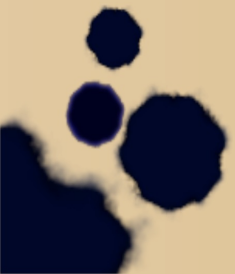


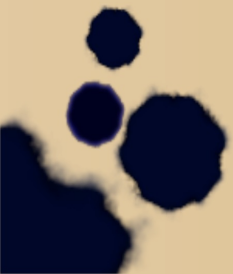
Métodos de Busca



Métodos de Busca




- Estratégias de Busca Cega
 - encontram soluções para problemas pela geração *sistemática* de novos estados, que são comparados ao objetivo;
 - são *ineficientes* na maioria dos casos:
 - são capazes de calcular apenas o custo de caminho do nó atual ao nó inicial (função g), para decidir qual o próximo nó da fronteira a ser expandido.
 - essa medida não necessariamente conduz a busca na direção do objetivo.
 - Como encontrar um barco perdido?
 - não podemos procurar no oceano inteiro...



Busca Heurística

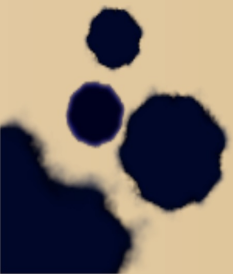


- Estratégias de Busca Heurística
 - utilizam ***conhecimento específico*** do problema na escolha do próximo nó a ser expandido
 - barco perdido
 - correntes marítimas, vento, etc...
 - Aplica de uma *função de avaliação* a cada nó na fronteira do espaço de estados
 - essa função ***estima o custo de caminho*** do nó atual até o objetivo mais próximo utilizando uma *função heurística*
 - Heurística
 - do grego heuriskein, encontrar, descobrir
 - introduzida por George Polya em 1957 (livro How to Solve It)
 - é conhecimento e, por isso, marcou quebra da IA com a pesquisa operacional
- 

Métodos de Busca Heurística



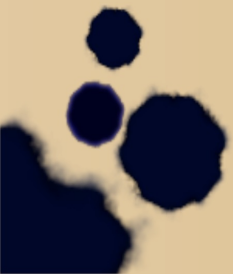
- Os métodos de busca vistos anteriormente fornecem uma solução para o problema de achar um caminho até um nó meta. Entretanto, em muitos casos, a utilização destes métodos é impraticável devido ao número muito elevado de nós a expandir antes de achar uma solução.
- Para muitos problemas, é possível estabelecer princípios ou regras práticas para ajudar a reduzir a busca.



Métodos de Busca Heurística



- A técnica usada para melhorar a busca depende de informações especiais acerca do problema em questão.
- Chamamos a este tipo de informação de INFORMAÇÃO HEURÍSTICA e os procedimentos de busca que a utilizam de MÉTODOS DE BUSCA HEURÍSTICA

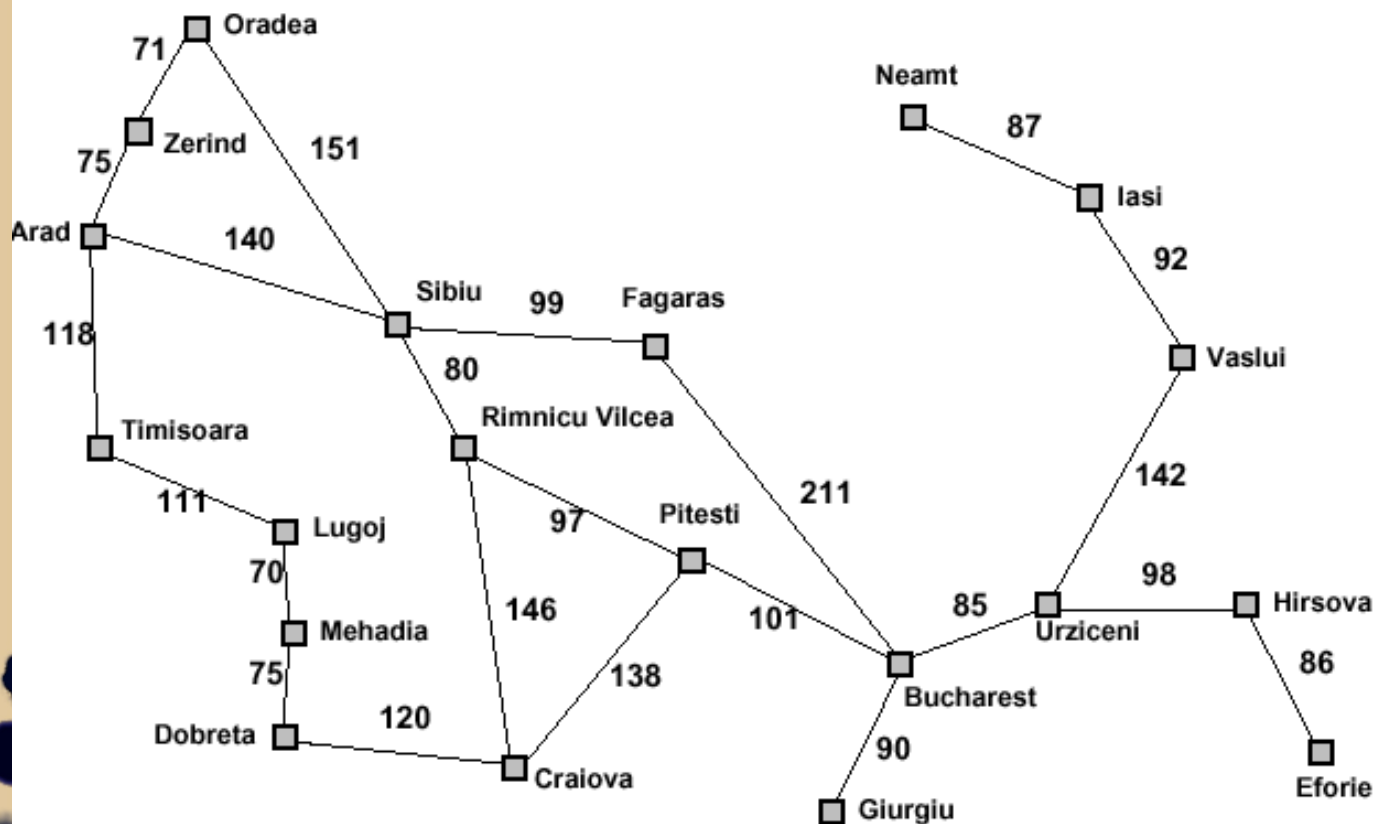


Métodos de Busca Heurística

- A informação que pode compor uma informação heurística é o Custo do Caminho.
- O CUSTO DO CAMINHO pode ser composto pelo somatório de dois outros custos:
 - O custo do caminho do estado inicial até o estado atual que está sendo expandido (função g); e
 - Uma estimativa do custo do caminho do estado atual até o estado meta (função heurística h).
- A filosofia geral que move a busca heurística é: O MELHOR PRIMEIRO. Isto é, no processo de busca deve-se primeiro expandir o nó "mais desejável" segundo uma função de avaliação.

Busca Heurística

Romania with step costs in km



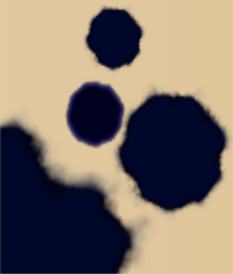
Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

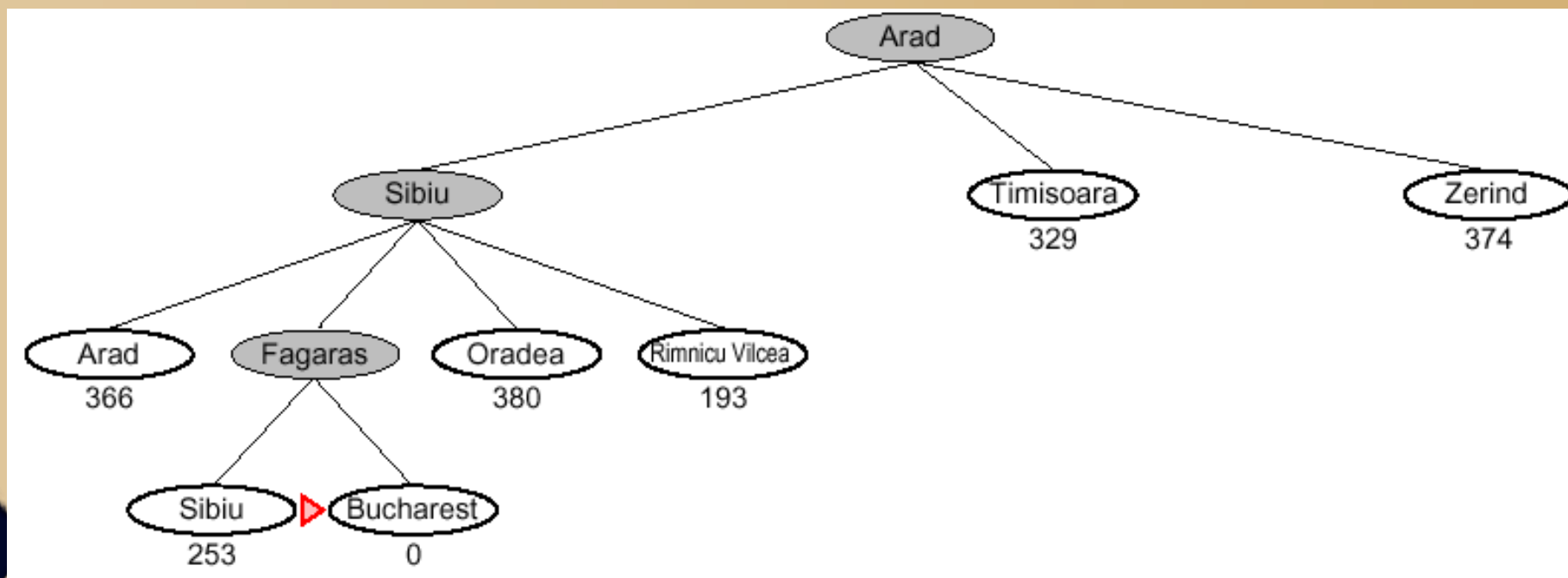
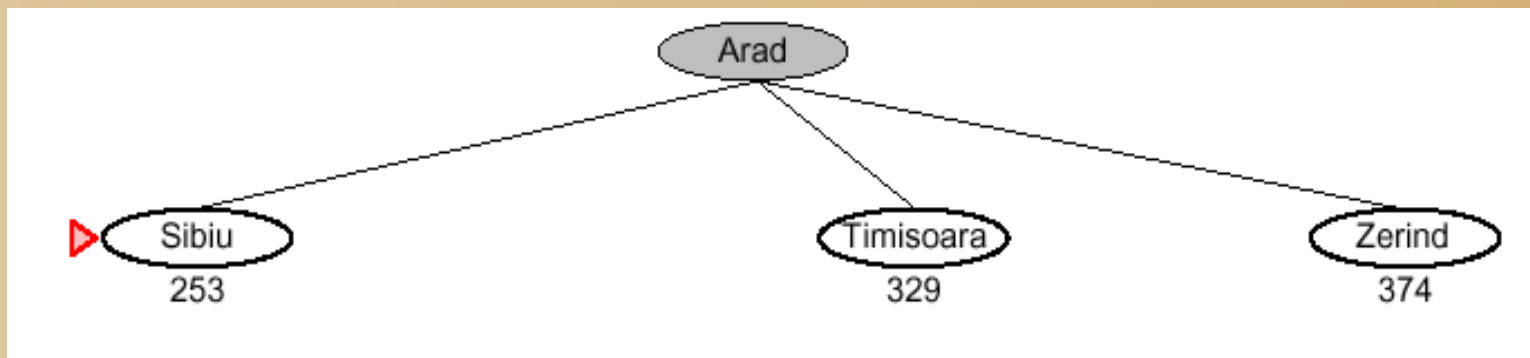
Busca Gulosa (Greedy Search)



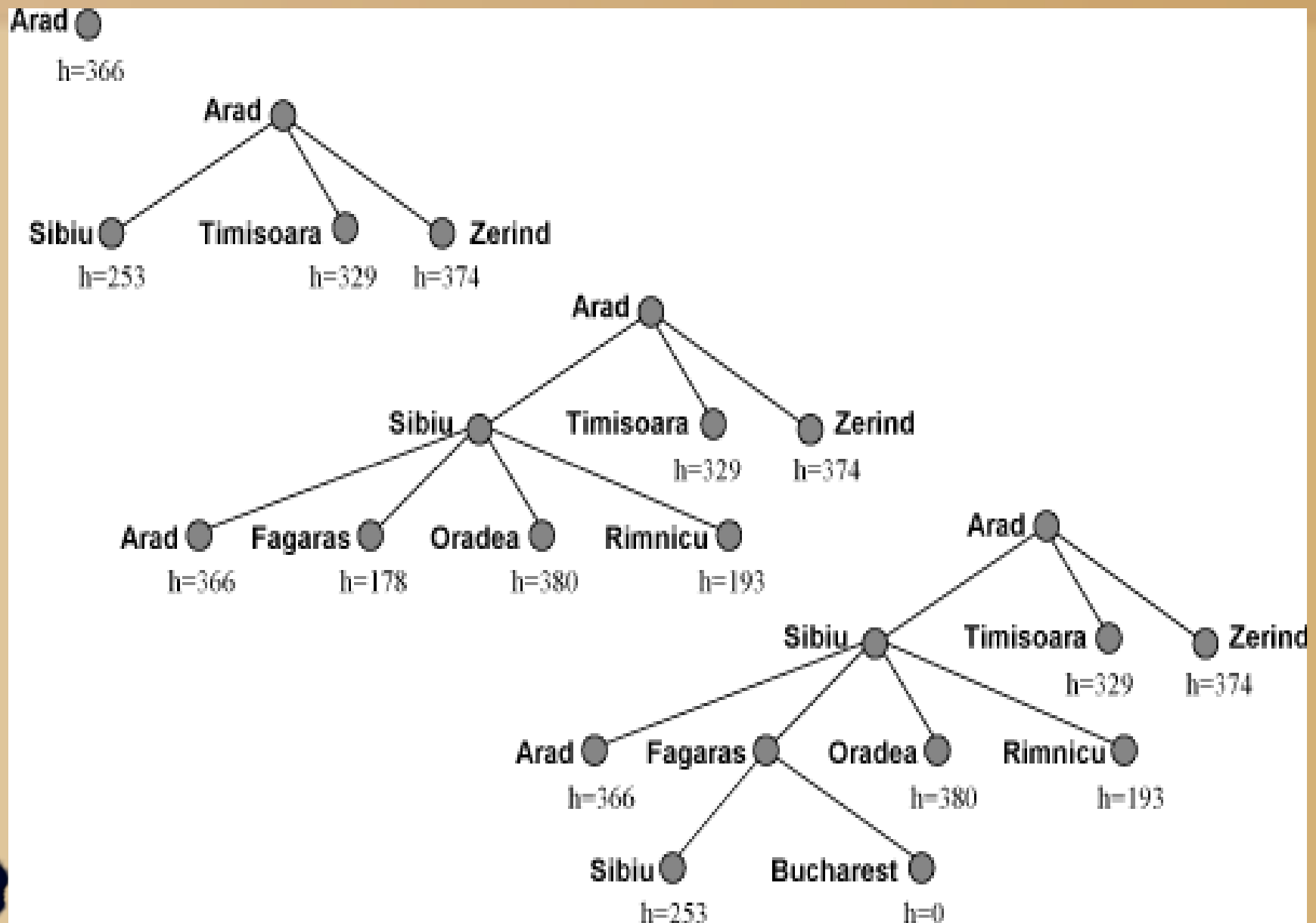
- Semelhante à busca em profundidade com backtracking.
- Tenta expandir o nó que parece mais próximo ao nó meta com base na estimativa feita pela função heurística h .
- No caso do mapa da Romênia, $h(n)$ é a distância em linha reta de n até Bucareste.



Busca Gulosa (Greedy Search)



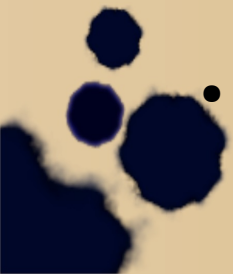
Busca Gulosa (Greedy Search)



Busca Gulosa



- Custo de busca mínimo!
 - No exemplo, não expande nós fora do caminho
- Porém *não* é ótima:
 - No exemplo escolhe o caminho que é mais econômico à primeira vista, via Fagaras
 - porém, existe um caminho mais curto via Rimnicu Vilcea
- Não é completa:
 - pode entrar em loop se não detectar a expansão de estados repetidos
 - pode tentar desenvolver um caminho infinito
- Custo de tempo e memória: $O(b^d)$



Busca A* (A estrela)

- Filosofia: procurar evitar expandir nós que já são “custosos”.
- É um método de busca que procura otimizar a solução, considerando todas as informações disponíveis até aquele instante, não apenas as da última expansão.
- Todos os estados abertos até determinado instante são candidatos à expansão.
- Combina, de certa forma, as vantagens tanto da busca em largura como em profundidade
- Busca onde o nó de menor custo “aparente” na fronteira do espaço de estados é expandido primeiro.

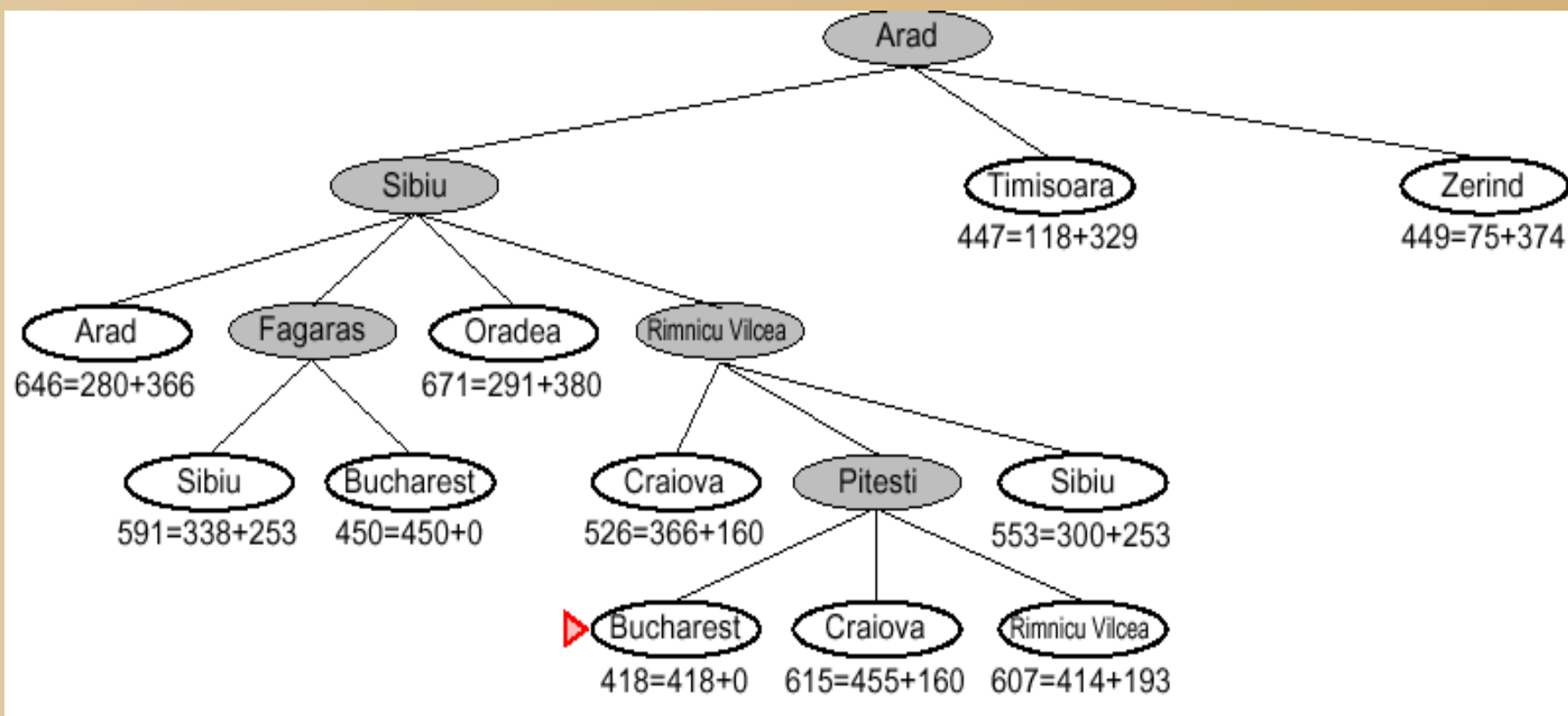
$f(n) = g(n) + h(n)$ onde

$g(n)$ = custo do caminho do nó inicial até o nó n .

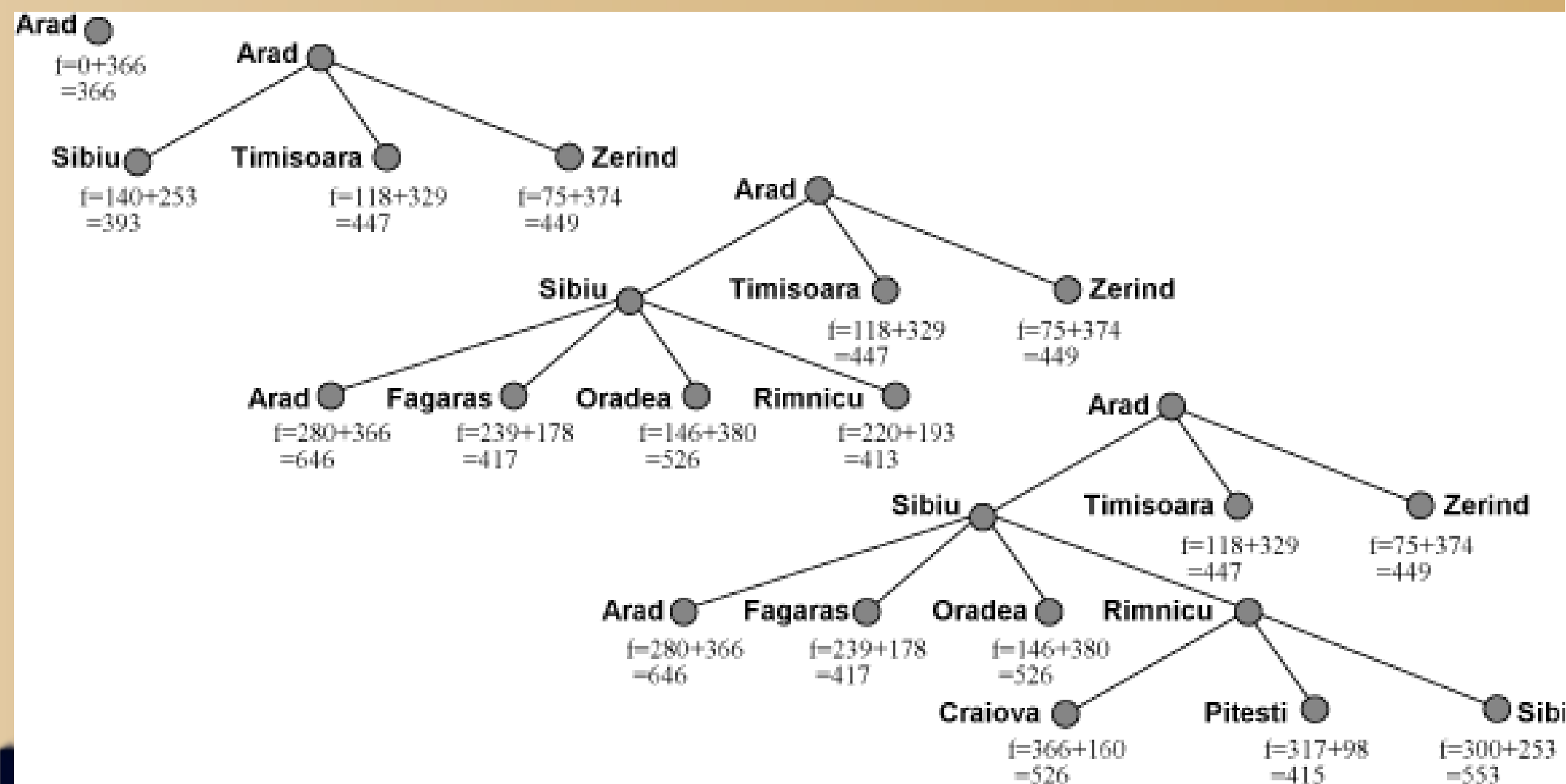
$h(n)$ = custo do caminho estimado do nó n até o nó final.

$f(n)$ = custo do caminho total estimado.

Busca A* (A estrela)

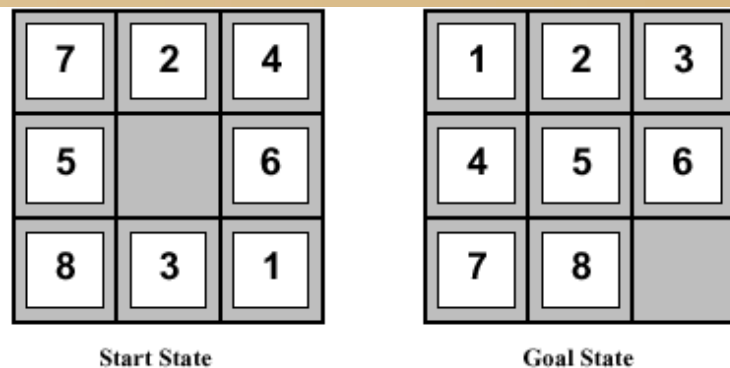


Busca A* (A estrela)



Busca A* (A estrela)

- Quanto mais admissível a heurística, menor o custo da busca.
- Exemplo: Para o jogo do oito
- $h_1(n)$: número de peças fora do lugar
- $h_2(n)$: distância Manhattan (número de casas longe da posição final em cada direção)



$$h_1(S) = ?? \quad 7$$

$$h_2(S) = ?? \quad 4+0+3+3+1+0+2+1 = 14$$

Busca com Limite de Memória

Memory Bounded Search



- IDA* (Iterative Deepening A*)
 - igual ao aprofundamento iterativo, porém seu limite é dado pela função de avaliação (f) (contornos), e não pela profundidade (d).
 - necessita de menos memória do que A* mas continua ótima
- SMA* (Simplified Memory-Bounded A*)
 - O número de nós guardados em memória é fixado previamente
 - conforme vai avançando, descarta os piores nós (embora guarde informações a respeito deles) e atualiza os melhores valores dos caminhos
 - É completa e ótima se a memória alocada for suficiente



Busca Subida da Encosta (Hill climbing)



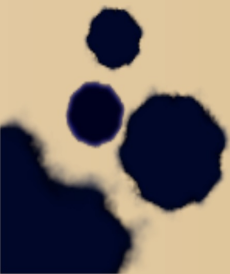
SUBIDA DA ENCOSTA

É a estratégia mais simples e popular. Baseada na Busca em Profundidade.

É um método de busca local que usa a idéia de que o objetivo deve ser atingido com o menor número de passos.

A idéia heurística que lhe dá suporte é a de que o número de passos para atingir um objetivo é inversamente proporcional ao tamanho destes passos.

Empregando uma ordenação total ou parcial do conjunto de estados, é possível dizer se um estado sucessor leva para mais perto ou para mais longe da solução. Assim o algoritmo de busca pode preferir explorar em primeiro lugar os estados que levam para mais perto da solução.



Busca Subida da Encosta (Hill climbing)



SUBIDA DA ENCOSTA

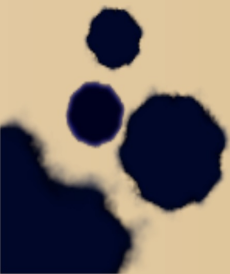
Há duas variações do método: a Subida de Encosta SIMPLES e a Subida de Encosta PELA TRILHA MAIS ÍNGREME.

SUBIDA DE ENCOSTA SIMPLES: Vai examinando os sucessores do estado atual e segue para o primeiro estado que for maior que o atual.

SUBIDA DE ENCOSTA PELA TRILHA MAIS ÍNGREME: Examina TODOS os sucessores do estado atual e escolhe entre estes sucessores qual é o que está mais próximo da solução.

Este método não assegura que se atinja o ponto mais alto da montanha.

Ele assegura somente que atingido um ponto mais alto do que seus vizinhos, então encontramos uma boa solução local.

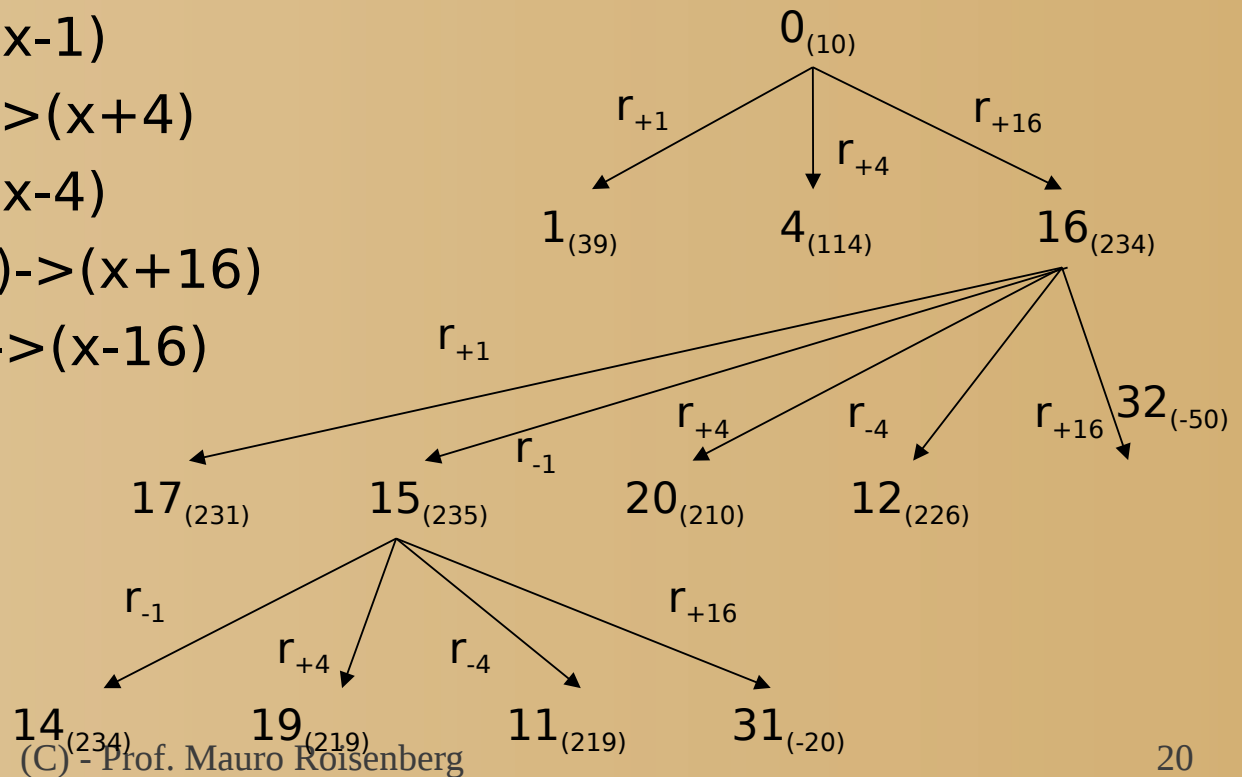


Busca Subida da Encosta (Hill climbing)

- Exemplo

- Achar o ponto máximo da função $f(x) = -x^2 + 30x + 10$ no intervalo $[0, 100]$.

- $r+1 = (x | x < 100) \rightarrow (x+1)$
 - $r-1 = (x | x > 0) \rightarrow (x-1)$
 - $r+4 = (x | x < 97) \rightarrow (x+4)$
 - $r-4 = (x | x > 3) \rightarrow (x-4)$
 - $r+16 = (x | x < 85) \rightarrow (x+16)$
 - $r-16 = (x | x > 15) \rightarrow (x-16)$



Busca por Têmpera Simulada ou Recozimento Simulado (Simulated Annealing)

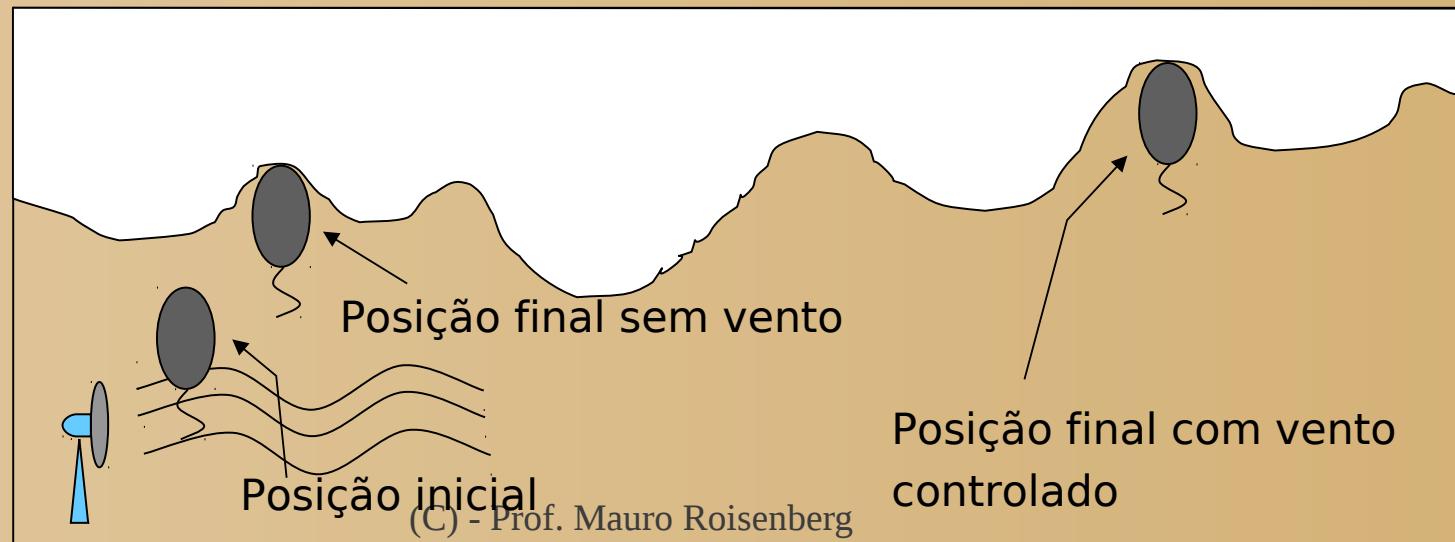


- É adequado a problemas nos quais a subida de encosta encontra muitos platôs e máximos locais.
- Não utiliza backtracking e Não garante que a solução encontrada seja a melhor possível.
- Pode ser utilizado em problemas NP-completos.
- É inspirado no processo de têmpera do aço. Temperaturas são gradativamente abaixadas, até que a estrutura molecular se torne suficientemente uniforme.



Busca por Têmpera Simulada ou Recozimento Simulado (Simulated Annealing)

- A idéia é permitir “maus movimentos” que com o tempo vão diminuindo de frequência e intensidade para poder escapar de máximos locais.
- O que o algoritmo de têmpera simulada faz é atribuir uma certa “energia” inicial ao processo de busca, permitindo que, além de subir encostas, o algoritmo seja capaz de descer encostas e percorrer platôs se a energia for suficiente.



Inventando Funções Heurísticas

- Como escolher uma boa função heurística h ?
- h depende de cada problema particular.
- h deve ser *admissível*
 - não superestimar o custo real da solução
- Existem estratégias genéricas para definir h :
 - 1) Relaxar restrições do problema;
 - 2) Usar informação estatística;
 - 3) Identificar os atributos mais relevantes do problema

(1) Relaxando o problema

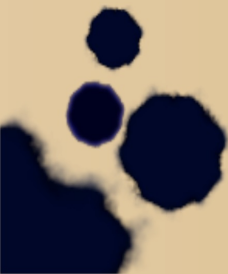


- Problema Relaxado:
 - versão simplificada do problema original, onde os operadores são menos restritivos
- Exemplo: jogo dos 8 números:
 - operador original: um número pode mover-se de A para B se A é adjacente a B e B está vazio
 - busca exaustiva $\approx 3^{20}$ estados possíveis

4	5	8
	1	6
7	2	3

Fator de ramificação ≈ 3 e $d \approx 20$ passos

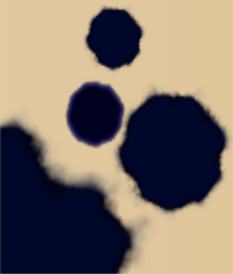
- Operadores relaxados:
 1. um número pode mover-se de A para B ($h1$)
 2. um número pode mover-se de A para B se A é adjacente a B ($h2$)



(2) Usando informação estatística



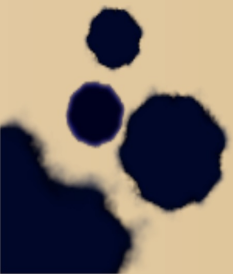
- Funções heurísticas podem ser “melhoradas” com informação estatística:
 - executar a busca com um conjunto de treinamento (e.g., 100 configurações diferentes do jogo), e computar os resultados.
 - se, em 90% dos casos, quando $h(n) = 14$, a distância real da solução é 18,
 - então, quando o algoritmo encontrar 14 para o resultado da função, vai substituir esse valor por 18.
- Informação estatística expande menos nós, porém elimina *admissibilidade*:
 - em 10% dos casos do problema acima, a função de avaliação poderá superestimar o custo da solução, não sendo de grande auxílio para o algoritmo encontrar a solução mais barata.



(3) Usando atributos/características



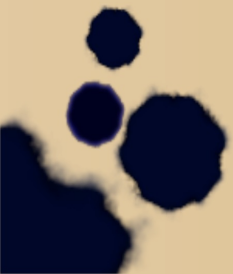
- Características do problema podem ser usadas para mensurar o quão se está próximo da solução
- ex. xadrez
 - número de peças de cada lado
 - somatório dos pesos das peças de cada lado (Peão-1, ..., Rainha-9)
 - número de peças sob ataque
- Quando não se conhece a importância das características, pode-se aprendê-las ($w_1f_1 + w_2f_2 + \dots + w_nf_n$)



Qualidade da função heurística



- Qualidade da função heurística: medida através do fator de expansão efetivo (b^*).
 - b^* é o fator de expansão de uma árvore uniforme com N nós e nível de profundidade d
 - $N = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$, onde
 - N = total de nós expandidos para uma instância de problema
 - d = profundidade da solução;
- Mede-se empiricamente a qualidade de h a partir do conjunto de valores experimentais de N e d .
 - uma boa função heurística terá o b^* muito próximo de 1.
- Se o custo de execução da função heurística for maior do que expandir nós, então ela *não* deve ser usada.
 - uma boa função heurística deve ser *eficiente*



Heurística... por toda IA



- A noção de heurística sempre foi além da busca e de uma formalização via função de um estado
- Heurística
 - escolha, prioridade, estratégia na busca de uma solução razoável onde não há solução ótima ou recurso para determiná-la
 - No dia a dia: heurística para dirigir, namorar, estudar,...
- Em IA: em todas as áreas como conhecimento de controle
 - ex. escolha de regras a serem disparadas (SBC)
 - ex. escolha de viés de generalização (aprendizagem)
 - ...

