

# Inteligência Artificial - Notas de aula

Raoni F. S. Teixeira

Aula 3 - Busca Local

## 1 Introdução

Nesta aula, vamos estudar agentes que procuram estados específicos sem examinar todo o ambiente. Em vez de analisar todas as possibilidades, eles focam apenas em partes do espaço de busca, movendo-se entre estados até encontrar uma solução ótima.

Se cada estado do ambiente  $s$  tiver uma pontuação  $f(s)$ , o agente buscará o estado com a maior pontuação possível:

$$\operatorname{argmax}_{s \in \mathcal{S}} f(s). \quad (1)$$

Esse processo é chamado de busca local.

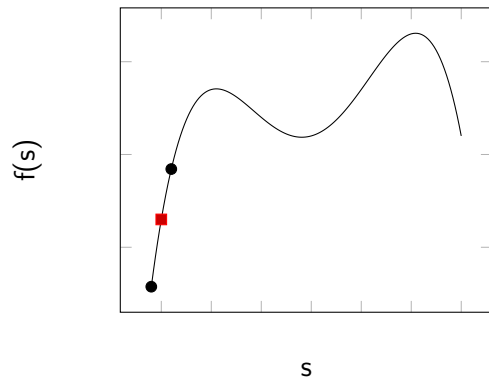
As seções seguintes examinam dois algoritmos. Outros exemplos podem ser encontrados em [RN09, KW19].

## 2 Subida da encosta

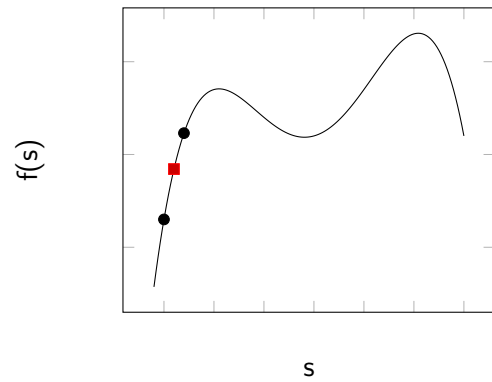
Pense em escalar uma montanha para alcançar o pico mais alto. A cada passo, você avalia os picos vizinhos e escolhe aquele com a maior elevação, até que não existam opções melhores.

Essa é essência do algoritmo de subida da encosta. Ele começa em um estado aleatório  $s$  e, a cada passo, move-se para o vizinho com a maior pontuação, analisando apenas os estados próximos. O processo termina quando atinge um ótimo ou quando o número máximo de iterações é alcançado.

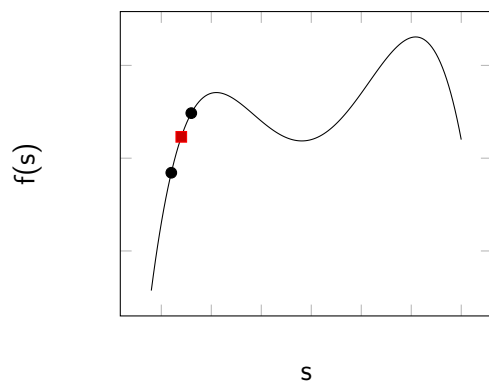
A Figura 1 mostra o funcionamento do algoritmo para uma função com dois máximos. A cor vermelha indica estado solução ( $s$ ) e a cor preta mostra os vizinhos imediatos. A cada iteração,  $s$  se move para o estado vizinho com a melhor pontuação. Quando o algoritmo termina (Figura 1f), a pontuação de  $s$  é maior que a pontuação de todos os seus vizinhos — máximo local.



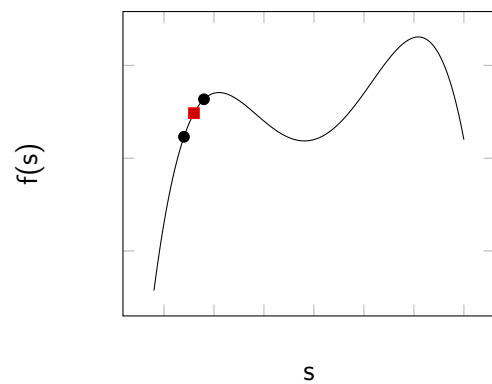
(a) Iteração 1.



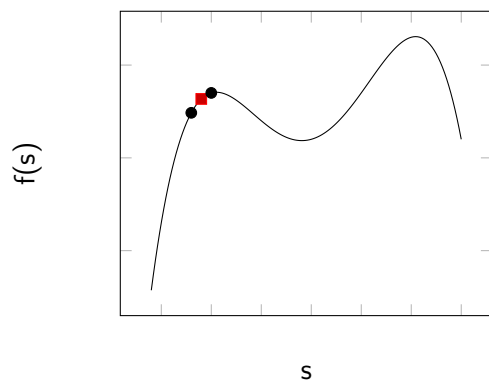
(b) Iteração 2.



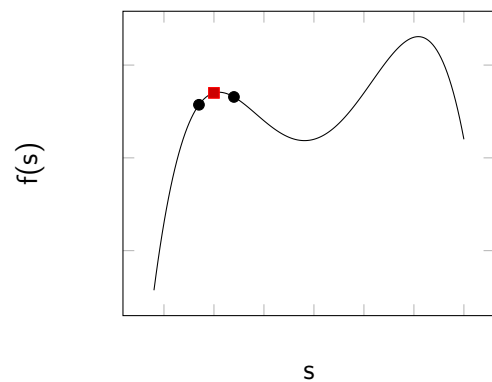
(c) Iteração 3.



(d) Iteração 4.



(e) Iteração 5.



(f) Iteração 6.

Figura 1: Seis iterações do algoritmo subida da encosta para um função  $f$  com dois mínimos. As cores vermelha e preta indicam respectivamente, estados em análise e vizinhos. A cada iteração, a solução se move para o estado vizinho com a melhor pontuação.

O pseudo-código a seguir apresenta esse algoritmo usando uma função *Vizinhanca* que devolve os vizinhos de um estado.

```
SUBIDAENCOSTA( $f$ , max-it)
1   $s \leftarrow$  estado aleatório
2   $it \leftarrow 0$ 
3  while  $it < \text{max-iter}$  do
4      melhor-vizinho  $\leftarrow s$ 
5      for each estado  $u \in \text{Vizinhanca}(s)$  do
6          if  $f(u) > f(\text{melhor-vizinho})$  then
7              melhor-vizinho  $\leftarrow u$ 
8          if  $f(\text{melhor-vizinho}) > f(s)$  then
9               $s \leftarrow \text{melhor-vizinho}$ 
10         else
11             break
12          $it \leftarrow it + 1$ 
13 return  $s$ 
```

Para aplicar esse algoritmo a problemas reais, você precisa definir três elementos:

1. **Representação dos estados:** como os estados serão descritos?
2. **Vizinhança:** quais estados próximos serão considerados?
3. **Função objetivo  $f$ :** como avaliar a qualidade de um estado?

O Roteiro 3 mostra um exemplo de definição para o problemas da *N-Damas*.

### 3 Algoritmo Genético

Algoritmo genético é um método iterativo de otimização que opera localmente sobre múltiplas regiões do espaço de busca, combinando as informações dos estados dessas regiões para evitar mínimos locais.

Inspirado na biologia evolutiva, o conjunto de estados analisados é chamado de **população** e cada ciclo de iteração é uma **geração**. Cada estado é chamado de *indivíduo* e é representado por uma sequência de *cromossomos* — um vetor de números. Operações denominadas *seleção*, *cruzamento* e *mutação* produzem uma nova população combinando e alterando os vetores da população atual.

O algoritmo consiste nos passos do fluxograma da Figura 2:

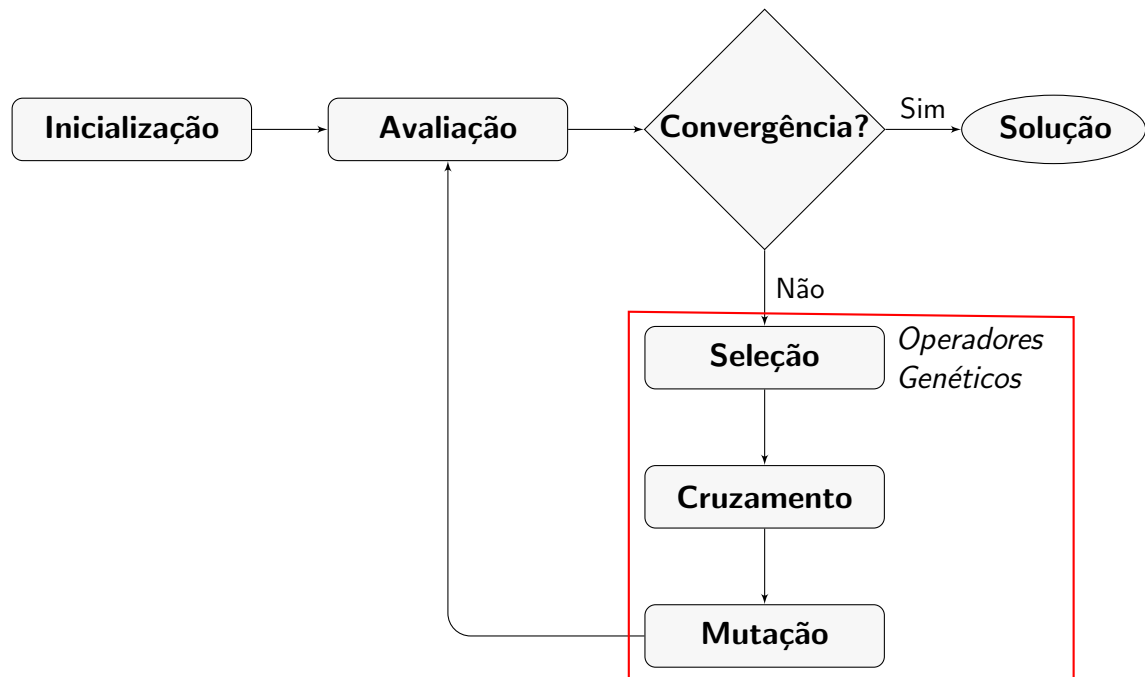


Figura 2: Algoritmo genético. O algoritmo termina quando um critério de convergência é atingido.

- **Inicialização:** define uma população inicial composta por  $M$  indivíduos gerados aleatoriamente. Essa população deve estar bem distribuída pelo espaço de busca, aumentando as chances de encontrar uma solução. Dois parâmetros são definidos nesta etapa: (a) a distribuição utilizada na amostragem e (b) o número de amostras  $M$ . Um valor maior de  $M$  melhora a representatividade do espaço, mas também aumenta o consumo de memória.
- **Avaliação:** cada indivíduo da população é avaliado por uma função  $f$ , que retorna um número real representando sua qualidade. O algoritmo termina quando encontra um indivíduo ótimo ou atinge o número máximo de iterações.
- **Seleção:** escolhe os indivíduos mais bem avaliados para atuarem como pais da próxima geração. A seleção é realizada aleatoriamente, mas com maior probabilidade para indivíduos com melhores valores de  $f$ .
- **Cruzamento:** combina os cromossomos dos indivíduos selecionados para criar novos descendentes. A Figura 3 ilustra essa operação em torno de um ponto de cruzamento, um número sorteado aleatoriamente no intervalo  $[0, N[$ . O cruzamento concatena os vetores  $v[0 \dots k]$  e  $v[k + 1 \dots N - 1]$  dos pais, formando os descendentes.
- **Mutação:** altera aleatoriamente o valor de algum cromossomo de um indivíduo. A taxa

de mutação define a probabilidade de essa alteração ocorrer. Uma taxa maior aumenta a chance de mutação. Tanto a posição do cromossomo quanto seu novo valor são determinados de forma aleatória e uniforme.

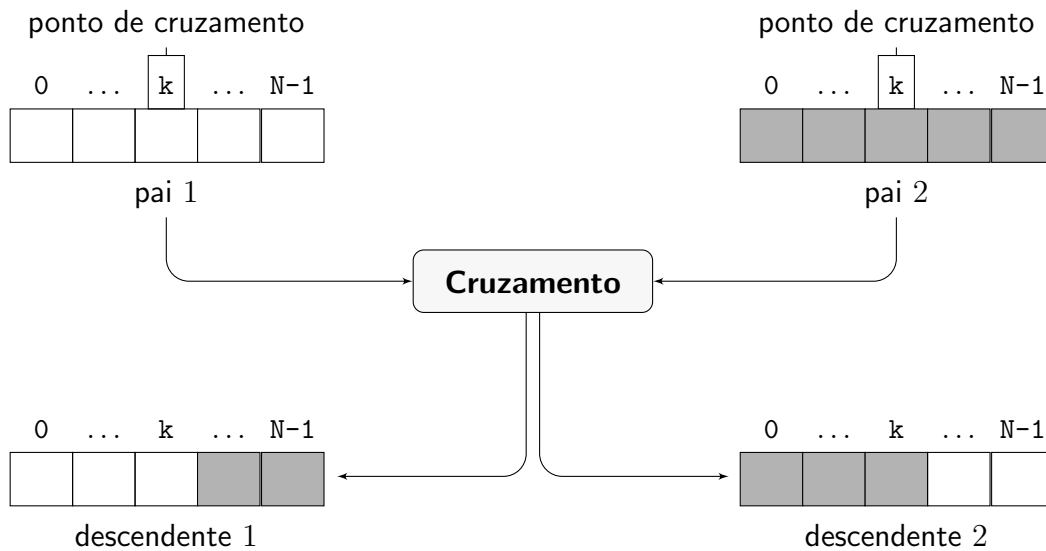


Figura 3: Operação de cruzamento. Os descendentes são uma combinação dos pais.

A aplicação dessas operações requer uma representação vetorial (cada estado é um vetor de números, *bits* etc) e uma função de avaliação, como mostra o exemplo do Roteiro 3.

## 4 Exercícios

Utilize o código do Roteiro 3 para fazer experimentos e responder as questões abaixo.

1. Por que você acha que a seleção é tão importante no algoritmo genético? Como ela pode influenciar o sucesso do processo evolutivo?
2. Por que a mutação é importante? Como você definiria a taxa de mutação se suspeitasse que existe uma solução ainda melhor que encontrada pelo algoritmo?

## Referências

- [KW19] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. The MIT Press, 2019.
- [RN09] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.