

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.

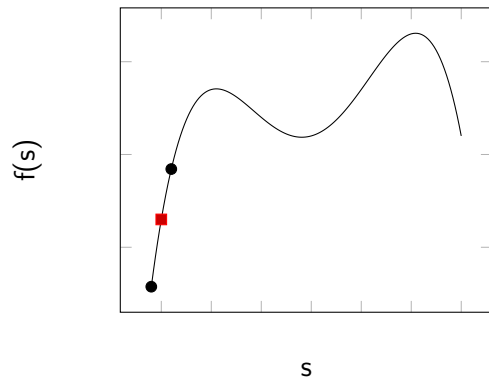
As seções seguintes examinam três algoritmos. Outros exemplos podem ser encontrados em [RN09, KW19].

2 Subida da encosta

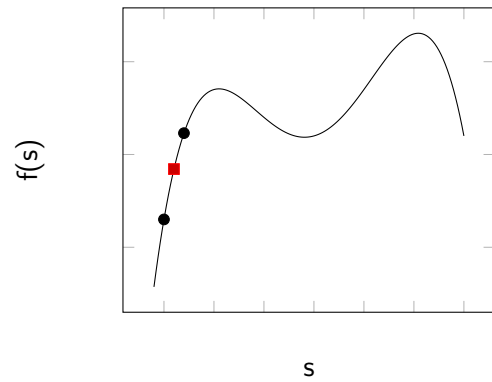
Pense em escalar uma montanha para alcançar o pico mais alto. A cada passo, você avalia os picos vizinhos e escolhe aquele com a maior elevação, até que não existam opções melhores.

Essa é a 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.

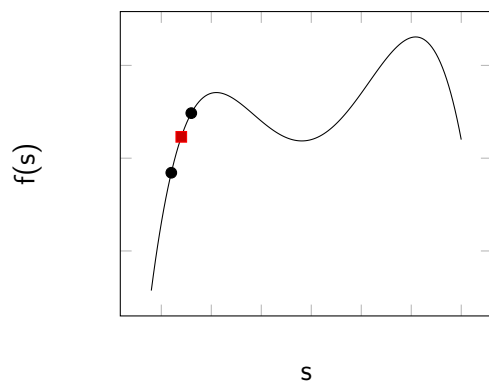
A Figura 1 mostra o funcionamento do algoritmo para uma função com dois máximos. A cor vermelha indica estado solução (s) e a cor preta mostra os vizinhos imediatos. A cada iteração, s se move para o estado vizinho com a melhor pontuação. Quando o algoritmo termina (Figura 1f), a pontuação de s é maior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



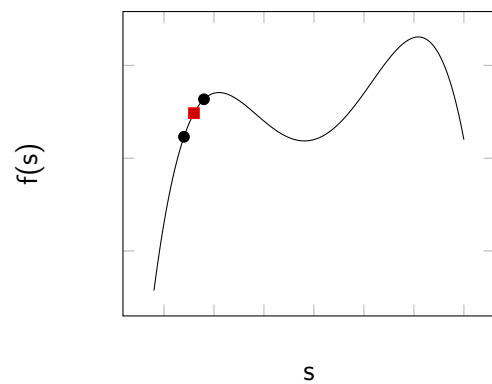
(a) Iteração 1.



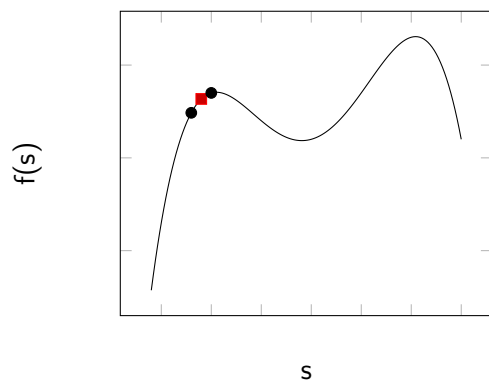
(b) Iteração 2.



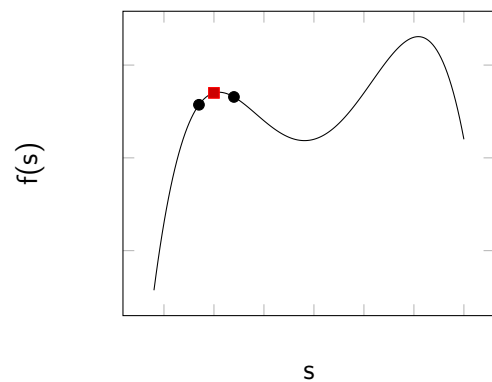
(c) Iteração 3.



(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 Tempera Simulada

O algoritmo de Tempera Simulada (*Simulated Annealing*) é uma técnica inspirada no processo de recozimento de metais, onde o material é aquecido e, em seguida, resfriado de forma controlada. O método é utilizado para encontrar soluções aproximadas em problemas de otimização, especialmente quando o espaço de busca possui múltiplos ótimos locais.

Ao contrário da Subida da Encosta, que pode ficar preso em ótimos locais, a Tempera Simulada permite movimentos que pioram a solução temporariamente, aumentando a chance de encontrar o ótimo global.

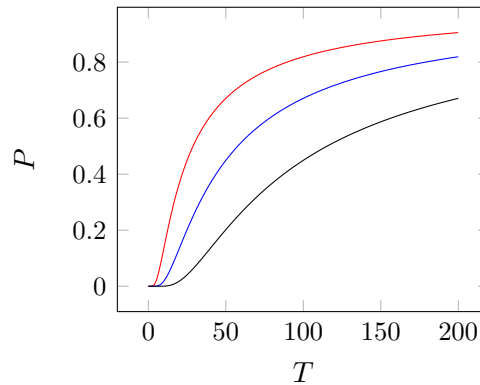


Figura 2: Probabilidade P em função da temperatura T para diferentes valores de ΔE . As curvas preta, azul e vermelha correspondem a $\Delta E = 20, 40$ e 80 . À medida que T diminui, P também reduz, tornando o algoritmo mais conservador na aceitação de estados piores.

A cada iteração, o algoritmo escolhe aleatoriamente um vizinho do estado atual. Se o vizinho apresentar uma pontuação melhor, ele é aceito como o novo estado. Caso contrário, o algoritmo pode aceitá-lo com base em uma probabilidade que depende da temperatura atual e da diferença de pontuação entre os estados.

Essa probabilidade é dada pela fórmula:

$$P(\Delta E, T) = e^{-\Delta E/T},$$

em que

- ΔE é a diferença de pontuação ($f(\text{vizinho}) - f(\text{atual})$) e
- T é a temperatura atual.

A Figura 2 exibe o gráfico da função P para diferentes valores de ΔE , variando T . As curvas preta, azul e vermelha correspondem a ΔE igual a 20, 40 e 80, respectivamente.

Conforme a temperatura diminui, a probabilidade de aceitar estados piores também se reduz, tornando o algoritmo mais conservador.

O algoritmo pode ser descrito da seguinte forma:

TEMPERASIMULADA($f, T_{\text{inicial}}, \text{taxa-resfriamento}, \text{max-it}$)

```
1   $s \leftarrow$  estado aleatório
2   $T \leftarrow T_{\text{inicial}}$ 
3   $it \leftarrow 0$ 
4  while  $it < \text{max-iter}$  do
5       $\text{vizinho} \leftarrow \text{selecionar-vizinho}(s)$ 
6       $\Delta E \leftarrow f(\text{vizinho}) - f(s)$ 
7      if  $\Delta E > 0$  then
8           $s \leftarrow \text{vizinho}$ 
9      else
10         if  $\text{RANDOM}(0, 1) < e^{-\Delta E/T}$  then // RANDOM gera um número entre  $[0, 1]$ 
11              $s \leftarrow \text{vizinho}$ 
12      $T \leftarrow T \times \text{taxa-resfriamento}$ 
13 return  $s$ 
```

Os parâmetros são:

- **Estado inicial** (s_{inicial}): Ponto de partida do algoritmo.
- **Temperatura inicial** (T_{inicial}): Define a probabilidade inicial de aceitar estados piores.
- **Taxa de resfriamento** ($\text{taxa}_{\text{resfriamento}}$): Controla a redução da temperatura a cada iteração.
- **Número máximo de iterações** (max_{iter}): Limita a execução do algoritmo.

4 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 3:

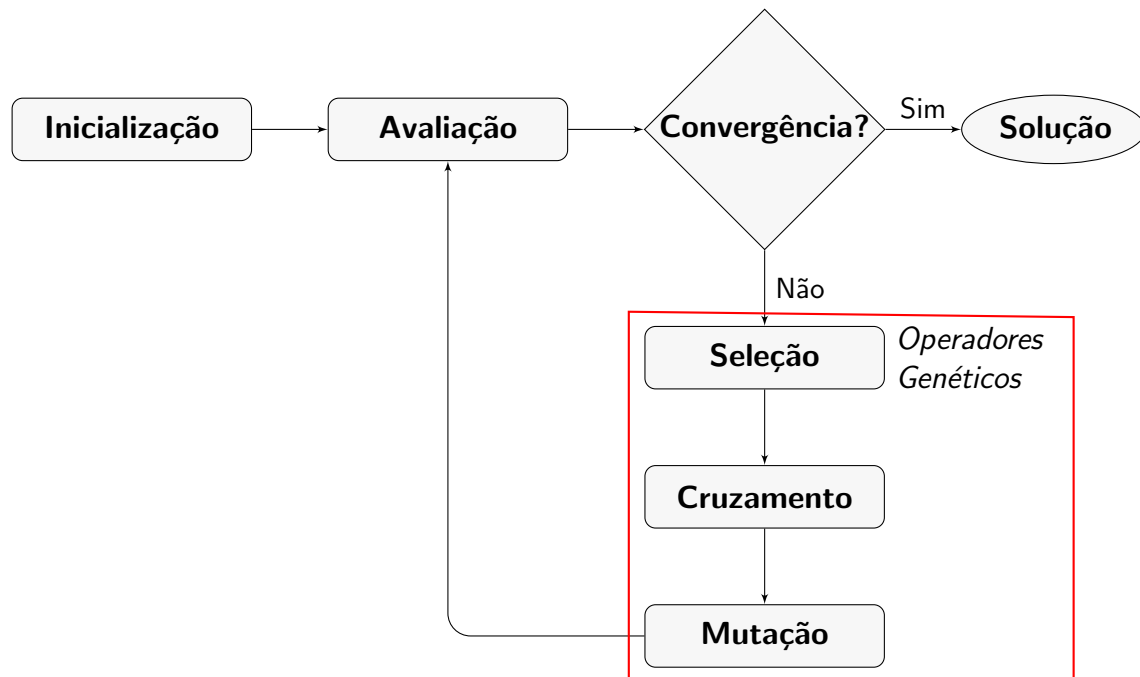


Figura 3: Algoritmo genético. O algoritmo termina quando um critério de convergência é atingido.

- **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 4 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.

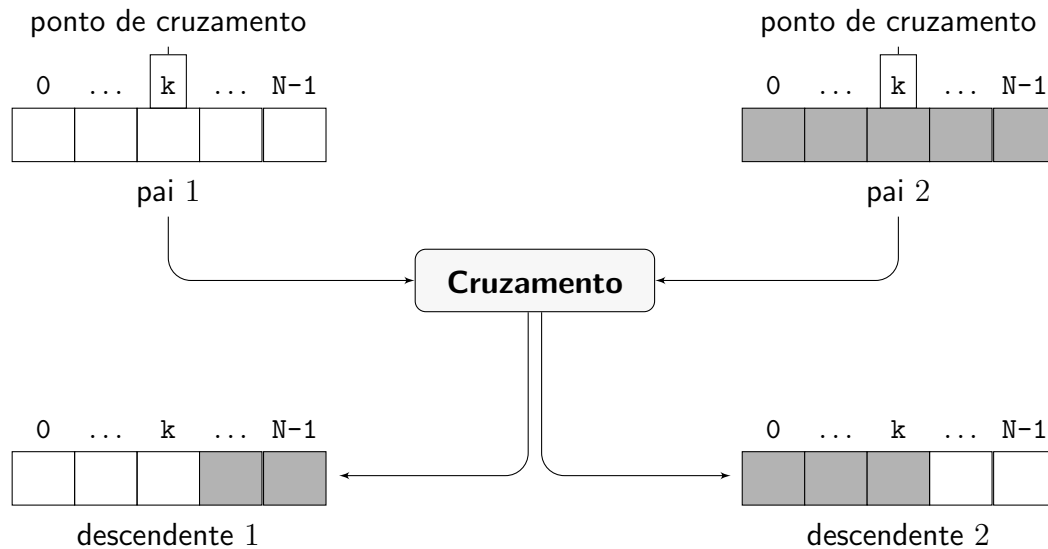


Figura 4: Operação de cruzamento. Os descendentes são uma combinação dos pais.

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.

5 Discussão Final

Cada um dos algoritmos analisamos nessa aula apresenta vantagens e desvantagens que influenciam sua eficácia em diferentes cenários.

A escolha do algoritmo depende das características do problema e das restrições práticas, como tempo disponível e precisão necessária. Subida da Encosta é ideal para problemas diretos e bem definidos. Já o Algoritmo Genético e a Tempera Simulada são mais indicados para desafios complexos que exigem maior flexibilidade e capacidade de exploração.

6 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.