



Universidade do Minho

Escola de Engenharia

Departamento de Informática

Procura Local e Otimização

Algoritmos de Solução única

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

Inteligência Artificial

2024/25

Agenda

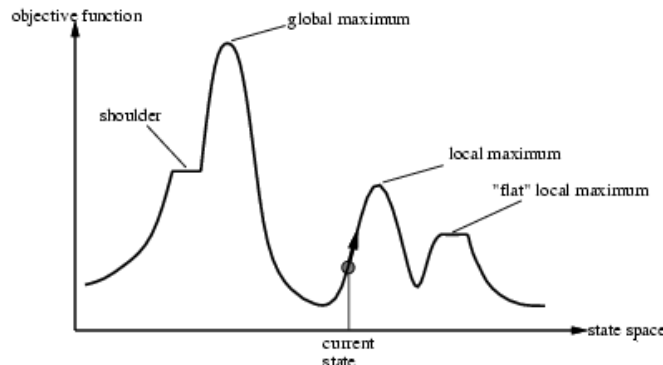
- Procura vs Otimização
- Exploration vs Exploitation
- Meta-heurísticas
- Procura local vs procura global
- Solução única vs Population-Based
- Solução única
 - Procura Subida da Colina (Hill-Climbing Search)
 - Arrefecimento Simulado (Simulated Annealing)
 - Procura Tabu (Tabu Search)
- Procura com ações não determinísticas



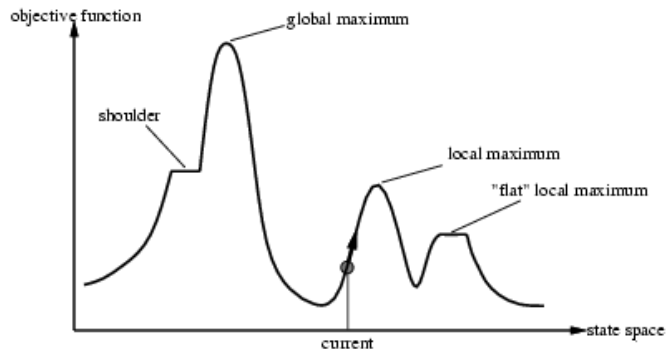
Problemas de Otimização

- Como é que um fabricante de talheres (cutelaria) pode obter o máximo de peças de um pedaço de chapa?
- Como é que uma transportadora pode se organizar para distribuir as encomendas num dado contentor ou transporte?
- Como é que uma operadora telefónica pode rotear chamadas para obter o melhor uso das suas linhas?
- Como uma universidade pode elaborar os horários das suas aulas tendo o melhor uso das salas de aula sem conflitos?

- Até agora abordamos, na essência, uma única categoria de problemas: ambientes observáveis, determinísticos e conhecidos, onde a solução é uma sequência de ações.
 - **Nem todos os ambiente são assim!**
- Algoritmos que executam procura puramente local no espaço de estados, avaliando e modificando um ou mais estados atuais, em vez de explorar sistematicamente caminhos a partir de um estado inicial.
- Esses algoritmos são adequados para problemas nos quais tudo o que importa é o estado da solução, não o custo do caminho para alcançá-lo.



- Não interessa o caminho até ao objetivo
ex: problema das rainhas, horários, otimização - de redes, de chão de fábrica, etc.
- Num problema de otimização - pode não se saber se já se atingiu o valor ótimo
se não conhecemos o ótimo da função que estamos a otimizar...



Exploration vs Exploitation

- **A resolução de problemas complexos requer a obtenção das melhores soluções possíveis em tempo útil;**
- **Para isso, ao invés da experimentação de todas as soluções possíveis (garantindo a solução ótima), é necessária a identificação de soluções o mais próximas quanto possível da solução ótima, num número limitado de tentativas;**
- **É então, essencial um balanceamento adequado entre:**
 - ***Exploration:*** exploração geral do espaço de procura;
 - ***Exploitation:*** procura focada nas zonas mais promissoras.

Exploration vs Exploitation

- ***Exploration* sem *Exploitation* permite ter uma visão geral do espaço de procura, mas sem chegar muito próximo do valor ótimo;**
- **Partir para a *Exploitation* de uma zona numa fase inicial do processo de procura pode levar a que a procura fique presa num ótimo local;**
- **Este balanceamento é normalmente gerido com sucesso através de Meta-heurísticos.**

- **Uma meta-heurística é um método heurístico para resolver de forma genérica problemas de otimização;**
- **Meta-heurísticas são geralmente aplicadas a problemas para os quais não se conhecem algoritmos eficientes;**
- **Utilizam uma combinação de escolhas aleatórias e de conhecimento histórico dos resultados anteriores adquiridos pelo método para se guiarem e realizar as suas procuras em vizinhanças dentro do espaço de procura, o que pode evitar ótimos locais.**

- **Normalmente inspiradas em fenómenos da natureza;**
- **Pela sua componente aleatória, são não-determinísticos;**
- **Não garantem a identificação da solução ótima:**
 - uma solução próxima;
 - No menor tempo de execução;
 - utilizando menos recursos computacionais que as técnicas tradicionais.

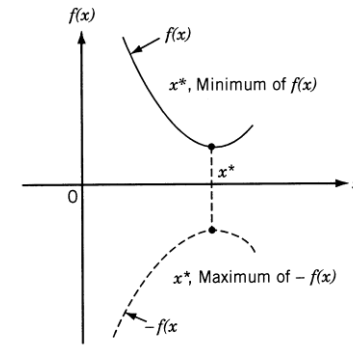
Problemas de Otimização

- **Problema de Otimização Matemática (minimização):**

minimize $f_0(x)$

subject to $g_i(x) \leq b_i, \quad i = 1, \dots, m$

- **$f_0: \mathbb{R}^n \rightarrow \mathbb{R}$:**
 - Função objetivo (calcula um valor)
- **$x = (x_1, \dots, x_n)$:**
 - Variáveis controláveis (linearmente independentes)
- **$g_i: \mathbb{R}^n \rightarrow \mathbb{R} \ (i = 1, \dots, m)$:**
 - Restrições (podem ser de outros tipos)



Para além da Procura Clássica Algoritmos de Melhoria Iterativa

- Em muitos problemas de otimização, o caminho para o objetivo é irrelevante.
- Espaço de Estados = conjunto das configurações completas.
- Algoritmos Iterativos mantêm um único estado (corrente) e tentam melhorá-lo.
- Algoritmos de Melhoria Iterativa:
 - Procura Subida da Colina (Hill-Climbing Search)
 - Arrefecimento Simulado (Simulated Annealing)
 - Procura Tabu (Tabu Search)
 - Algoritmos Genéticos (Genetic Algorithms)
 - Colónia de Formigas (Ant Colony Optimization)
 - Enxame de Partículas (Particle Swarm Optimization)
- **Estratégia:** Começar como uma solução inicial do problema e fazer alterações de forma a melhorar a sua qualidade

▪ **Procura local vs Procura global**

- Algumas meta-heurísticas aplicam métodos de procura local, onde as novas soluções exploradas são “vizinhas” de soluções anteriores (e.g. *Simulated Annealing, Tabu Search*);
- Outras meta-heurísticas distribuem o processo de procura por todo o espaço de procura (normalmente através de abordagens baseadas em populações).

▪ **Solução única vs *Population-based***

- As abordagens de solução única, são iterativas, e orientam o processo de procura através da melhoria da solução anterior;
- As abordagens baseadas em populações utilizam uma procura em paralelo por parte de vários membros da população, podendo, ou não, existir a troca de informação entre os indivíduos (e.g. *Particle Swarm optimization, Genetic Algorithms, Ant Colony optimization*).

Algoritmos de Melhoria Iterativa

Solução única

“Individual Based” (solução única)!

- Hill-Climbing Search:
 - Escolher um estado aleatoriamente do espaço de estados
 - Considerar todos os vizinhos desse estado
 - Escolher o melhor vizinho
 - Repetir o processo até não existirem vizinhos melhores
 - O estado atual é a solução
- Simulated Annealing:
 - Semelhante ao Hill-Climbing Search mas admite explorar vizinhos piores
 - Temperatura que vai sendo sucessivamente reduzida define a probabilidade de aceitar soluções piores
- Tabu Search:
 - Semelhante ao Hill-Climbing Search, explora os estados vizinhos mas elimina os piores (vizinhos tabu)
 - Algoritmo determinístico

Procura Subida da Colina (Hill-Climbing Search)

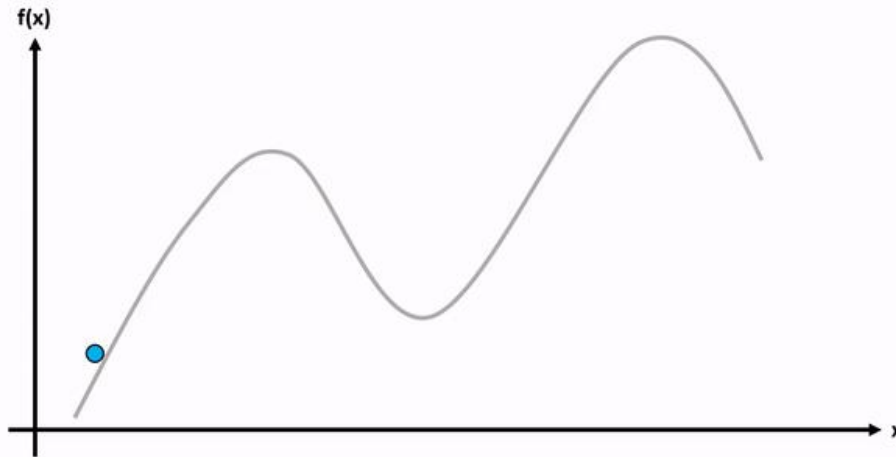
- **O Hill Climbing** é um algoritmo clássico, sendo bastante eficiente na tarefa de encontrar máximos ou mínimos locais, pela exploração.
- **Estratégia:**
 - Inicia-se num ponto aleatório X e avalia-se esse ponto;
 - Move-se desse ponto X original para um novo ponto Y vizinho do ponto X;
 - se esse novo ponto Y for uma solução melhor do que o ponto original X, fixa-se o ponto Y e inicia-se o processo novamente, porém caso seja inferior, volta-se para nosso ponto inicial X e tenta-se visitar um outro vizinho.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                  neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] > VALUE[current] then return STATE[neighbor]
    current ← neighbor
```

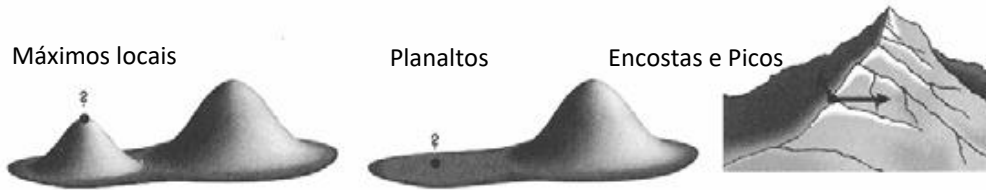
Procura Subida da Colina (Hill-Climbing Search)

- Hill Climbing é ótimo para encontrar boas soluções (mínimo/máximo locais) mas dificilmente vai encontrar a melhor solução (a menos que se tenha “sorte” na inicialização do ponto inicial).



Fonte: <https://www.globalsoftwaresupport.com/wp-content/uploads/2018/04/ezgif.com-video-to-gif-47.gif>

Procura Subida da Colina (Hill-Climbing Search)



- Nos casos apresentados, o algoritmo chega a um ponto de onde não tem mais progresso.
- **Solução:** *reinício aleatório (random restart)*
 - O algoritmo realiza uma série de procuras a partir de estados iniciais gerados aleatoriamente.
- Cada procura é executada:
 - Até que um número máximo estipulado de iterações seja alcançado, ou
 - Até que os resultados encontrados não apresentem uma melhoria significativa.
- O algoritmo escolhe o melhor resultado obtido com as diferentes procuras.

- O sucesso depende muito do morfologia (formato) da superfície do espaço de estados:
 - Se há poucos máximos locais, o reinício aleatório encontra uma boa solução rapidamente
 - Caso contrário, o custo de tempo é exponencial.

Arrefecimento Simulado (Simulated Annealing)

- O **Simulated Annealing** é um algoritmo inspirados pela natureza, assim como as Redes Neurais Artificiais e os Algoritmos Genéticos, entre outros.
- Pressuposto: Escapar do mínimo local permitindo alguns “maus” movimentos mas gradualmente diminuindo a sua dimensão e frequência!
- **Estratégia:**
 - Semelhante ao Hill Climbing, inicia-se num ponto aleatório X e avalia-se;
 - o algoritmo faz um movimento até um dos seus vizinhos Y e avalia esse novo ponto;
 - Se os resultados melhoraram nesse novo ponto Y então mover para Y e refazer o processo anterior, todavia se o ponto Y for inferior, mover para esse ponto Y caso a **probabilidade** de ir para um ponto negativo seja superior a um número aleatório.

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
         schedule, a mapping from time to "temperature"
  local variables: current, a node
                  next, a node
                  T, a "temperature" controlling prob. of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
     $\Delta E \leftarrow \text{VALUE}[\textit{next}] - \text{VALUE}[\textit{current}]$ 
    if  $\Delta E > 0$  then current ← next
    else current ← next only with probability  $e^{\Delta E / T}$ 
```

$$\text{probabilidade}(p) = \text{Exp}(Y - X / T)$$

Essa função exponencial calcula a diferença do ponto Y subtraída pelo ponto X (posição anterior), dividido pela temperatura (variável T). Com isso, nas primeiras iterações, quando temperatura T está mais elevada, existe uma probabilidade maior de aceitar valores negativos e vai diminuindo com iterações posteriores, a probabilidade vai diminuindo.

Com o tempo (diminuição da temperatura), o algoritmo passa a comporta-se como do de Hill-Climbing (Subida de Colinas)

Procura Tabu (Tabu Search)

- A ideia básica da procura Tabu é penalizar movimentos que levam a solução para espaços de procura visitados anteriormente (também conhecidos como tabu).
- A Procura Tabu, no entanto, aceita de forma determinística soluções que não melhoram para evitar ficar presa em mínimos locais.
- **Estratégia:**
 - Ideia chave: manter a sequência de nós já visitados (Lista tabu);
 - Partindo de uma solução inicial, a procura move-se, a cada iteração, para a melhor solução na vizinhança, não aceitando movimentos que levem a soluções já visitadas, esses movimentos conhecidos ficam armazenados numa lista tabu;
 - A lista permanece na memória guardando as soluções já visitadas (tabu) durante um determinado espaço de tempo ou um certo número de iterações (prazo tabu). Como resultado final é esperado que se encontre um valor ótimo global ou próximo do ótimo global.



Procura Tabu (Tabu Search)

Algorithm 1 Tabu search algorithm

```
Set  $x = x_0$ ;                                ▷ Initial candidate solution
Set  $length(L) = z$ ;                          ▷ Maximum tabu list length
Set  $L = \{\}$ ;                                ▷ Initialize the tabu list
repeat
  Generate a random neighbor  $x'$ ;
  if  $x' \notin L$  then
    if  $length(L) > z$  then
      Remove oldest solution from  $L$ ;          ▷ First in first out queue
      Set  $x' \in L$ ;
    end if
  end if
  if  $x' < x$  then
     $x = x'$ ;
  end if
until (Stopping criteria satisfied)           ▷ e.g. Number of iterations
return  $x$ ;                                    ▷ Best found solution
```

Procura com ações não determinísticas Falhas no mecanismo de aspiração

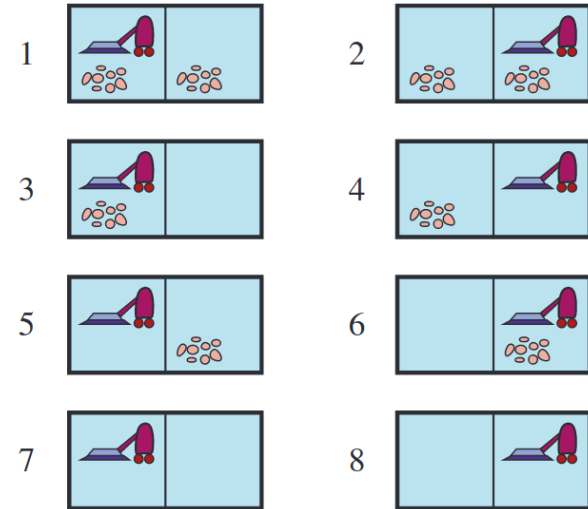
Supondo há existência de falhas ao aspirar:

- pode limpar duas células
- pode depositar sujeira

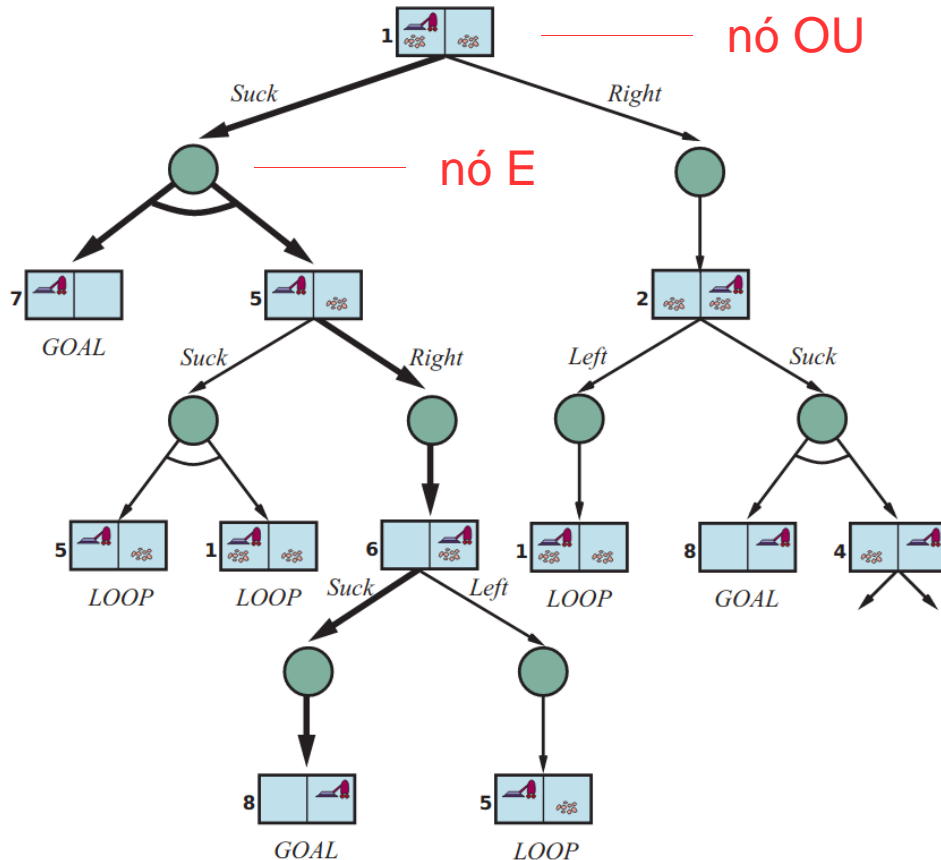
■ começando no estado 1

Plano de contingência

- aspirar
- **se** Estado = 5 **então**
Direita, Aspira



Árvores E-OU (*AND-OR*)

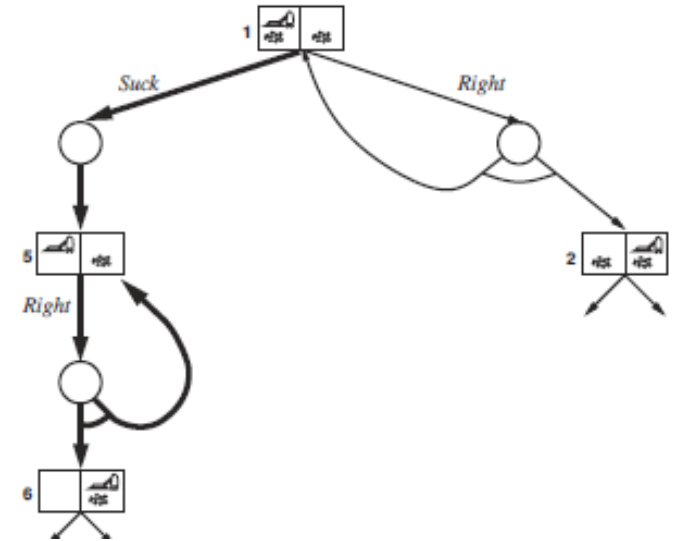


- ações do agente – nós OU
- result. no mundo – nós E
- Plano genérico
 - um objetivo em cada folha
 - uma ação em cada nó OU
 - todos os resultados em nós E

Procura com ações não determinísticas Falhas no mecanismo de movimentação

Supondo há existência de movimentação do agente (aspirador):

- as ações de movimento por vezes falham, deixando o agente na mesma localização.
- Por exemplo, mover-se para a direita no estado 1 leva ao conjunto de estados {1, 2}.
- Solução cíclica que consiste em continuar a tentar a Direita até funcionar.



- Abordou-se com os métodos de procura clássica, na essência, uma única categoria de problemas: ambientes observáveis, determinísticos e conhecidos, onde a solução é uma sequência de ações;
- Neste contexto, analisou-se o que acontece quando essas suposições são relaxadas. Usando algoritmos que realizam uma procura puramente local no espaço de estados, avaliando e modificando um ou mais estados ao invés de explorar sistematicamente caminhos a partir de um estado inicial;
- Estes algoritmos são adequados para problemas em que tudo o que importa é o estado da solução, não o custo do caminho para alcançá-lo;
- Esta família de algoritmos de procura local inclui métodos inspirados na física estatística (*Simulated annealing*) e na biologia evolutiva (*Genetic algorithms*) que iremos abordar mais tarde.

Bibliografia Recomendada

- Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach, 3rd edition, ISBN-13: 9780136042594, chapter 4.





Universidade do Minho

Escola de Engenharia

Departamento de Informática

Procura Local e Otimização

Algoritmos de Solução única

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

Inteligência Artificial

2024/25