Inteligência Artificial

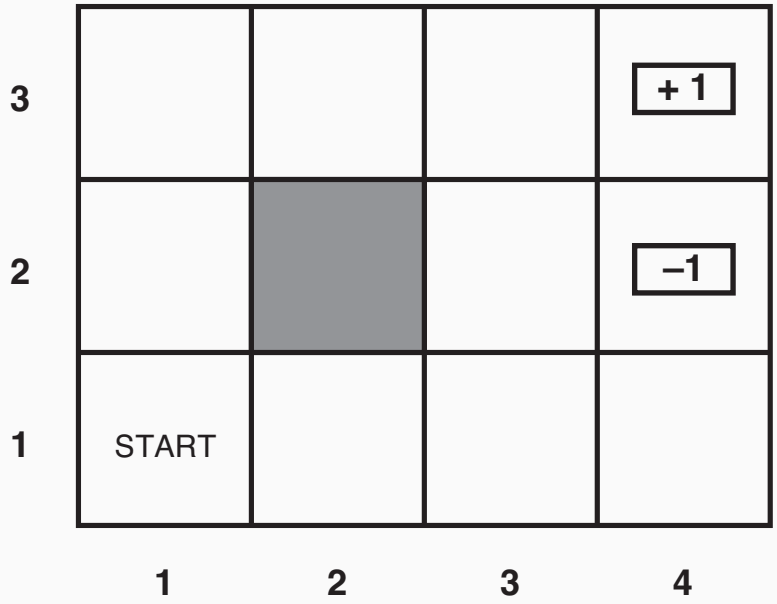
Prof. Fabrício Olivetti de França Prof. Denis Fantinato

3º Quadrimestre de 2019

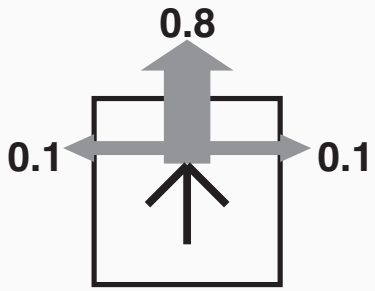
Problemas de Decisão Sequencial (prévia)

Problemas de Decisão Sequencial

Imagine que um agente esteja no seguinte ambiente:



(a)



(b)

Problemas de Decisão Sequencial

Começando do *início* o agente deve escolher uma sequência de ações até chegar a um dos estados objetivos: $+1$ ou -1.

Problemas de Decisão Sequencial

Nesse exemplo $Acoes(s) = \{Cima, Baixo, Esquerda, Direita\}$.

Vamos assumir um ambiente totalmente observável.

Problemas de Decisão Sequencial

Se o ambiente for determinístico, a solução é simples: *{Cima, Cima, Direita, Direita, Direita}*.

Problemas de Decisão Sequencial

Digamos que cada **tentativa** de executar uma ação faz a ação requisitada com probabilidade $p = 0.8$.

Com $p = 0.1$ o agente executa uma ação perpendicular a ação requisitada, e $p = 0.1$ para a outra ação perpendicular.

Problemas de Decisão Sequencial

Exemplo: ao requisitar que o agente ande para cima, ele terá 80% de chances de executar tal ação, 10% de chances de andar para a esquerda e 10% de chances de andar para a direita.

Problemas de Decisão Sequencial

Caso o agente bata em uma das paredes, ele permanece no local atual.

O quadrado em cinza é uma parede e não pode ser acessada.

Heurísticas e Meta-Heurísticas

Heurística

Heurística, derivada do grego “encontrar” ou “descobrir”, são técnicas para encontrar a solução de um problema sem garantia de obter algum ótimo (nem mesmo local) ou que seja racional.

Heurística

Objetivo de obter um alvo imediato (ex.: colocar a rainha da coluna *i* na fileira de menor número de ataques).

Heurística

George Pólya enumera algumas dicas para criar uma heurística e tentar resolver um problema (*How to Solve it, 1945*):

- Se não consegue entender o problema, desenhe uma imagem ou diagrama representativo.
- Se não consegue chegar do estado inicial até uma solução, tente partir da solução e chegar ao estado inicial.
- Se o problema é abstrato, crie um exemplo concreto.
- Tente resolver um problema mais genérico primeiro (possivelmente menos restritivo).

Heurística

Métodos heurísticos são particularmente vantajosos em ambientes:

- Parcialmente ou não observáveis
- aleatórios
- Número elevado de estados (ou espaço de estados muito grande)
- sem objetivo definido (dada uma métrica de qualidade)

Heurística

Uma forma simples de busca heurística é a **Busca Local** que, em certas condições, garante encontrar um ótimo local em tempo tratável e utilizando memória constante.

Busca Local

Busca Local

A **Busca Local** parte de uma solução inicial qualquer e, iterativamente, caminha para soluções vizinhas até um ponto de convergência.

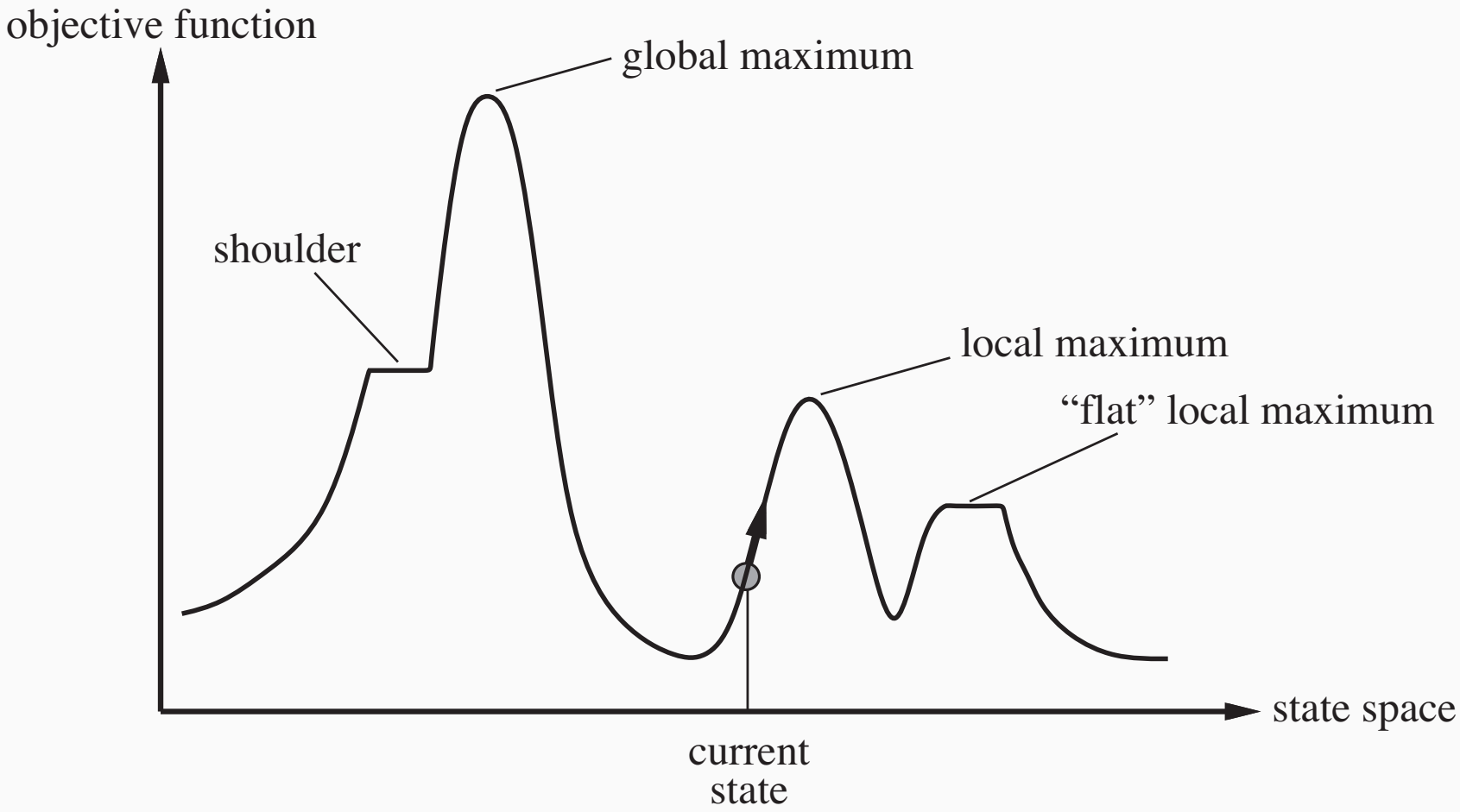
Busca Local

Na Busca Local, os passos para chegar até a solução tipicamente não são guardados em memória, ou seja, apenas **tem por objetivo encontrar uma solução** sem saber os passos que levaram até ela.

Busca Local

Para aplicar um algoritmo de busca local precisamos definir o espaço de estados como uma função $f : S \mapsto \mathbb{R}$ que mapeia um estado a um valor numérico indicando a qualidade desse estado.

Busca Local


objective function
global maximum
shoulder
local maximum
“flat” local maximum
state space
current
state

Busca Local

Um algoritmo de busca local é dito **completo** se sempre encontra um estado final (se existir);

É dito **ótimo** se sempre encontra o melhor dos estados finais.

Exercício

Imagine que em nosso Grid World aplicamos a seguinte busca local:

- Inicia com uma política aleatória
- Verifica cada solução vizinha como a alteração da política para apenas um estado
- Se pelo menos um vizinho for melhor que a política atual, substitua a política atual pelo melhor vizinho

Exercício

A avaliação é feita pela simulação da política partindo do estado inicial limitando o número de passos em 100.

Ela é uma busca local completa? Ela é ótima?

Exercício

Considere agora um algoritmo que gera uma nova política aleatória em cada passo.

Ela é uma busca local completa? Ela é ótima?

Hill-climbing

O algoritmo de busca local **hill-climbing** ou de **maior subida** simplesmente repete iterativamente os passos:

- Avalia todos os vizinhos do estado atual
- Caminha para o estado vizinho de maior valor, se maior que o atual

E para quando não existem mais vizinhos melhores.

Hill-climbing

A ideia geral do algoritmo é o de escalar uma montanha pelo lado mais íngrime, ou seja, de subida mais rápida.

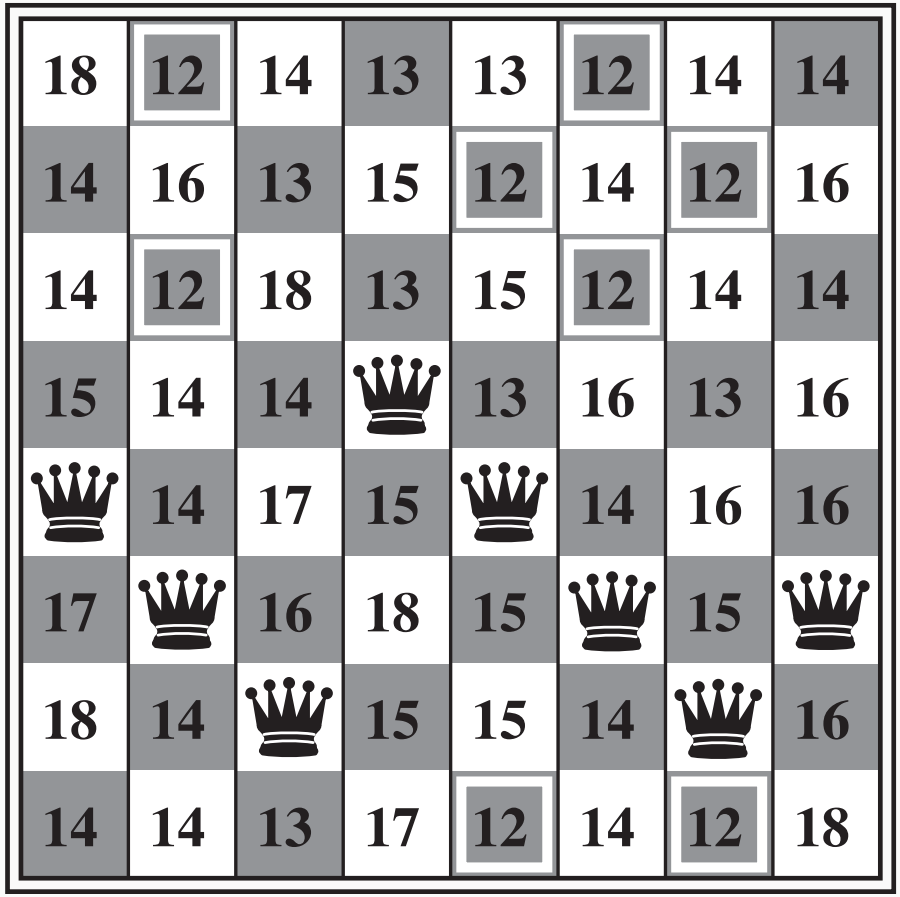
Hill-climbing para 8-Rainhas

Vamos definir o estado do problema das 8-rainhas como a lista de fileiras contendo uma rainha em cada coluna:

1	2	3	4	5	6	7	8
5	6	7	4	5	6	7	6

Hill-climbing para 8-Rainhas

Esse estado representa esse tabuleiro:



Hill-climbing para 8-Rainhas

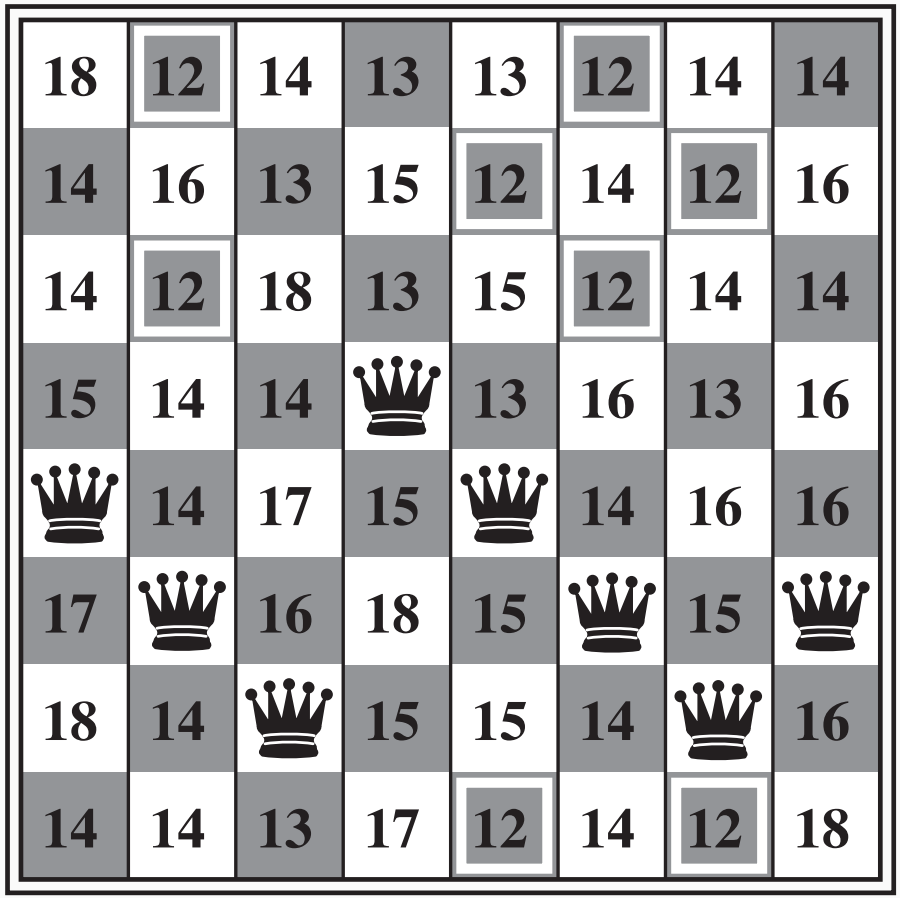
Um estado vizinho é definido como alterar um dos números de nosso estado, movendo uma rainha para outra fileira:

1	2	3	4	5	6	7	8
5	6	7	4	5	6	7	6

1	2	3	4	5	6	7	8
5	1	7	4	5	6	7	6

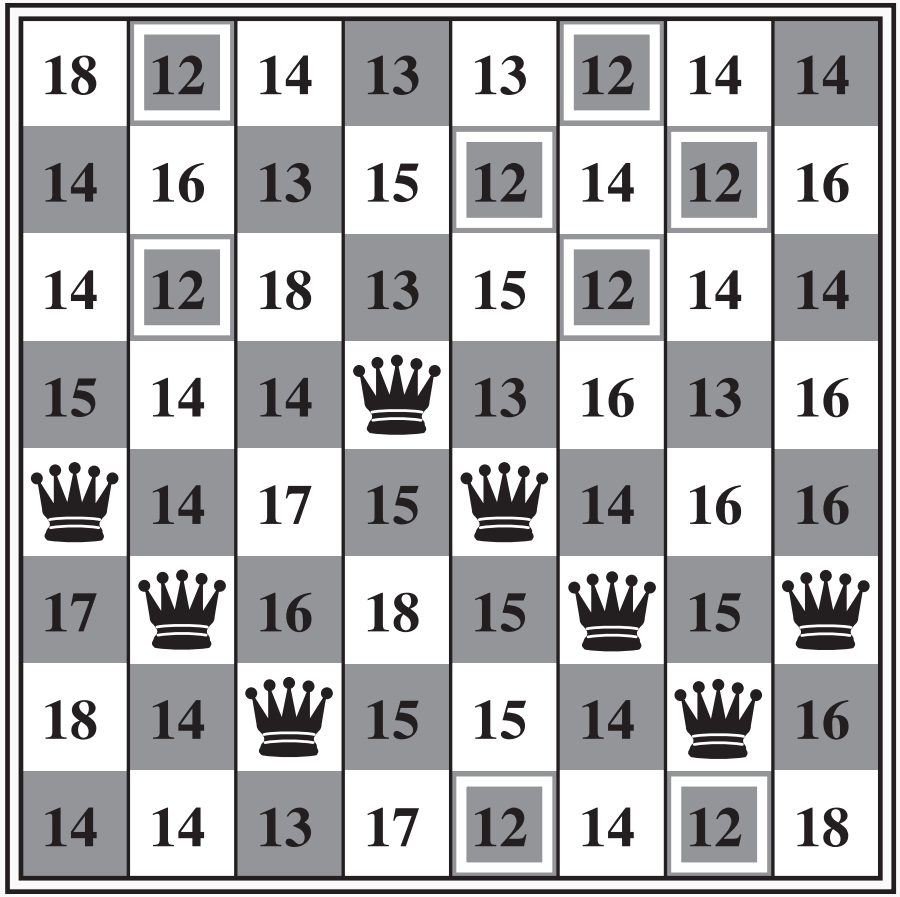
Hill-climbing para 8-Rainhas

Definimos como função-objetivo o número de pares de rainhas se atacando (direta ou indiretamente), com isso conseguimos mensurar o valor de cada estado vizinho:



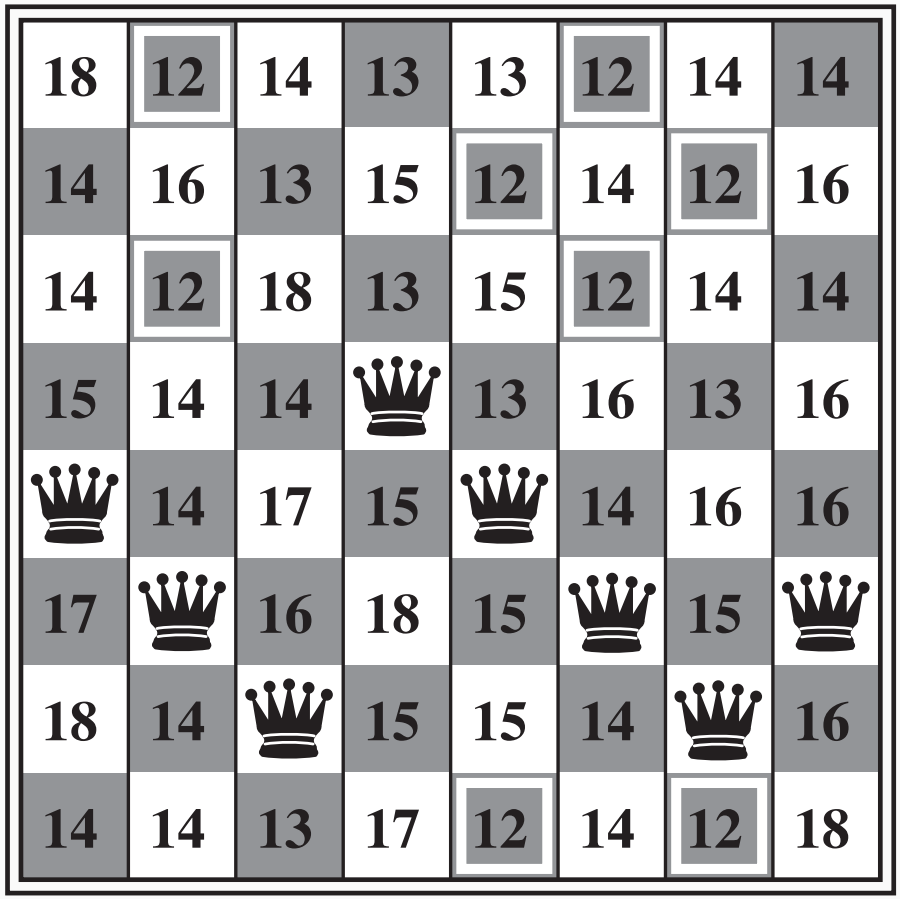
Hill-climbing para 8-Rainhas

Qual o valor da função-objetivo para esse estado?



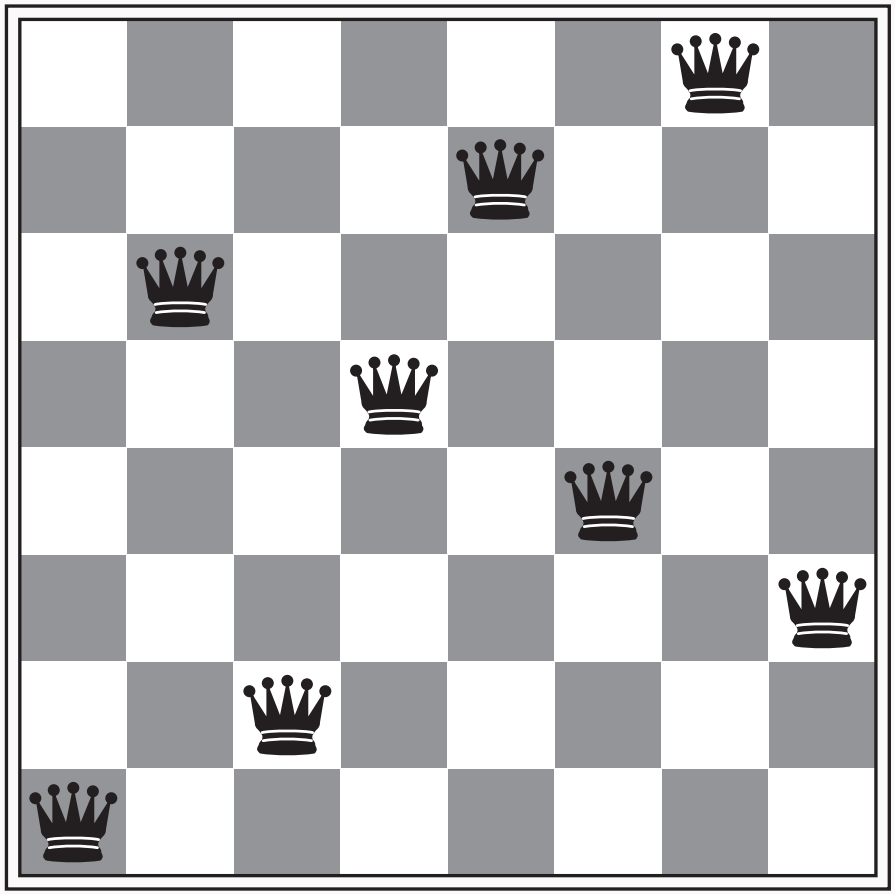
Hill-climbing para 8-Rainhas

Escolhendo sucessivamente o vizinho de menor valor (estamos minimizando), qual o estado final?



Hill-climbing para 8-Rainhas

Escolhendo sucessivamente o vizinho de menor valor (estamos minimizando), eventualmente chegamos em um ótimo local com valor f(s) = 1:



Hill-climbing para 8-Rainhas

Para esse problema, considerando diferentes estados inicias, ele consegue chegar na solução em 14% dos casos.

Ou seja, em 86% dos casos, o Hill-climbing não consegue achar uma solução.

Por outro lado, quando ele chega em um estado final, utiliza, em média, apenas 4 passos (lembrando que o espaço de busca é de $8^8 \approx 17$ milhões de estados.

Hill-climbing para 8-Rainhas

Alterando o algoritmo para permitir fazer uma mudança para um estado vizinho de mesmo valor quando nenhum vizinho melhor é encontrado, aumenta as chances de sucesso para 94%, porém elevando o número de passos médios para 21 (e 64 passos para casos sem sucesso).

Hill-climbing Estocástico

Variações

Escolhe aleatoriamente um dos vizinhos dentre aqueles melhores que o estado atual.

A probabilidade de escolha pode ser proporcional a quanto melhora.

Hill-climbing com Reinício Aleatório

Aplica o Hill-climbing em diferentes estados iniciais e retorna a melhor solução obtida.

Essa estratégia consegue resolver o problema de 3 milhões de rainhas em menos de um minuto!

Hill-climbing

O sucesso do Hill-climbing dependerá da forma da função objetivo no espaço de busca (presença de máximos/mínimos locais, platôs, pontos de cela)

Hill-climbing

O algoritmo de Hill-climbing é incompleto pois, dado um estado inicial, está limitado a vizinhança de seu ótimo local.

Hill-climbing

Por outro lado, um algoritmo de **caminhante aleatório** é completo pois pode chegar em qualquer estado do espaço de estados, porém é ineficiente.

Recozimento Simulado

Um meio termo é definido pelo algoritmo de **Recozimento Simulado** (**Simulated Annealing**).

Recozimento Simulado

A ideia é que a escolha do próximo estado seja aleatória, se esse estado for melhor que o atual, é aceito, caso contrário ele substitui o atual com probabilidade $e^{\frac{\Delta E}{T}}$, sendo ΔE a diferença entre a função-objetivo desse estado com o estado atual e uma T a *temperatura* que é reduzida a cada iteração.

Recozimento Simulado

O comportamento inicial do algoritmo é de ser mais permissivo quanto a piora do estado mas, com o passar das iterações, ele tende a aceitar apenas estados que apresentam melhoras.

Se a temperatura for diminuída devagar o suficiente, esse algoritmo encontra o ótimo global com probabilidade se aproximando de 1.

Recozimento Simulado

```
def SimulatedAnnealing(problema):
    s0 = solucaoInicial()
    T = 100
    while T > eps:
        s = estadoVizinhoAleatorio(s0)
        if f(s) > f(s0):
            s0 = s
        else:
            if random() <= exp((f(s)-f(s0))/T):
                s0 = s
        T = reduz(T)
```

Parte II

Meta-Heurísticas

Algoritmo Genético

O **Algoritmo Genético** é uma meta-heurística populacional da família de **Algoritmos Bio-Inspirados**.

Os **Algoritmos Bio-Insprados** são técnicas heurísticas inspiradas em fenômenos biológicos ou comportamentos observados na natureza.

Algoritmo Genético

Especificamente, o Algoritmo Genético tem como inspiração o processo de recombinação e mutação do material genético durante o processo de reprodução.

Algoritmo Genético

Cada solução é denominada de **indivíduo** e codifica um estado do problema. Essa codificação é denominada **cromossomo**.

O algoritmo trabalha com um conjunto de soluções denominada **população**.

Algoritmo Genético

Em cada passo do algoritmo, os indivíduos da população passam por um processo de recombinação (também denominado cruzamento).

Os novos indivíduos passam por um processo de mutação, que altera uma pequena parte da solução e, finalmente, o conjunto de indivíduos pais e filhos são combinados e selecionados para gerar uma nova população.

Cruzamento

O cruzamento tem o objetivo de formar novas soluções com a melhor parte de duas ou mais soluções da população. Ela seleciona $n > 1$ indivíduos e gera n novos indivíduos (denominados filhos) como a recombinação das partes dos indivíduos *pais*:

0	**1**	1	0	1	0
1	1	**0**	**0**	**1**	**1**

0	**1**	**0**	**0**	**1**	**1**
1	1	1	0	1	0

Mutação

No processo de mutação, alguns poucos elementos da solução são alteradas aleatoriamente. Esse procedimento tem o objetivo de escapar de possíveis ótimos locais causando perturbações aleatória no estado representado pela solução:

0	1	0	**0**	1	1
0	1	0	**1**	1	1

Seleção

Dada uma população de n indivíduos, o procedimento de seleção junta a população atual com a população filha gerada nos passos anteriores formando um conjunto de $m > n$ indivíduos.

Dessa população, são selecionados n indivíduos para formar a população da próxima iteração.

Seleção

O procedimento mais utilizado de seleção é o torneio, nele dois ou mais indivíduos são selecionados aleatoriamente e o melhor deles é escolhido para fazer parte da próxima população. Esse processo é repetido n vezes até completar o conjunto de soluções.

Schema

A ideia básica do funcionamento do Algoritmo Genético é a de que o processo de recombinação e a pressão seletiva do procedimento de seleção faz com que, com o passar das iterações, indivíduos com **boas soluções parciais** sejam mantidos na população e sua recombinação eventualmente conduza a uma solução boa.

Schema

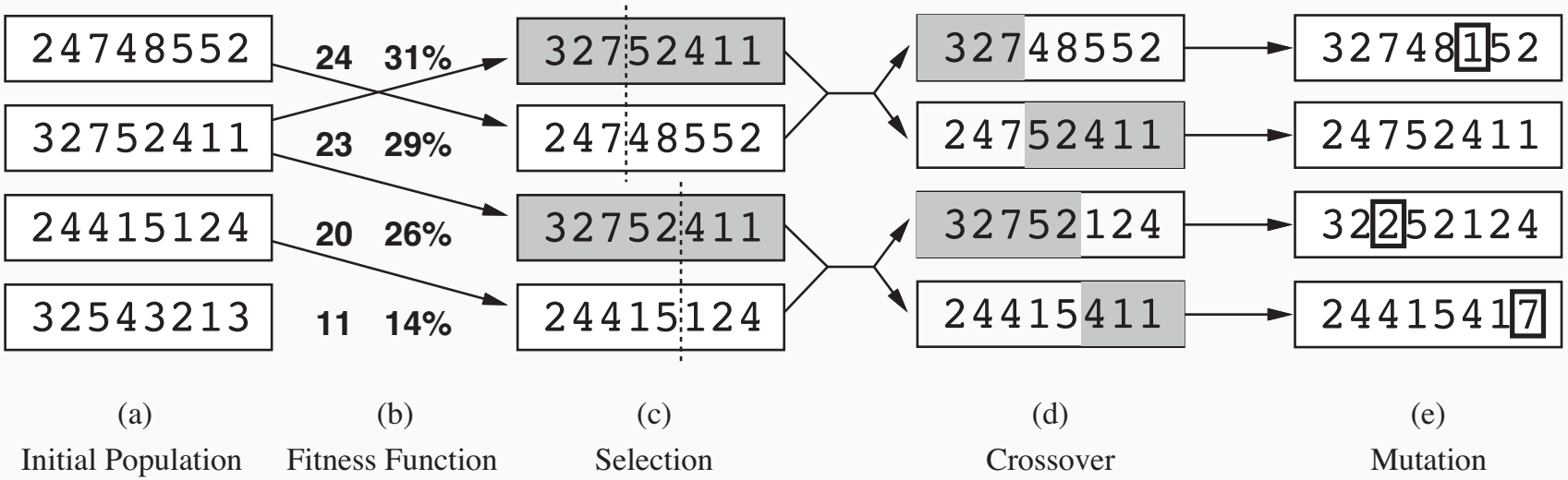
No problema das 8 Rainhas, imagine a solução parcial $2, 4, 6, *$, essa solução não contém nenhum par de rainhas se atacando. Tal solução pode ser um bom bloco construtor para soluções completas que resolvam o problema.

Schema

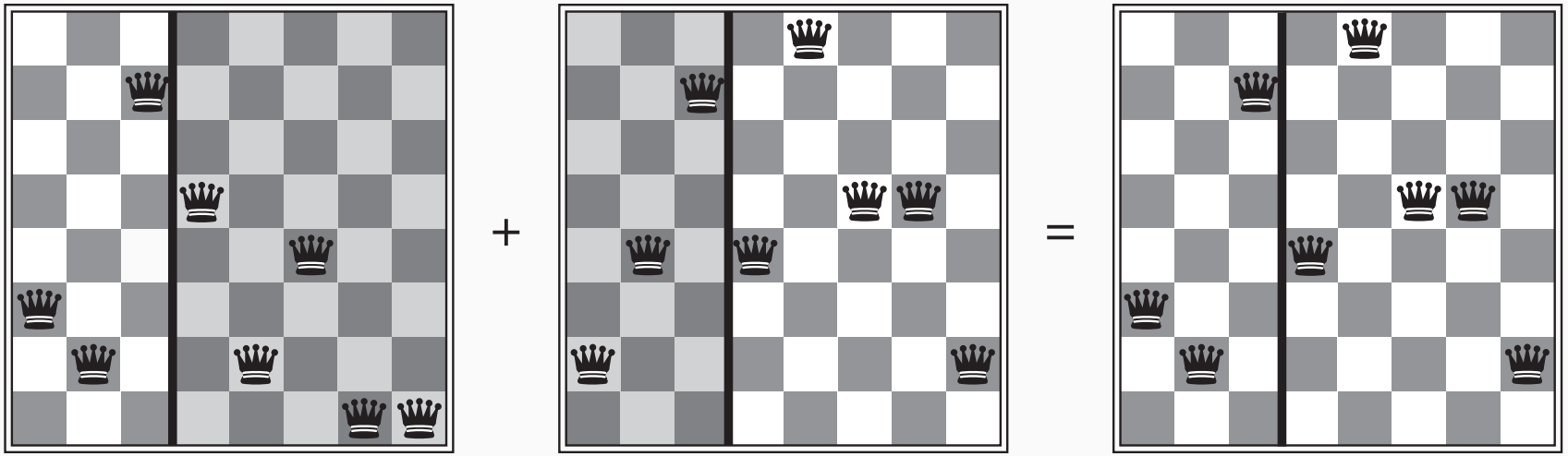
Imagine agora a solução parcial $*, 1, 5, 7$, da mesma forma nenhum par de rainhas se atacam. Considere a combinação da solução parcial anterior com essa, teremos $2, 4, 6, *, *, 1, 5, 7$. Nenhum par de rainhas se atacam, com essa solução parcial precisaríamos apenas resolver a posição das duas rainhas do meio para então encontrar a solução do problema.

8 Rainhas

fitness: número de pares de raínhas que não se atacam (sucesso com fitness = 28)



8 Rainhas



Algoritmo Genético

Em resumo o Algoritmo Genético pode ser definido como:

```
def GA():
    P = solucoesAleatorias(n)
    while naoConvergiu():
        P'  = cruzamento(P)
        P'' = mutacao(P')
        P   = seleciona(P, P'')
    return melhor(P)
```

8-Puzzle

Ao definirmos a representação, o procedimento de cruzamento e mutação para um determinado problema de busca, não só podemos implementar um algoritmo genético como também qualquer um dos algoritmos de busca local descritos anteriormente.

8-Puzzle

Defina a representação, cruzamento e mutação para o problema 8-Puzzle.

8-Puzzle

Representação: a quantidade de movimentos no pior caso desse problema é 31, o conjunto de ações pode ser definido utilizando dois bits: 00 (para cima), 01 (para baixo), 10 (para esquerda), 11 (para direita).

8-Puzzle

Nosso cromossomo pode ser representado por um vetor de 62 bits, com isso o cruzamento é feito da mesma forma como ilustrada acima e a mutação é a simples troca de um bit.

8-Puzzle

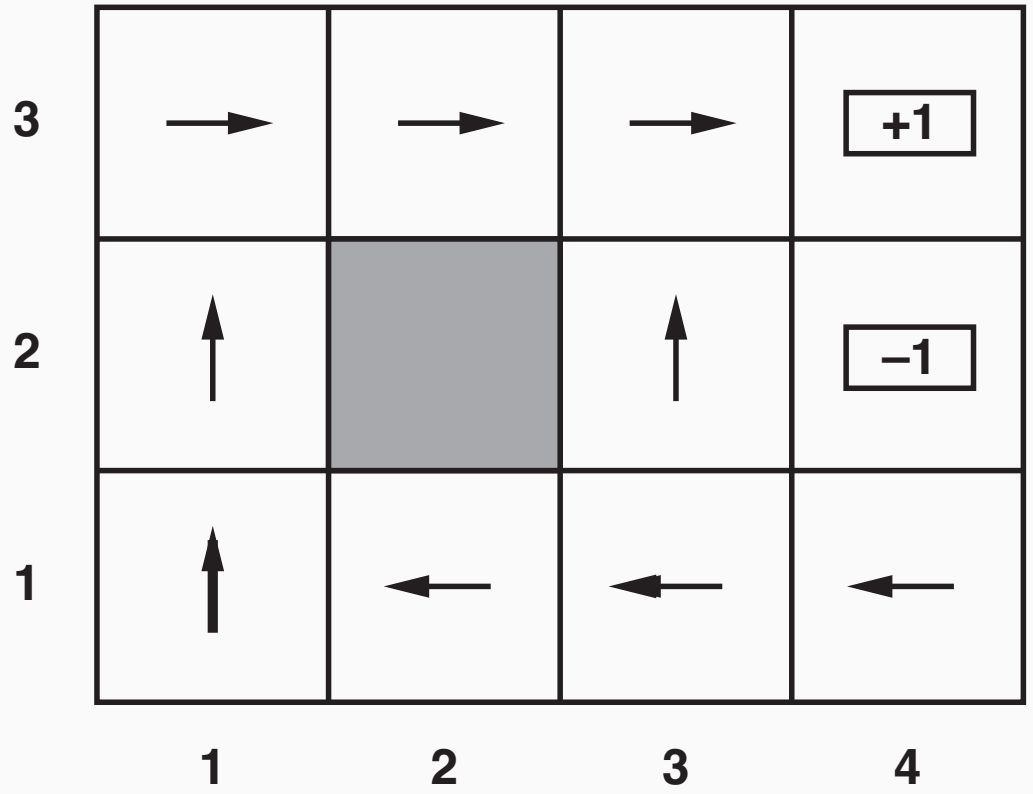
Outra possibilidade é a de codificar simbolicamente os movimentos em um vetor de tamanho 31, sendo que o cruzamento permanece o mesmo, mas a mutação se torna a troca de um símbolo por um outro. Qual a vantagem em relação a anterior?

8-Puzzle

A função-objetivo pode ser a distância do estado final após seguir a sequência de instruções ao estado-alvo.

Grid World

No nosso problema do Grid World podemos buscar por uma política ótima representando nossa solução como um vetor de 9 posições equivalente a cada coordenada com ação. Cada posição apresenta uma direção que deve ser seguida.



Grid World

Nossa função-objetivo é definida pela utilidade esperada ao seguir essa política em n simulações.

Super Mario World

Como vocês fariam para o jogo Super Mario World?

Estratégias Evolutivas

Um algoritmo relacionado ao Algoritmo Genético é a heurística Estratégias Evolutivas, criada inicialmente para trabalhar com problemas cujos estados são representados por valores contínuos.

Estratégias Evolutivas

As Estratégias Evolutivas seguem um padrão similar ao anterior, elas são caracterizadas como $ES(\mu/\rho, \lambda)$ ou $ES(\mu/\rho + \lambda)$. Sendo:

- μ o número de soluções em uma população
- ρ quantas soluções serão escolhidas para o cruzamento
- λ o número de soluções-filhas criadas

A ‘,’ indica que apenas os filhos participarão do processo de seleção enquanto na estratégia ‘+’ tanto a população pai como a filha são consideradas durante o processo.

Estratégias Evolutivas

Uma solução no algoritmo ES é representado por um vetor de variáveis contínuas que correponde as variáveis do problema:

x1	x2	x3
0.5	-1.1	3.2

O cruzamento mais comum é feito como a média ponderada de diferentes soluções, para duas solucões-pais:

$$f = \alpha p_1 + (1 - \alpha) p_2$$

Sendo $0 \leq \alpha \leq 1$ escolhido aleatoriamente.

Estratégias Evolutivas

A mutação é uma perturbação aleatória pequena na solução, comumente com a soma de um vetor aleatório com distribuição gaussiana:

$$f = f + \mathcal{N}(0, 1)$$

Grid World

Para o Grid World, podemos utilizar um vetor contínuo com 12 elementos representando o **valor de utilidade** de cada estado.

A função-objetivo é definida como a execução de n tentativas de seguir a política ótima para aquele vetor de utilidades.

Conclusão

Em situações onde lidamos com **incerteza** ou com problemas cujo **espaço de busca** é intratável, podemos utilizar algoritmos de busca local ou busca heurística para encontrar uma solução, mesmo que sem garantia de obter o ótimo global.