Algoritmos genéticos: aplicação ao problema de roteamento de veículos

Realizado por Pedro Paulo, Kaue Sales e Mayara

por Pedro Paulo

Representação do Espaço de Estados

O que é um estado?

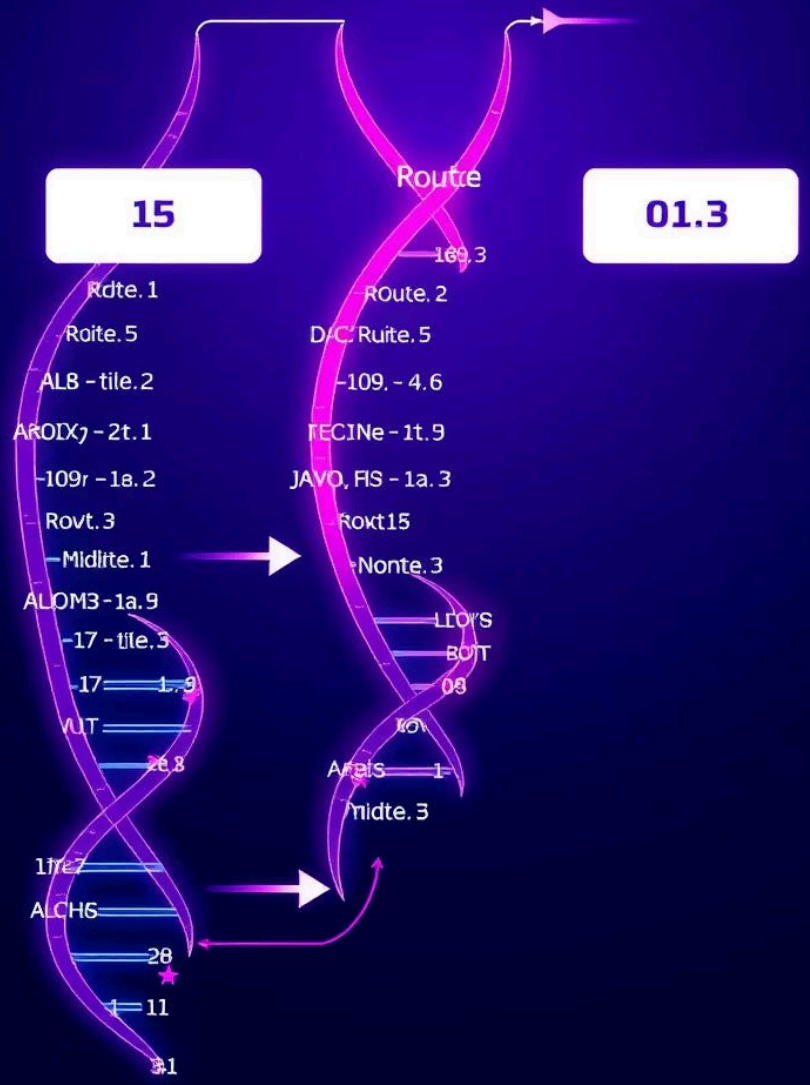
Uma **solução completa**: para 3 caixeiros, cada um com uma rota que começa e termina no ponto inicial (30, 30) cobrindo as 15 cidades selecionadas.

Tamanho do espaço

Combinações de 15 cidades distribuídas em 3 rotas **ordenadas** ↔️ imensa quantidade de permutações.

Impacto do tamanho da população (N)

- **Pop pequena (ex: N=30):** pouca diversidade → risco de convergir para ótimos locais.
- **Pop grande (ex: N=100+):** mais diversidade, converge levemente mais rápido, mas custo computacional cresce O(N).
- **Valor adotado (N=50):** bom equilíbrio entre diversidade e tempo de execução em nossos testes iniciais.



Indivíduo, Gene e Domínio de Cada Gene

Indivíduo (class Individuo)

Representa um vetor de 3 rotas: rotas = [$rota_1$, $rota_2$, $rota_3$].

Gene

Cada gene é o **índice de uma cidade** (0…29) presente em exatamente uma das rotas.

Domínio do gene

Todos os índices das 30 cidades, **mas** restringimos às 15 cidades pré-selecionadas (indices_escolhidos).

Garante que todo gene seja um valor válido e que cada cidade apareça **exatamente uma vez** na atribuição inicial.

Função de Geração de População

Objetivo

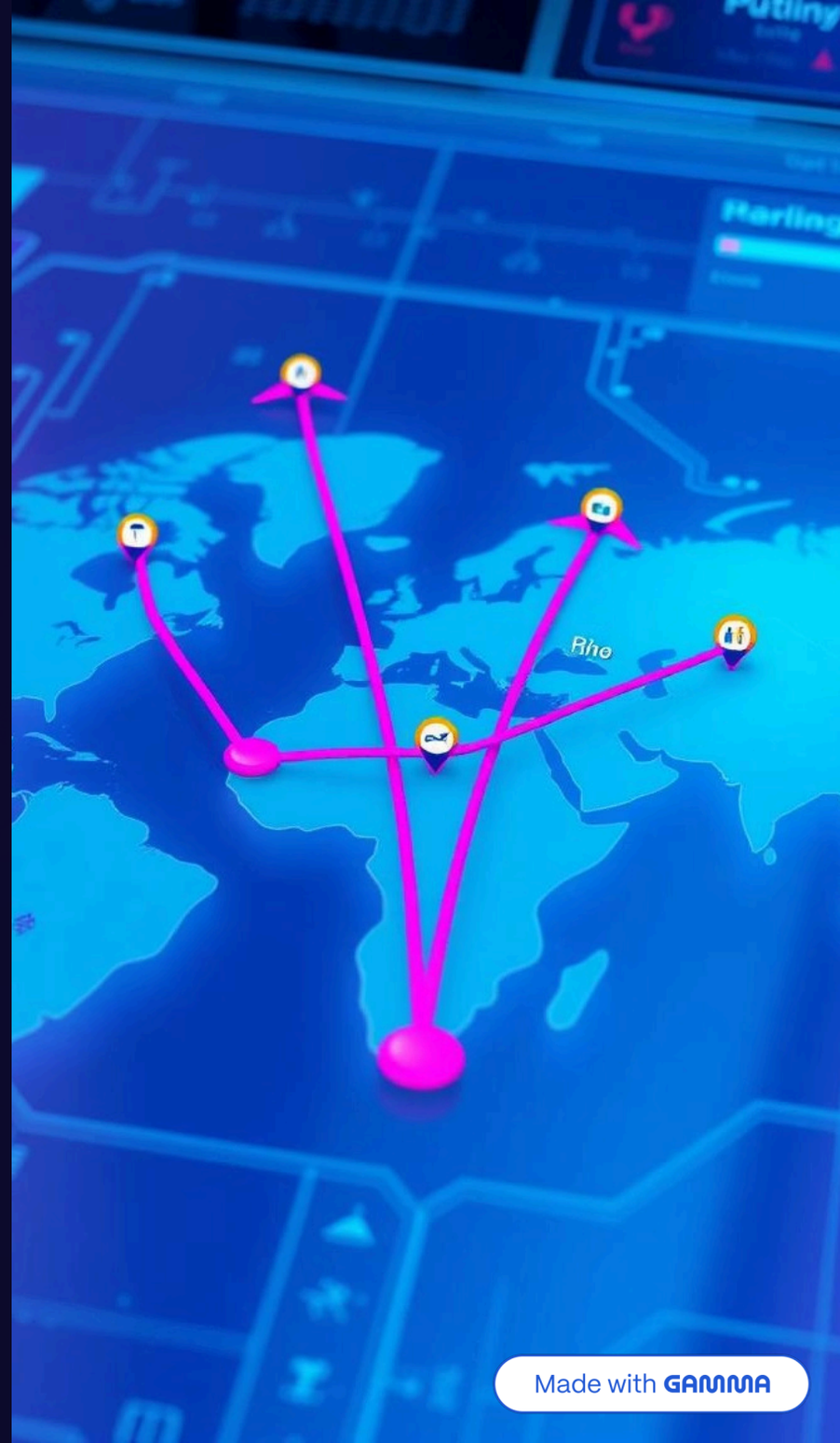
Criar N indivíduos válidos e diversos para iniciar a evolução.

Como implementamos

1. Recebe lista selecionadas = $[i_1,\ldots,i_{15}]$.
2. Para cada novo indivíduo:
   - Embaralha (random.shuffle) essa lista.
   - Itera sobre as 15 cidades, escolhendo aleatoriamente um dos 3 caixeiros para inserir cada cidade.
   - Garante que todas as cidades sejam alocadas e que nenhuma se repita.

Por quê?

Embaralhar + destino aleatório introduz variação inicial suficiente.



Função de Fitness e Seleção

Função de Fitness

Objetivo: quantificar quão boa é uma solução (mais curto e balanceado).

Cálculo: Para cada rota, monta sequência de coordenadas [ponto_inicial] + [coordenadas das cidades na ordem da rota] + [ponto_inicial]. Soma distâncias euclidianas entre pontos consecutivos.

Distância total = soma das 3 rotas. **Balanceamento:** Calcula média μ e desvio-padrão σ das distâncias individuais. Penalidade = 2·σ (rotas muito desiguais pioram o custo).

Custo final = distancia_total + penalidade. **Fitness** = 1 / (1 + custo_final) → normalizado em (0,1], maior é melhor.

Função de Seleção

Meta: escolher indivíduos "pais" com probabilidade maior para bons fitness, mas preservando diversidade.

Implementação: torneio de tamanho k=3

1. Sorteia k indivíduos aleatórios da população.
2. Compara fitness() de cada um; retorna o de maior fitness.

Uso no fluxo: Para gerar cada filho: faz dois torneios independentes (pai1 e pai2).

Por que torneio? Simples de implementar, não exige normalização por soma total de fitness. Permite controlar pressão seletiva via k.

Observações Finais sobre Parâmetros

- **População maior:** promove mais exploração, mas aumenta custo computacional.
- **Tamanho do torneio (k):** maior k aumenta pressão seletiva, diminui diversidade.
- **Validação prática:** monitore evolução do melhor fitness para ajustar N e k.
- **Penalidades:** devem ser calibradas para balancear rotas e melhorar soluções.