Parte II: IA Simbólica ou Good Old-Fashioned Artificial Intelligence (GOFAI).

Prof. Fabiano Araujo Soares, Dr. / FGA 0221 - Inteligência Artificial

Universidade de Brasília

2025

Resolvendo problemas por busca: Ambientes complexos

Por que encontrar soluções eficientes em ambientes complexos é um dos maiores desafios da inteligência artificial moderna?

Resolvendo problemas por busca: Ambientes complexos

Por que encontrar soluções eficientes em ambientes complexos é um dos maiores desafios da inteligência artificial moderna?

Buscar soluções em ambientes complexos não é apenas um desafio computacional – é o próprio exercício de transformar incerteza em inteligência.

Busca em Ambientes Complexos

- Muitos problemas de inteligência artificial — como otimização de layout de circuitos, programação automática, logística, design de portfólios e alocação de recursos — focam na obtenção de uma **configuração final ótima**, conforme critérios de desempenho ou custo.
- Nesses cenários, o **caminho percorrido** até a solução é irrelevante; importa apenas a qualidade do estado final alcançado.
- Ambientes complexos normalmente envolvem um espaço de busca muito grande, características estocásticas, múltiplos objetivos e restrições, o que torna métodos tradicionais de busca sistemática inviáveis ou caros computacionalmente.

Busca em Ambientes Complexos

- Algoritmos de **busca local** e otimização — como *hill climbing*, *simulated annealing*, algoritmos genéticos e *beam search* — avaliam apenas o estado corrente e seus vizinhos, sem armazenar trajetórias completas.
- Essas estratégias são essenciais em aplicações que requerem decisões rápidas, atuam em ambientes dinâmicos ou lidam com incerteza, como robótica, indústria, jogos, planejamento logístico e sistemas embarcados.

Busca Local - Introdução

- A busca local inicia a partir de um **estado inicial** e explora sistematicamente apenas seus **estados vizinhos imediatos**, ou sucessores, sem recuperar ou armazenar o caminho percorrido.
- Ao contrário da busca sistemática, que mantém memória de todos os estados visitados para evitar redundâncias e ciclos, a busca local consome **muito menos memória**, focando na melhoria incremental da solução atual.
- A busca local é geralmente **estocástica** e não sistemática, o que significa que pode:
  - Não explorar todas as regiões do espaço de estados;
  - Ficar presa em máximos ou mínimos locais, ou mesmo áreas planas (platôs);
  - Não garante encontrar a solução global ótima.

Busca Local - Introdução

- Essas características a tornam adequada para problemas com espaços de estados muito grandes, onde o custo de expandir todo o espaço é proibitivo.
- Exemplos comuns incluem algoritmos como **hill-climbing**, **simulated annealing** e **algoritmos genéticos**, amplamente utilizados em otimização e resolução de problemas reais complexos.

Exemplos de Algoritmos de Busca Local

- **Hill-climbing (ou Gradient Descent)**,
- **Random-restart hill-climbing**,
- **Simulated annealing**,
- **Beam search**,
- **Algoritmos genéticos**.

Hill-Climbing - Introdução

- Hill-Climbing é um algoritmo de busca local para otimização.
- A ideia principal é iterativamente mover para o vizinho que melhora a solução atual.
- O processo continua até que não existam vizinhos melhores, indicando um ótimo local.

Hill-Climbing - Funcionamento

1. Comece com uma solução inicial (estado qualquer).
2. Avalie os estados vizinhos da solução atual.
3. Se algum vizinho for melhor, mova para ele.
4. Repita até que nenhum vizinho seja melhor.

Exemplo: Encontrar o máximo na função $f(x) = -x^2 + 4x + 1$

- Espaço de estados: valores inteiros de x entre 0 e 5.
- Avaliação: $f(x)$ para cada x.

x	$f(x)$
0	1
1	4
2	5
3	4
4	1
5	-4

Exemplo (continuação): Passos do Hill-Climbing

- Estado inicial: $x = 0$, valor $f(0) = 1$.
- Vizinhos: $x = 1$ (vizinho à direita), valor $f(1) = 4$.
- Melhor vizinho: $x = 1$, mover para $x = 1$.

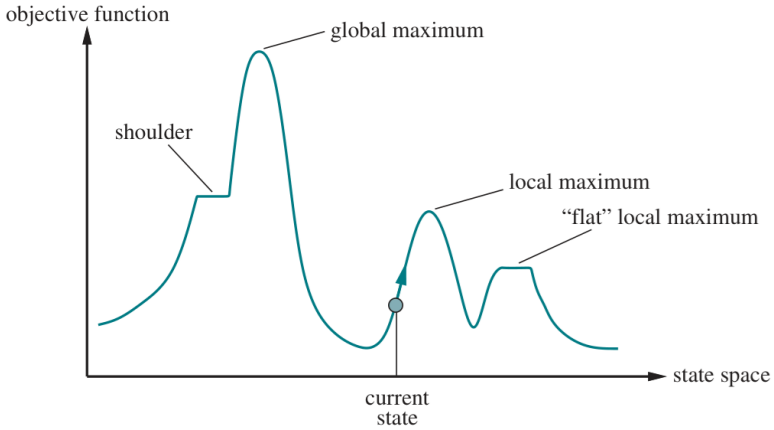
Exemplo (continuação): Passos do Hill-Climbing

- Estado atual: $x = 1$, valor $f(1) = 4$.
- Vizinhos: $x = 0$ ($f(0) = 1$), $x = 2$ ($f(2) = 5$).
- Melhor vizinho: $x = 2$, mover para $x = 2$.

Exemplo (continuação): Passos do Hill-Climbing

- Estado atual: $x = 2$, valor $f(2) = 5$.
- Vizinhos: $x = 1$ ($f(1) = 4$), $x = 3$ ($f(3) = 4$).
- Nenhum vizinho é melhor, pois $5 > 4$.
- Parada: ótimo local encontrado em $x = 2$ com valor 5.

Considerações sobre Hill-Climbing



Considerações sobre Hill-Climbing

- Converge rápido ao ótimo local.
- Pode ficar preso em máximos locais, sem garantir a solução global.
- Estratégias como reinício aleatório e simulated annealing ajudam a superar ótimos locais.

Simulated Annealing - Introdução

- Metaheurística de busca local probabilística inspirada no processo físico de recozimento (annealing) na metalurgia.
- Permite aceitar soluções piores temporariamente para escapar de mínimos locais.
- A "temperatura"controla a probabilidade de aceitar soluções ruins e vai diminuindo ao longo do tempo.

Simulated Annealing - Funcionamento

1. Comece com uma solução inicial e uma temperatura alta.
2. Em cada iteração, escolha um estado vizinho aleatório.
3. Se o vizinho for melhor, aceite-o.
4. Se for pior, aceite-o com probabilidade $p = e^{-\Delta E/T}$, onde ΔE é a piora da função objetivo e T a temperatura atual.
5. Diminua a temperatura gradualmente (resfriamento).
6. Repita até a temperatura estar muito baixa ou critério de parada ser atingido.

Exemplo: Maximizar $f(x) = -x^2 + 4x + 1$ com $x \in [0, 5]$

x	$f(x)$
0	1
1	4
2	5
3	4
4	1
5	-4

- Inicialize em $x = 0$, $f(0) = 1$, temperatura alta.
- Escolha um vizinho aleatório, por exemplo, $x = 1$.
- Como $f(1) = 4$ é melhor, aceite-o.

Exemplo (continuação): Aceitando solução pior

- De $x = 2$ ($f = 5$) para $x = 3$ ($f = 4$) — piora!
- Aceite com probabilidade $p = e^{-\Delta E/T}$.
- Se a temperatura for alta, há boa chance de aceitar para evitar mínimos locais.
- Conforme a temperatura diminui, a chance de aceitar soluções piores também diminui.

Benefícios e Limitações

- Evita ficar preso em mínimos ou máximos locais ao permitir movimentos para soluções piores.
- A escolha adequada da temperatura inicial e resfriamento é fundamental.
- Pode ser computacionalmente mais custoso que Hill-Climbing puro.

Variações

- **Random-restart hill-climbing**: Executa várias vezes o hill-climbing, guarda o melhor resultado.
- **Beam Search**: mantém k estados, seleciona os melhores vizinhos para próxima geração.

Algoritmos Genéticos - Introdução

- Algoritmos Genéticos (AG) são métodos de busca e otimização inspirados no processo de evolução natural.
- Utilizam uma **população** de candidatas a solução (chamadas indivíduos).
- O objetivo é **evoluir** essa população ao longo de gerações para encontrar soluções cada vez melhores.

Representação dos Indivíduos

- Cada indivíduo é representado por um **DNA**, uma codificação das variáveis da solução.
- Exemplo: posições das rainhas no problema das 8 damas podem ser representadas por um vetor onde cada posição indica a linha da rainha em uma coluna específica.
- A escolha da representação é crítica para o desempenho do algoritmo.

Parâmetros do Algoritmo Genético

- **Tamanho da população:** número de indivíduos por geração.
- **Número de parentes (ρ):** indivíduos que irão gerar prole (comum $\rho = 2$).
- **Seleção:** método para escolher quais indivíduos serão pais.
- **Recombinação (crossover):** combinação do DNA dos pais para criar descendentes.
- **Mutação:** alteração aleatória em parte do DNA para introduzir diversidade.
- **Nova geração:** como substituir ou manter indivíduos antigos (elitismo, descarte, etc.).

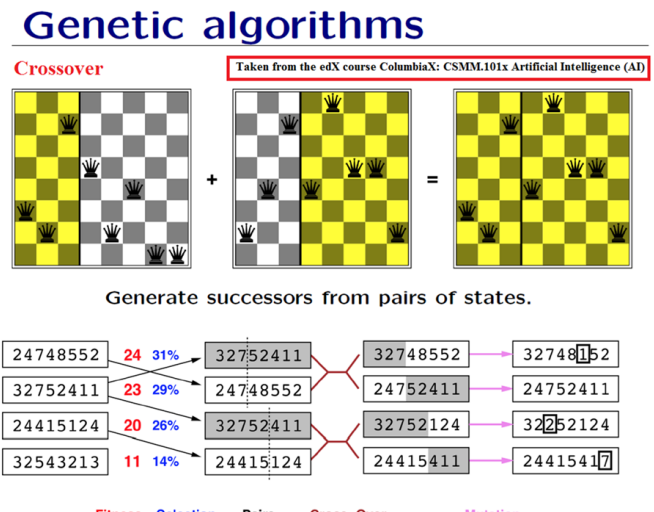
Ciclo do Algoritmo Genético

1. Inicialização: cria a população inicial aleatoriamente ou baseada em heurísticas.
2. Avaliação: calcula o **fitness** de cada indivíduo (qualidade da solução).
3. Seleção: escolhe pais para gerar descendentes.
4. Recombinação: cria novos indivíduos combinando os pais.
5. Mutação: aplica mutações nos descendentes.
6. Substituição: forma a próxima geração com descendentes e possivelmente alguns pais.
7. Repetição até condição de parada (número de gerações ou critério de convergência).

Problema das 8 Damas - Representação

- O problema consiste em colocar 8 damas em um tabuleiro de xadrez para que nenhuma ataque outra.
- Representação típica: vetor de tamanho 8, cada elemento indica a linha da dama na respectiva coluna.
- Exemplo: $[1, 3, 5, 7, 2, 0, 6, 4]$.
- Fitness pode ser baseado no número de pares de damas que não se atacam.

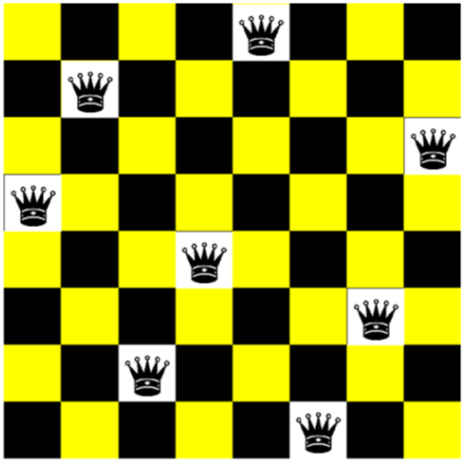
Algoritmo Genético no Problema das 8 Damas



Algoritmo Genético no Problema das 8 Damas

- População inicial: gera várias configurações aleatórias.
- Avaliação: conta conflitos entre damas para definir fitness.
- Seleção: pais com melhor fitness têm maior chance de reprodução.
- Recombinação: mistura as posições das rainhas entre pares de pais.
- Mutação: altera aleatoriamente a posição de uma rainha para aumentar diversidade.
- Nova geração: repete o processo até encontrar solução sem conflitos.

Algoritmo Genético no Problema das 8 Damas - Solução



Algoritmos Genéticos: Vantagens e Desafios

- **Vantagens:**
  - Efetivo para problemas com grandes espaços de busca.
  - Robusto contra mínimos locais.
  - Fácil paralelização.
- **Desafios:**
  - Definir boa função de fitness.
  - Ajustar parâmetros (população, taxa de mutação, etc.).
  - Pode convergir prematuramente (diversidade insuficiente).

Referência do Código do Algoritmo

- Algoritmo extraído de `https://github.com/chengxi600/RLStuff/blob/master/Genetic%20Algorithms/8Queens_GA.ipynb`
- Disponível em Python, inclui todos os passos descritos.
- Pode ser usado para estudo e adaptação para outros problemas.

Resumo

- Estratégias locais e informadas são essenciais para IA eficiente em grandes espaços de busca.
- Variantes de busca local (hill-climbing ou gradient descent, simulated annealing, beam search, algoritmos genéticos) são ferramentas práticas.
- Escolha da técnica depende do problema, do custo computacional e da natureza do objetivo.

Referências

- Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach.* 4ª ed.
- Slides oficiais: *aima.cs.berkeley.edu*
- Outros: materiais complementares em IA.

Obrigado!

E-mail: fabianosoares@unb.br

LinkedIn: https://www.linkedin.com/in/fabiano-soares-06b6a821a/

Site do curso: https://www.fabianosoares.eng.br/fga0221-inteligência-artificial