

Engenharia de Computação
Inteligência Artificial
14 - Sistemas Imunológicos Artificiais
Prática Guiada: CLONALG e NSA

Retomando e avançando

O que já estudamos

- AG: seleção, recombinação e mutação.
- ACO: construção por feromônios.
- BPSO: velocidade probabilística e memória.
- CLONALG: seleção, clonagem e hipermutação.

O que faremos

- Executar um CLONALG completo.
- Interpretar escolhas do código e parâmetros.
- Reutilizar o mesmo problema de 16 bits.
- Planejar a adaptação ao Trabalho 02.

Questão central

Como mudar o mecanismo de busca mantendo a representação e o problema?

Roteiro de 100 minutos

Parte 1: execução e compreensão (50 minutos)

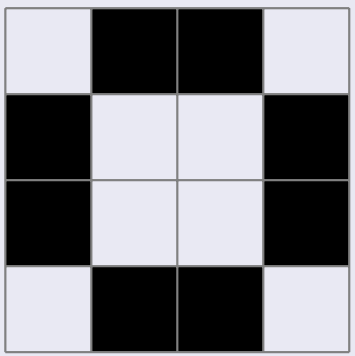
Etapa	Tempo
Relembrar a imagem-alvo e o fitness	5 min
Conhecer o código, a entrada e a saída	10 min
Acompanhar seleção, clones e hipermutação	15 min
Experimentos com parâmetros e seeds	15 min
Discussão de resultados e limites	5 min

Parte 2: adaptação ao Trabalho 02 (50 minutos)

Etapa	Tempo
Identificar requisitos e representação	10 min
Mapear mudanças no código	15 min
Adaptar e executar um teste mínimo	20 min
Registrar pendências e decisões	5 min

Mesmo problema de AG, ACO e BPSO

Imagem-alvo 4×4



Representação binária

`0110 1001 1001 0110`

- Um candidato contém 16 bits.
- Fitness: número de posições iguais ao alvo.
- Escala: 0 (pior) até 16 (ótimo).

$$f(x) = \sum_{i=1}^{16} \mathbf{1}[x_i = t_i]$$

Observação

É um problema didático: o algoritmo não consulta diretamente os bits do alvo para construir candidatos; usa apenas o retorno do fitness.

Arquivos e execução

Pacote entregue

- `clonalg.py`: implementação completa do algoritmo principal.
- `testes.py`: verificações básicas de funcionamento.
- `nsa.py`: demonstração complementar de detecção de anomalias.
- `README.md`: comandos e roteiro de interpretação.

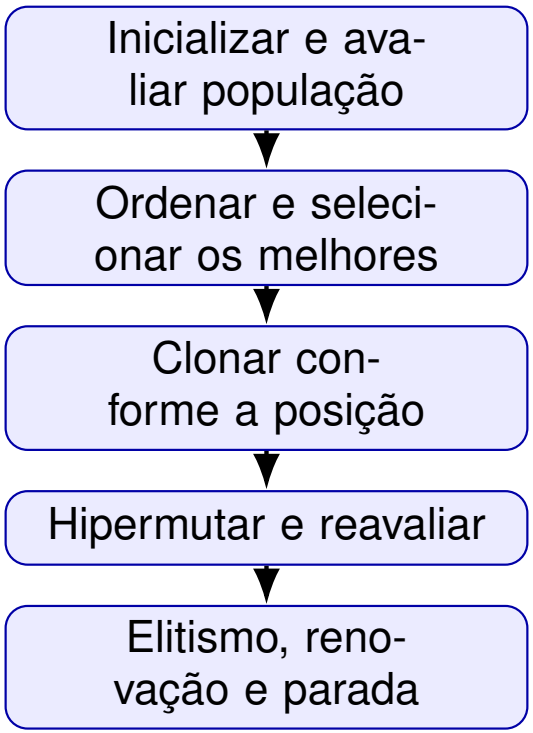
Primeiro checkpoint: execute, sem modificar

1. Execute `python testes.py`.
2. Execute `python clonalg.py -seed 42 -avaliacoes 1000`.
3. Registre melhor fitness, avaliações utilizadas e imagem produzida.

Observação

Somente biblioteca padrão do Python. O código está pronto: o objetivo é compreender e adaptar, não preencher lacunas.

CLONALG: de onde vêm as soluções?



Leitura do código

Em qual etapa a função `fitness` é chamada? Por que precisamos contar essas chamadas?

Afinidade: a qualidade da solução

Neste problema de maximização

$$a(x) = \frac{f(x)}{16}, \qquad 0 \leq a(x) \leq 1.$$

Fitness	Afinidade	Interpretação
16	1,00	todos os bits corretos
12	0,75	12 coincidências
8	0,50	8 coincidências
0	0,00	nenhuma coincidência

Conexão com o trabalho

Se o problema for minimizar custo, faz sentido utilizar $f/16$? Que transformação será necessária?

Clonagem: quantos clones gerar?

Regra utilizada na implementação

$$N_i = \left\lceil \frac{\beta\, n_s}{i} \right\rceil, \quad i = 1, \ldots, n_s.$$

Posição	i	$\beta = 1; n_s = 5$	Clones
Melhor	1	$\lceil 5/1 \rceil$	5
Segundo	2	$\lceil 5/2 \rceil$	3
Terceiro	3	$\lceil 5/3 \rceil$	2
Quarto	4	$\lceil 5/4 \rceil$	2
Quinto	5	$\lceil 5/5 \rceil$	1

Observação

Esta é uma variante didática de clonagem por posição. Não é uma fórmula única ou obrigatória de CLONALG.

Hipermutação: quanto modificar?

Probabilidade de inversão para cada bit

$$p_m(a) = p_{\min} + (p_{\max} - p_{\min})(1 - a).$$

Afinidade a	p_m ($p_{\min} = 0{,}02$, $p_{\max} = 0{,}35$)	Efeito
1,00	0,0200	pouca alteração
0,50	0,1850	alteração intermediária
0,00	0,3500	maior alteração

Interpretação

Um clone de alta afinidade pode ser alterado? Esta probabilidade garante exatamente um número de inversões?

Uma mutação não é necessariamente uma inversão

Exemplo de alterações independentes por bit

$$1100 \rightarrow 1000 \rightarrow 1001$$

- O primeiro passo inverte o segundo bit.
- O segundo passo inverte o quarto bit.
- O resultado final apresenta **duas inversões** em relação ao original.
- No código, cada bit tem sua própria chance de inverter: podem ocorrer zero, uma ou várias inversões.

Verifique no código

Localize a função `hipermutar`. Explique a condição que decide a inversão de cada bit e o papel da semente.

Elitismo, diversidade e orçamento

Explorar boas soluções

- Avaliar clones.
- Manter os melhores candidatos.
- Registrar o melhor já encontrado.

Evitar perda de diversidade

- Reservar vagas para candidatos aleatórios.
- Monitorar repetição de soluções.
- Comparar diferentes sementes.

Observação

A execução usa orçamento máximo de chamadas à função de avaliação e também pode encerrar ao atingir fitness 16. Para comparar métodos, registre o número efetivo de avaliações.

Pergunta

Por que comparar somente “50 gerações de AG” e “50 gerações de CLONALG” não é suficiente?

Checkpoint: experimentos controlados (15 min)

Rode o código e registre os resultados

1. Execute com seeds 0, 1 e 42, mantendo os parâmetros.
2. Compare `-p-max 0.10` e `-p-max 0.60`.
3. Teste `-beta 0.5` e `-beta 2.0`.
4. Registre: melhor fitness, avaliações, solução e ocorrência do ótimo.

Configuração	Seed	Fitness	Avaliações
Base	0, 1, 42		
$p_{\max} = 0{,}10$	42		
$p_{\max} = 0{,}60$	42		
$\beta = 0{,}5$	42		
$\beta = 2{,}0$	42		

Discussão: interpretar antes de concluir (5 min)

Resultados

1. O que ocorreu quando aumentamos $p_{\max}$?
2. Mais clones implicam mais avaliações por geração?
3. Uma única semente basta para concluir qual método é melhor?
4. O histórico de melhor fitness pode diminuir? Por quê?

Observação

O problema de 16 bits é pequeno, e soluções ótimas podem aparecer rapidamente. Isso não comprova desempenho em instâncias maiores ou com restrições.

NSA: demonstração complementar

Outra finalidade dentro dos Sistemas Imunológicos Artificiais

- `nsa.py` gera detectores que não reconhecem exemplos conhecidos como normais.
- Em seguida, examina novas cadeias e sinaliza aquelas reconhecidas por detectores.
- O critério didático é coincidência de $r = 4$ posições consecutivas.

Se houver tempo, execute

`python nsa.py`. Explique por que "não detectada" não significa que a amostra seja seguramente normal.

Observação

NSA detecta padrões; CLONALG busca soluções de otimização. Não devem ser comparados como se resolvessem a mesma tarefa.

Trabalho 02: reconhecer o que deve mudar (10 min)

Releia o enunciado do seu problema

1. Qual é a solução candidata? Vetor binário, permutação ou vetor real?
2. Qual função mede a qualidade? Maximização ou minimização?
3. Quais restrições determinam a viabilidade?
4. Quais são os operadores de variação admissíveis?
5. Que métricas permitem comparar AG, enxames e CLONALG?

Observação

O código fornecido resolve somente a imagem de 16 bits. Adaptá-lo ao trabalho exige modelar o problema; trocar apenas a imagem-alvo não é suficiente para todos os temas.

Mapa de adaptação: do exemplo ao seu tema (15 min)

Componente	Decisão necessária no Trabalho 02
`ALVO` / representação	Como codificar uma solução válida?
`criar_candidato`	Como criar soluções iniciais?
`fitness`	Como avaliar objetivo e restrições?
`hipermutar`	Como modificar uma solução sem corromper a representação?
`afinidade`	Como normalizar a qualidade, inclusive na minimização?
`executar`	Como selecionar, preservar diversidade e parar?
Histórico / métricas	O que registrar para comparar os métodos?

Pergunta orientadora

O mesmo operador de inversão de bits serve para roteamento, seleção de atributos e ajuste de parâmetros reais?

Adaptação orientada (20 min)

Em grupo, comece pelo menor teste possível

1. Crie uma representação mínima do tema e duas soluções válidas.
2. Implemente ou ajuste a avaliação e teste valores conhecidos.
3. Defina um operador de mutação compatível com a representação.
4. Só depois conecte o ciclo CLONALG; preserve orçamento e seed.
5. Registre o que funcionou e o que falta implementar.

Observação

Se o algoritmo ainda não estiver completo, apresente ao menos os testes da representação, da avaliação e das restrições. Essas partes são pré-requisitos para uma comparação confiável.

Comparação com AG, ACO e BPSO

Método	Mecanismo característico	Elemento comum
AG	Seleção, crossover e mutação	Mesma avaliação
ACO	Amostragem por feromônios	Mesma representação viável
BPSO	Velocidades e melhores posições	Mesma função objetivo
CLONALG	Clones, hipermutação e renovação	Mesmo orçamento de avaliações

Discussão

Quais adaptações são necessárias para que todos resolvam *o mesmo problema*, com a mesma função objetivo e tratamento das restrições?

Encerramento: entregáveis da aula (5 min)

Registre o progresso do grupo

1. Resultado de pelo menos uma execução de `clonalg.py`.
2. Uma observação fundamentada sobre clonagem ou hipermutação.
3. Definição da representação e da função objetivo do Trabalho 02.
4. Um teste mínimo documentado e a lista de pendências.

Síntese

O código está pronto para o *exemplo didático*; no trabalho, o principal desafio é a modelagem correta e a avaliação comparável entre métodos.

Próximo conteúdo

Como combinar diferentes modelos para melhorar a robustez das previsões? Introdução a métodos ensemble.

Dúvidas

