Algoritmo genético para o Problema do Caixeiro Viajante

Alexandre de Sousa Martins[1], Clara Carolina Amaro[2]
[1,2]Institute of Technological Sciences, Universidade Federal de Itajubá, Itabira, Minas Gerais, Brazil
alexandresmartins@unifei.edu.br, claracamaro@unifei.edu.br

***Resumo*—O problema do caixeiro viajante é um problema Não-Polinomial, considerado de grande complexidade, que tem como ponto principal descobrir o menor caminho entre uma quantidade finita de pontos com uma distância conhecida entre si, com a condição de não passar duas vezes pelo mesmo ponto e voltar ao ponto inicial passando por todos. Um método que se pode utilizar para fazer esta implementação é o algoritmo genético, que tem como objetivo otimizar a busca chegando em um conjunto de soluções. O objetivo deste trabalho é implementar um algoritmo genético utilizando a linguagem de programação *R* para encontrar o melhor caminho e apresentá-lo.**

***Palavras-chave*—Algoritmo genético, Caixeiro viajante.**

I. Introdução

O algoritmo genético é uma ferramenta de busca utilizada para encontrar soluções aproximadas de otimização e busca, implementado por John Holland [1]. São baseados na teoria evolucionista de Darwin, representam cada indivíduo de uma população como sendo um conjunto de caracteres ou bits e associam a ele uma aptidão ou adequação *(fitness)*. O algoritmo genético faz parte de uma classe particular dos algoritmos evolutivos que usam como base a biologia evolutiva como mutação e seleção natural, entre outras. A implementação do algoritmo genético acontece em forma de simulação de computador em que se utiliza uma população de representações abstratas. Ele se diferencia dos outros métodos de otimização tradicionais pelo fato de utilizar transições probabilísticas e não regras determinísticas e não necessita de conhecimento do problema, mas sim a forma de avaliação do resultado. Os resultados são apresentados como uma população de soluções, ou seja com várias soluções e por basear-se em um conjunto de soluções possíveis sem levar em conta os parâmetros de otimização. O algoritmo genético tem como os principais componentes: função-objetivo, indivíduo, seleção e reprodução.

Um bom exemplo de onde se pode utilizar o algoritmo genético é o problema do caixeiro viajante. O problema do caixeiro viajante é um problema onde tenta-se determinar a menor rota para percorrer uma série de cidades, retornando para a cidade de origem, porém, não se pode visitar a mesma cidade duas vezes, deixando assim o problema com uma complexidade elevada, sendo considerado um problema de otimização NP-difícil.

978-1-7281-7539-3/20/$31.00 ©2020 IEEE

II. Revisão bibliográfica

Com base no objetivo deste trabalho, será feita uma breve explicação dos componentes principais do problema e da solução implementada. Primeiramente , será descrita a estrutura de um Algoritmo Genético e o que é o problema do caixeiro viajante.

A. Algoritmo Genético

Quando se trata de algoritmo genético o ponto inicial é a criação da população de indivíduos, após a criação da população alguns indivíduos serão selecionados para sofrerem algumas operações, serão estas operações que determinarão os descendentes para a nova população. Estas operações se repetirão até que se chegue em uma solução ideal ou um critério de parada seja alcançado. Antes de dar prosseguimento é importante ressaltar o significado de alguns termos utilizados em algoritmos genéticos, sendo eles:

- População: é o conjunto de indivíduos que são soluções de determinado problema;
- Indivíduo: é o elemento da população formada pelo conjunto de genes;
- Gene: é o cromossomo do indivíduo;
- Cruzamento: é o processo de troca ou combinação de gene de dois ou mais indivíduos, obtendo assim novos indivíduos;
- Mutação: operação que modifica os genes do indivíduo [2];
- Função objetivo: é onde permite-se o cálculo da aptidão de cada indivíduo;

A estrutura de um algoritmo genético é fazer a entrada com o tamanho da população e obter na saída a população de soluções, durante a execução do algoritmo serão seguidos os passos de selecionar um indivíduo fazer o cruzamento, ter a mutação dos descendentes gerados e avaliar os novos indivíduos e por fim verificar se satisfaz a condição proposta para a solução do problema. Ao utilizar o algoritmo genético tem-se algumas vantagens como: realizar buscas simultâneas, para a sua implementação pode se utilizar tanto parâmetros discretos ou contínuos e tem flexibilidade para otimizar as funções objetivas.

B. Problema do caixeiro viajante

O problema do caixeiro viajante é um problema que tenta determinar a menor rota para percorrer uma série de cidades

(visitando uma única vez cada uma delas), retornando à cidade de origem.

Ele é um problema de otimização NP-difícil inspirado na necessidade dos vendedores em realizar entregas em diversos locais (as cidades) percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível [3], podendo ser aplicado em diversos segmentos como manufatura, telecomunicações, entre outros. Por ser um NP o problema do caixeiro viajante não tem um algoritmo eficiente que o resolva.

III. Materiais e métodos

Para a implementação, primeiro foi definido o conjunto de dados que seria utilizado através das instâncias presentes na *TSPLIB* [4] que é uma biblioteca que armazena diferentes instâncias para o Problema do Caixeiro viajante, após isso foi definida a linguagem que é a *R*, com as instâncias e linguagem definidas são escolhidos e implementados os componentes do algoritmo genético.

A. *Linguagem* R

A linguagem *R*, é uma linguagem de programação de código aberto que possui grande versatilidade e surgiu da necessidade de implementar soluções de análise, manipulação e visualização de dados. Contando com vários pacotes auxiliares é possível realizar implementações dos mais variados tipos, desde técnicas de redução dimensional como a Análise de componentes principais até algoritmos como o algoritmo genético. Para este projeto não foram utilizados pacotes auxiliares.

B. *Codificação do indivíduo*

A codificação do indivíduo é definida por meio de uma classe chamada "Individuo"definida pela função *setClass* do *R*, os atributos, com excesão do cromossomo devem ser inicializados com o valor zero, sendo eles:

- noscidade: Que é um número entre 1 e 29 onde cada um representa uma cidade diferente do conjunto de dados;
- distCidades: que é uma lista que armazena a matriz de distâncias do conjunto de instâncias *bays.29.tsp* referente a 29 cidades que foi escolhido para essa aplicação.
- distt: é um atributo numérico que armazena a distância total da rota presente no indivíduo;
- notaAvaliacao: é um atributo numérico definido pela função de avaliação que atribui uma nota para a rota do indivíduo;
- geracao: representa numericamente a geração do indivíduo em questão;
- cromossomo: é uma vetor de caracteres que armazena 30 números, onde o primeiro e o segundo são iguais para representar o nó inicial e final da rota que devem ser iguais e os demais representam as outras cidades;

C. *Definição da população*

A população do algoritmo genético é armazenada numa classe também definida pela função /emphsetClass denominada "algoritmoGenetico", nessa estão implementados os seguintes parâmetros:

- tamanhoPopulacao: definido pelo valor escolhido como parâmetro para o tamanho da população;
- populacao: É uma lista que armazena valores do tipo da classe "Individuo", é neste parâmetro que de fato estão armazenados todos os indivíduos que compõem uma população
- geracao: representa numericamente a geração da população em questão;
- melhorSolucao: Representa a melhor nota dentre as calculadas para os indivíduos presente na geração;

D. *Parâmetros*

Os parâmetros usados na construção do algoritmo são os seguintes:

- Tamanho da população: 50 indivíduos;
- Probabilidade de mutação 5%;
- Número de gerações: 50 gerações;
- Taxa de cruzamento: 100%;
- Número de cromossomos: 30, 29 deles representando cada cidade da rota e o último repetindo a cidade inicial que representa o retorno ao início;

A definição desses parâmetros se deu por diferentes motivos, onde foi levada em conta o tempo de execução do algoritmo, a definição do conjunto de instâncias, e os resultados obtidos ao longo das execuções do código.

E. *Função de* Fitness

A seleção dos indivíduos se dá por meio de uma função de avaliação que é denominada *fitness* ou objetivo, por meio dela cada um dos indivíduos será avaliado de maneira individual e a ele será atribuída uma nota [5]. No caso do caixeiro viajante, cada nota será definida como o inverso do somatório das distâncias de cada rota de um indivíduo, onde cada número do cromossomo representa o índice de cada cidade. Para fins de visualização, a nota é multiplicada por 10.000, onde a melhor nota possível é igual a 1 dividido pela menor distância possível multiplicada por 10.000 e a pior nota se dá da mesma maneira mas com a distância da maior rota possível substituindo o valor da menor.

F. *Operadores genéticos*

Os operadores genéticos são de suma importância na execução de um algoritmo genético, dentre eles estão os operadores de seleção, mutação e cruzamento. Cada operador genético foi implementado por uma função específica e a forma como esses operadores são definidos é a seguinte:

- Seleção: Esse operador é definido basicamente pelo elitismo e também pelo método da roleta, onde em um escala menor, indivíduos com avaliação mais baixa também são selecionados mas a prioridade é dada a aqueles com melhor avaliação;
- Mutação: Este operador se dá por meio da mutação de inversão entre os alelos 17 e 22;

- Cruzamento: Ocorre o cruzamento no ponto de corte entre a primeira metade de um pai com a segunda metade do outro e os genes se cruzam, porém esse método é implementado de forma que não haja repetições de valores nos filhos;

G. Demais funções

Além das funções específicas para implementar a função de avaliação e operadores genético, também existem funções auxiliares a essas que realizam operações como a geração de cromossomos, somatórios, ordenações, exibições e inicializações para garantir o funcionamento correto do código;

IV. Resultados

Nesta seção é mostrado de maneira detalhada a análise dos resultados, primeiramente é plotado um gráfico para demonstrar a evolução do algoritmo ao longo de cada geração até sua convergência para uma única solução e após isso é feita durante 50 vezes a execução do algoritmo para os mesmos parâmetros da execução única com o valor do melhor *fitness* sendo exposto em relação ao número de execuções em um gráfico. Também será explicada a forma como executar o algoritmo tanto para uma única execução quanto para 50.

A. Execução única

Para os parâmetros definidos anteriormente, o algoritmo é executado de maneira única para verificar o valor do melhor *fitness* de cada geração entre as 50, por meio do gráfico da Figura 1 é possível ver a evolução do algoritmo até sua convergência, onde a maior nota significa que o caminho encontrado possui a menor distância entre as gerações. Nesta única execução o *fitness* encontrado foi de 2,20 para uma distância de 4544.

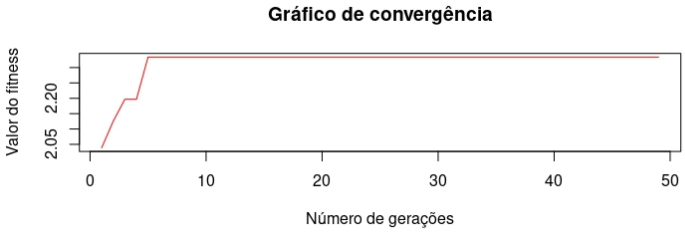

Figura 1. Evolução do algoritmo até sua convergência.

B. Análise estatística

Para uma análise estatística do algoritmo genético, com os mesmos parâmetros da execução única e sem alterações no código do algoritmo genético, foram feitas 50 execuções para determinar o melhor valor do *fitness*, dentre elas, como mostra o gráfico da Figura 2. Com os valores obtidos nessas execuções podem ser determinados a média, desvio padrão e variância desses resultados que são, respectivamente, 2,171175, 0,09282296 e 0,008616101 e mostram que a melhor, ou seja menor, rota tem uma distância de 4239.

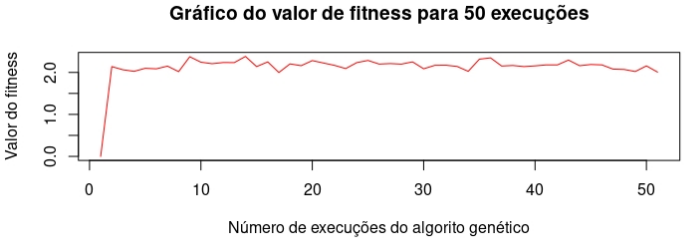

Figura 2. Gráfico do valor de *fitness* para 50 execuções.

C. Problemas no desenvolvimento

Durante a elaboração do algoritmo foi verificado nas execuções uma baixa variabilidade dos indivíduos, para amenizar este efeito foram feitas alterações no código de forma a aumentar essa variação por meio de ajuste nos parâmetros de seleção, mutação e cruzamento. Outro problema encontrado na questão do desenvolvimento foi a questão do tempo de execução para uma quantidade grande de indivíduos, por conta disso as execuções foram realizadas com 50 indivíduos, porém resultados de *fitness* melhores foram obtidos com 500 ou mais indivíduos na população de cada geração.

D. Execução

Primeiramente, é necessário um dispositivo onde a implementação da linguagem *R* esteja devidamente instalada e é recomendado que o *RStudio* também esteja presente. Para instalar as duas aplicações existem variações de acordo com o sistema operacional usado.

Com o *R* e *RStudio* devidamente instalados e configurados, para executar o algoritmo genético deve-se decidir entre o arquivo de execução única denominado *Single.R* na pasta do projeto ou o arquivo de execução múltipla denominado *Multi.R*, a única diferença entre os códigos presentes nesses dois arquivos é a confecção do gráfico onde o primeiro retorna um gráfico das melhores soluções de cada geração para uma única execução e o segundo retorna um gráfico das melhores soluções de cada execução do código dentre 50 execuções diferentes. Após a escolha do arquivo, basta abrir o *RStudio*, clicar em *File*, seguido de *Open File*, navegar até a pasta do projeto, selecionar o arquivo e clicar no botão *Source* que o algoritmo será executado totalmente e tanto os resultados no *Console* quanto o gráfico serão exibidos.

V. Conclusão

Para este trabalho foi de extrema importância compreender o funcionamento do algoritmo genético em sua totalidade, de forma que cada parâmetro e método implementado tem uma influência significativa nos resultados obtidos. Também é válido citar que implementando tal algoritmo por meio da linguagem *R* foi possível observar a versatilidade da linguagem na implementação de algoritmos para diferentes finalidades. Por fim, com os resultados obtidos e a melhor solução conhecida para o arquivo *bays29.tsp* da *TSPLIB* que é uma distância

de 2020 foi possível verificar que para encontrar um valor próximo dessa solução é necessário um aumento considerável no tamanho da população de indivíduos e no número de gerações, o que causaria um aumento significante no tempo de execução.

Referências

[1] ICMC Instituto de Ciências Matemáticas e de Computação. Algoritmos genéticos. Disponível em: https://sites.icmc.usp.br/andre/research/genetic/ Acesso em: 15 de Julho de 2022.

[2] Tutorials point. Genetic algorithms - mutation. Disponível em: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.html Acesso em: 17 de Julho de 2022.

[3] Vinicius dos Santos. O problema do caixeiro viajante. Disponível em: https://www.computersciencemaster.com.br/o-problema-do-caixeiro-viajante/ Acesso em: 17 de Julho de 2022.

[4] Universität Heidelberg. Tsplib. Disponível em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Acesso em: 17 de Julho de 2022.

[5] Vijini Mallawaarachchi. How to define a fitness function in a genetic algorithm? Disponível em: https://towardsdatascience.com/how-to-define-a-fitness-function-in-a-genetic-algorithm-be572b9ea3b4 Acesso em: 17 de Julho de 2022.