O**(n!) para o algoritmo sem PD e **O**(n^2 * 2n) para o algoritmo com a PD aplicada. Matematicamente, à medida que a quantidade de valores de entrada aumenta, a função n! cresce mais que a função n^2 * 2n, isso pode ser observado na Figura 1.

Nessa Figura o tempo (notado em segundos) foi obtido experimentalmente, por meio da execução dos algoritmos em um computador e pode-se observar que o resultado tende ao comportamento assintótico das funções da notação do *big* ***O***.

Figura 1: Gráfico de tempo de execução dos algoritmos



5. CONCLUSÕES

Na prática, foi visto que o algoritmo solucionador sem PD era mais rápido que o com PD quando o número de cidades era pequeno, mas crescia muito rapidamente em tempo de execução quando o número de vértices aumentava. Já o algoritmo solucionador com PD se mostrou mais lento com um número de cidades pequeno, mas muito mais estável em tempo de execução quando o número de cidades aumentava.

Os resultados observados nos testes de execução vão de encontro com a análise algorítmica prevista pela teoria da complexidade, que inclui o PCV na categoria matemática dos Problemas NP-Completos. A otimização desse tipo de problema é muito difícil, visto que ainda não foi descoberto um modo de resolvê-los em tempo polinomial - um tempo viável computacionalmente.

O primeiro algoritmo usado trata o PCV de um ponto de vista reducionista, sem utilizar a programação dinâmica, calculando todas as possibilidades de combinações. Essa abordagem tem como complexidade O(n!). O segundo algoritmo também calcula todas as possibilidades, porém, ao utilizar a PD e armazenar os cálculos na matriz de memorização, essa abordagem elimina cálculos repetidos e, consequentemente, diminui a complexidade do algoritmo de **O**(n!) para **O**(n^2 * 2n).

É importante perceber que, mesmo com o uso da Programação Dinâmica, o PCV continua sendo NP-Completo. Porém, com o uso de PD, pode-se ir de um tempo fatorial para um tempo exponencial, gerando um impacto positivo na execução do algoritmo. Com isso, é possível afirmar que a técnica de PD detém um impacto significativo no algoritmo solucionador do PCV no que diz respeito ao seu tempo de execução, apesar de não solucioná-lo por completo.

REFERÊNCIAS

BELLMAN, Richard. Dynamic programming treatment of the travelling salesman problem. **Journal of the ACM (JACM)**, v. 9, n. 1, p. 61-63, 1962.

CONTE, Nelson. O problema do caixeiro viajante, teoria e aplicações. 2002.

DANTZIG, G.; FULKERSON, D; JOHNSON, S. Solutions of a large-scale traveling salesman problem. **Operations Research**, 1954.

NILSSON, Nils John – Principles of artificial intelligence. New York: edição de Birkhauser, 1982. **ISBN 978-3-540-11340-9**.

SILVEIRA, J. P. da. **O Problema do Caixeiro Viajante**. 2000. Disponível em: <http://www.mat.ufrgs.br/~portosil/caixeiro.html/>. Acesso em: 29 de mai. de 2020.