

Proj 1 - Genetic Algorithm

Davi Dupin - 17/0140148, Matheus Azevedo - 17/0110940

Resumo: Este projeto visa demonstrar como o Travelling Salesman Problem (TSP) pode ser resolvido utilizando um Algoritmo Genético. Utilizando-se de conceitos da Inteligência Artificial, podemos resolver uma série de problemas complexos, como veremos a seguir. Apesar da amostra de dados ser relativamente pequena, conseguimos implementar um algoritmo com escalabilidade razoável.

Palavras-chave: Inteligência Artificial; Algoritmo Genético; Python; Processamento de Dados.

1 Introdução

Este projeto foi feito para entendermos melhor como funciona a implementação de um algoritmo genético. A ideia é trazer a experiência de código para melhor compreensão do assunto. Assim, devemos ficar atentos ao detalhes de implementação, como os tipos de técnicas usadas para solucionar problemas dentro do campo da inteligência artificial.

2 Materiais e métodos

A respeito das tecnologias, utilizamos Python 3. Tentamos usar Python 2 em um primeiro momento, porém, após alguns erros, resolvemos alterar a versão da linguagem. O método de avaliação sobre o projeto foi bem visual: tentamos buscar uma solução própria e depois analisamos se o código produzia algo parecido ou até melhor.

Sobre a implementação, podemos dividir o algoritmo em 4 etapas:

1 – População inicial: primeiro criamos uma população base, utilizando os dados disponíveis no projeto. Essa população é um array de Rotas (começamos com 100 Rotas distintas), onde uma rota contém 11 cidades, sendo a primeira e a última a mesma cidade – Brasília. A ordem das outras cidades é aleatória.

2 – Refinar população: nessa etapa iremos avaliar a distância que deve ser percorrida em uma Rota. Esse é o parâmetro principal para avaliar se uma rota é “melhor” que a outra. Neste projeto, a melhor rota é a rota tal qual se faz um percurso por todas as cidades com a menor distância percorrida. Assim, “refinar” uma população nada mais é que criar um ranking das melhores Rotas e deixá-las disponíveis para os próximos passos.

3 – Selecionar as melhores rotas: agora que temos um ranking, basta selecionar essas rotas. Essa seleção utiliza dois critérios: ranking e o total de rotas já escolhidas. É possível limitar a quantidade de rotas que serão selecionadas para se criar uma nova geração de rotas

4 – Procriar: agora que temos as melhores rotas disponíveis, iremos “procriar” essas rotas, ou seja, iremos produzir outras rotas utilizando duas rotas que estão dentro da nossa seleção anterior. Esse passo está em conjunto com o próximo passo.

5 – CrossOver: no momento de procriar as rotas, iremos dar uma chance de uma pequena mutação dentro de uma rota. Uma mutação seria uma alteração na ordem das cidades de uma rota. Esse parâmetro visa aumentar a diversidade entre rotas, mesmo que os “pais” sejam os mesmos. Também é possível alterar a taxa de mutação de rotas.

6 – Gerar próxima geração: após o passo 5, temos uma nova geração de rotas, onde a chance de existir um aprimoramento na distância aumentou.

7 – Repetir: agora basta repetir o mesmo procedimento até um número onde acreditamos que não exista mais aprimoramento ou até onde nosso programa possa produzir um resultado satisfatório

3 Resultados

O programa possui duas saídas:

```
matheus@matheus-pc:~/Documents/dev/ila-proj1$ python3 main.py
Distância Inicial: 187.0
Distância Final: 128.0
```

Figura 1 – Saída do programa no Terminal Linux

Nesse caso, a saída mostra o valor da melhor distância da primeira geração de uma população aleatória, e logo depois que o algoritmo terminar sua execução, será mostrada a segunda mensagem, que é a melhor distância da última geração dessa mesma população.

Logo depois, temos esse gráfico:

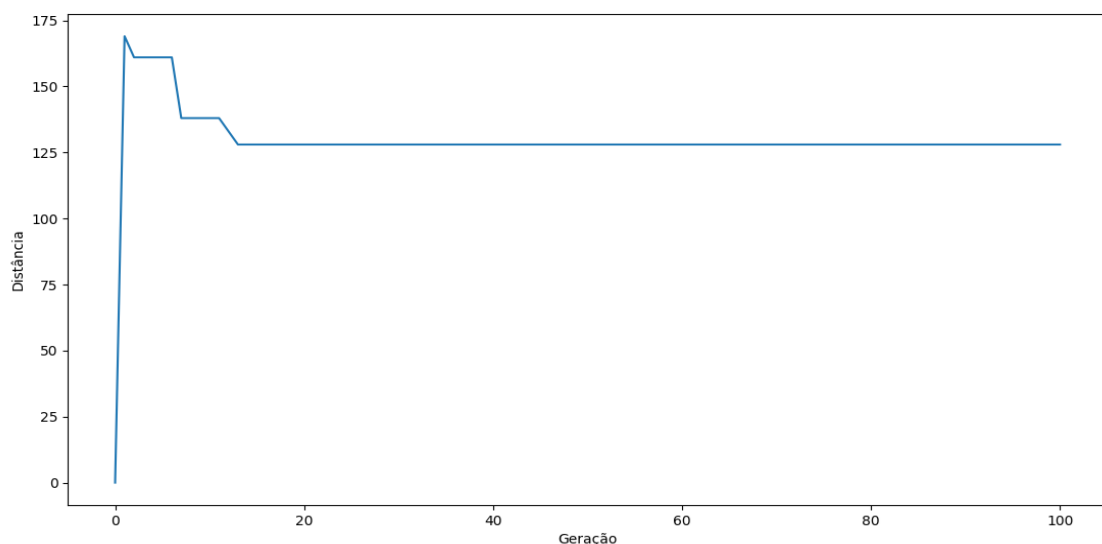
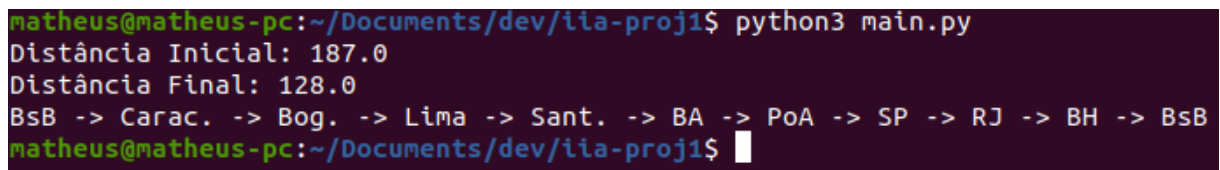


Figura 2 – Gráfico das gerações e a melhor distância de cada geração, mostrando o progresso de cada geração.

Ao fechar esse gráfico, temos esse último output:



```
matheus@matheus-pc:~/Documents/dev/ia-proj1$ python3 main.py
Distância Inicial: 187.0
Distância Final: 128.0
BsB -> Carac. -> Bog. -> Lima -> Sant. -> BA -> PoA -> SP -> RJ -> BH -> BsB
matheus@matheus-pc:~/Documents/dev/ia-proj1$
```

Figura 3 – Último output do programa, mostrando o caminho (rota) que foi usado para chegar na melhor distância.

Esse caminho (ou rota) é sempre o mesmo, mudando apenas a ordem das cidades.

4 Análise de Resultados

Podemos perceber o quão importante são as variáveis que compõem a população. No caso, passamos alguns minutos com dificuldade para encontrar um pequeno erro de distâncias entre duas cidades, o que não permitia um cálculo exato das rotas. Outro fator que influencia muito no resultado final é a relação entre a população inicial e as novas gerações. Tentamos realizar o cálculo utilizando poucas gerações, 50 por exemplo, e algumas vezes o resultado era correto, outras vezes não... E isso acontecia com outras variantes também, como a taxa de mutação, o tamanho da população inicial ou o tamanho da seleção de rotas.

5 Considerações Finais/Conclusões

A criação desse projeto nos ajudou a compreender melhor como um algoritmo genético funciona, quais as dificuldades de se implementar o mesmo e quais as facilidades também. Acredito que o campo de IA é bem amplo, e para quem não possui vivência ou experiência no assunto, parece até fantasia. Um ponto importante que não havíamos cogitado, mas que foi prontamente demonstrado nesse projeto, é a importância de dados coesos e, obviamente, corretos. Não adianta possuir o melhor algoritmo do mundo, porém com dados corrompidos.