Aplicação de Algoritmo Genético no Problema do Caixeiro Viajante

CCO-727 Otimização Inteligente de Sistemas Produtivos – Profº Edilson Kato

Diego Luiz Cavalca – diego.cavalca@dc.ufscar.br
Joel David Costa Júnior – joel.costa@dc.ufscar.br

RESUMO

O presente trabalho visa analisar e comparar, em termos de desempenho computacional, a qualidade das soluções obtidas usando Algoritmos Genéticos (AG) para solução do Problema do Caixeiro Viajante (PCV). O PCV foi o primeiro problema apresentado na literatura com o intuito de buscar um melhoramento nas rotas de veículos, no entanto trata-se de um problema NP-hard, ou seja, não é possível encontrar uma solução ótima em tempo computacional válido. Visto que soluções ótimas não são ideais para esse tipo de problema, métodos heurísticos são estudados para buscar resultados satisfatórios. Uma heurística muito utilizada são os AG's que tomam como base a teoria da evolução e a genética para o seu desenvolvimento. Este trabalho descreve dois tipos de desenvolvimento de um AG, aqui denominados de 'clássico' e 'sem *crossover'*, trazendo bons resultados em tempo computacional aceitável. Para o problema em questão, utilizou-se para teste a instância eil51.tsp que está disponível na biblioteca TSPLib. O melhor resultado conhecido para a base de dados TSPLib EIL51.tsp foi 426. Os algoritmos propostos nesse trabalho - tradicional e sem *crossover* - atingiram respectivamente 442.28 e 428.98 para o dado problema.

1. INTRODUÇÃO

O Problema do Caixeiro Viajante (PCV) consiste em estabelecer uma rota que passe por cada nó de um grafo, uma única 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 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 (Holland, 1975), que utiliza técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação (ou *crossing over* - *crossover*).

A segunda abordagem desenvolvida neste trabalho consiste em um AG levemente modificado, no qual não possui operação de cruzamento (*crossover*), onde a evolução das gerações ocorre apenas por uma derivação das operações de mutação.

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, 1990), os biólogos consideram a operação de mutação, e não a de cruzamento, como a principal fonte de *matéria-prima* na evolução genética.

2. APLICAÇÃO DO ALGORITMO GENÉTICO PARA O PROBLEMA DO CAIXEIRO VIAJANTE

2.1. ALGORITMO GENÉTICO CLÁSSICO

Os algoritmos genéticos são inspirados em modelos biológicos, e se fundamentam na Genética de Mendel (1865) e também na Teoria da Evolução de Darwin (1859), usando uma medida para avaliar a capacidade que os indivíduos de uma população têm para sobreviver e se reproduzir" (ARTERO, 2008. p 153). Com isto, busca-se transferir as características de indivíduos mais aptos e com fácil chance de reprodução, para a próxima geração, enquanto as características dos indivíduos menos aptos e, consequentemente, com menos chance de reprodução, são perdidas. Assim, espera-se a obtenção de novas gerações cada vez mais próximas da perfeição. Néia et. al. (2013) destaca que quanto mais um indivíduo se adapta ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes. Artero (2008) usa como exemplo o aprimoramento genético de animais, como os bovinos, onde são cruzados animais com maior capacidade de produção de leite, objetivando-se, nas novas gerações, melhorias nesta característica.

Coppin (2013) descreve o funcionamento dos algoritmos genéticos da seguinte maneira:

1. Gere uma população aleatória de cromossomos (esta será a população inicial).
2. Se o critério de terminação for satisfeito, pare. Caso contrário, siga para a etapa 3.
3. Determine a aptidão de cada cromossomo.
4. Aplique cruzamento e mutação a cromossomos selecionados, a partir da geração atual, para gerar uma nova população de cromossomos - a próxima geração.
5. Retorne à etapa 2

Quadro 1 – *Pseudocódigo* Algoritmo Genético

Fonte: COPPIN. 2013, p. 335-336.

2.2. ALGORITMO GENÉTICO SEM OPERAÇÃO DE CRUZAMENTO *(CROSSOVER)*

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 cruzamento, 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 através do código-fonte disponível no ANEXO I.

1. Gere uma população aleatória de cromossomos (esta será a população inicial).
2. Se o critério de terminação for satisfeito, pare. Caso contrário, siga para a etapa 3.
3. Determine a aptidão de cada cromossomo.
4. Seleciona iterativamente grupo de 4 indivíduos para toda a população; Separa o melhor destes e aplica a mutação nos 3 indivíduos restantes para gerar uma nova população de cromossomos - a próxima geração.
5. Retorne à etapa 2.

Quadro 2 – *Pseudocódigo do AG sem crossover.*

Fonte: *Autores.*

Em linhas gerais, em cada geração, após o cálculo da aptidão individual, toda a população passa pela seleção, agrupados em quatro indivíduos por vez; destes 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 utilizadas três operações de mutações distintas, uma para cada indivíduo em questão. Esta abordagem se mostra interessante quando se visa a variabilidade da população.

Neste sentido, observa-se que a operação de *crossover* proposta no algoritmo genético clássico é subtraída neste AG, sendo implementada uma operação hibrida visando gerar uma maior diversidade para a população, o que de fato se torna útil dada a característica complexa do problema, conforme abordado por (Seratna, 2010).

Partindo do pressuposto que os indivíduos estão divididos em pontos I e J, as operações derivadas de mutação são aplicadas nos três piores indivíduos da subpopulação selecionada:

A. ***Flip*** – os blocos I e J são rearranjados em ordem inversa entre si entre o intervalo de posições destes;
B. ***Swap*** – realiza a simples inversão de posições dos blocos I e J no vetor do indivíduo;
C. ***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.

3. 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.

3.1. DEFINIÇÕES E REGRAS DO PCV

O PCV não permite rotas com valores duplicados, ou seja, cada cidade deve ser visitada apenas uma vez e a rota deve ser encerrada na cidade de partida inicial.

3.2. PROJETO DE ALGORITMOS PROPOSTOS

3.2.1. ALGORITMO CLÁSSICO

O operador de cruzamento PMX faz a combinação dos cromossomos sem permitir que genes iguais apareçam na mesma prole, portanto, foi o usado na aplicação.

O cruzamento é feito assim que os pais são selecionados, estes são passados como parâmetro para o método *crossoverPMX* que retorna os filhos resultantes da combinação de genes. Esse processo é realizado até preencher toda a nova população que deve ter o mesmo tamanho da população de progenitores.

A mutação não é um operador fundamental para o algoritmo, mas seu uso pode ser fundamental para se alcançar bons resultados. Os indivíduos da aplicação têm 3% de chance de sofrerem mutação, e esta aplicada foi por inversão.

3.2.2. ALGORITMO SEM *CROSSOVER*

Para este AG, 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.

Em tempo, para este algoritmo, 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.

4. RESULTADOS

Para a avaliação do PCV os dados usados foram os do problema EIL51.tsp, disponibilizado pela TSPLIB (TSPLIB, 2010), que possui tamanho de 51 nós. Os resultados obtidos variavam de acordo com mudanças na quantidade de gerações e no tamanho da

população de indivíduos. Custos menores puderam ser alcançados com o aumento de gerações e de indivíduos das populações, entretanto, por conta desses acréscimos o tempo de execução da aplicação tornou-se mais longo.

4.1. AG CLÁSSICO

O melhor resultado alcançando pelo algoritmo foi **442.281**, atingindo um resultado abaixo de 10% em comparação com o melhor resultado obtido na literatura para este conjunto de dados, atingindo assim à meta proposta. Este resultado foi alcançado com uma população de **10000** indivíduos e um limite máximo definido de **200** gerações; o resultado em si foi encontrado na geração número **125**.

A tabela 1 mostra um resumo geral dos resultados do AG clássico.

Min.	*Median*	*Mean*	*Stand. Desv.*	*Max.*
442.28	567.00	578.00	75.00	795.00

Tabela 1: *Resumo geral de resultados do AG clássico.*

A taxa de mutação foi definida em 3%. Se esta taxa for muito alta as características boas de alguns filhos gerados no cruzamento podem ser perdidas, o que não é o objetivo do método, por isso o ideal para o algoritmo é manter uma taxa menor do que 5%. Menores custos puderam ser alcançados com o aumento de gerações e de cromossomos das populações. No entanto, quando as gerações chegavam à faixa de 150 até 200, a população não sofria evolução significativa, tomando isto como concluiu-se que, gerações com número superior a 200 não eram interessantes para o estudo.

A seguir, as figuras 1 e 2 mostram o comportamento deste resultado em relação as gerações:

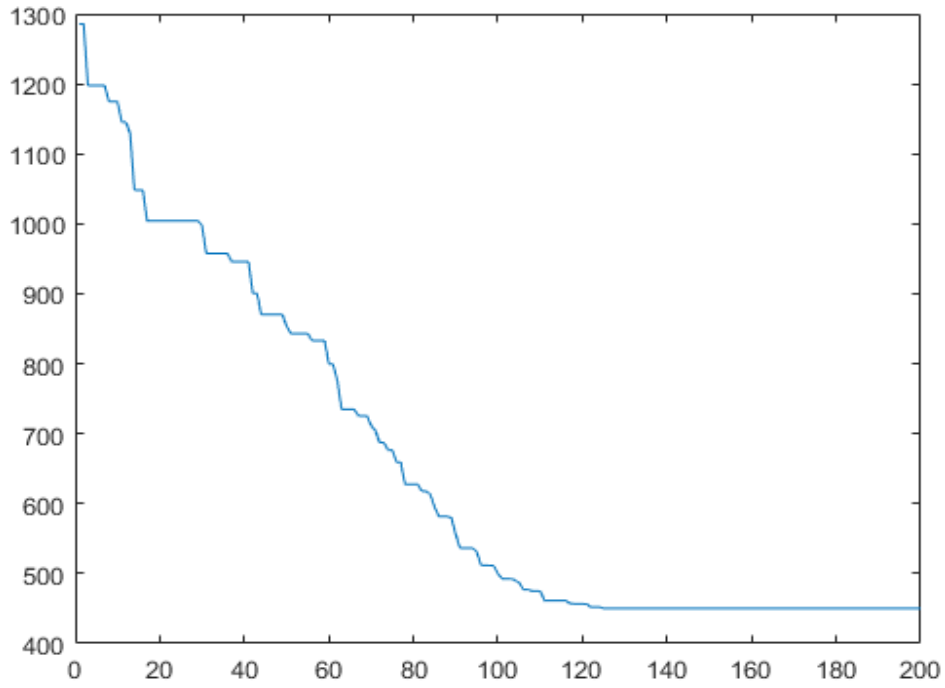

Figura 1: Relação entre fitness (caminho) e gerações do melhor resultado.

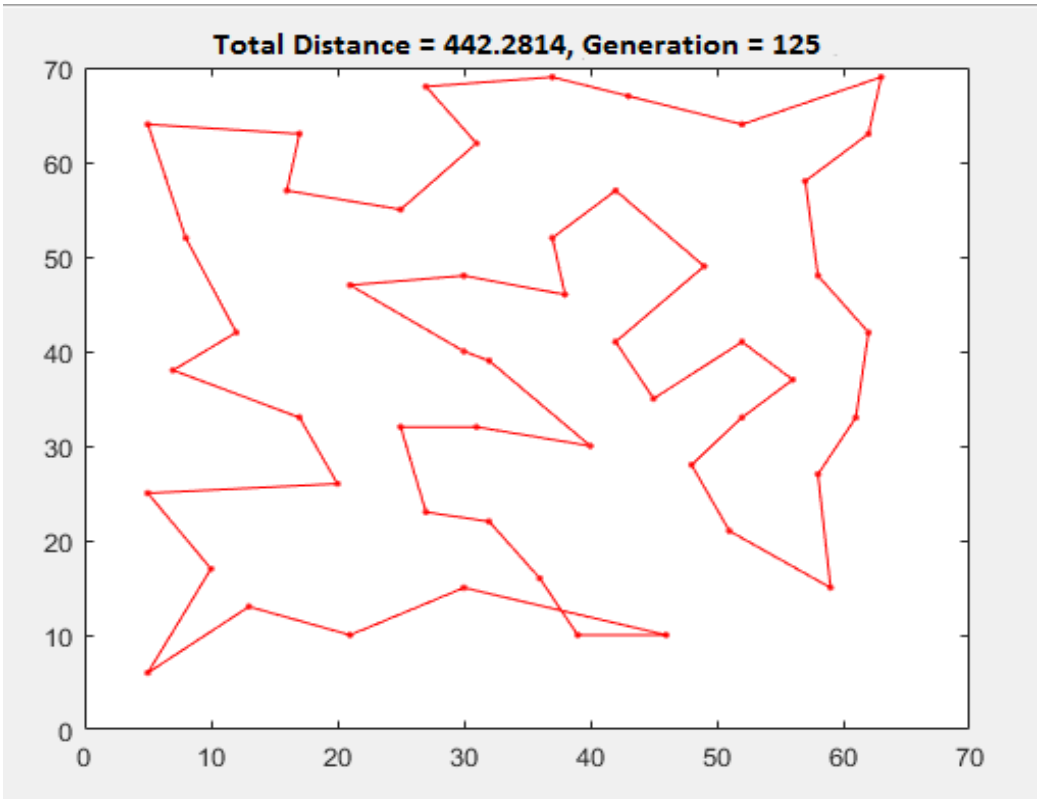

Figura 2: Representação do melhor caminho encontrado pelo AG clássico

Abaixo, a figura 3 mostra a distribuição dos resultados obtidos nos testes do AG:

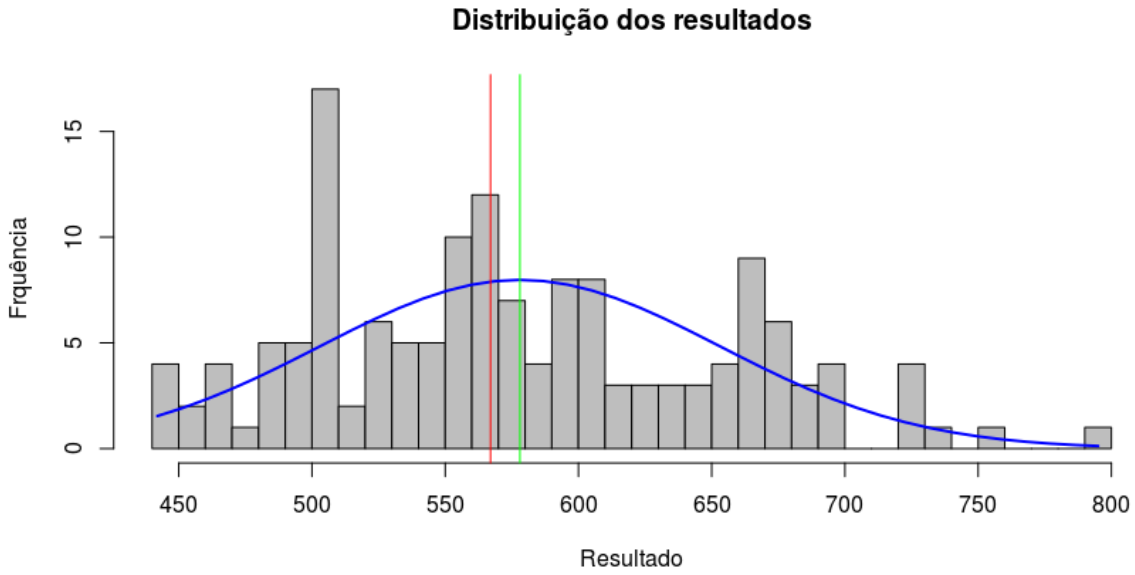

Figura 3 - *Distribuição dos resultados do AG*

Portanto, fica evidente que o comportamento geral deste AG é estável, com pouca diferença entre a média (linha verde) e a mediana (vermelha) para todos os casos, assumindo assim uma distribuição normal dos resultados.

4.2. AG SEM *CROSSOVER*

Conforme o resumo geral sobre a performance deste AG apresentado abaixo, (reportado de forma completa no ANEXO I), o melhor resultado obtido para o PCV aplicado ao conjunto de dados EIL51.tsp foi **428.98**:

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max*
428.98	439.0	445.6	492.1	477.3	860.0

Tabela 2: *Resumo geral de resultados do AG sem crossover.*

É interessante notar que este resultado está a apenas **2.98** pontos do melhor resultado conhecido para este conjunto de dados. Além disso, a variância observada teve valor de 96.3, mostrando uma consistência no desempenho geral deste AG.

Também vale ressaltar que a mediana dos resultados obtido situou em **445.6**, atingindo um resultado abaixo de 10% em comparação à meta, conforme pode ser vista representada pela linha vertical vermelha no gráfico de distribuição dos resultados abaixo:

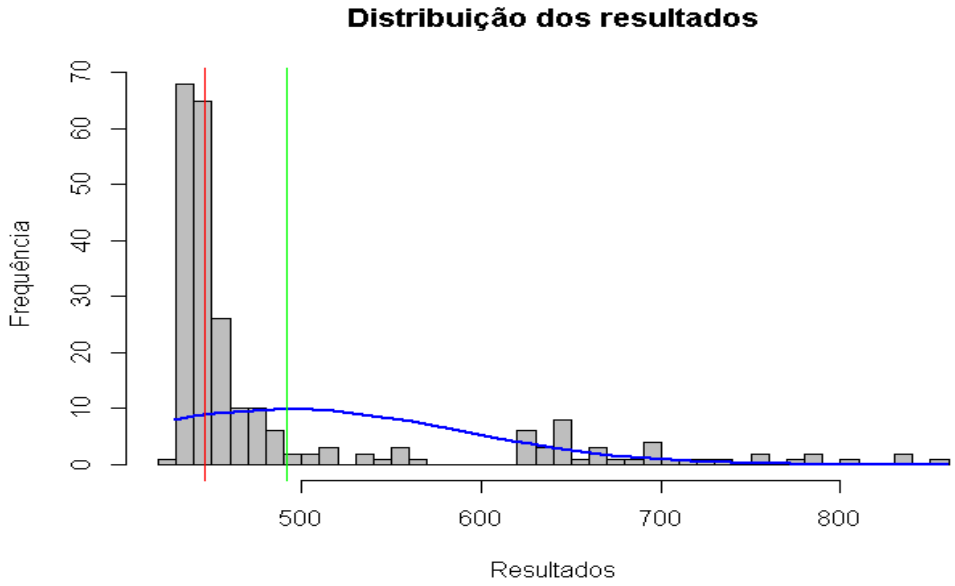

Figura 4 - Distribuição dos resultados do AG

Vale ressaltar que o melhor resultado deste AG foi encontrado com o número populacional de **100** indivíduos e um limite máximo definido de **3000** gerações; o resultado em si foi encontrado na geração número **1948**. O comportamento deste resultado em relação as gerações pode ser visualizado abaixo:

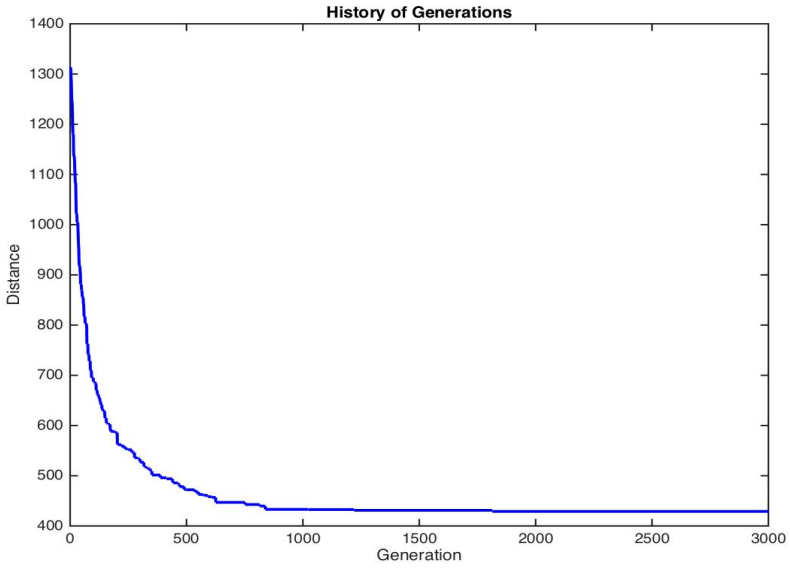

Figura 5: Relação entre fitness (caminho) e gerações do melhor resultado.

Em tempo, a distribuição visual do caminho encontrado pode ser visualizada a seguir:

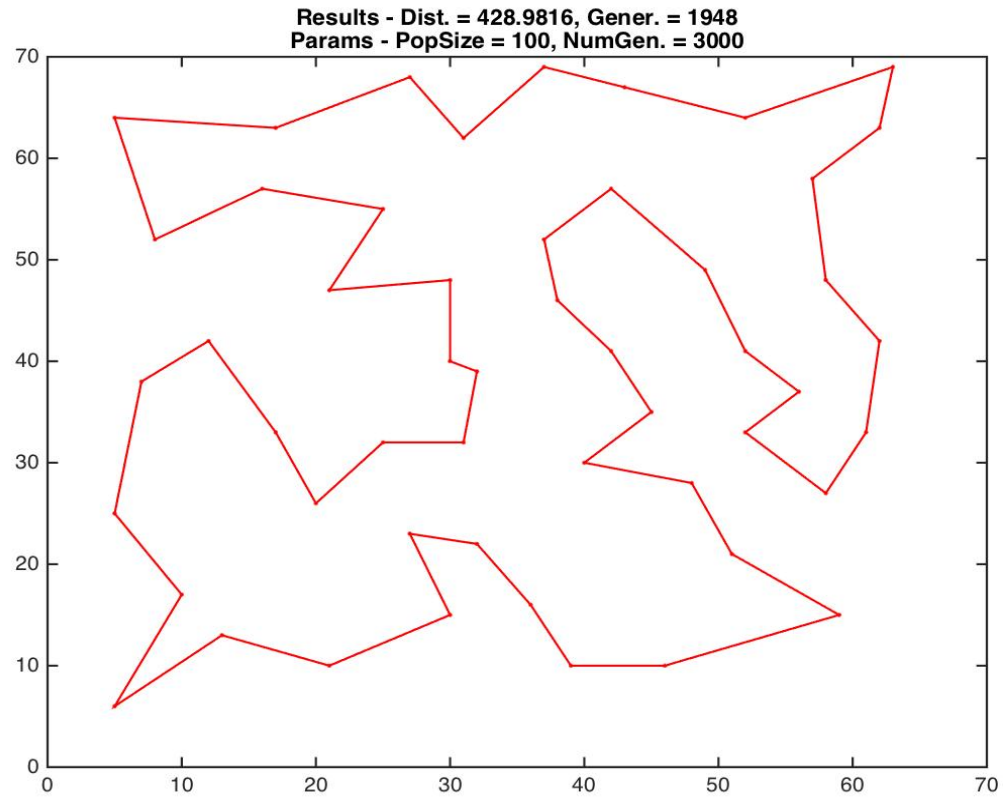

Figura 6: Representação do melhor caminho encontrado pelo AG.

Portanto, o AG sem *crossover* se mostra eficiente para o PCV, obtendo um resultado extremamente próximo ao melhor encontrado, ainda que testado em sob uma pequena quantidade de configuração de parâmetros.

Dada a característica complexa do PCV, este resultado corrobora a tese de que ao remover a operação de cruzamento do AG obtém-se uma maior variabilidade populacional, apoiando a aproximação a rota de menor custo a cada geração, se mostrando mais eficiente quando comparado ao AG clássico, conforme abordado neste trabalho.

5. CONCLUSÃO

As regras que estabelecem o tamanho da população e da utilização da operação de *crossover* são extremamente importantes para o desenvolvimento do AG. Esta definição é atingida de maneira natural 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 exatas quando se utiliza a população maior. É evidente que a rápida acumulação de estatísticas precisas permitiria que o melhor dos dois mundos.

Além disso, fica evidente que diante da característica complexa do PCV, com restrições quanto a composição do cromossomo, é válido analisar diferentes estratégias na implementação do AG a fim de lidar com a diversidade da população entre as gerações, o que leva assumir mecanismos híbridos nas operações do algoritmo genético, como foi proposto neste trabalho, ao abordar um AG sem operação de *crossover*, no qual garante a diversidade da população através da mutação em larga escala dos indivíduos a cada geração.

Como próximos trabalhos, podem ser analisados os tempos computacionais envolvidos, a fim de mensurar o esforço estrutural nas várias simulações distintas. No AG sem *crossover*, podem ser exploradas novas formas de mutação, incluindo a própria diminuição do número de mutações realizadas. Novos estudos podem ser direcionados a fim de explorar a correlação entre as variáveis envolvidas nas configurações de cada AG proposto no trabalho, assim definindo com exatidão quais parâmetros, e em quais escalas, afetam diretamente os resultados de maneira eficiente.

REFERÊNCIAS

ARTERO, A. O. **Inteligência artificial teórica e prática.** São Paulo: São Paulo, 2008.

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*.

COPPIN, B. Algoritmos genéticos. In: _____. **Inteligência artificial.** Rio de Janeiro, 2013. cap. 14, p. 334-360.

HOLLAND, J. H. **Adaptation in natural and atificial system.** An Arbor. Universidade of Michingan Press, 1975.

NÉIA, S. S.; ARTERO, A. O.; CANTÃO, L. A. P.; CUNHA, C. B. Roteamento de veículos utilizando otimização por colônia de formigas e algoritmo genético. In: LOPES, et al. (Eds.), **Meta-heurísticas em pesquisa operacional**. cap. 14, p. 219-236, 2013.

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

TSPLIB - Traveling Salesman Problem Library. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/> Acesso em 30 ago. 2016.

ANEXO I - ALGORITMOS DESENVOLVIDOS E DEMAIS INSUMOS

O código-fonte, resultados, análise dos dados (linguagem R), e demais arquivos de cada AG desenvolvido neste trabalho estão disponíveis em https://github.com/diegocavalca/mestrado/tree/master/CCO-727/Trabalho1-AG_TSP

A fim de organização e objetividade, cada AG esta disponível nos seguintes endereços:

1. **AG clássico:** https://github.com/diegocavalca/mestrado/tree/master/CCO-727/Trabalho1-AG_TSP/AG1

2. **AG sem *crossover*:** https://github.com/diegocavalca/mestrado/tree/master/CCO-727/Trabalho1-AG_TSP/AG2