

Algoritmos Evolutivos: Otimizando Soluções Inspiradas na Natureza

Explorando técnicas de otimização que imitam a evolução natural para resolver problemas complexos do mundo moderno

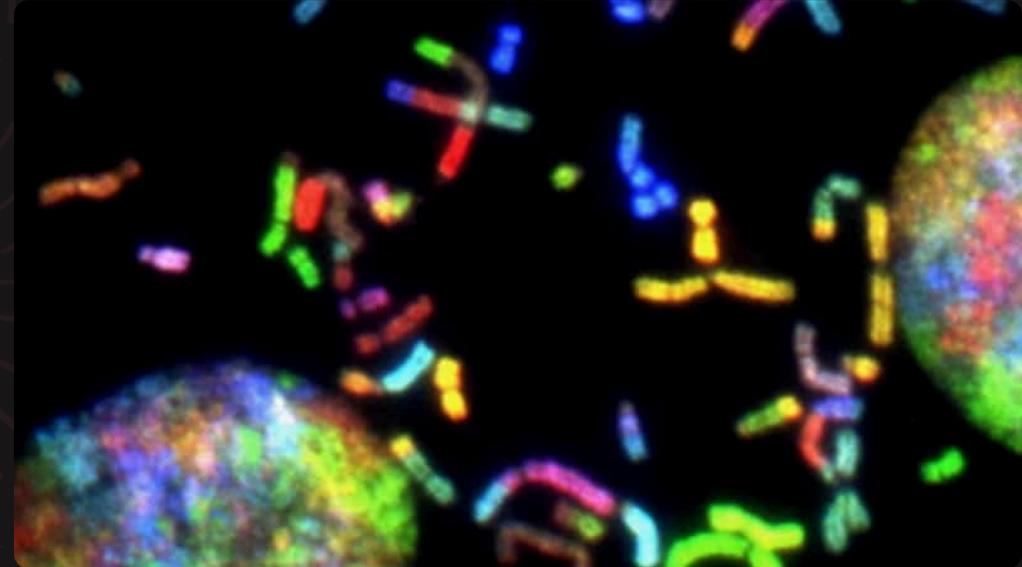


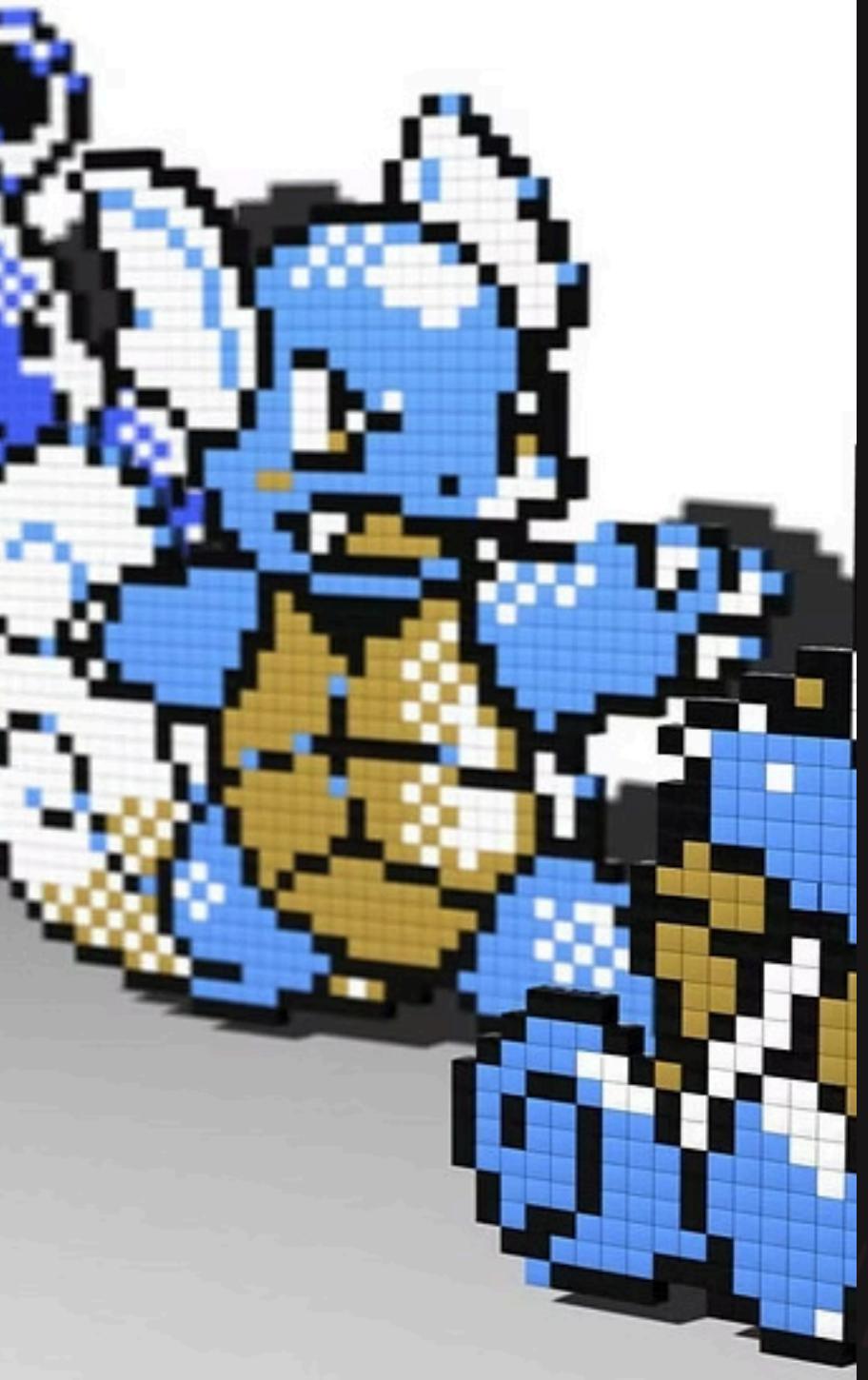
O que são Algoritmos Evolutivos?

Algoritmos evolutivos são técnicas computacionais poderosas inspiradas nos processos da evolução biológica e seleção natural de Darwin. Eles funcionam como uma simulação digital da natureza, onde soluções "competem" para sobreviver.

Através de ciclos iterativos de seleção, mutação e recombinação, esses algoritmos exploram o espaço de soluções possíveis, identificando progressivamente candidatos cada vez melhores.

São particularmente valiosos para problemas complexos de otimização onde métodos tradicionais falham ou são computacionalmente inviáveis.





Base Biológica e Conceitos Fundamentais



Cromossomos e Genes

Representam soluções candidatas codificadas. Cada cromossomo contém genes que definem características específicas da solução.



Função Fitness

Medida quantitativa da qualidade de cada solução. Determina quais indivíduos têm maior probabilidade de sobreviver e reproduzir.



Operadores Evolutivos

Seleção natural escolhe os melhores, mutação introduz variabilidade, e cruzamento combina características de soluções parentais.

Principais Tipos de Algoritmos Evolutivos

1

Algoritmos Genéticos (AG)

Baseados em representações binárias ou de valores reais, utilizam operadores clássicos de cruzamento e mutação para evoluir populações de soluções.

- Codificação flexível de soluções
- Cruzamento de um ou múltiplos pontos
- Mutação bit a bit ou gaussiana

2

Evolução Diferencial (ED)

Especializado em otimização contínua, usa diferenças vetoriais entre membros da população para gerar novos candidatos. Simples e altamente eficaz.

- Operador de mutação diferencial
- Estratégias de cruzamento binomial
- Excelente para espaços contínuos

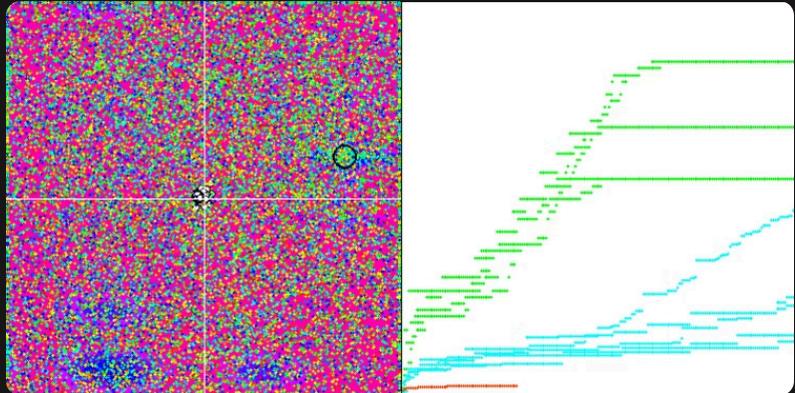
3

Programação Evolutiva (PE)

Evolui tanto parâmetros quanto estruturas de soluções. Muito popular em aprendizado de máquina para otimização de arquiteturas neurais.

- Mutação adaptativa de estruturas
- Sem operador de cruzamento
- Foco em auto-adaptação

Evolução Diferencial: Destaque e Aplicações



□ **Criado em 1995** por Rainer Storn e Kenneth Price, a Evolução Diferencial rapidamente se tornou uma das técnicas mais respeitadas em otimização contínua.

Por que a ED se destaca?

Reconhecida mundialmente pela sua robustez, simplicidade de implementação e desempenho excepcional em problemas de otimização não linear e multidimensional.

Aplicações em Destaque

- **Projeto de filtros digitais:** otimização de coeficientes para resposta ideal
- **Redes neurais:** treinamento e arquitetura de modelos deep learning
- **Sistemas de reservatórios:** gestão otimizada de recursos hídricos
- **Engenharia elétrica:** design de circuitos e sistemas de potência
- **Problemas financeiros:** otimização de portfólios de investimento

Algoritmos Genéticos: História e Funcionamento



Os Algoritmos Genéticos são especialmente poderosos para **problemas combinatórios** como o clássico problema do caixeiro viajante, onde encontrar a rota ótima entre múltiplas cidades se torna exponencialmente complexo com métodos tradicionais.

Aplicações Práticas dos Algoritmos Evolutivos

Arquitetura Inteligente

Geração automática de layouts otimizados para espaços residenciais e comerciais, considerando luz natural, fluxo e eficiência.



Robótica

Controle e navegação autônoma

Inteligência Artificial

Treinamento de redes neurais profundas e ajuste automático de hiperparâmetros para máximo desempenho.



Visão Computacional

Reconhecimento de padrões



Engenharia

Design estrutural otimizado

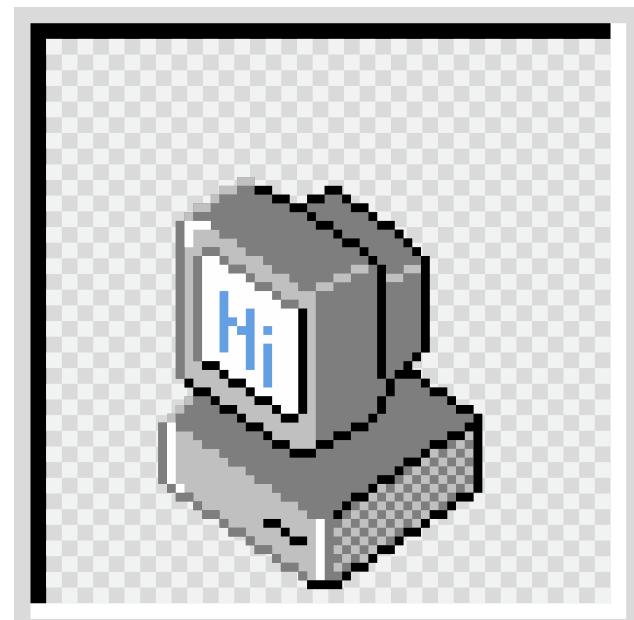
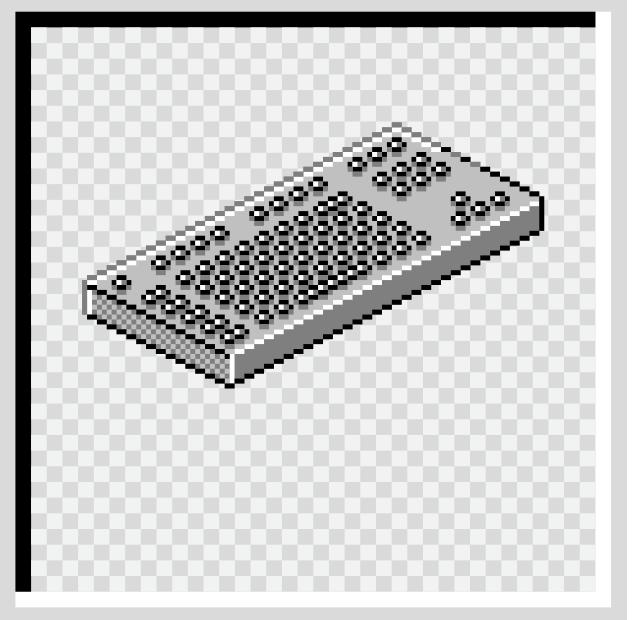


Bioinformática

Análise de sequências genéticas

[Home](#)[Content](#)[Contact](#)

- Biologia (Natureza)
- Indivíduo
- População
- Cromossomo (DNA)
- Gene
- Fitness (Aptidão)
- Ambiente
- Computação
- Uma solução candidata
- O conjunto de todo teste
- A estrutura de dados
- Um elemento específico significante
- A função objetivo
- O problema

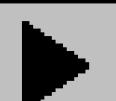


Home

Content

Contact

A metodologia do AG, trabalha com uma população de soluções (rotas), onde cada solução contém um genoma (uma sequência de locais).



Home Content Contact

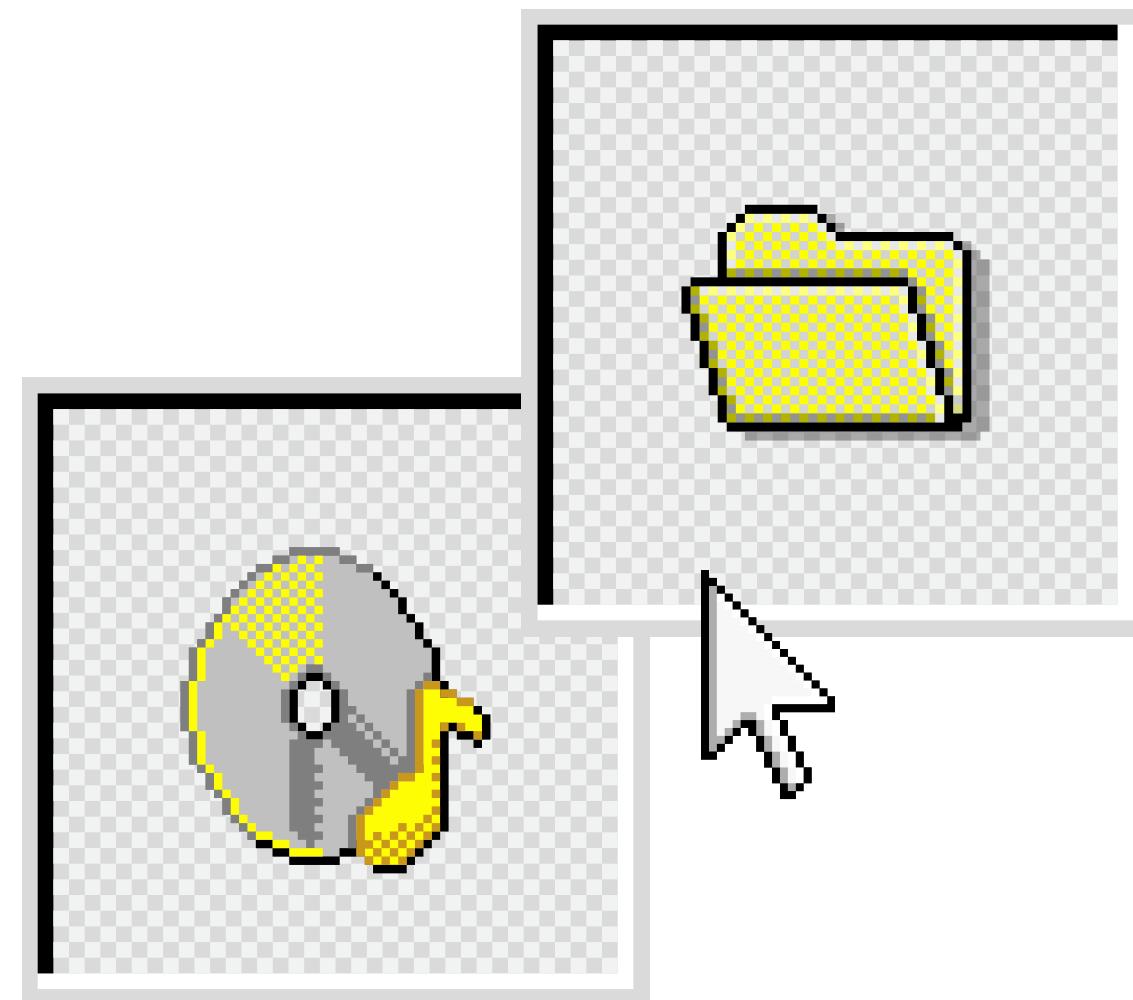
- Seleção: Escolhe as melhores filhos (baseado na aptidão).
- Crossover (Cruzamento): Combina partes de dois parentes para criar um filho melhor.
- Mutação: Altera aleatoriamente os filhos para manter a diversidade e evitar ótimos locais.



Home

Content

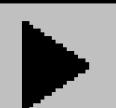
Contact



i

PARTE PRÁTICA IMPLEMENTAÇÃO

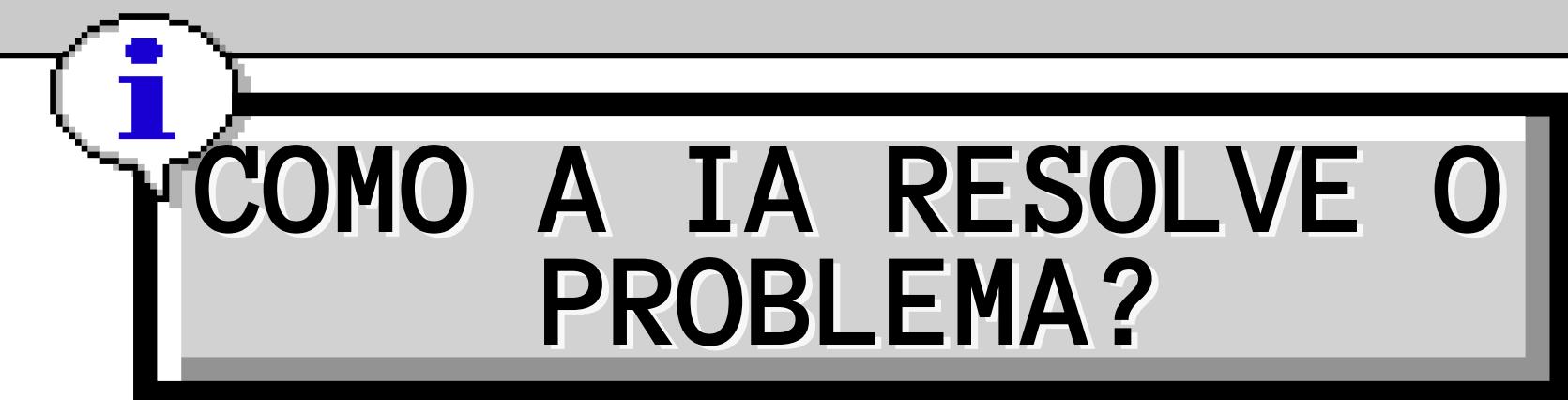
Algoritmos Genéticos e Busca
A: Uma Implementação Híbrida
para Navegação em Ambientes
Simulados



Home

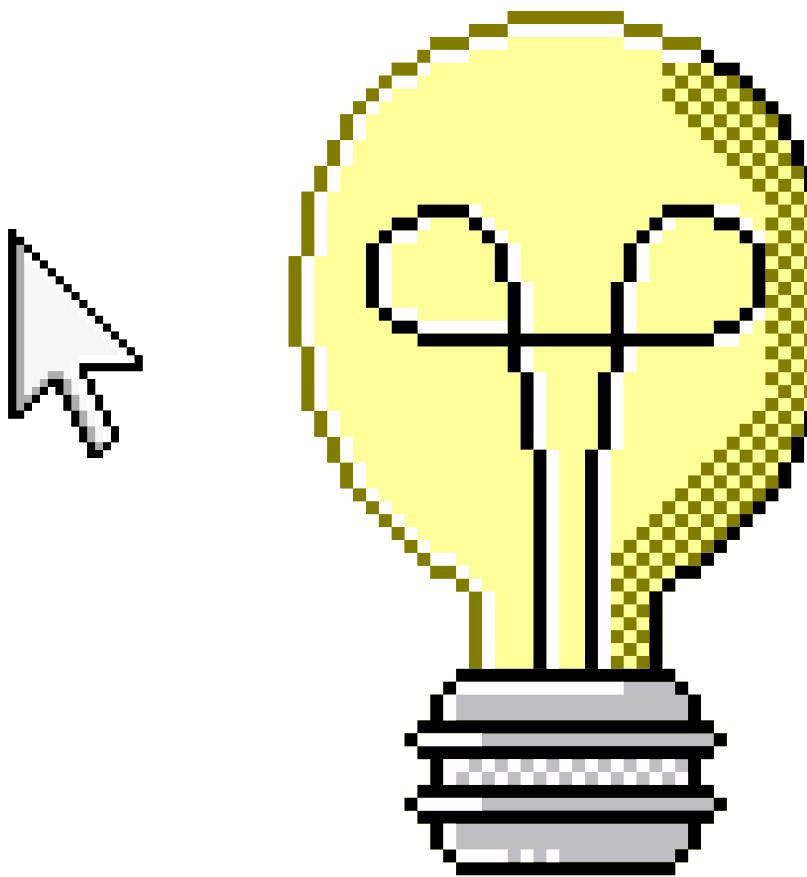
Content

Contact



- Nível MACRO (o Estrategista): Usa o Algoritmos Genético, sendo responsável pela ordem de visita, com o objetivo de maximizar o lucro (recompensa) em relação ao tempo total.
- Nível MICRO (o Navegador): Usa Busca A* (A-Star) sendo responsável pelo trajeto físico, com o objetivo de encontrar o caminho mais curto desviando de terrenos custosos (água, montanhas).

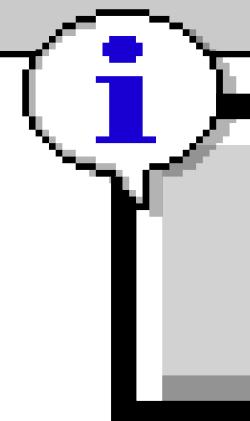
A implementação foca em duas camadas que combina estratégia global e tática local.



Home

Content

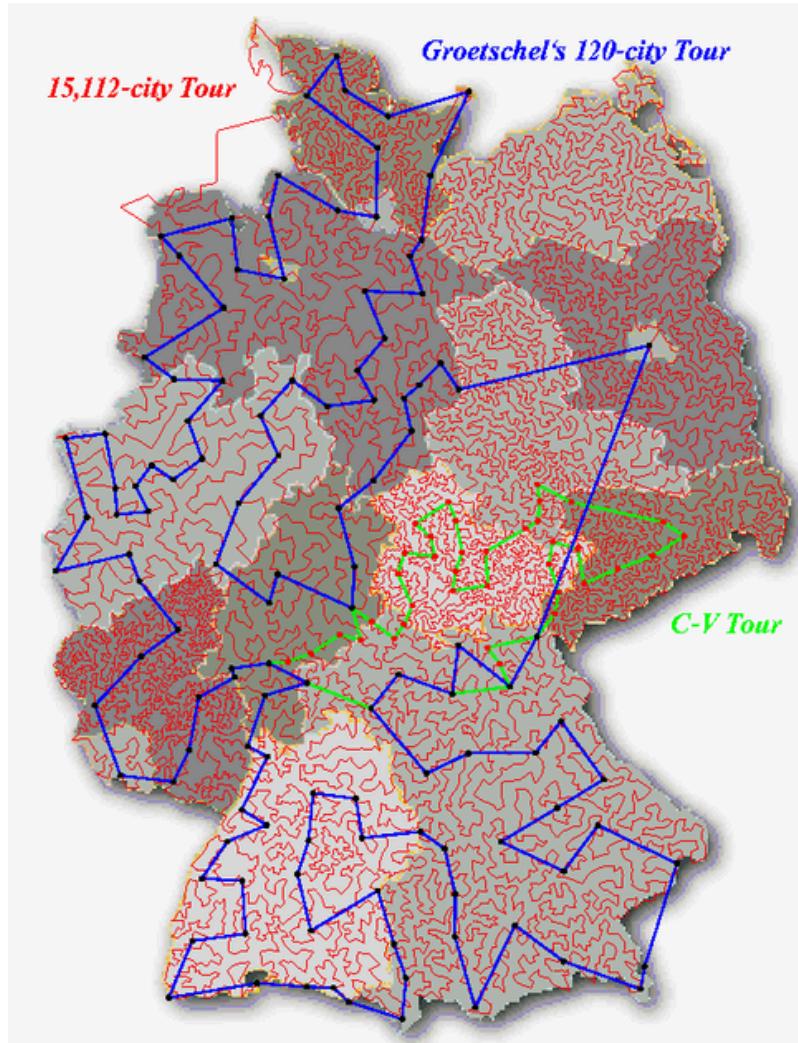
Contact



PROBLEMA DO CAIXEIRO VIAJANTE

Encontrar a rota mais curta para visitar um conjunto de cidades e retornar ao ponto de origem.

- O objetivo é, na essência, uma variação do Problema do Caixeiro Viajante.
- Como não podemos calcular matematicamente todas as rotas possíveis em tempo real, usamos a Computação Evolucionária para aproximar a solução ideal em um terreno simulado.



Home

Content

Contact



CENÁRIO: VÍDEOGAME

O projeto utiliza como ambiente de simulação o jogo PokéMon Legends: Arceus (2022), um título da franquia japonesa que introduziu um mundo semi-aberto e novas mecânicas de exploração.

Um dos elementos centrais do endgame deste jogo é o fenômeno dos Mass Outbreaks (Surtos em Massa).

- Eventos temporários onde alcateias de PokéMon da mesma espécie aparecem em locais específicos do mapa simultaneamente.
- Jogadores viajam de surto em surto procurando por versões raras dos monstrinhos.



[Home](#)[Content](#)[Contact](#)

CENÁRIO: VÍDEOGAME

O jogador busca variações raras de espécimes, que podem ser categorizadas em três níveis de interesse:

Pokémon Shiny (Brilhante), uma variação genética extremamente rara que altera a coloração original da criatura e emite um efeito visual de brilho ao entrar em batalha.

Pokémon Alpha, uma variação fisiológica caracterizada por gigantismo, e atributos de combate superiores

Pokémon Alpha Shiny, a interseção dos dois eventos anteriores. É um Pokémon que possui tanto o gigantismo e força do Alpha quanto a coloração rara do Shiny.



Home

Content

Contact



TEORIA DA PROBABILIDADE E EVENTOS ESTOCÁSTICOS

Para um computador, a "sorte" não existe. Existe apenas a Probabilidade (P). Para modelar o problema de otimização, tratamos cada visita a um Mass Outbreak como um experimento estatístico (Tentativa de Bernoulli).

Utilizamos as taxas de aparecimento (Odds) do método Mass Outbreak com o item Shiny Charm (o cenário ideal de caça):

Cálculo das probabilidades conjuntas para a definição das recompensas:

Probabilidade de Shiny (P_S): $1/128.49 \approx 0.00778$

Probabilidade de Alpha (P_A): $2\% = 0.02$

Tentativas por Outbreak (N_T): 19



[Home](#)[Content](#)[Contact](#)

TEORIA DA PROBABILIDADE E EVENTOS ESTOCÁSTICOS

Calculamos a probabilidade de cada um dos quatro resultados possíveis por tentativa:

1. Probabilidade de Alpha Shiny (P_{AS}):

$$\begin{aligned} P_{AS} &= P_S \times P_A \\ &= (1/128.49) \times 0.02 \\ &\approx 0.0001556 \end{aligned}$$

2. Probabilidade de Shiny Comum (SC, não-Alpha):

$$\begin{aligned} P_{SC} &= P_S \times (1 - P_A) \\ &= (1/128.49) \times (1 - 0.02) \\ &\approx 0.007627 \end{aligned}$$

3. Probabilidade de Alpha Comum (AC, não-Shiny):

$$\begin{aligned} P_{AC} &= (1 - P_S) \times P_A \approx 1 \\ &\quad \times 0.02 = 0.02 \end{aligned}$$

O Valor Esperado ($E_{tentativa}$) é a soma de cada resultado possível multiplicado pelo seu valor:

$$\begin{aligned} E_{tentativa} &= (P_{AS} \times V_{AS}) \\ &+ (P_{SC} \times V_{SC}) + (P_{AC} \times V_{AC}) \\ &\quad + (P_C \times V_C) \end{aligned}$$

substituindo:

$$\begin{aligned} E_{tentativa} &\approx (0.0001556 \times 1000) \\ &+ (0.007627 \times 100) \\ &+ (0.02 \times 20) + (0.98 \times 0) \\ E_{tentativa} &\approx 0.1556 + 0.7627 + 0.4 \\ &\quad + 0 \\ E_{tentativa} &\approx 1.3183 \end{aligned}$$

Cálculo do Valor Esperado final por Outbreak:

$$\begin{aligned} R_{base} &= E_{tentativa} \times N_T \\ R_{base} &\approx 1.3183 \times 19 \\ \mathbf{R}_{base} &\approx \mathbf{25.04} \end{aligned}$$



Home

Content

Contact



Como o Algoritmo Genético não pode interpretar valor de um Shiny, precisamos traduzir a raridade probabilística em um valor numérico determinístico: a Recompensa (R)

Para evitar a necessidade de simular milhões de batalhas (Método de Monte Carlo), utilizamos o conceito de Valor Esperado (E).

Cada ponto no mapa recebe um valor R baseado no que ele pode oferecer em média.

- Outbreak Comum:
- Possui apenas a chance base de Shiny (P_s).
- Recompensa: Baixa/Média.
- Outbreak de Alta Demanda
- Representa surtos de espécies que são valiosas competitivamente ou esteticamente
- Recompensa: Definimos heuristicamente um multiplicador alto (no código, 19.5 o valor base).



Home

Content

Contact



Isso cria a base para a nossa Função de Fitness:

Fitness =

$$\frac{\sum \text{Recompensa} (\text{Valor de Alpha Shiny})}{\text{Custo Total} (\text{Obtido do } A^*)}$$

Ou seja, o algoritmo só decidirá escalar uma montanha (alto custo) para visitar um ponto se a Probabilidade Combinada de encontrar um Alpha Shiny naquele local for alta o suficiente para justificar o gasto de tempo.

Se a recompensa for baixa (Pokémon comum), a IA preferirá o caminho plano e fácil.



[Home](#)[Content](#)[Contact](#)

No Nível Micro do nosso sistema, o algoritmo A* precisa de uma regra para calcular o custo de ir do ponto A ao ponto B.

Testamos duas abordagens clássicas de geometria para simular diferentes mecânicas de jogo.

1. Distância de Manhattan

A distância é a soma das diferenças absolutas das coordenadas.

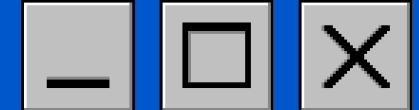
$$d = |x_1 - x_2| + |y_1 - y_2|$$

2. Distância Euclidiana

A distância em linha reta entre dois pontos, baseada no Teorema de Pitágoras.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



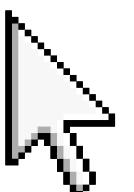


Home

Content

Contact

EXECUTANDO



Home

Content

Contact

CONCLUSÃO

