

Busca Heurísticas e Meta-Heurística

Profa. Dra. Sarajane Marques Peres

Escola de Artes, Ciências e Humanidades – Universidade de São Paulo

<http://each.uspnet.usp.br/sarajane/>

Introdução

- ▶ Estratégias de buscas heurísticas utilizam algum conhecimento específico sobre o problema a fim de encontrar soluções de maneira eficiente.

- ▶ Organização:
 - ▶ Resolução de problemas por meio de busca
 - ▶ Estratégias de busca com informação (heurísticas)
 - ▶ Construção de funções heurísticas
 - ▶ Algoritmo de busca local
 - ▶ Algoritmos de busca distribuída
 - ▶ Meta-Heurística

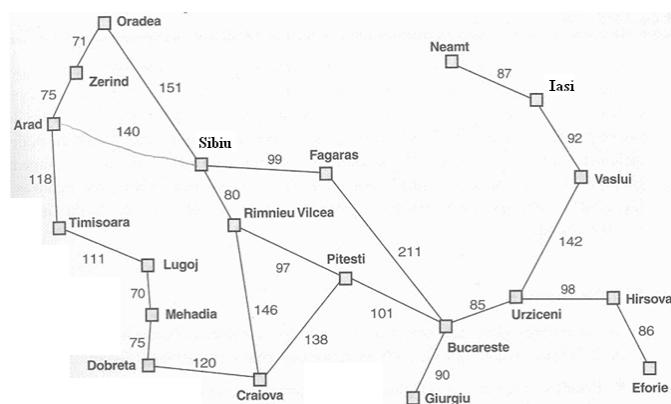


Resolução de problemas por meio de busca

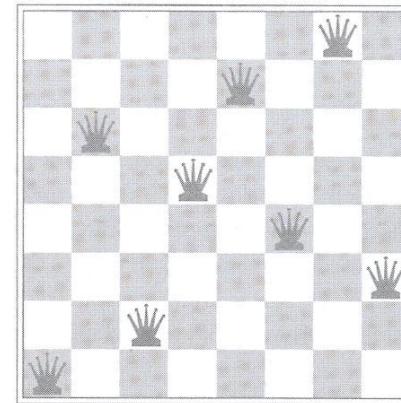
- ▶ Definição de um problema (representação):
 - ▶ Estado inicial
 - ▶ Ações possíveis
 - ▶ Estado inicial + realização de ações possíveis = espaço de estados (espaço de busca)
 - ▶ Teste de objetivo: determina se um dado estado é um estado objetivo (alcance do objetivo buscado na resolução do problema)
 - ▶ Pode ser um único objetivo (expresso de uma única forma ou de diferentes formas)
 - ▶ Ou um conjunto de vários objetivos
 - ▶ Função de custo: medida de desempenho da busca dentro do processo de resolução do problema



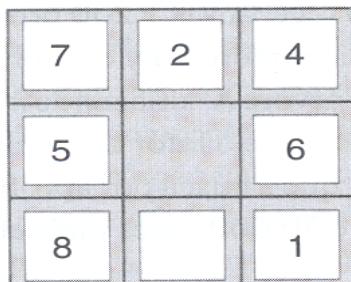
Exemplos de problemas



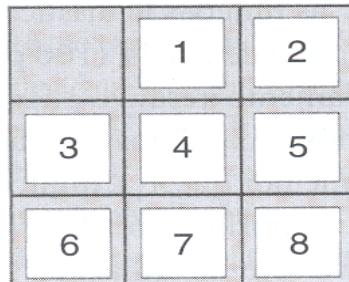
A solução é o caminho entre uma cidade e outra.
A busca retorna o caminho enquanto o encontra.



A solução é um posicionamento específico das 8 rainhas.
A busca retorna o tabuleiro “solução”.



Estado inicial



Estado objetivo

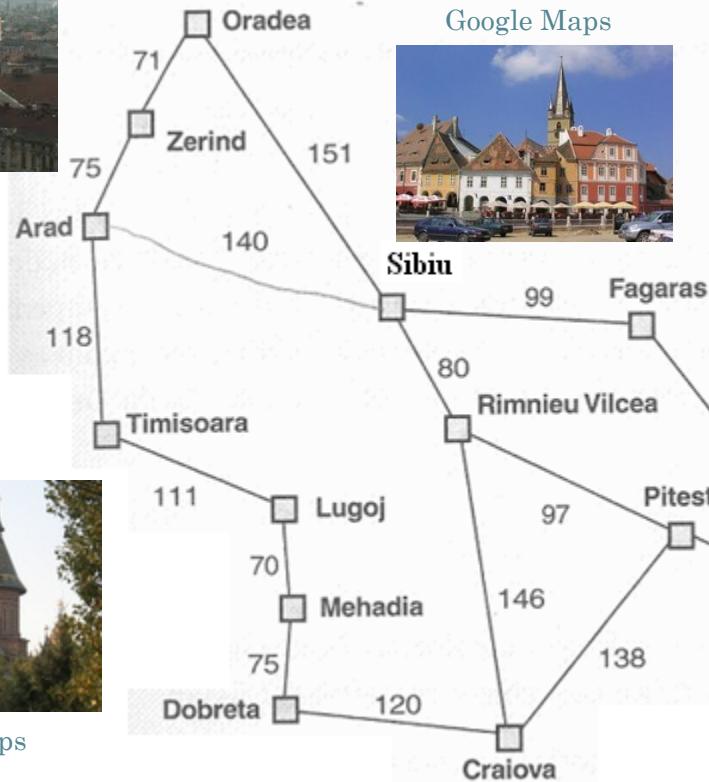
A solução é um posicionamento específico das pedras do tabuleiro.

A busca retorna o processo para encontrar a solução.

CUSTOS DE CAMINHOS NO MAPA RODOVIÁRIO DA ROMÊNIA.

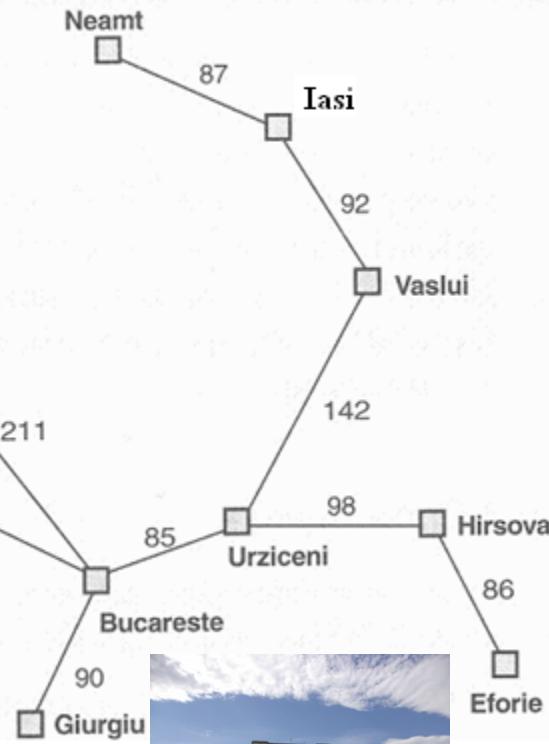


tennisforum.com



Google Maps

Achar o melhor caminho
para ir de Arad até
Bucareste!



Google Maps



Busca em árvore

função BUSCA-EM-ÁRVORE (*problema, estratégia*) **retorna** uma solução ou uma resposta sobre a “falha”

Iniciar a árvore de busca usando o estado inicial de *problema*
repita

se não existe nenhum candidato para expansão

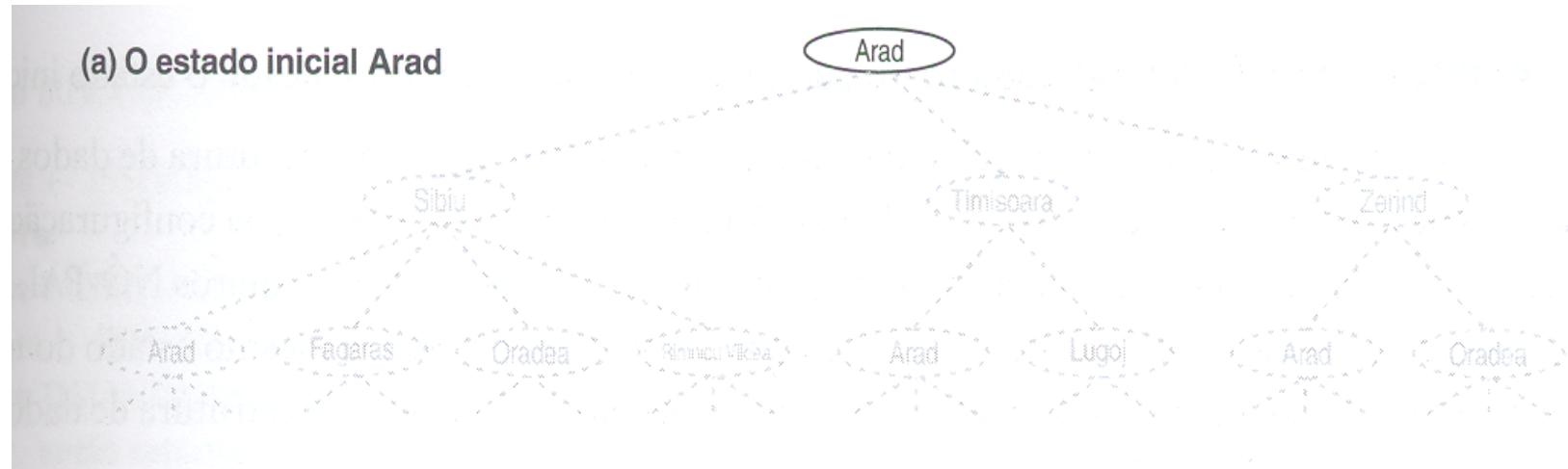
então retornar falha; // (exit)

 escolher um nó folha para expansão de acordo com *estratégia*

se o nó contém um *estado objetivo* então retornar a *solução* correspondente

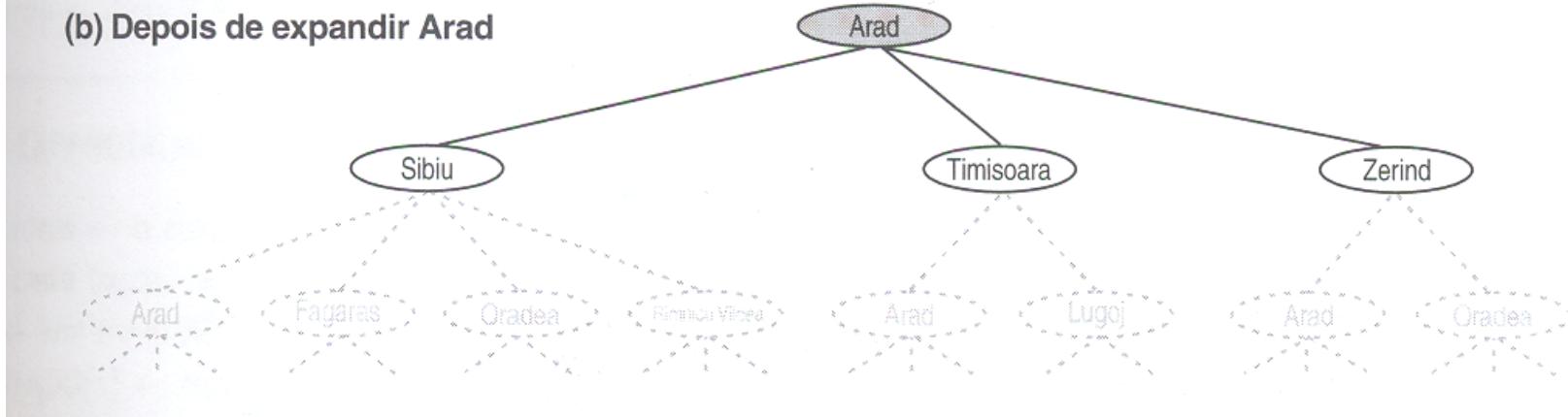
senão expandir o nó e adicionar os nós resultantes à árvore de busca

OBSERVAÇÃO: PROCESSO DE BUSCA EM ÁRVORE

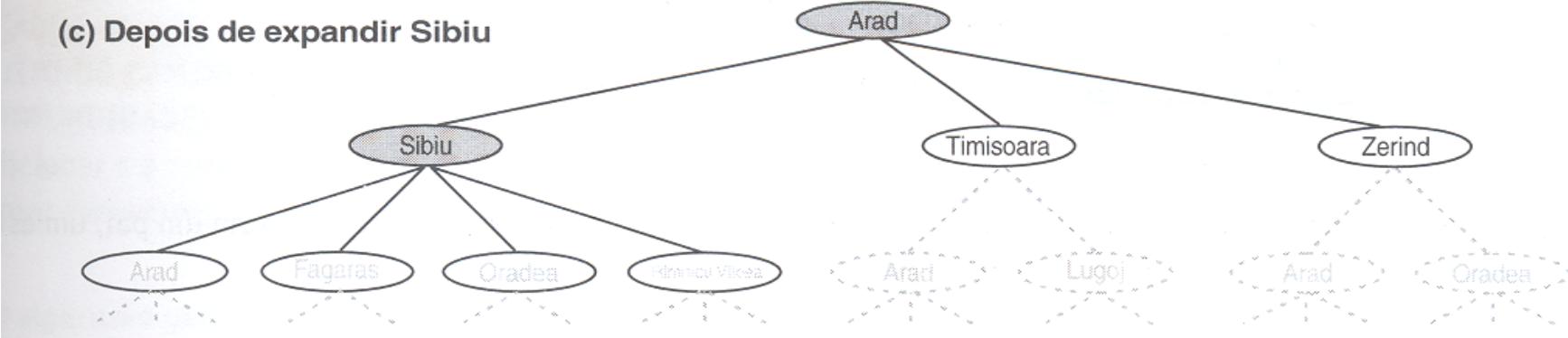


OBSERVAÇÃO: PROCESSO DE BUSCA EM ÁRVORE

(b) Depois de expandir Arad



(c) Depois de expandir Sibiu



E assim por diante



Estratégia de busca com informação (heurística)

- ▶ **Busca com informação:**
 - ▶ Uma estratégia que usa conhecimento específico sobre um problema, além da definição do próprio problema.
 - ▶ Pode encontrar soluções de forma mais eficiente que uma estratégia sem informação.

- ▶ **Busca pela melhor escolha:**
 - ▶ Trata-se de um tipo de busca em árvore ou um tipo de busca em grafo no qual um nó é selecionado para expansão (ou exploração) com base em uma **função de avaliação** $f(n)$.
 - ▶ Tradicionalmente, o nó com a avaliação **mais baixa** é selecionado para expansão porque a **avaliação** mede a **distância** dele até o objetivo (de alguma forma – usando por exemplo, algum tipo de custo).



Estratégia de busca com informação (heurística)

- ▶ Cuidado com a interpretação do nome “busca pela melhor escolha”!!
 - ▶ Se pudéssemos realmente expandir o **melhor nó** primeiro, isso não seria uma busca e sim uma **“marcha direta”** ao objetivo.
 - ▶ O que fazemos é escolher o nó que **parece ser** o melhor de acordo com a função de avaliação.
 - ▶ Se a função de avaliação for (exatamente) precisa, esse será de fato o melhor nó!
- ▶ Componente fundamental: a **FUNÇÃO HEURÍSTICA**

Exemplo 1 :A estimativa do custo do caminho mais econômico de uma cidade a outra pode ser a distância em linha reta entre estas duas cidades.

Exemplo 2 :A avaliação de uma grade de distribuição de horário é a soma de condições desejáveis que ela atende, uma vez que ela atende a todas as restrições.

- ▶ Denotada por $h(n)$:
 1. Custo estimado do caminho mais econômico do nó n até um nó objetivo, ou
 2. Avaliação do próprio nó como uma solução para o problema
- ▶ Funções arbitrárias, específicas para um problema, como uma restrição: **se n é o nó objetivo então $h(n) = 0$** .
- ▶ Seu valor depende do estado (condições) do nó que lhe é passado como entrada.



Busca Gulosa (pela melhor escolha)

- ▶ Estratégia que expande o nó mais próximo à meta, na suposição de que isso provavelmente levará a uma solução rápida e eficiente.
 - ▶ Avalia os nós usando apenas a função heurística:

$$f(n) = h(n)$$

(função de avaliação = função heurística)

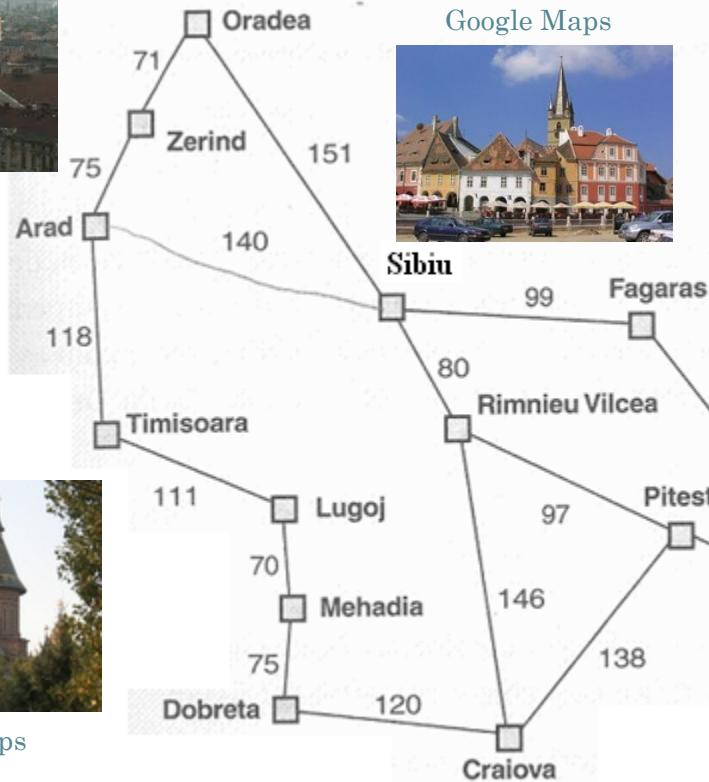
- ▶ Para o problema de planejamento de rotas na Romênia:
 - ▶ Heurística da distância em linha reta (h_{DLR})
 - ▶ Se o objetivo é chegar em Bucareste, é necessário conhecer as distâncias em linha reta até Bucareste.



CUSTOS DE CAMINHOS NO MAPA RODOVIÁRIO DA ROMÊNIA.

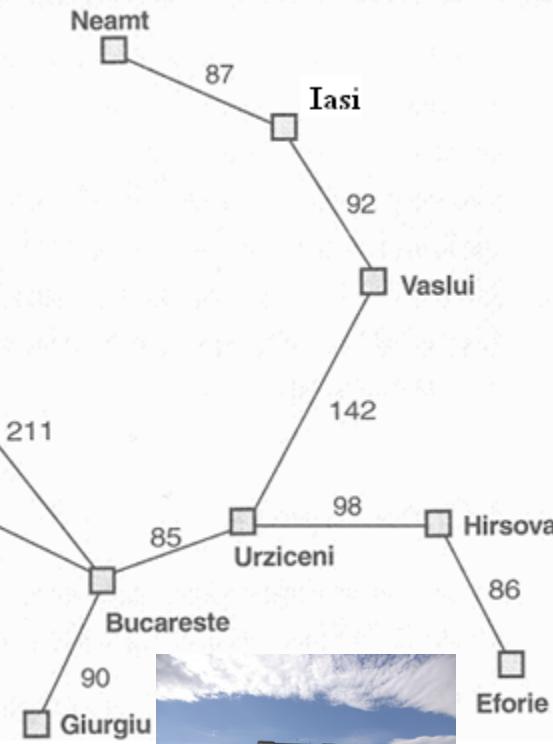


tennisforum.com



Google Maps

Achar o melhor caminho
para ir de Arad até
Bucareste!



Google Maps



VALORES DE h_{DLR}

Arad	366	Mehadia	241
Bucareste	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Os valores h_{DLR} não são dados na definição do problema e é necessário **experiência no contexto do problema** para saber que h_{DLR} está relacionada com distâncias reais e que, portanto, é uma heurística útil.

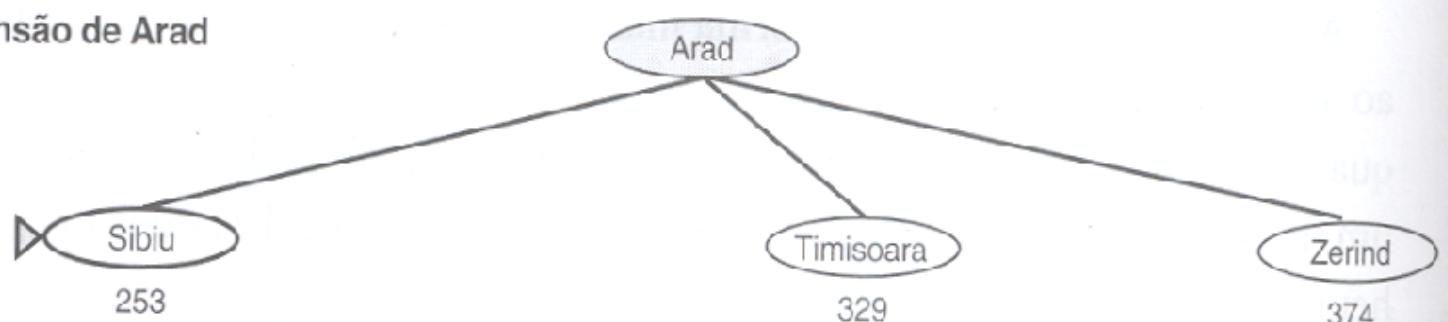


O PROCESSO EXECUTADO POR UMA BUSCA GULOSA

(a) O estado inicial

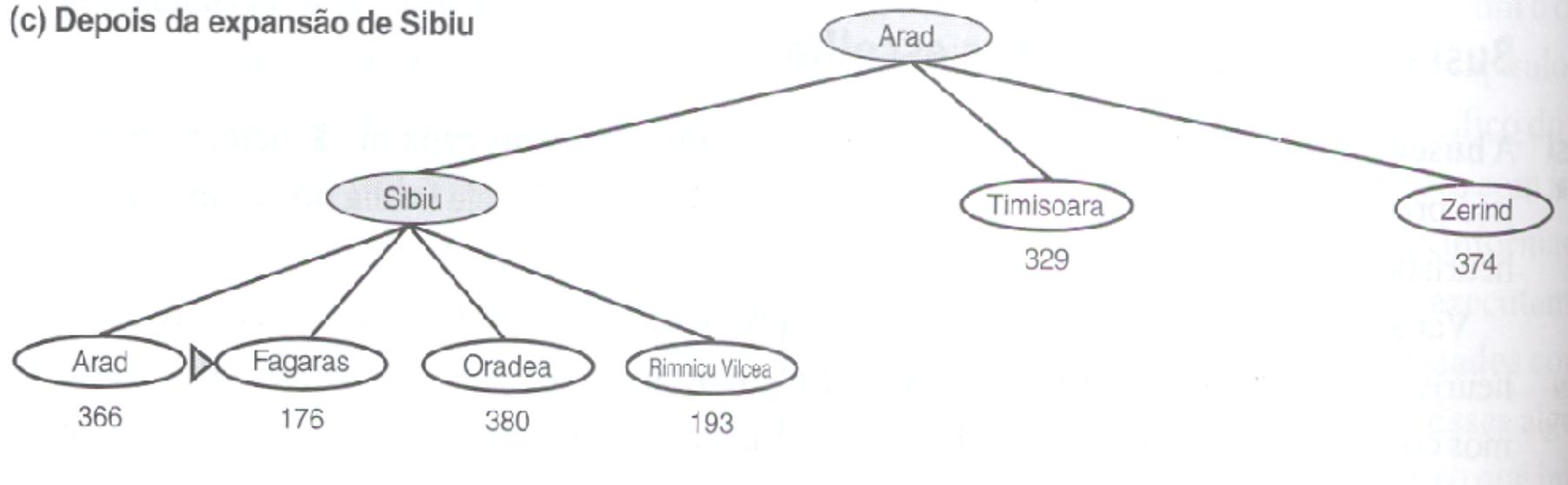


(b) Depois da expansão de Arad



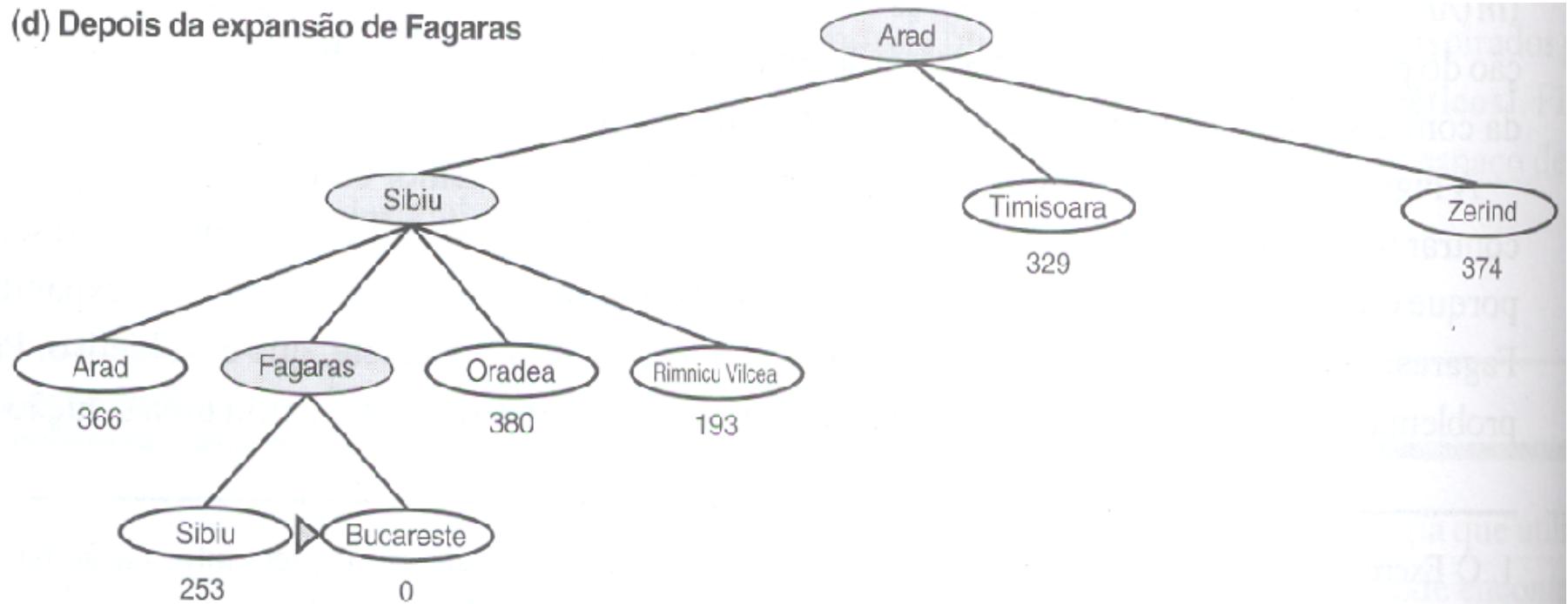
O PROCESSO EXECUTADO POR UMA BUSCA GULOSA

(c) Depois da expansão de Sibiu



O PROCESSO EXECUTADO POR UMA BUSCA GULOSA

(d) Depois da expansão de Fagaras



A heurística não é ótima: O caminho até Bucareste passando por Sibiu e Fagaras é 32 quilômetros mais longo que o caminho por Rimnicu Vilcea e Pitesti.



Busca A*

- ▶ É um tipo de busca pela melhor escolha que minimiza o **custo total estimado** da solução.
- ▶ Ela avalia os nós combinando:
 - ▶ $g(n)$ – o custo para alcançar cada nó
 - ▶ $h(n)$ - o custo para ir do nó até o objetivo

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

(função de avaliação = custo até o nó + custo estimado para ir do nó até o objetivo)

- ▶ Sabendo que $g(n)$ fornece o custo do caminho desde o nó inicial até o nó n , e que $h(n)$ é o custo estimado do caminho de custo mais baixo desde n até o objetivo, tem-se
- ▶ $f(n) = \text{custo estimado}$ da solução de custo mais baixo passando por n .
- ▶ **A heurística deve ser admissível:**
 - ▶ $h(n)$ é admissível se ela nunca superestimar o custo para alcançar o objetivo.

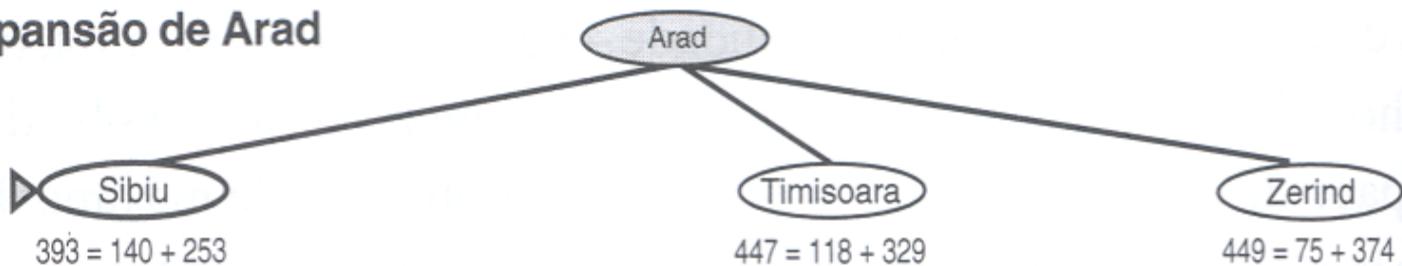


O PROCESSO EXECUTADO POR UMA BUSCA A*

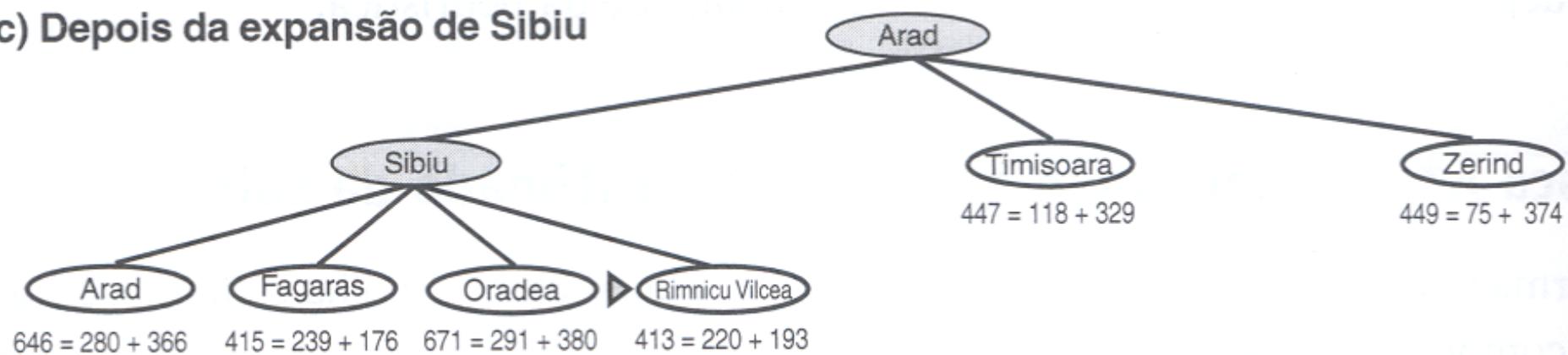
(a) O estado inicial



(b) Depois da expansão de Arad

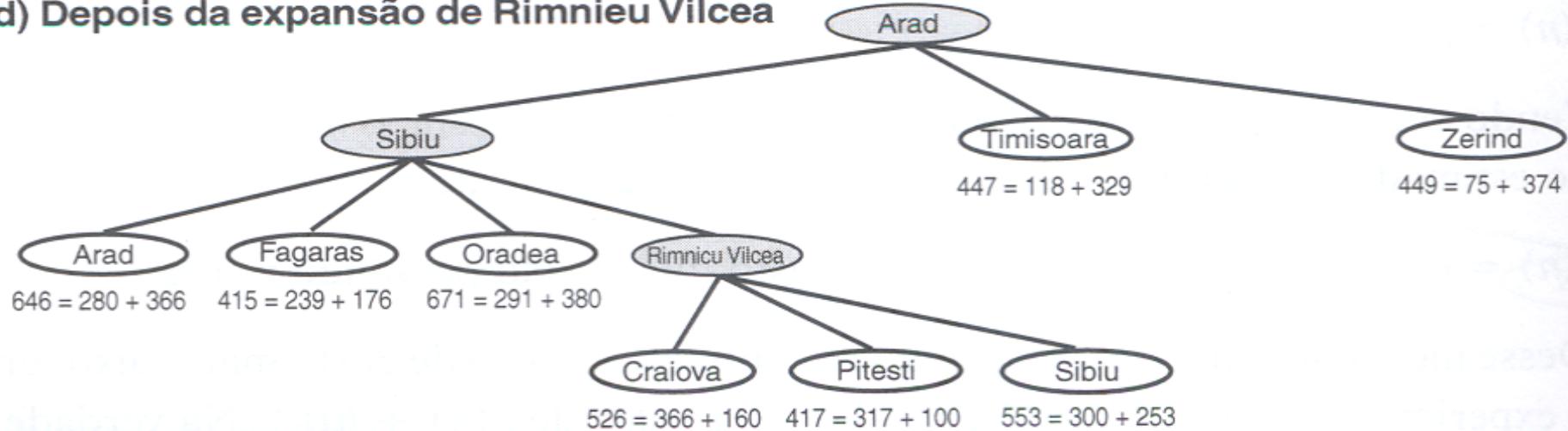


(c) Depois da expansão de Sibiu

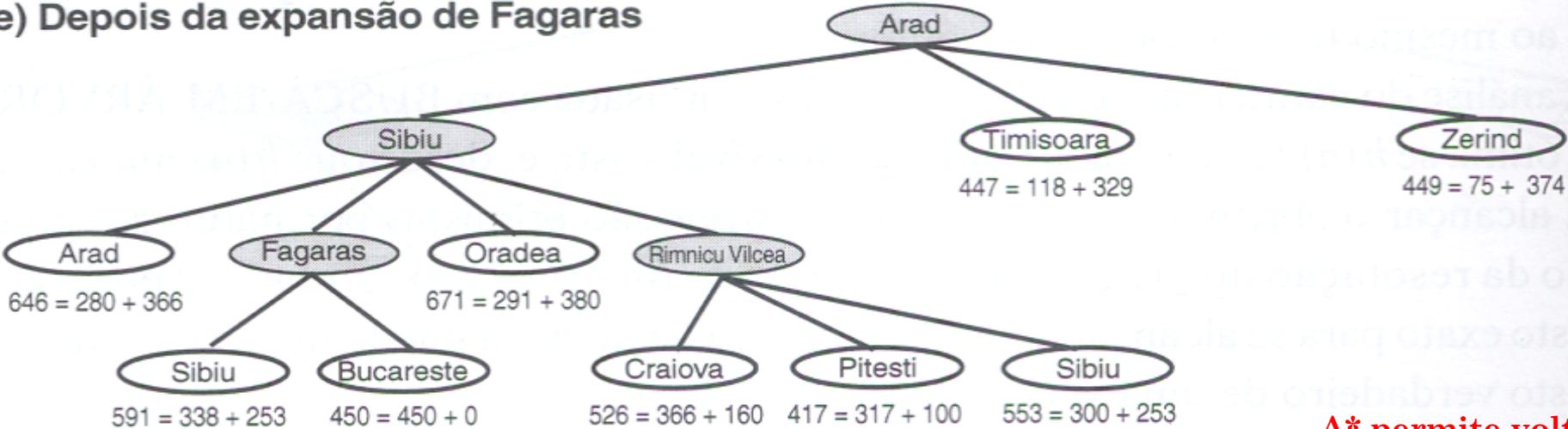


O PROCESSO EXECUTADO POR UMA BUSCA A*

(d) Depois da expansão de Rimnieu Vilcea



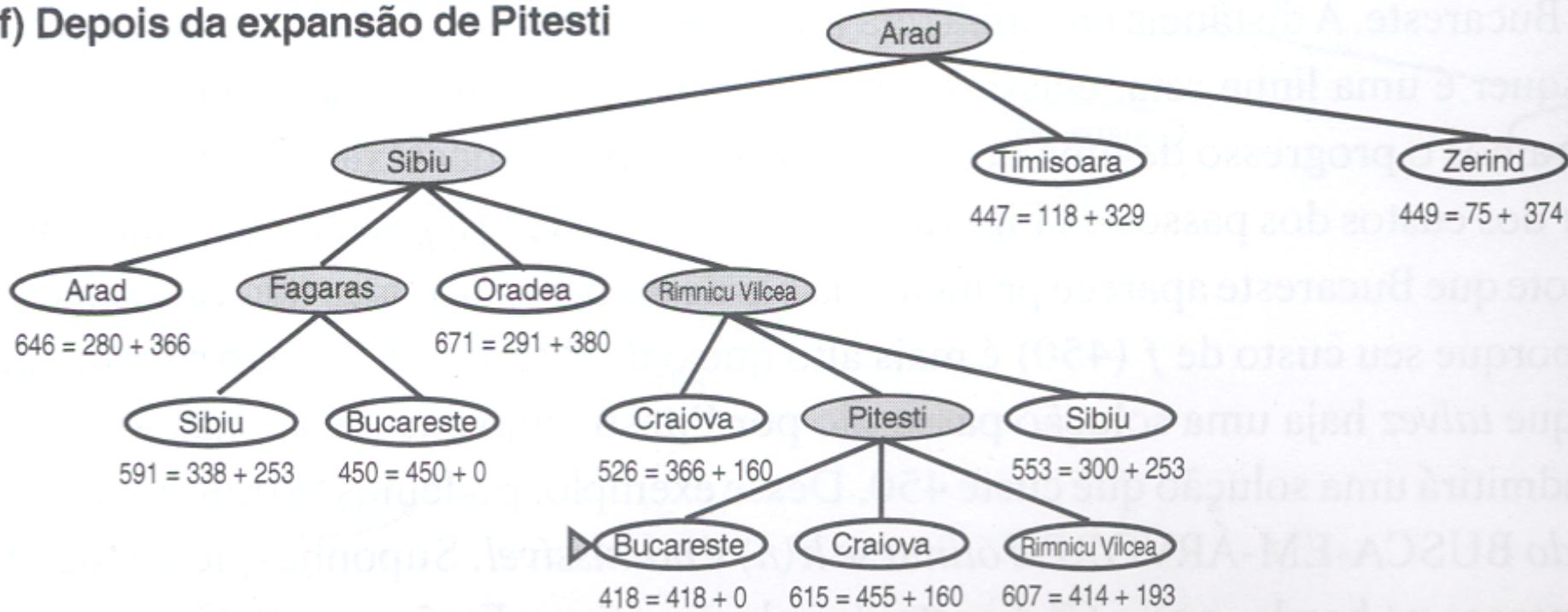
(e) Depois da expansão de Fagaras



A* permite volta

O PROCESSO EXECUTADO POR UMA BUSCA A*

(f) Depois da expansão de Pitesti

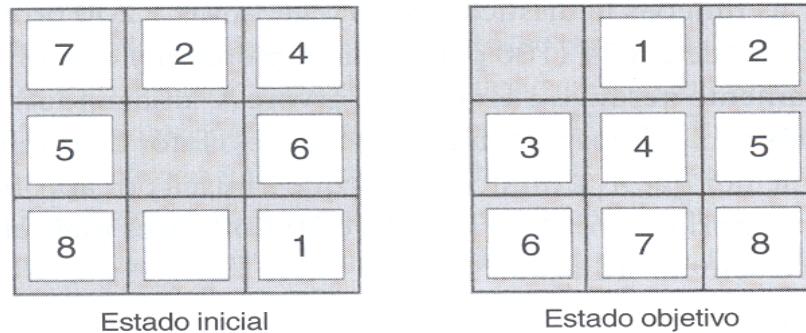


A* usando BUSCA-EM-ÁRVORE é ótima se $h(n)$ é admissível.



Funções Heurísticas

- ▶ Considere o problema do quebra-cabeças de 8 peças:



- ▶ **Objetivo:**

- ▶ deslizar os blocos no sentido horizontal ou vertical para o espaço vazio, até a configuração coincidir com a configuração objetivo.
- ▶ O fator de ramificação é aproximadamente igual a 3.
 - ▶ Quando o espaço vazio está no meio, há quatro movimentos possíveis
 - ▶ Quando o espaço vazio está em um canto, são possíveis dois movimentos;
 - ▶ Quando o espaço vazio está em uma borda, há três movimentos;

Funções Heurísticas

- ▶ Se quisermos descobrir soluções usando A*, precisaremos de uma função heurística que nunca superestime “o número de passos até o objetivo”.
- ▶ Candidatas:
 - ▶ h_1 = o número de blocos em posições erradas. Se todos os blocos estão em posições erradas então $h_1=8$. h_1 é uma heurística admissível porque é claro que qualquer bloco que esteja fora do lugar deve ser movido pelo menos uma vez.
 - ▶ h_2 = a soma das distâncias dos blocos de suas posições objetivo. Como os blocos não podem se mover em diagonal, a distância que levaremos em conta é distância de Manhattan. h_2 é admissível porque o resultado de qualquer movimento é deslocar um bloco para uma posição mais próxima do objetivo. No exemplo:
$$h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$$
- ▶ O custo (real) da solução é 26



CRIAÇÃO DE FUNÇÕES HEURÍSTICAS ADMISSÍVEIS

h_1 e h_2 são estimativas do comprimento de caminho restante para o quebra-cabeça de 8 peças, mas também são comprimentos de caminho perfeitamente precisos para versões simplificadas do quebra-cabeça.

Se as regras do quebra-cabeça fossem alteradas de forma que um bloco pudesse se deslocar para qualquer lugar, e não apenas para o quadrado vazio adjacente, então h_1 daria o número exato de passos da solução mais curta.

De modo semelhante, se um bloco pudesse se mover para um quadrado em qualquer direção, então h_2 forneceria o número exato de passos na solução mais curta.

Um problema com menos restrições sobre as ações é chamado **problema relaxado** e o custo de uma solução ótima para um problema relaxado é uma heurística admissível para o problema original.



Busca Local

- ▶ Algoritmos de busca do tipo que estudamos até agora realizam uma exploração sistemática do espaço de busca.
 - ▶ Este caráter sistemático é alcançado mantendo-se um ou mais caminhos na memória e registrando-se as alternativas que foram exploradas em cada ponto ao longo do caminho e quais delas não foram exploradas;
 - ▶ Quando um objetivo é encontrado, o caminho até esse objetivo também constitui uma solução para o problema.
- ▶ Em alguns casos, o caminho até o objetivo não é relevante.
 - ▶ Veja o caso do problema das 8 rainhas: o que importa é a configuração final.
 - ▶ Outros exemplos:
 - Layout de instalações industriais, projeto de circuitos integrados, **escalonamento de jornadas de trabalho**, roteamento de veículos, etc.



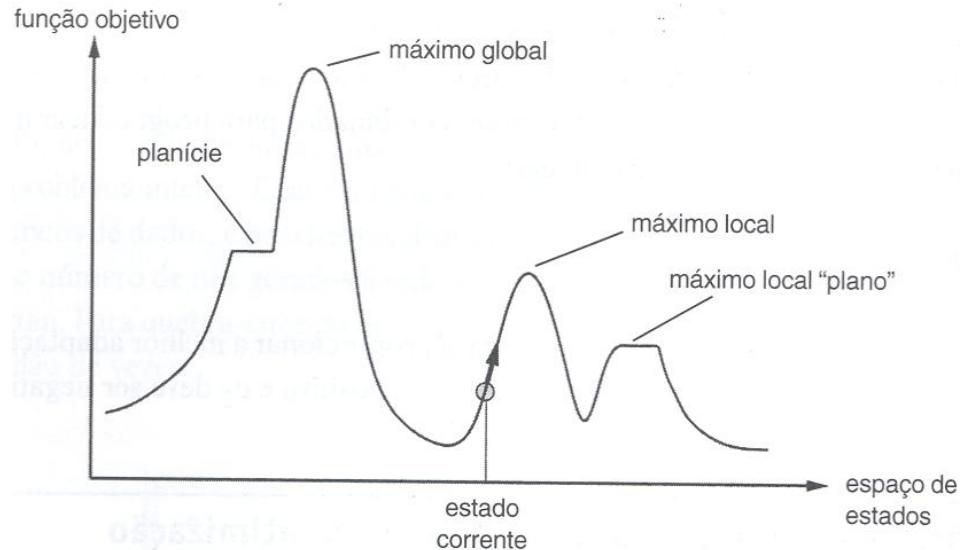
Busca local

- ▶ Os algoritmos de busca local operam usando um único estado corrente (em vez de vários caminhos) e em geral se movem apenas para os **vizinhos** desse estado.
- ▶ Vantagens;
 - ▶ Usam pouquíssima memória – quase sempre um valor constante;
 - ▶ Freqüentemente podem encontrar soluções razoáveis em espaços de estados grandes ou infinitos.
- ▶ São úteis para resolver problemas de otimização puros, nos quais o objetivo é encontrar o melhor estado de acordo com uma função objetivo.



Conceitos Relacionados

► Topologia do espaço de estados:



- Mínimo local e global
- Máximo local e global

- Um algoritmo de busca local completo sempre encontra um objetivo, caso ele exista.
- Um algoritmo de busca local ótimo sempre encontra um mínimo/máximo global.
- Efeitos de uma busca de **subida de encosta** e uma busca de **têmpera simulada** (ou simulated annealing)

Busca de **subida de encosta** (busca gulosa local)

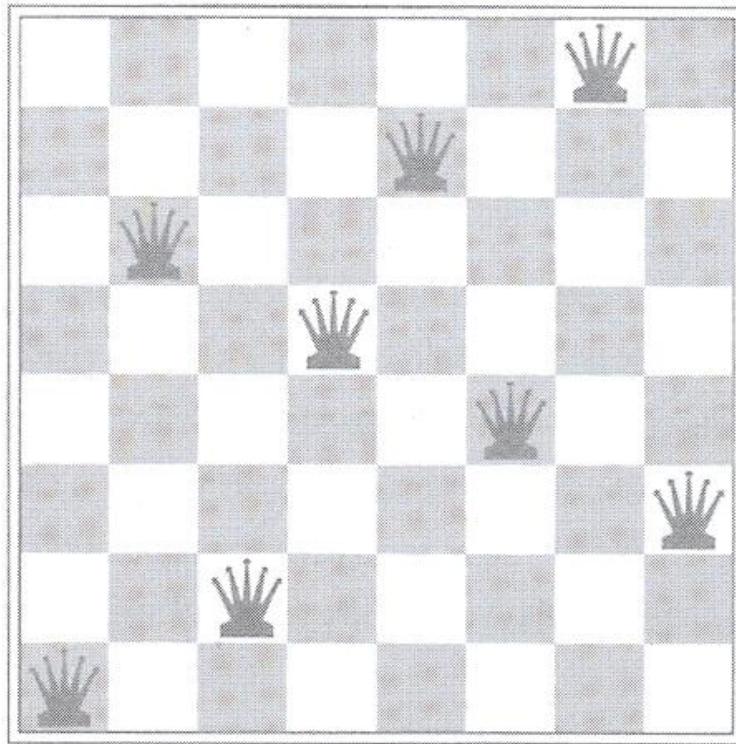
- ▶ Um laço repetitivo que se move de forma contínua no sentido do valor crescente, isto é, encosta acima.

- **função** SUBIDA-DE-ENCOSTA (problema) **retorna** um estado que é um máximo local
- **entradas:** problema //um problema
- **variáveis locais:** *corrente*, vizinho // nós
- *corrente* \leftarrow CRIAR-NÓ (ESTADO-INICIAL[problema]);
- **repita**
- *vizinho* \leftarrow um sucessor de *corrente*;
- **se** VALOR[vizinho] \leq VALOR[*corrente*]
- **então retornar** ESTADO[*corrente*]; // possibilidade de término
- **senão** *corrente* \leftarrow *vizinho*;



Contextualizando – 8 rainhas

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	18	13	16	13	16
18	14	17	15	15	14	16	16
17	18	16	18	15	15	15	18
18	14	18	15	15	14	18	16
14	14	13	17	12	14	12	18



- Melhor dos melhores;
- Primeiro melhor

Busca de subida de encosta

- ▶ Razões que paralisar a busca:
 - ▶ Máximos locais: é um pico mais alto que cada um de seus estados vizinhos, embora seja mais baixo que o máximo global. O estado da última figura do slide anterior é um mínimo local
→ todo movimento de uma única rainha piora a situação.
 - ▶ Picos: **resultam em uma seqüência de máximo locais que torna muito difícil a navegação.**
 - ▶ Platôs: área da topologia de espaço de estados em que a função de avaliação é plana. Pode ser um máximo local plano ou uma planície.



Têmpera simulada (Simulated Annealing)

- ▶ Um algoritmo de subida de encosta que nunca faz movimentos “encosta abaixo” em direção a estados com valor mais baixo (ou de custo mais alto) sem dúvida é incompleto, porque pode ficar paralisado em um máximo (ou mínimo) local.
- ▶ O algoritmo de Têmpera Simulada combina a subida de encosta com um percurso aleatório, resultando, de algum modo, em eficiência e completeza.
- ▶ Introduz-se uma modificação no algoritmo de subida de encosta:
 - ▶ em vez de escolher o melhor movimento, escolhe-se um movimento aleatório;
 - ▶ Se o movimento melhorar a situação, ele será aceito;
 - ▶ Caso contrário, o algoritmo aceitará o movimento com alguma probabilidade menor que 1;
 - ▶ A probabilidade diminui exponencialmente de acordo com a má qualidade do movimento;
 - ▶ A probabilidade também diminui à medida que a temperatura T se reduz (movimentos ruins tem maior probabilidade de serem aceitos no início);

Têmpera simulada

- **função TEMPERA-SIMULADA** (problema, escalonamento) **retorna** um estado solução
- **entradas:** problema // um problema
- escalonamento // um mapeamento de tempo para “temperatura”
- **variáveis locais:** corrente, próximo // dois nós
- T // uma “temperatura” que controla a probabilidade de passos descendentes
- corrente \leftarrow CRIAR-NÓ (ESTADO-INICIAL[problema]);
- **Para** t \leftarrow 1 **até** ∞ **faça**
- T \leftarrow escalonamento[t];
 se T = 0 **então** **retorna** corrente; // término
 próximo \leftarrow um sucessor de corrente selecionado ao acaso;
 $\Delta E \leftarrow$ VALOR[próximo] – VALOR[corrente];
 se $\Delta E > 0$ **então** corrente \leftarrow proximo;
 senão corrente \leftarrow próximo somente com probabilidade $e^{\Delta E/T}$;

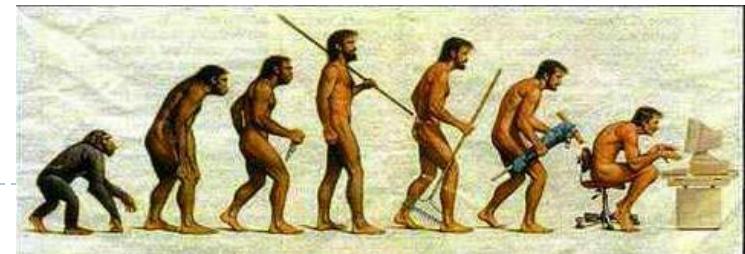
Algoritmos Genéticos (Goldberg) (Linden)

- ▶ São algoritmos de busca baseados no mecanismo de seleção natural e na genética natural.
 - ▶ Desenvolvidos por John Holland e sua equipe na Universidade de Michigan.
 - ▶ A pesquisa em AGs possui duas metas:
 - ▶ Abstrair e explicar o processo adaptativo dos sistemas naturais;
 - ▶ Projetar sistemas de software artificiais que emulem os mecanismos dos sistemas naturais.
- ▶ É um sistema robusto:
 - ▶ Faz o balanço entre a eficiência e a eficácia necessária para sobrevivência em diferentes ambientes.
- ▶ Faz parte de uma área de estudo que usa modelos computacionais dos processos naturais de evolução como uma ferramenta para resolver problemas – Algoritmos Evolucionários ou Computação Evolutiva
 - ▶ Outras técnicas: Estratégias Evolucionários e Programação Genética



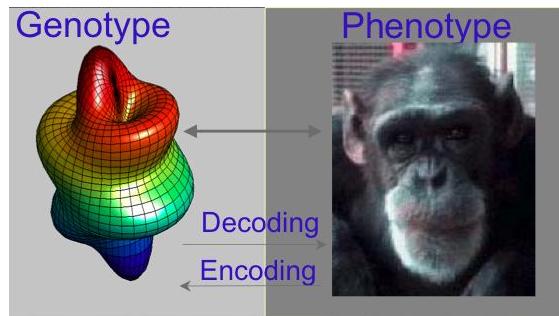
Algoritmos Genéticos (Linden)

- ▶ Técnica de busca baseada em uma metáfora do processo biológico de evolução natural
- ▶ São técnicas heurísticas de otimização global (pode ser considerada uma meta-heurística).
- ▶ Utilizam:
 - ▶ População de estruturas, denominadas indivíduos ou cromossomos, sobre as quais são aplicados operadores genéticos. Cada indivíduo recebe uma avaliação que é uma quantificação de sua qualidade como solução do problemas em questão.
 - ▶ Operadores genéticos: aproximações computacionais de fenômenos vistos na natureza, como reprodução sexuada, mutação genética etc...



Terminologia

- ▶ Cromossomo X indivíduo X string de bits
- ▶ Um cromossomo é formado por genes, que podem ter um determinado valor entre vários possíveis (*alelos*). A posição do gene é o *locus*.
- ▶ Genótipo X Fenótipo
 - ▶ Genótipo: é a estrutura do cromossomo, e pode ser identificada na área de GA como o termo *estrutura*.
 - ▶ Fenótipo: é a interação do conteúdo genético com o ambiente.



www.lania.mx/~asanchez/IA/GAapuntes/gen50.html

- ▶ Função fitness: mede a adaptabilidade de uma solução a um problema (é a medida de avaliação).

Algoritmos Genéticos

▶ Representação de conhecimento

▶ Representando o problema

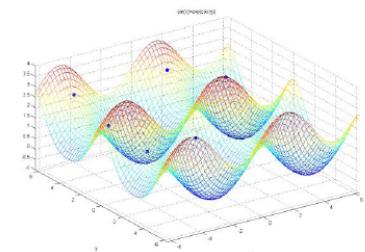
▶ O cromossomo

- A representação deve ser o mais simples possível;
- Se houver soluções infactíveis, é interessante que elas não sejam representadas
- Restrições do problema devem estar preferencialmente implícitas na representação

▶ Representando a avaliação no problema

▶ A função fitness (ou objetivo) – de avaliação

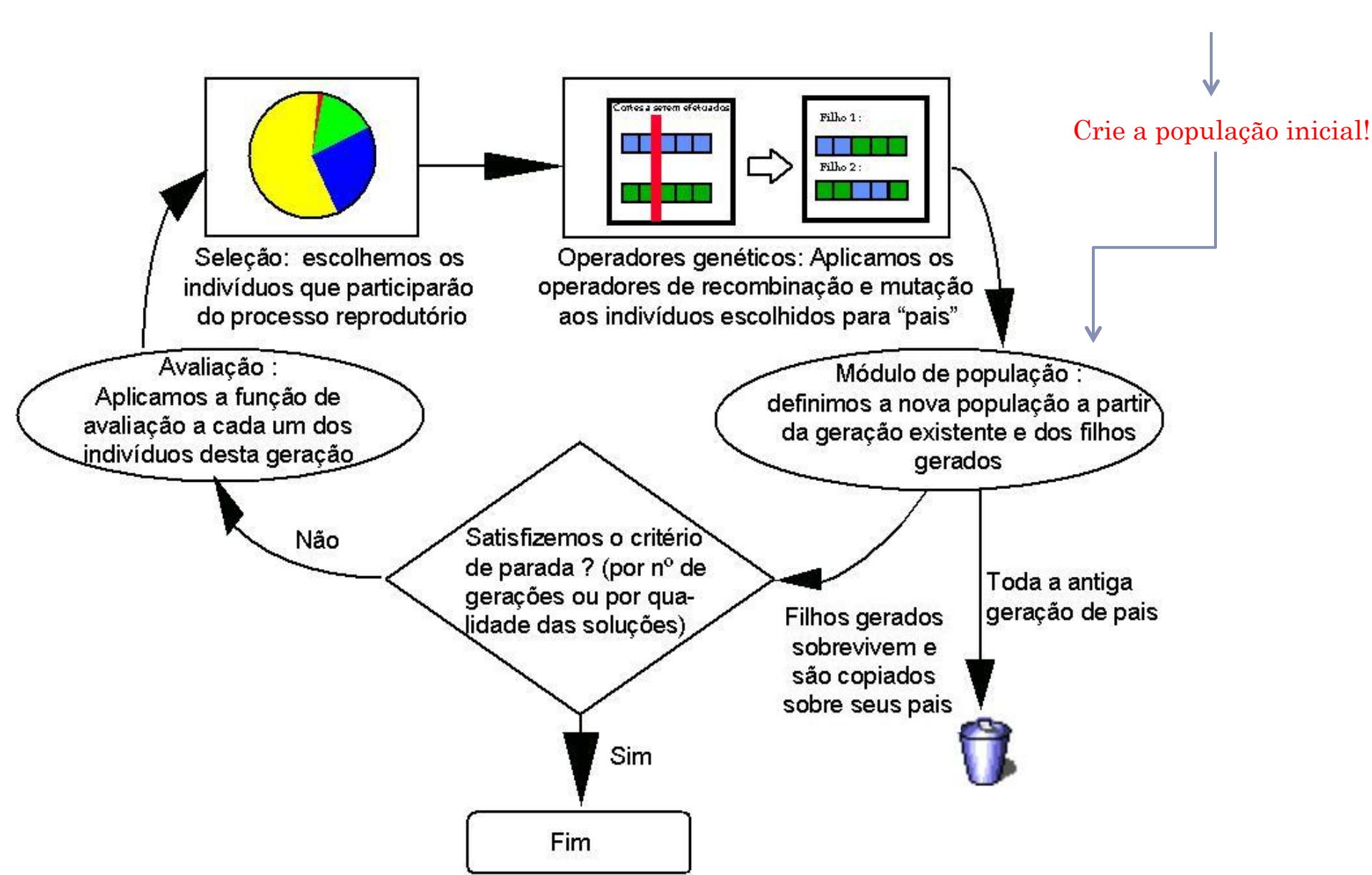
- Caso soluções infactíveis sejam permitidas na representação, ou restrições não estejam representadas, tente usar a função fitness para “matar” tais indivíduos.



Algoritmo Genético (Linden)

- ▶ Convergência genética:
 - ▶ Se traduz em uma população com baixa diversidade genética que, por possuir genes similares, não consegue evoluir, a não ser pela ocorrência de mutações aleatórias que sejam positivas.
- ▶ Perda de diversidade:
 - ▶ Pode ser definida como sendo o número de indivíduos que nunca são escolhidos pelo método de seleção de pais.





Um Algoritmo Genético simples

- ▶ Simplicidade de operação e poder de efeito são das duas principais atrações da abordagem de algoritmos genéticos.
- ▶ Depois que a codificação do problema está pronta, ou seja, a representação da solução está construída, a população inicial pode ser gerada.
- ▶ Retornemos ao nosso problema da caixa preta
 - ▶ A população é composta por 4 indivíduos, ou seja, por 4 possíveis configurações de interruptores (lembrem-se que queremos encontrar a configuração que optimiza uma função f)

0 1 1 0 1
1 1 0 0 0
0 1 0 0 0
1 0 0 1 1

População inicial instanciada aleatoriamente.



Um algoritmo genético simples

- ▶ Cada indivíduo da população representa uma solução para o problema, e cada um deles tem uma avaliação sobre a sua adequabilidade ao problema.
- ▶ A avaliação de cada indivíduo é o valor da função f quando aplicada sobre ele.
- ▶ A função f pode ser chamada de função objetivo ou função *fitness* – representa o quanto adaptado o indivíduo está ao seu ambiente (ao problema). Quanto mais a solução que o indivíduo representa estiver próxima da ótima, mais adaptada está esta solução ao problema.
- ▶ Suponha que a função *fitness* retorne a adequabilidade de cada indivíduo de acordo com as informações da tabela.

No.	String	Fitness	% do Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0



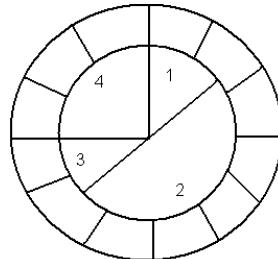
Um algoritmo genético simples

- ▶ Um algoritmo genético simples, que rende bons resultados na prática, é composto por três operadores: Seleção (ou reprodução), Crossover e Mutação
- ▶ A reprodução é um processo no qual indivíduos são copiados de acordo com seus valores para a função *fitness*. Também chamada de elitismo ou reprodução assexuada.
 - ▶ Copiar indivíduos de acordo com sua função *fitness* significa que indivíduos com um valor mais alto tem maior probabilidade de contribuir com descendentes nesta ou na próxima geração.
 - ▶ Este operador, naturalmente, é uma versão artificial da seleção natural, a sobrevivência Darwiniana dos indivíduos mais adaptados ao meio.
 - ▶ Na natureza, o *fitness* é determinado pela habilidade de um indivíduo a sobreviver contra predadores, pestes e outros obstáculos.
 - ▶ A função *fitness* é, na realidade, o árbitro que decide quem vive e quem morre.



Um algoritmo genético simples

- ▶ Uma implementação simples para o operador de (seleção) reprodução.
 - ▶ A roleta:
 - ▶ Cada indivíduo na população tem um “slot” na roleta de tamanho proporcional ao seu *fitness*.
- ▶ Para processar o operador de reprodução basta “rodar a roleta” e verificar qual indivíduo é selecionado.
- ▶ Cada vez que uma descendência é requerida, a roleta é acessada para indicar o candidato.
- ▶ Quando um indivíduo é escolhido para reproduzir, faz-se uma réplica exata dele.
- ▶ Esse novo indivíduo comporá o conjunto de indivíduos que são candidatos a compor uma nova população (geração) para o algoritmo.



Um algoritmo genético simples

- ▶ Depois da seleção, o operador de crossover pode ser executado.
- ▶ O crossover pode ser de vários tipos, pode ocorrer sob uma determinada probabilidade de ocorrência e é também conhecido como reprodução sexuada.
- ▶ Procede-se em dois passos:
 - ▶ Membros dos da nova população (indivíduos reproduzidos) são escolhidos randomicamente;
 - ▶ Cada par de indivíduos sofre recombinação:
- ▶ Seleciona-se, randomicamente, uma posição K da string (k pode variar de 1 ao tamanho da string - 1);
- ▶ Dois novos indivíduos *filhos* são criados por meio da combinação dos genes dos indivíduos *pais*.



Um algoritmo genético simples

- ▶ Ilustrando:

- ▶ Pai1 = 0 | | 0 |
- ▶ Pai2 = | | 0 0 0

- ▶ Posição sorteada: $k = 4$

- ▶ Pai1 = 0 | | 0 | |
- ▶ Pai2 = | | 0 0 | 0
- ▶ Filho1 = 0 | | 0 0
- ▶ Filho2 = | | 0 0 |



Um algoritmo genético simples

▶ Mutação:

- ▶ É um operador ocasional que provoca alterações randômicas dos valores de algumas posições na string (em alguns genes dos indivíduos).
- ▶ Na codificação binária do problema da caixa preta, a mutação simplesmente muda de 1 para 0 o valor de um bit da string.
- ▶ A freqüência da mutação deve ser baixa pois ela representa uma busca aleatória no espaço de busca.
- ▶ A implementação dela se dá como segue:
 - ▶ Se a mutação deve acontecer (de acordo com uma probabilidade de mutação)
 - ▶ Então um indivíduo é randomicamente escolhido
 - ▶ E um alelo deste indivíduo é randomicamente escolhido e
 - ▶ O valor para este alelo é randomicamente escolhido dentro do alfabeto correlato.



Um algoritmo genético simples

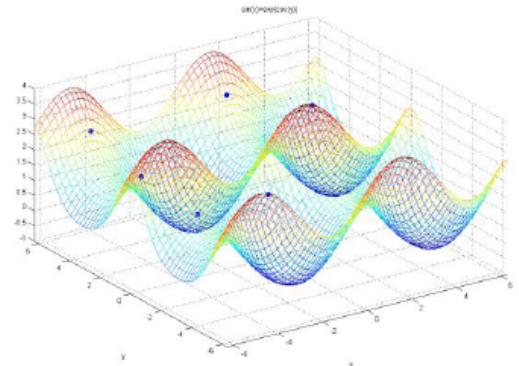
String no.	População inicial Gerada randomicamente	Valor de x Inteiro associado	f(x) X^2	%	Seleções
1	0 1 1 0 1	13	169	0,14	1
2	1 1 0 0 0	24	576	0,49	2
3	0 1 0 0 0	8	64	0,06	0
4	1 0 0 1 1	19	361	0,31	1
			1170	1,00	

Pool de reprodução	Seleção de pares	Ponto de crossover	Nova população	Valor de x	f(x)
0 1 1 0 1	2	4	0 1 1 0 0	12	144
1 1 0 0 0	1	4	1 1 0 0 1	25	625
1 1 0 0 0	4	2	1 1 0 1 1	27	729
1 0 0 1 1	3	2	1 0 0 0 0	16	256



Algoritmos Genéticos (Goldberg)

- ▶ Diferenciando dos métodos tradicionais
 - ▶ AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
 - ▶ AGs realizam busca a partir de uma população de pontos (soluções) e não a partir de um único ponto;
 - ▶ AGs usam informação de retorno – *feedback* da função objetivo - e não informação derivativa ou outro tipo de conhecimento auxiliar;
 - ▶ AGs usa regras de transição probabilísticas e não regras determinísticas.



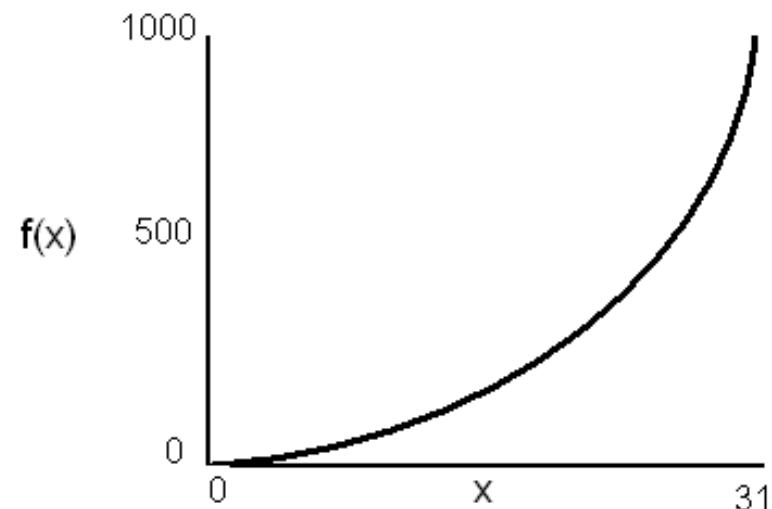
Explorando ... (Goldberg)

- ▶ AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros.
 - ▶ O conjunto de parâmetros natural do problema deve ser codificado como uma string de comprimento finito, instanciada por meio de um alfabeto finito.

▶ Exemplo:

- ▶ Maximização de $f(x) = x^2$, no intervalo $[0,31]$.
- ▶ Representação binária para o parâmetro x .

0	1	1	1	1
$x = 15$				



Explorando (Goldberg)

- ▶ Considere a caixa preta de interruptores.



- ▶ Este problema consiste em um recurso do tipo “caixa preta” com um seqüência de 5 interruptores de entrada.
- ▶ Para cada configuração dos 5 interruptores existe um sinal de saída **f**, matematicamente $f = f(s)$, onde s é uma configuração específica.
- ▶ O objetivo do problema é posicionar os interruptores de maneira a obter o maior valor possível para **f**.
- ▶ Codificação dos interruptores: Uma string de comprimento 5, valorado com 0s ou 1s onde cada um dos 5 interruptores é representado por 1 se o interruptor está ligado e 0 se o interruptor está desligado.

Explorando (Goldberg)

- ▶ AGs realizam busca a partir de uma população de pontos (soluções) e não a partir de um único ponto.
 - ▶ Em muitos métodos de otimização, move-se de um estado para o próximo, ou de uma solução para a próxima, usando alguma regra de transição para determinar o próximo estado ou situação (ponto).
 - ▶ Este método ponto-a-ponto é perigoso porque pode cair em mínimos ou máximos locais.
 - ▶ Os AGs trabalham com uma população de soluções, buscando o máximo ou mínimo de forma paralela.
- ▶ No nosso problema da caixa preta de interruptores, um método clássico começaria com a determinada configuração de interruptores, aplicaria a regra de transição e geraria a nova configuração.
 - ▶ Um algoritmo genético começaria com uma população de strings e geraria sucessivas populações de strings.
- ▶ Por exemplo:
 - ▶ Uma população inicial de 4 strings seria gerada randomicamente;
 - ▶ Em seguida, a partir desta, sucessivas populações poderiam ser geradas



Explorando (Goldberg)

- ▶ AGs usam informação de retorno – *feedback* da função objetivo - e não informação derivativa ou outro tipo de conhecimento auxiliar;
- ▶ Muitas técnicas exigem muita informação auxiliar sobre o problema para trabalhar adequadamente.
- ▶ Técnicas baseadas no gradiente, por exemplo, precisam de derivadas para serem capazes de “subir um morro”.
- ▶ Técnicas gulosas requerem acesso à maioria, senão a todos, os parâmetros informacionais –por exemplo, as possibilidades a serem avaliadas.
- ▶ Algoritmos genéticos não exigem informações diretas sobre o problema (eles são ditos cegos) eles requerem apenas o *feedback*.



Explorando (Goldberg)

- ▶ AGs usa regras de transição probabilísticas e não regras determinísticas.

- ▶ A escolha por gerar um novo estado é feita baseada em probabilidades que ajudar o algoritmo a caminha por regiões promissoras no espaço de busca.



Exemplo

- ▶ <http://www.obitko.com/tutorials/genetic-algorithms/portuguese/example-function-minimum.php>
- ▶ <http://www.obitko.com/tutorials/genetic-algorithms/portuguese/example-3d-function.php>
- ▶ <http://www.obitko.com/tutorials/genetic-algorithms/portuguese/tsp-example.php>



Bibliografia

- ▶ RUSSEL, S. e NORVIG, P. Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.
 - ▶ Pgs: 63-68; 70-71; 95-99; 104-106; 113-117.
- ▶ GOLDBERG, D. Genetic Algorithms. Addison Wesley, 1988
- ▶ LINDEN, R. Algoritmos Genéticos: uma importante ferramenta da Inteligência Computacional. Ed. Brasport, 2006.

