Tópico 8
Estruturas e Estratégias de Busca

1 Introdução

- A IA Clássica, voltada para processamento simbólico, pode ser dividida em representação do conhecimento e busca.
- Em problemas de busca, o espaço de busca e suas propriedades topológicas são definidos pela técnica de representação do conhecimento (codificação da solução).
- Em espaços discretos, deve-se percorrer um grafo em busca de um nó.
- Em espaços contínuos, deve-se percorrer o espaço denso em busca de um ponto.
- Em ambos os casos, a noção de vizinhança do nó ou ponto atual é de grande relevância.
- Existem buscas:
  - ✓ Não-informadas (usam apenas informação de ordem 0) e informadas;
  - ✓ Populacionais e não-populacionais;
  - ✓ Com decisões determinísticas e com decisões estocásticas;

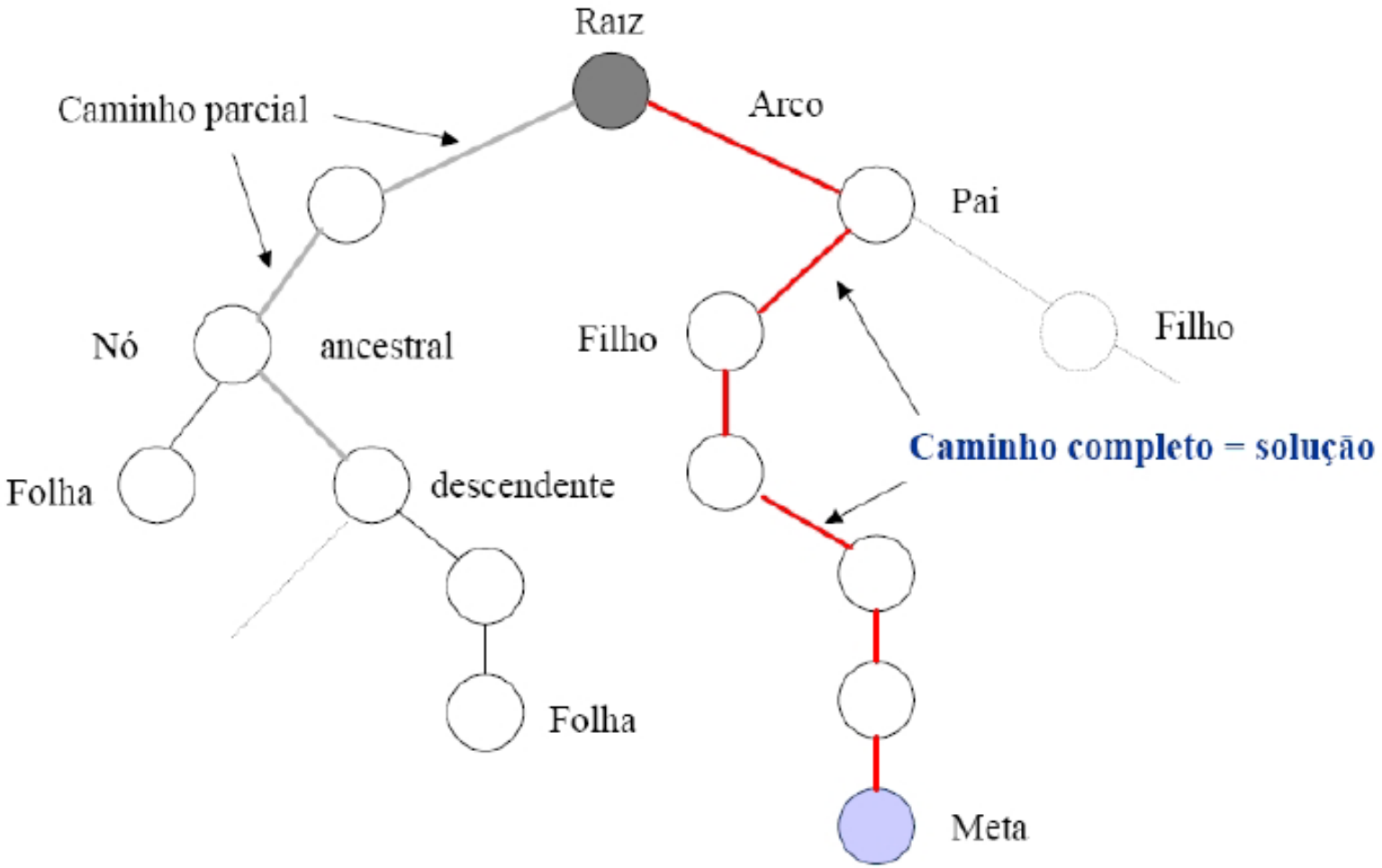
- ✓ Com e sem garantia de convergência para a solução ótima (embora essa propriedade pode depender do tipo de problema);
- ✓ Com e sem busca local.

- Uma busca não-informada é também chamada de busca cega;
- Qualquer que seja a estratégia de busca, é desejável estruturar a busca de modo a evitar ciclos. Essa é a motivação principal para a utilização de árvores de busca em espaços discretos.

2 Algoritmos de busca não-informados

- Dentre as várias estratégias para busca cega em árvore, será dada ênfase aqui a apenas algumas delas.
- É importante salientar que, embora o custo de cada uma seja distinto, todas as propostas aqui apresentadas se tornam computacionalmente intratáveis quando o número de folhas é muito grande.

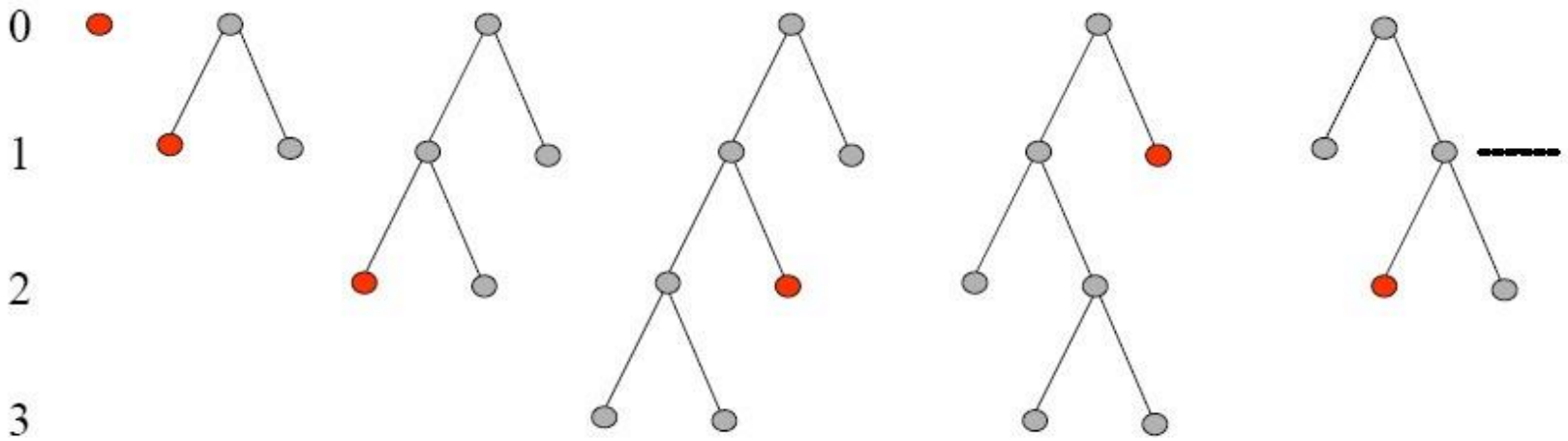
2.1 Nomenclatura



Nota: Parte das figuras desta Seção 2 foram tomadas de Notas de Aula do Prof. Fernando Gomide (DCA/FEEC/Unicamp).

- Tomando uma árvore *n*-ária (todos os pais, ou seja, todos os nós que não são folha, têm *n* filhos) e equilibrada, com altura *h*, então o número de caminhos possíveis é dado por n^h.

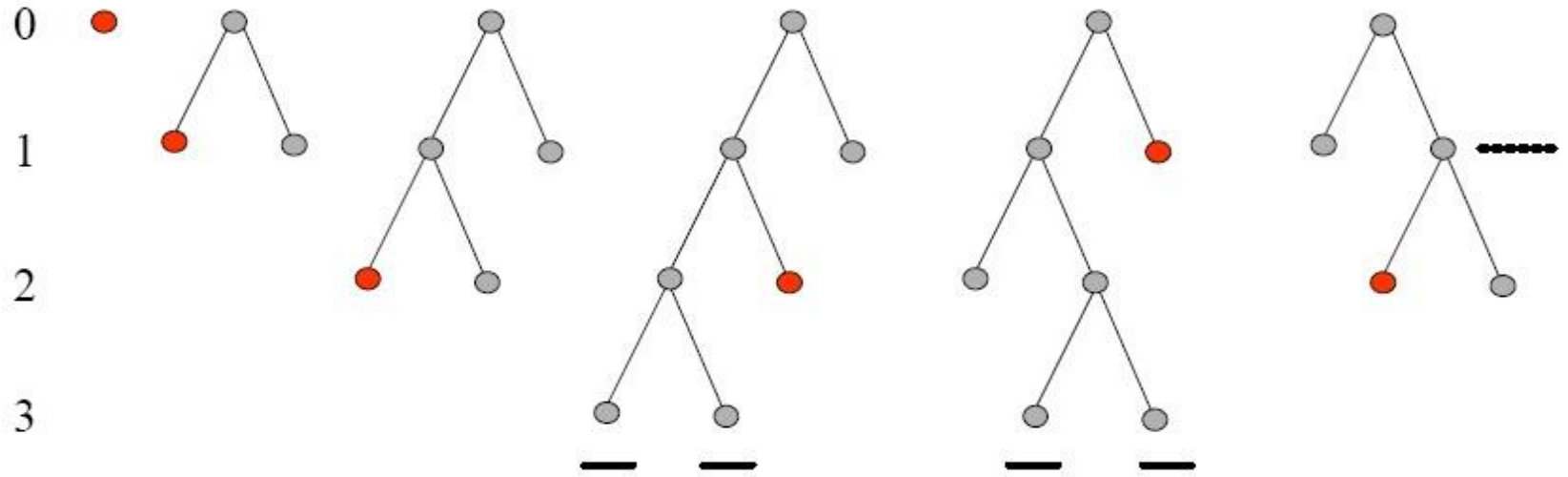
2.2 Busca em profundidade (*depth-first search*)



- Este exemplo considera que a altura máxima da árvore é três.
- Na priorização dos nós-filhos, é utilizada a estratégia *left-first*.

- Desvantagem 1: não garante encontrar a melhor solução, considerando a melhor solução como aquela que se encontra mais próxima da raiz, ou seja, que requer o menor número de operações para ser alcançada.
- Desvantagem 2: não pode ser aplicada quando a árvore tem altura infinita.

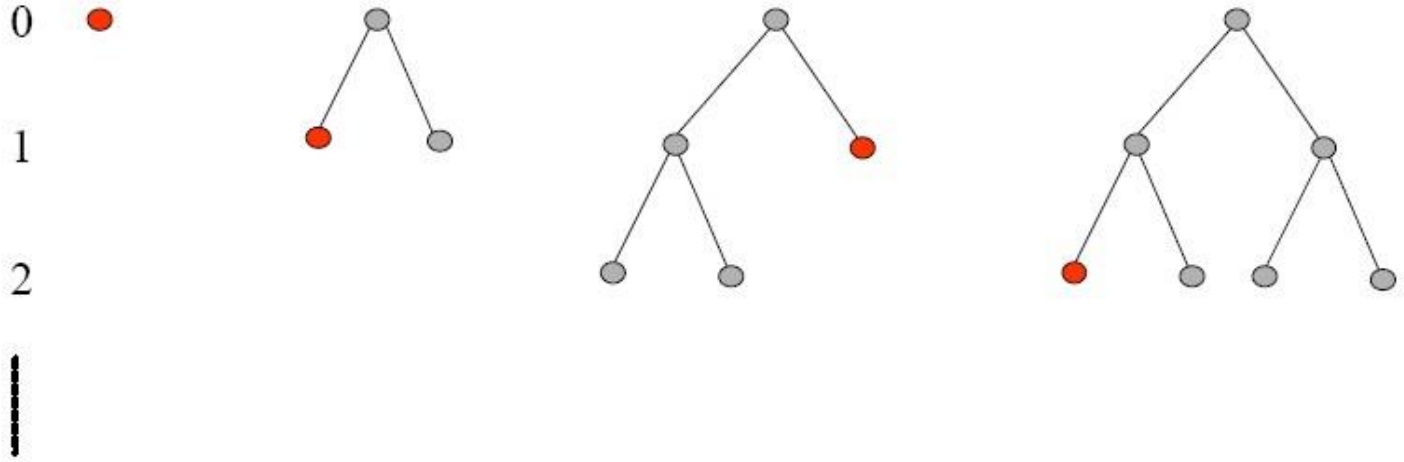
2.3 Busca em profundidade com altura limitada (*depth-limited search*)



- Impõe-se uma altura máxima para a busca (igual a três, na figura acima), mesmo que a altura da árvore seja maior ou até ilimitada.

- A definição de altura máxima para a busca deve levar em conta a limitação de recursos de processamento e memória, além de requisitos do problema (deve haver uma chance não desprezível de encontrar a solução).
- Na priorização dos nós-filhos, é utilizada a estratégia *left-first*.

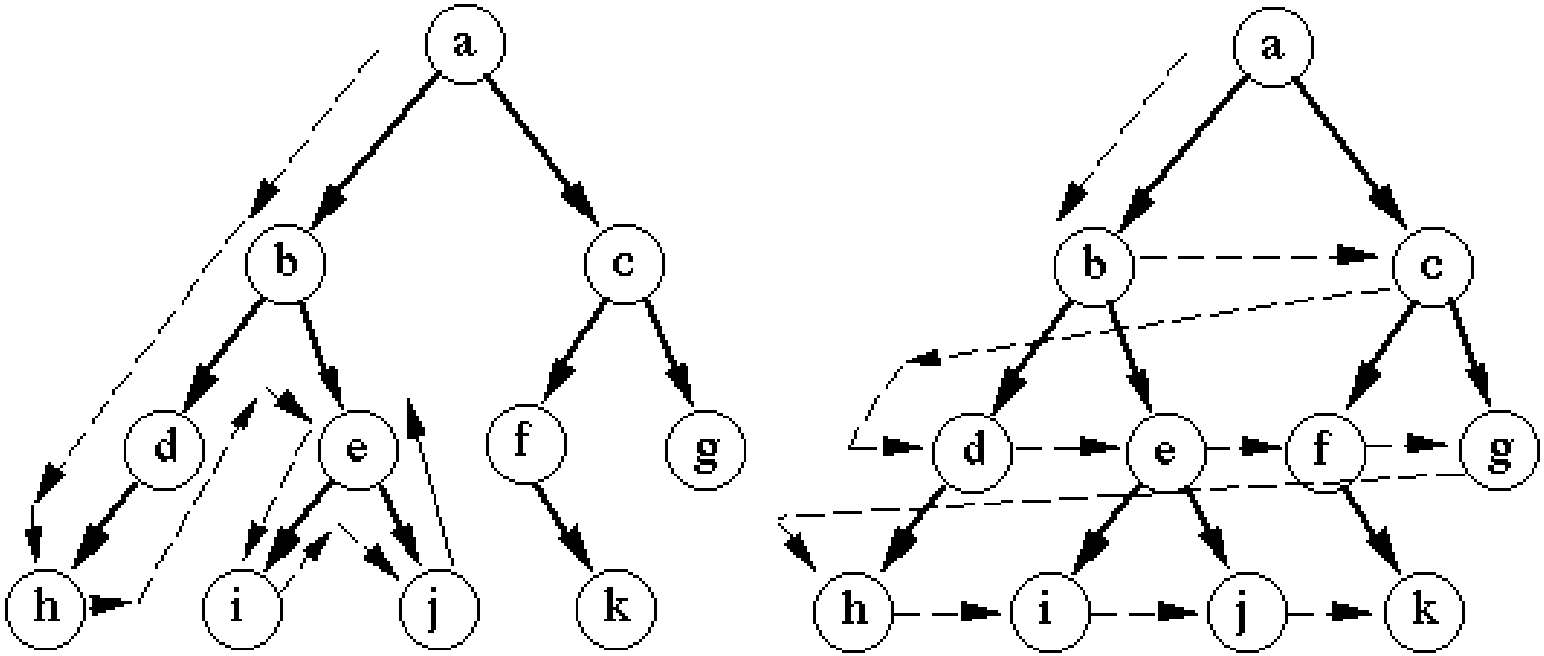
2.4 Busca em amplitude ou largura (*breadth-first search*)



Na priorização dos nós-filhos, é utilizada a estratégia *left-first*.

- Desvantagem: produz um aumento expressivo no uso de memória.
- É um método de força bruta que garante encontrar a melhor solução.

- Contraste entre as duas técnicas:



Fonte: https://moodle-arquivo.ciencias.ulisboa.pt/1516/mod/page/view.php?id=92138

2.5 Comparação das três estratégias

- n: ordem do nó;
- h: altura da árvore;
- $h_{\max}$: altura máxima para a busca
- h_{sol}: altura da solução

Critério	Tempo	Memória	Ótimo?	Completo?
Profundidade	n^h	$n \cdot h$	Não	Não
Profundidade limitada	$n^{h_{\max}}$	$n \cdot h_{\max}$	Não	Não
Amplitude	$n^{h_{sol}+1}$	$n^{h_{sol}+1}$	Sim	Sim

2.6 Exemplo de busca em árvore

- Quebra-cabeça de 8 (Eight-puzzle): considere o problema de partir de uma configuração qualquer para os números inteiros de 1 a 8 numa grade 3 × 3 e chegar à configuração final:

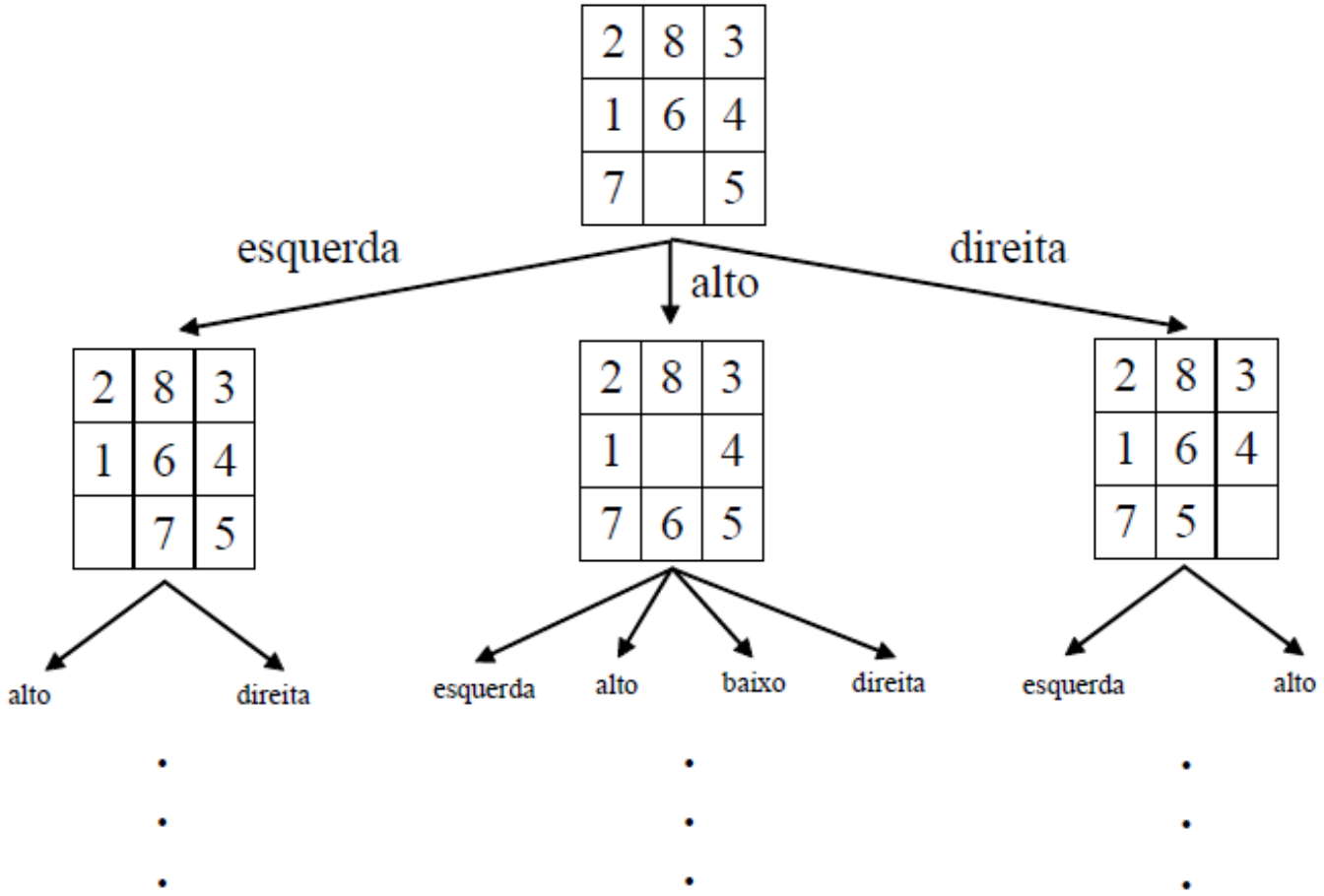
1	2	3
8		4
7	6	5

- Inicialmente, é necessário definir uma representação em árvore para este problema, seguida da definição de uma forma eficiente de buscar a solução percorrendo a árvore.
- Considere a configuração inicial na forma:

2	8	3
1	6	4
7		5

a qual corresponde ao nó-raiz da árvore.

- Há três movimentos possíveis:
  - ✓ mover o 6 para baixo;
  - ✓ mover o 7 para a direita;
  - ✓ mover o 5 para a esquerda.
- Em outras palavras, a configuração inicial tem três configurações sucessivas a ela.
- Embora a sequência de passos acima se mostre adequada ao raciocínio humano, para a formulação do processo de busca é mais adequado pensar no movimento do quadrado vazio e não no movimento de um número de um quadrado ocupado para o quadrado vazio.
- Neste caso, o quadrado vazio pode expressar no máximo 4 movimentos possíveis (quando está no centro) e no mínimo 2 movimentos possíveis (quando está nas bordas).
- Com isso, a árvore de busca fica totalmente caracterizada e um fragmento dela pode ser visualizado a seguir.

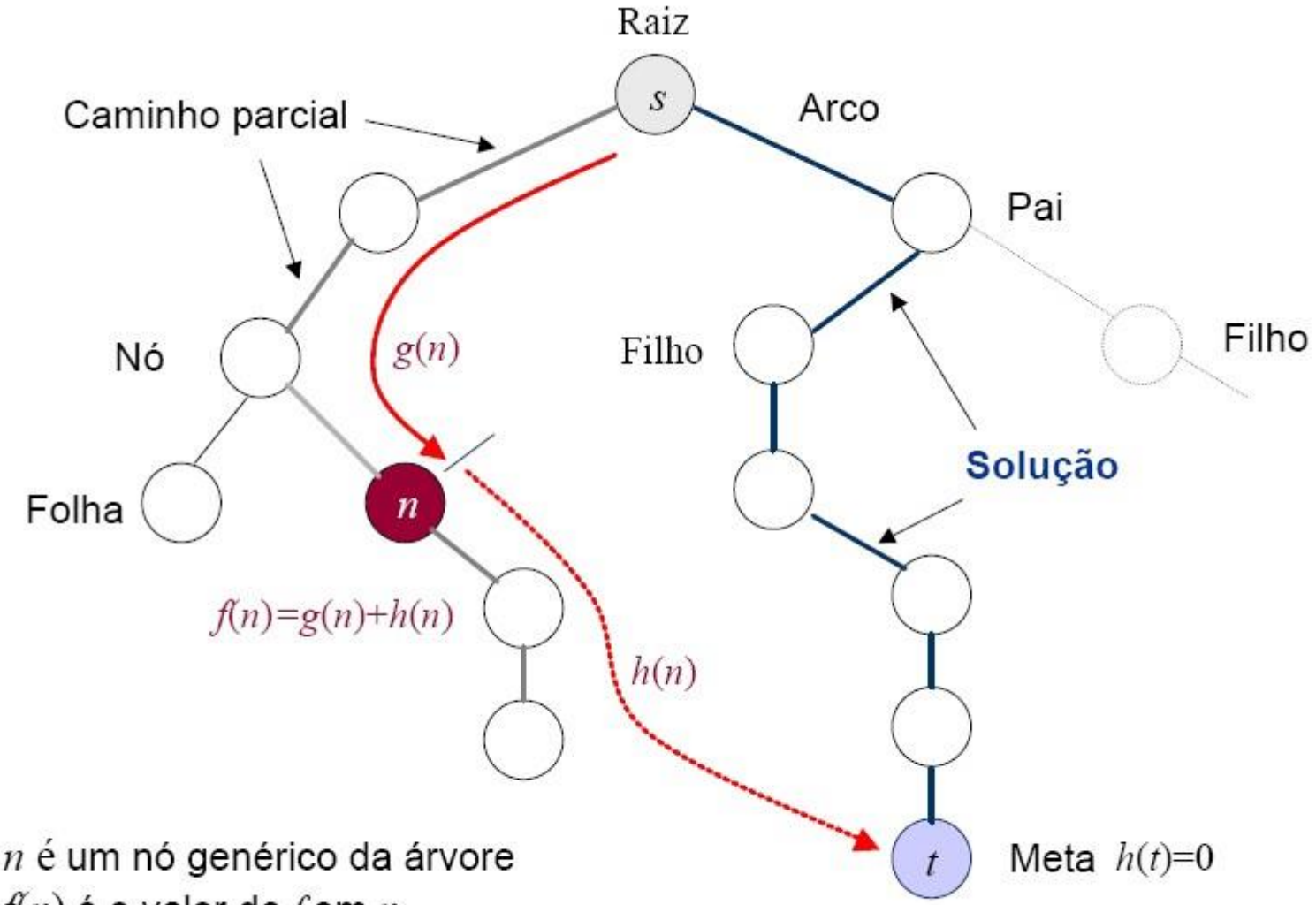


Árvore de busca para o quebra-cabeça de 8

- Por ser uma árvore com raiz, cada nó desta árvore de busca possui um único pai e existe apenas um nó-raiz, que corresponde à configuração inicial.

3 Algoritmos de busca informados

- São também conhecidos como busca *best-fit* ou busca heurística.
- As buscas informadas são parecidas com a busca em amplitude, com a exceção de que a busca não procede de forma uniforme a partir do nó-raiz da árvore. De acordo com heurísticas e/ou informações específicas do problema, é possível definir uma ordem de preferência entre os caminhos possíveis a partir do nó-raiz.
- Logo, recorrendo a um pouco mais de informação, é possível ser mais eficiente.
- Em outras palavras, o melhor caminho até a solução é obtido pela adoção de decisões do tipo *best-fit* a cada passo. Para tanto, estimar o custo de se chegar a cada nó – $g(n)$ – e/ou o quão distante se está da solução – $h(n)$ – passam a ser de grande relevância.



n é um nó genérico da árvore
$f(n)$ é o valor de f em n
$g(n)$ é o custo da trajetória até n
$h(n)$ é uma *estimativa* do custo de n até a meta

- Assim, a decisão para que nó seguir é determinada:
  - ✓ Na busca de custo uniforme: pelo custo de se chegar àquele nó; ou
  - ✓ Na busca gananciosa: pelo custo estimado de se alcançar a solução a partir do nó corrente; ou
  - ✓ Na busca A: pela adição de ambos os custos.
- Seguem definições mais completas para as métricas envolvidas:
  - ✓ $g(n)$ (fator de altura): é o custo do caminho de custo mínimo entre o nó-raiz e o nó n.
  - ✓ $h(n)$ (fator heurístico): é o custo *estimado* do caminho de custo mínimo entre o nó n e o nó-solução, considerando todos os possíveis nós-solução e todos os possíveis caminhos entre esses dois nós.
  - ✓ $f(n) = g(n) + h(n)$: é o custo do caminho de custo mínimo entre o nó-raiz e um nó-solução, considerando todos os caminhos que passam pelo nó n.

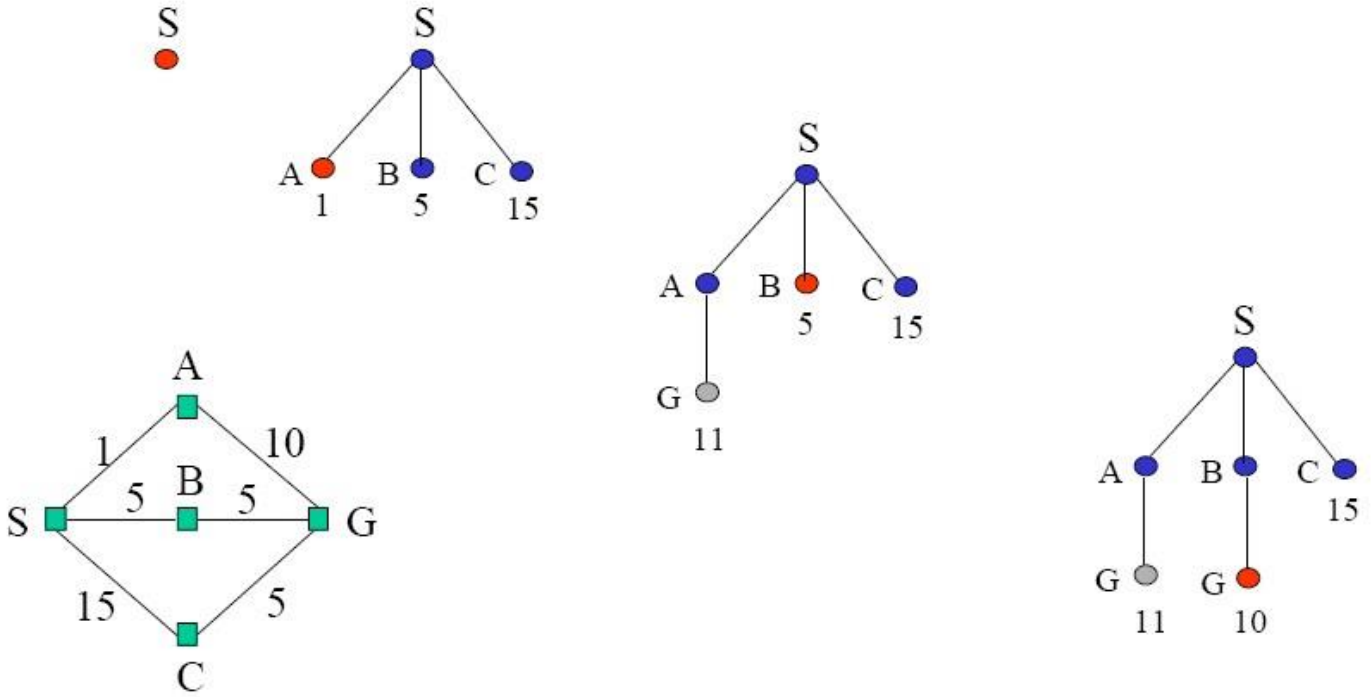
- Note que $f(n_0) = h(n_0)$ é o custo *estimado* do caminho de custo mínimo entre o nó-raiz e um nó-solução.
- Esses algoritmos de busca só são válidos junto a problemas para os quais, conforme aumenta a altura da árvore, o custo da solução vai aumentando monotonicamente, ou seja, o custo junto a cada aresta da árvore é não-negativo.
- Uma outra restrição mandatória para a validade das técnicas é que o número de filhos de cada nó-pai da árvore de busca seja finito.
- Quando o fator heurístico $h(n)$ nunca sobre-estima o caminho de custo mínimo real entre o nó n e o nó-solução, $h_{real}(n)$, ou seja, $h(n) \leq h_{real}(n)$, então $h(n)$ é chamado de heurística otimista ou heurística admissível e o algoritmo de busca é denominado A*.

3.1 Busca de custo uniforme (*uniform-cost search*)

- Leva em conta apenas o fator de altura $g(n)$, escolhendo o nó com o menor g a cada passo.

3.2 Exemplo de aplicação da busca de custo uniforme

- Considere o problema de encontrar o menor caminho entre *S* e *G* num grafo.



3.3 Busca gananciosa (*greedy search*)

- Leva em conta apenas o fator heurístico $h(n)$.

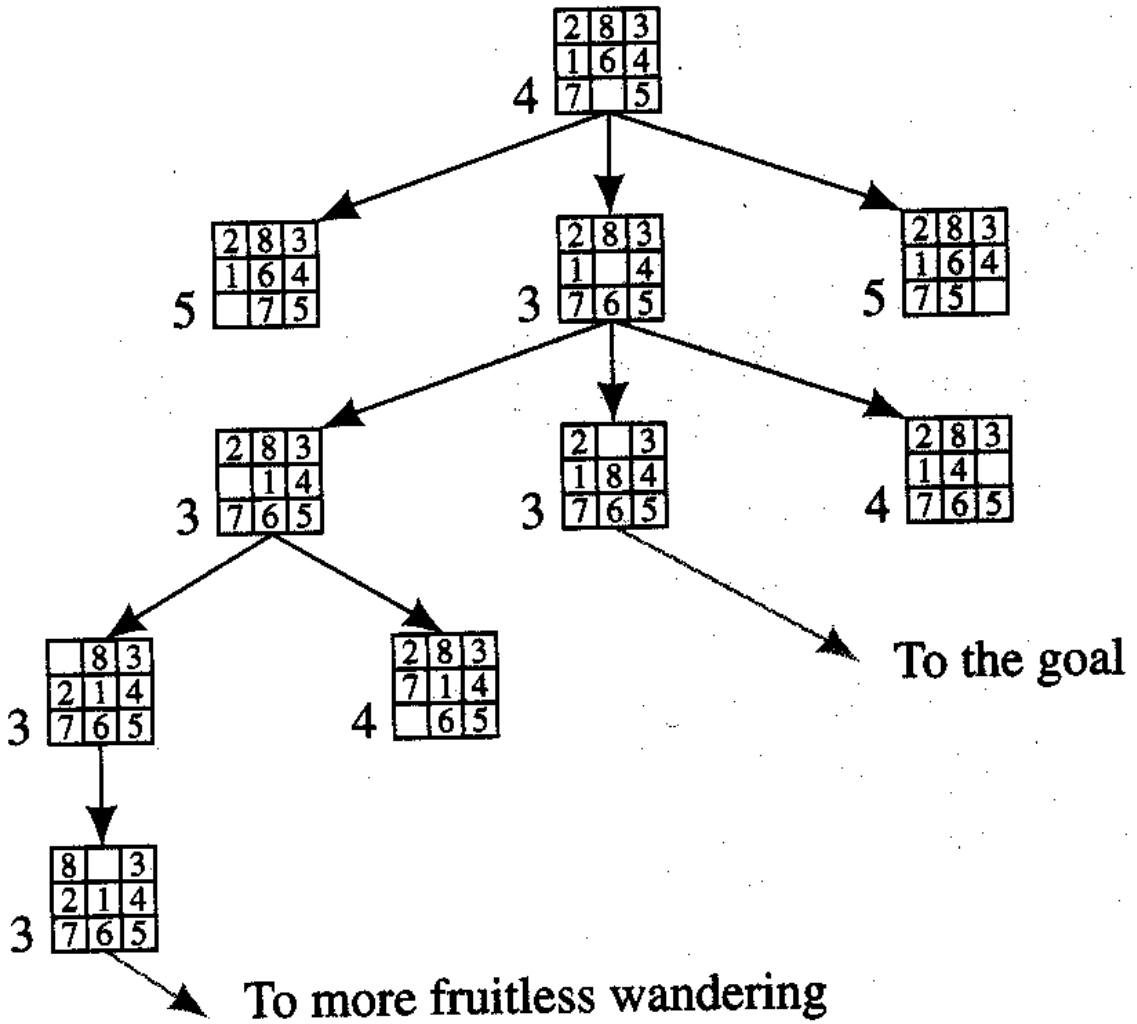
3.4 Exemplo de aplicação da busca gananciosa

- Considere novamente o problema de partir da configuração

2	8	3
1	6	4
7		5

para os números inteiros de 1 a 8 numa grade 3 × 3 e chegar à configuração final:

1	2	3
8		4
7	6	5


4
5
3
5
3
3
4
3
4
To the goal
3
To more fruitless wandering

- Este exemplo mostra que a busca gananciosa pode se perder em certas aplicações, geralmente porque não considera o quanto já se caminhou na árvore.

3.5 Busca A*

- Esta técnica requer que a estimação do custo $f(n) = g(n) + h(n)$ seja monotônica, ou seja, nunca caia quando se afasta da raiz da árvore de busca. Diferente das duas técnicas anteriores, sob esta hipótese, sempre é possível encontrar a solução ótima com a busca A*.
- A busca A* degenera para a busca em amplitude quando a heurística não aponta preferência por nenhum caminho ao longo da busca.
- A busca A* é completa, ótima e eficiente (HART *et al.*, 1968).
- No entanto, sua complexidade ainda é exponencial e o seu uso de memória é intenso.
- Existem na literatura adaptações da busca A* visando torná-la mais tratável junto a classes específicas de problemas:

- ✓ Iterative-Deepening A* (KORF, 1985)
- ✓ Simplified Memory Bound A*

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

$h(n)$ otimista $\rightarrow$ heurística admissível

$f(n)$ nunca decresce ao longo de um caminho $\rightarrow$ monotonicidade

n *pai de* n', $f(n') \leq f(n) \rightarrow f(n') = \max[f(n), g(n') + h(n')] \rightarrow$ *pathmax*

A^* expande todos nós com $f(n) < f^*$

Primeira solução encontrada é a solução ótima

A^* é otimamente eficiente $\rightarrow$ nenhum outro algoritmo expande menos nós que A^*

3.6 Exemplo de aplicação da busca A*

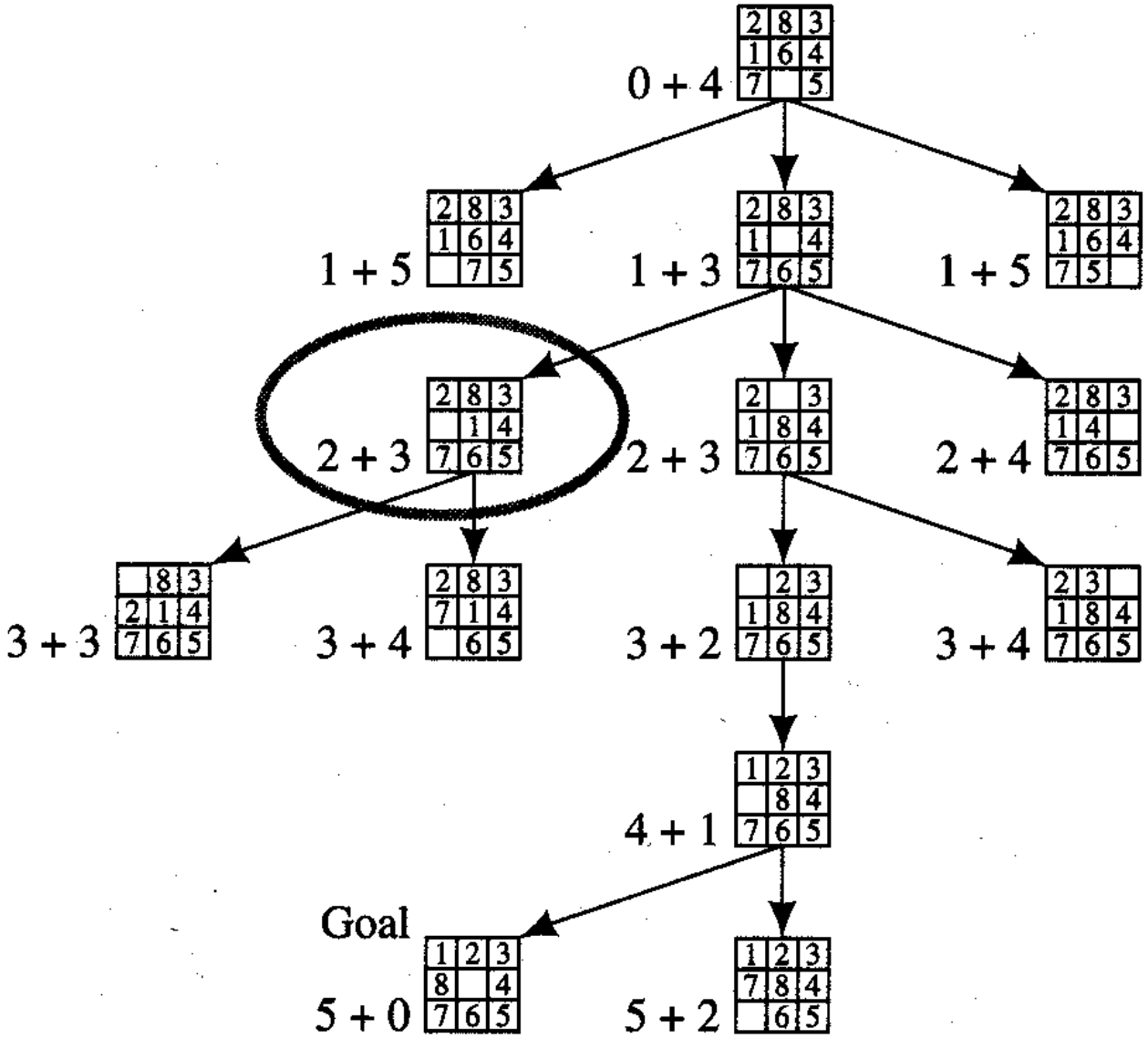
- Considere novamente o problema de partir da configuração

2	8	3
1	6	4
7		5

para os números inteiros de 1 a 8 numa grade 3 × 3 e chegar à configuração final:

1	2	3
8		4
7	6	5

- Heurísticas admissíveis [$h(n)$]:
  - ✓ **Quantidade de números fora de posição;**
  - ✓ Somatória das distâncias dos números que estão em posição incorreta até a posição correta.


0 + 4
1 + 5
1 + 3
1 + 5
2 + 3
2 + 3
2 + 4
3 + 3
3 + 4
3 + 2
3 + 4
4 + 1
Goal
5 + 0
5 + 2

3.7 Definição de heurísticas admissíveis

- Mesmo sem conhecer o custo real, muitas vezes é possível garantir que o custo estimado não o ultrapassa.
- Exemplos:
  - ✓ Quantidade de números fora de posição no quebra-cabeça de 8: para cada número fora de posição, é necessário ao menos um movimento para colocar o número na posição correta;
  - ✓ Rota mínima entre duas cidades: qualquer caminho entre duas cidades é maior ou igual ao comprimento da reta que une as duas cidades.

3.8 Relações com branch-and-bound

- A busca A* é mais poderosa e eficiente que o algoritmo branch-and-bound, mas requer mais informação sobre o problema.
- A seguir, é apresentado o princípio de operação do algoritmo branch-and-bound.

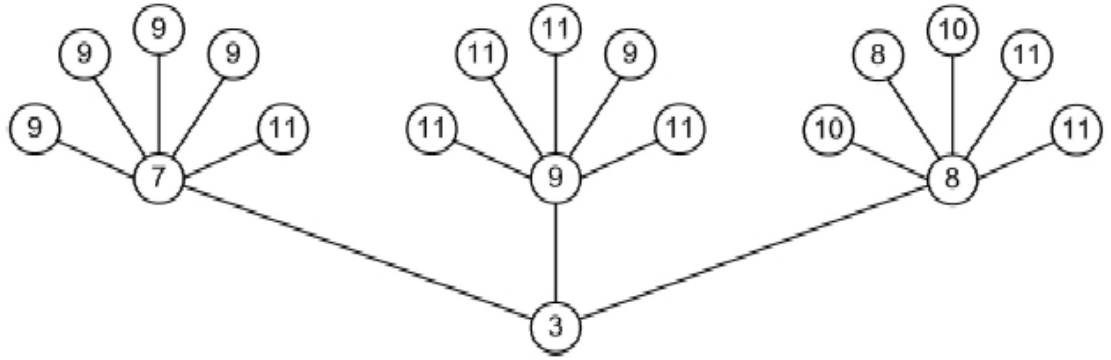


Figure 5.4. *Parsimony scores.*

Obtain a good global bound g as follows:

visit ③ ⟶ ⑦ ⟶ ⑨ now, set $g = 9$

visit ③ ⟶ ⑧ ⟶ ⑧ now, set $g = 8$

visit ③ ⟶ ⑨ no need to go further, as$l > g$

In this example, brach and bound allows us to avoid $\frac{1}{3}$ of all complete trees. In larger datasets we can save more, but it depends on compatibility of data.

4 Algoritmos de busca com decisões estocásticas

4.1 Simulated Annealing

- O material desta seção está baseado em GOMES (2006), e é aplicável tanto para espaços de busca contínuos como discretos.
- O recozimento simulado (GOFFE *et al.*, 1994) é a meta-heurística mais antiga e um dos primeiros algoritmos a adotar uma estratégia explícita para escapar de pontos de ótimo local. O método é inspirado no processo de aquecimento de metais e vidros, que assumem um estado de baixa energia quando resfriados de maneira lenta e gradual.
- A ideia básica é contornar pontos de ótimo local permitindo a realização de movimentos que conduzam temporariamente a soluções de pior qualidade que a atual. A probabilidade de se aceitarem movimentos que levem a soluções de pior qualidade decresce ao longo do processo de busca.

- O algoritmo inicia com a geração de uma solução inicial *s*, que pode ser obtida aleatoriamente ou de acordo com alguma outra heurística. Em seguida, é inicializado um parâmetro de temperatura *T*, o principal parâmetro de controle do recozimento simulado. A cada iteração, uma solução vizinha *s*' é tomada aleatoriamente do conjunto de soluções vizinhas $N(s)$ da solução atual.
- A aceitação ou não da nova solução *s*' leva em consideração três elementos: o custo da solução corrente *s*, o custo da solução *s*' em análise e o valor do parâmetro *T*.
- Considerando um problema de minimização, caso o valor da função-objetivo da solução em análise seja menor que o valor da função-objetivo da solução atual, a solução em análise é aceita. Caso contrário, a solução será aceita com uma probabilidade *P*, geralmente calculada de acordo com a distribuição de Boltzmann:

$$P(s, s', T) = \exp\left(-\frac{f(s') - f(s)}{T}\right)$$

- A seguir, é apresentado um pseudocódigo para o recozimento simulado:

Procedimento RecozimentoSimulado

Gerar solução inicial *s*

Inicializar temperatura *T*

Enquanto não atingir condição de parada **faça**

Selecionar solução aleatoriamente *s*' de $N(s)$

Se $f(s') < f(s)$ então $s \leftarrow s'$

senão aceitar *s*' com probabilidade $P(s, s', T)$

Atualizar temperatura *T*

Fim Enquanto

Fim Procedimento

Recozimento simulado (*simulated annealing*)

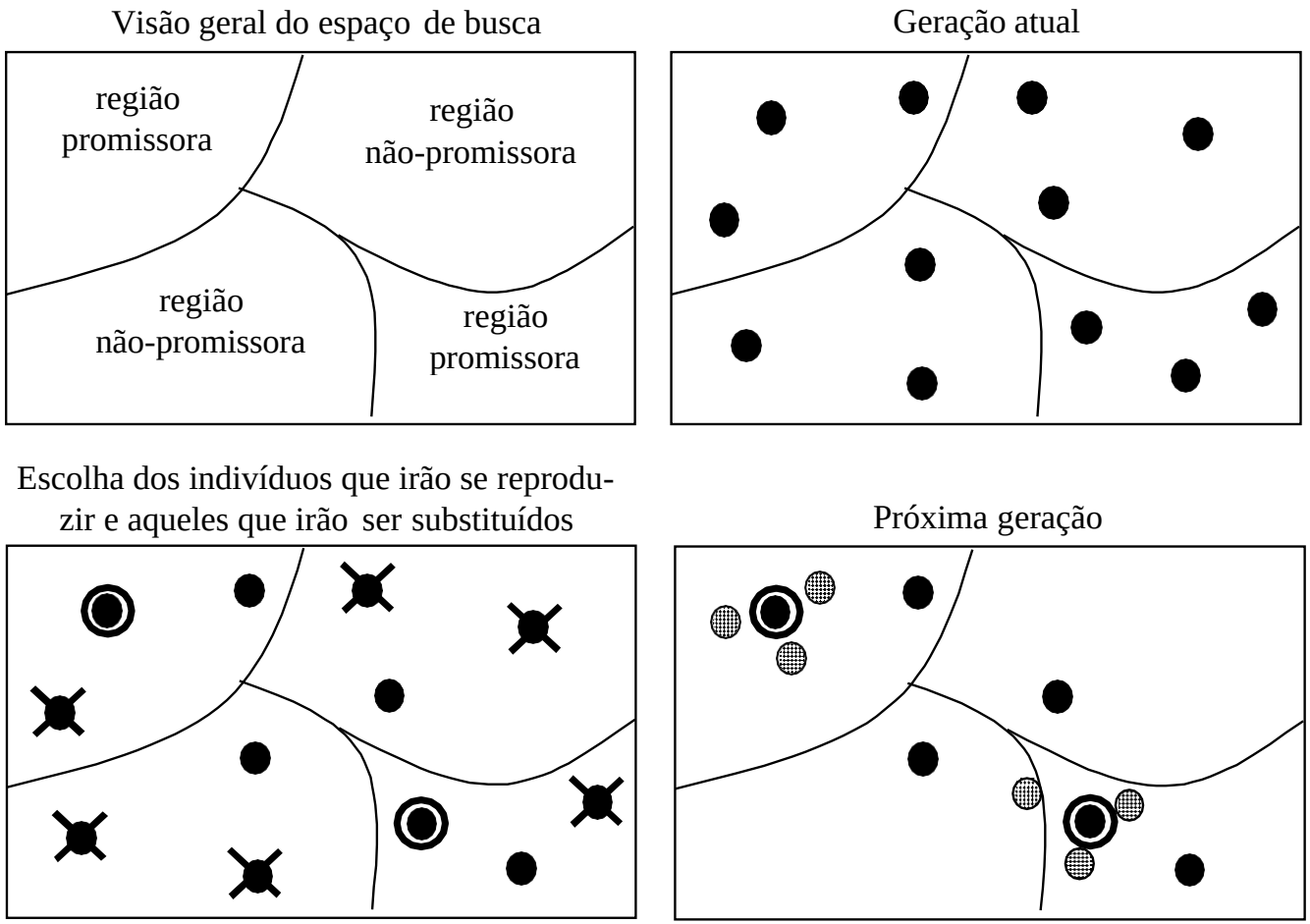
4.2 Busca Tabu

- O material desta seção está baseado em GOMES (2006), e é aplicável tanto para espaços de busca contínuos como discretos.
- A busca tabu (GLOVER & LAGUNA, 2004) é uma meta-heurística de escopo local provida da habilidade de contornar pontos de ótimo local prematuros. Isso é realizado via uma memória seletiva que reflete o histórico da busca, considerando soluções obtidas no decorrer do processo de busca. Assim, restrições que delimitam o espaço de busca são sistematicamente impostas de modo que certas soluções se tornem proibidas ou tabu, no intuito de varrer regiões ainda inexploradas do espaço de soluções.
- As origens da busca tabu datam de meados da década de 1960, mas sua versão atual foi formulada por Fred Glover em 1986. Desde então, o método vem sendo aplicado com sucesso a uma ampla variedade de problemas combinatórios de alta complexidade (GLOVER & LAGUNA, 2004).

- A busca tabu trabalha com base em uma única solução. O método de busca parte de uma solução inicial, e a cada iteração a vizinhança da solução corrente é analisada. Na sequência, uma nova solução é selecionada do conjunto de vizinhos. Esse processo é repetido até que um critério de parada seja alcançado. A principal característica desse método é que ele opera com o objetivo de transcender a otimalidade local, permitindo a degradação da solução durante o processo de busca e até a ocorrência temporária de soluções infactíveis.
- A busca tabu conserva um histórico do processo de busca com o objetivo de contornar pontos de ótimo local. Durante a busca, certas soluções (ou movimentos) são considerados proibidos ou tabu. Em geral, soluções ou partes de soluções recentemente visitadas ou frequentemente obtidas são classificadas como tabu. Existem variações da busca tabu propondo melhorias, tais como estratégias de diversificação e memórias adaptativas.

5 Algoritmos de busca populacionais

- Algoritmos evolutivos



6 Algoritmos para busca local

- Requerem a definição de uma vizinhança;
- Tipos de algoritmos de busca local em espaços discretos:
  - ✓ First improvement;
  - ✓ Best improvement;
  - ✓ Best improvement with limited resources.

7 Referências Bibliográficas

HART, P., NILSSON, N. & RAPHAEL, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. Syst. Science and Cybernetics, vol. SSC-4, no. 2, pp. 100-107, 1968.

GOMES, L.C.T. "Inteligência Computacional na Síntese de Meta-heurísticas para Otimização Combinatória e Multimodal", Tese de Doutorado, FEEC/Unicamp, 2006.

GLOVER, F.W. & LAGUNA, M. "Tabu Search", Springer, 2004.

LAWLER, E.L. & WOOD, D.E. "Branch-and-Bound Methods: A Survey", Operations Research, vol. 14, no. 4, pp. 699-719, 1966.

MITTEN, L.G. "Branch-and-Bound Methods: General Formulation and Properties", Operations Research, vol. 18, no. 1, pp. 24-34, 1970.

GOFFE, W.L., FERRIER, G.D. & ROGERS, J. "Global optimization of statistical functions with simulated annealing", Journal of Econometrics, vol. 60, pp. 65-99, 1994.

KORF, R. "Depth-First Iterative Deepening: An Optimal Admissible Tree Search Algorithm", Artificial Intelligence, vol. 27, pp. 97-109, 1985.

NILSSON, N.J. "Artificial Intelligence: A New Synthesis", Morgan Kaufmann Publishers, 1998.

PEARL, J. "Heuristics: Intelligent Search Strategies for Computer Problem Solving", Addison Wesley, 1984.

RUSSEL, S. & NORVIG, P. "Artificial Intelligence – A Modern Approach", Prentice Hall, 2003.

WINSTON, P. "Artificial Intelligence", 3rd edition, Addison Wesley, 1993.