# Aplicação de Algoritmo Genético no Problema do Caixeiro Viajante

Diego Luiz Cavalca – [diego.cavalca@dc.ufscar.br](mailto:diego.cavalca@dc.ufscar.br)

Joel David Costa Júnior – [joel.costa@dc.ufscar.br](mailto:joel.costa@dc.ufscar.br)

## **Resumo**

*(COMPLETAR RESUMO – DIEGO/JOEL)*

O presente trabalho visa analisar e comparar, em termos de desempenho computacional e qualidade das soluções obtidas usando Algoritmos Genéticos para construção das rotas para o PCV. Utilizou-se dados disponíveis na biblioteca TSPLib.

## **Introdução**

O Problema do Caixeiro Viajante (PCV) consiste em estabelecer uma única rota que passe por cada nó de um grafo, uma e apenas uma vez, retornando ao nó inicial no final do percurso.

Este roteiro Hamiltoniano deve ser feito de modo que a distância total percorrida seja mínima. O conjunto de rotas possíveis para o PCV Simétrico, isto é, se a distância do ponto "a" ao ponto "b" é igual ao do ponto "b" ao ponto "a",é "a", é o resultado de todas as combinações possíveis e pode ser calculado por ( n − 1 )! , sendo "n" o número de nós.

Este problema pertence a classe de problemas conhecida por NP-Hard, isto é, não existem algoritmos com limitação polinomial capazes de resolvê-lo. Assim a quantidade de passos de um algoritmo que possa solucioná-lo otimamente não pode ser dada por uma função polinomial do tamanho de sua entrada. Logo, apenas os problemas de pequeno porte podem ser solucionados de forma ótima.

Problemas maiores tornam-se tornam se inviáveis através dos métodos exatos, haja vista o esforço computacional que seria exigido para resolvê-los. Muitas abordagens de algoritmos heurísticos, que fornecem soluções factíveis próximas da ótima, têm sido desenvolvidas para resolver os problemas NP-Hard, apresentando soluções aproximadas e as algumas vezes ótimas para o problema.

O objetivo deste trabalho é analisar o Algoritmo Genético (AG) aplicado ao Problema do Caixeiro Viajante (PCV) a fim de alcançar um resultado eficiente em tempo viável. Neste contexto, serão apresentadas duas abordagens distintas do AG: clássica e sem operação de cruzamento (crossover).

Segundo (Benevides, Konowalenko, Costa, Nunes, & Barboza, 2011)⁠, existem diversas abordagens para o problema do caixeiro viajante utilizando algoritmos genéticos. Elas diferem entre si não apenas na questão dos parâmetros, mas também na forma de representar as soluções viáveis, de selecionar os indivíduos para reprodução e na maneira de definir os operadores genéticos.

Dentre das diversas abordagens existentes para a implementação do AG, este trabalho visa desenvolver duas variações destes algoritmos, sendo o primeiro um AG clássico, fundamentado principalmente pelo americano John Henry Holland, que utiliza técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação (ou crossing over). A segunda abordagem desenvolvida neste trabalho consiste em um AG levemente modificado, no qual não possui operação de cruzamento (cross-over).

Como defende (Senaratna, 2005)⁠, existem muitos problemas práticos que foram melhor resolvidos desta forma. Por exemplo, (Spears & Anand, 2015)⁠ dizem que para os módulos de rede neural e seus circuitos de controle, algoritmos genéticos, sem cruzamento, um desempenho muito melhor do que aqueles com crossover. Vale observar que há muitos casos na natureza onde organismos complexos evoluíram sem qualquer tipo de cruzamento (por exemplo rotíferos Bdelloid). Na verdade, conforme afirma (DE GARIS, 1990a),os biólogos consideram a operação de mutação, e não a de cruzamento, como a principal fonte de "matéria-prima" na evolução genética.

## **Aplicação do Algoritmo Genético para o Problema do Caixeiro Viajante**

### *(Algoritmo Genético Clássico - JOEL)*

….

### *(Completar Algoritmo Genético sem Crossover - DIEGO)*

Em seu experimento, (DE GARIS, 1990a) relata que um AG sem cruzamento supera o resultado de um AG clássico. Este fato é surpreendente, uma vez que um elemento chave considerado no estudo do AG é a operação de cruamento, conforme indica a literatura clássica. Entretanto, nota-se que De Garis modelou este AG com uma população extremamente pequena (20) neste estudo. (Spears & Anand, 2015)⁠ indicam que, uma vez que o AG é modelado sem operação de crossover, populações pequenas produzem resultados altamente variáveis, devido à amostragem extremamente escassa do espaço de busca.

Assim, um segundo algoritmo genético foi desenvolvido, no qual se extraiu a operação de crossover e aplicou-se a operação de mutação em larga escala, como se observa no pseudocódigo abaixo, e mais detalhadamente no Anexo 2.

Em linhas gerais, em cada geração, toda a população passa pela seleção, agrupados em quatro indivíduos por vez; deste quatro indivíduos, é selecionado o com menor aptidão (menor caminho, no contexto do problema). Uma vez selecionado o melhor indivíduo, os outros três sofrem o processo de mutação, aumentando assim a diversidade da população para a próxima geração. Vale notar que são utilizados três operações de mutações distintas, uma para cada indivíduo em questão. Esta abordagem se mostra interessante quando se visa a variabilidade da população.

Partindo do pressuposto que os indivíduos estão divididos em pontos I e J, as operações de mutação são:

1. ***Flip*** – os blocos I e J são rearranjados em ordem inversa entre si entre o intervalo de posções destes;
2. ***Swap*** – realiza a simples inversão de posições dos blocos I e J no vetor do indivíduo;
3. ***Slice*** – captura o bloco no intervalo I:J, desloca o último elemento para a primeira posição e rearranja os demais uma célula adiante.

Como supracitado, a abordagem do algoritmo genético sem crossover neste trabalho visa realizar uma comparação ao modelo clássico de AG existente na literatura para o Problema do Caixeiro Viajante, analisando se esta é uma alternativa viável para este contexto, dada a complexidade do problema em questão.

## **Metodologia**

Os algoritmos foram desenvolvidos utilizando a ferramenta MATLAB 2015a, testados sob o sistema operacional OSX El Captain 10.11.6, numa máquina com processador de 2.26GHz Intel Core 2 Duo, memória RAM de 8gb 1067MHz DDR3 e placa de vídeos NVIDIA GeForce 9400 256MB e 1Tb de HD.

Os parâmetros para configuração dos algoritmos utilizados nos testes se diferenciaram para cada Algoritmo Genético desenvolvido, de acordo com suas especificações, tentando manter uma métrica similar a fim de obter um resultado comparativo de ambas as abordagens ao final do trabalho.

Em ambos os casos, foram realizados testes com populações de 20, 100, 200 e 400 indivíduos. Cada população foi submetida a 100, 500, 1000, 3000, 5000 e 10000 gerações possíveis.

( COMPLETAR METODOLOGIA CLÁSSICO - JOEL )

Para os testes do Algoritmo Genético sem operação de *crossover*, não serão necessários outros parâmetros, uma vez que, conforme já citado, este realiza a operação de mutação em larga escala, abrangendo sob todo a população, em todas as gerações.

## **Resultados**

(ALGORITMO GENETICO CLASSICO - JOEL)

(ALGORITMO GENETICO SEM CROSSOVER - DIEGO)

Como próximos trabalhos, podem ser analisados os tempos computacionais envolvidos, a fim de mensurar o esforço computacional envolvido nas várias simulações distintas. No AG sem crossover, pode ser explorada novas formas de mutação, incluindo a diminuição do número de mutações realizadas.

## **Conclusão**

As regras que estabelecem o tamanho da população e da utilização da operação de cross-over são extremamente importantes para o desenvolvimento do AG. Recentes teorias indicam que a operação de cross-over é mais efetiva quando os exemplos amostrais sendo procurados.

Este princípio é atingido de maneira natual usando uma população de tamanho considerável, de acordo com o problema estudado. Portanto, pode ser deduzido que o cruzamento é mais efetivo quando aplicado em grandes populações. Os resultados deste estudo indicam esta tese, mesmo este sendo limitado a um problema em particular de programação genética.

É também interessante notar que embora as melhorias de desempenho rápido ocorrem com um tamanho da população menor, uma população maior ajuda o algoritmo genético encontrar melhores soluções. Isto é causado pela acumulação mais lenta de estatísticas mais exactas quando se utiliza a população maior. É evidente que a rápida acumulação de estatísticas precisas permitiria que o melhor dos dois mundos.

(REVISAR E COMPLETAR – DIEGO/JOEL)

## **Referências**

de Garis, H. (1990a). Genetic Programming: Building Nanobrains with Genetically Programmed Neural Network Modules, Proceedings of the International Joint Conference on Neural Networks, San Diego, CA, Junho 1990

Benevides, P. F., Konowalenko, F., Costa, D. M. B., Nunes, L. F., & Barboza, A. O. (2011). Aplicação de Algori itmos Genéticos e Simulated Annealing para o Problema do Caixeiro Viajante em uma Situação Real de Distribuição de Produtos. In *I Congresso Brasileiro de Engenharia de Produção*.

Senaratna, N. I. (2005). Genetic Algorithms: The Crossover-Mutation Debate. *Bachelor of Computer Science(Special) of the University of Colombo*, 1–22. Retrieved from http://www.geocities.ws/nis\_nisco/docs/GA.pdf

Spears, W. M., & Anand, V. (2015). A Study of Crossover Operators in Genetic Programming. *Statewide Agricultural Land Use Baseline 2015*, *1*. http://doi.org/10.1017/CBO9781107415324.004