DCA0121 – INTELIGÊNCIA ARTIFICIAL APLICADA
Aula 8 – Métodos de Busca Não Informada

Prof. Marcelo Augusto Costa Fernandes

mfernandes@dca.ufrn.br

Introdução

- O termo busca possui um aspecto importante dentro da disciplina de IA
  - “Resolver um problema de IA é basicamente resolver um problema de busca”

Métodos de Busca (*Search Strategies*)

- Métodos de busca não informada - *Uninformed (or Blind) Search Strategies*
- Métodos de busca informada ou Heurística (*Informed Search Strategies*)

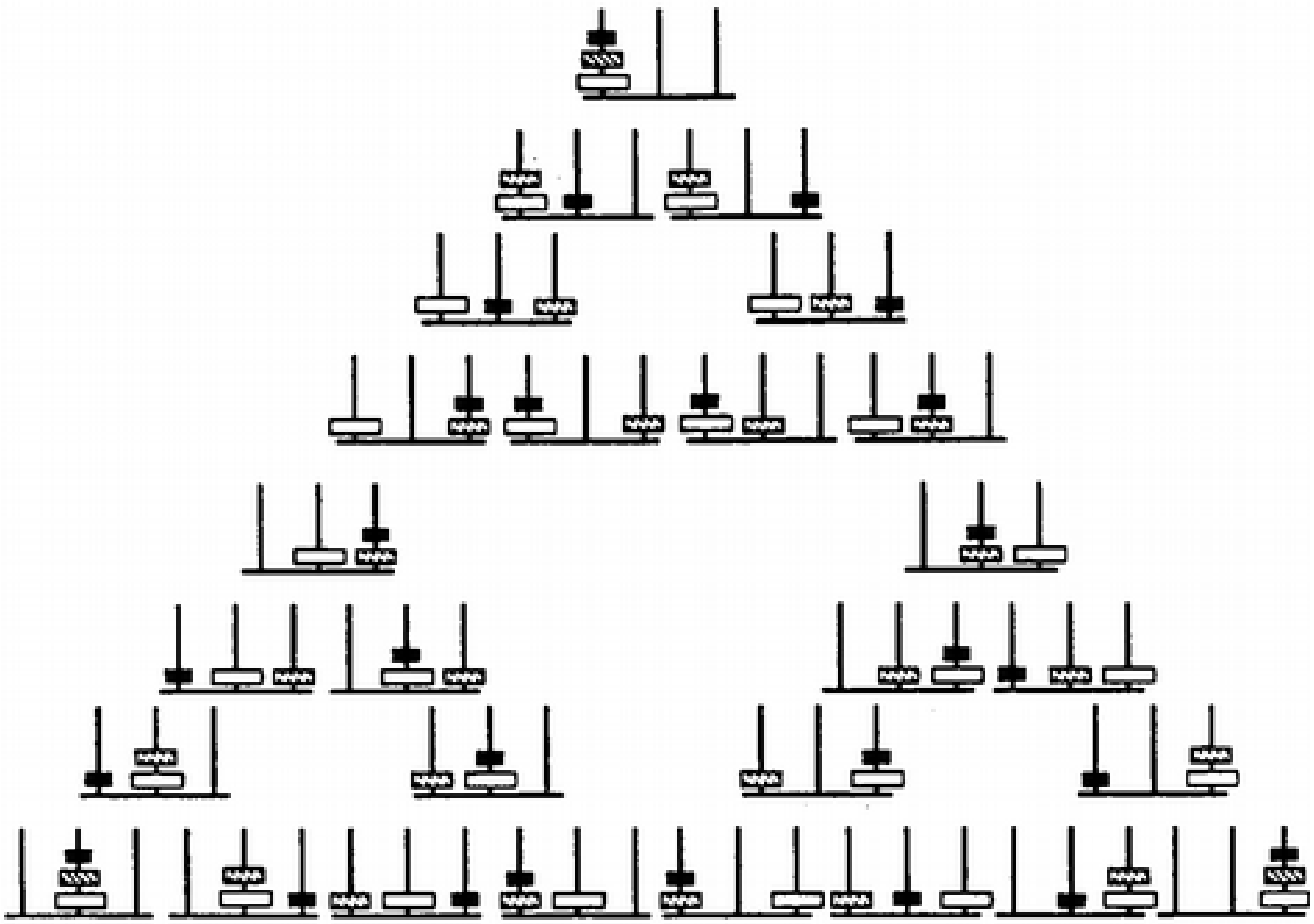
Métodos de busca não informada

- *Uninformed (or Blind) Search Strategies*
- Também chamados de estratégias de busca simples
- Podem ser aplicados a um grupo grande de problemas que envolvem busca
- Porém, não utilizam dados contextuais do problema
  - São cegos (*blind*)
- São mais ineficientes quando comparados aos as técnicas de busca informada

Espaço de Busca (*Search Space*)

- Todas as possíveis ações que devem ser feitas para resolver um dado problema
- As ações dependem do ambiente
  - Alguns ações são proibidas
- Toda ação possui um custo associado
- Alguns caminhos levam a becos sem saída onde outros podem levar a soluções.
- Podem existir múltiplas soluções porém algumas melhores do que outras.
- Os problemas de busca caracterizam por encontrar todas as ações que levam melhor solução.

Exemplo – Jogo Torre de Hanoi



Busca em um espaço de estados

- Outra forma de ver o problema de busca
- Formado por estados e operadores
- Cada estado pode ser caracterizado com uma possibilidade
- Um operador transforma ou leva um estado para outro seguindo uma estratégia de busca
- O objetivo é o estado final

Grafos e Árvores

- Grafos
  - Conjunto finito de nós conectados por arcos.
  - Um loop ou um ciclo existe, quando um arco pode levar de volta para o nó original.
  - Podem ser não direcionados ou direcionados (Dígrafo)
  - Os arcos podem ter pesos associados indicando o custo do caminho.
  - Um grafo sem ciclos é caracterizado com uma árvore

Eficiência em algoritmos

- Eficiência é uma das características mais importantes dos algoritmos. Existem 3 aspetos para escolher um algoritmo:
  - Tempo gasto para a codificação, teste e depuração
  - Tempo de máquina necessário para executar o algoritmo (tempo de execução)
  - O espaço ou memória necessário para a execução
- O algoritmo pode ser:
  - Completo: se existir uma solução, ela certamente é encontrada.
  - Ótimo: A busca encontra a solução de menor custo
- Complexidade temporal: quanto tempo demora para encontrar a solução?
- Complexidade espacial: quanto de memória é usada para realizar a busca

Notação O (Big-O)

- Operadores matemáticos para calcular complexidade de algoritmos
- Seja uma função *f*, pode-se dizer que quando *f*(n) é O(h) diz-se
  - *f*(n) é da ordem de O(n)
  - *f*(n) é assintoticamente limitada por O(n)
- Exemplos
  - $f = n^2+1$ -> *f* é $O(n^2)$
  - $f = 100$ -> *f* é O(1)
  - $f = 5 + 2\log(n) + 3\log(n)^2$ -> f é $O(\log(n)^2)$

Notação O (Big-O)

- O(1): Constante
- O(n): Linear
- O(log n): logarítmica
- $O(n^2)$: Quadrática
- $O(c^n)$: Geométrica
- O(n!): Combinacional
- Apresenta o pior caso quanto a complexidade do algoritmo e é comumente utilizada na comparação entre os algoritmos

Algoritmos

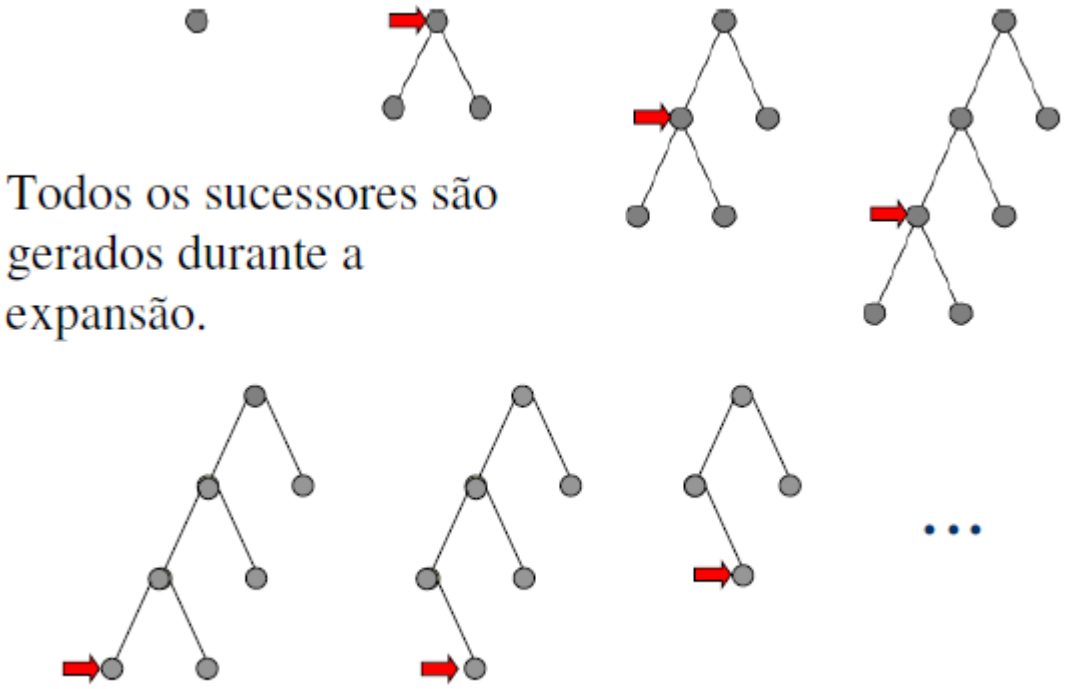
- Busca em profundidade (*DFS - Depth-First Search*)
- Busca em Profundidade Limitada (*DLS -Depth-Limited Search*)
- Busca em Largura (*BFS - Breadth-First Search*)
- Busca Uniforme (*UCS - Uniform-Cost Search*)
- Busca em Profundidade Interativa (*IDS - Iterative Deepening Search*)
- Busca Bidirecional (*Bidirectional Search*)

Busca em profundidade (*DFS - Depth-First Search*)

- Faz uma busca exaustiva do nó raiz ao ramo mais profundo.
- Todo caminho deve ser armazenado em uma estrutura LIFO (pilha)
- Complexidade espacial: O(bd)
- Complexidade temporal: $O(b^d)$
- d - Profundidade da solução encontrada
- b - Fator de Ramificação (*Branching Factor*)
  - número máximo de sucessores de um nó
- É completa se o grafo não possuir ciclos
- Não é ótima
  - Pode encontrar uma solução que não será a melhor

Busca em profundidade (*DFS - Depth-First Search*)

- Pode-se gerar o espaço de busca em tempo de execução



Busca em profundidade (*DFS - Depth-First Search*)

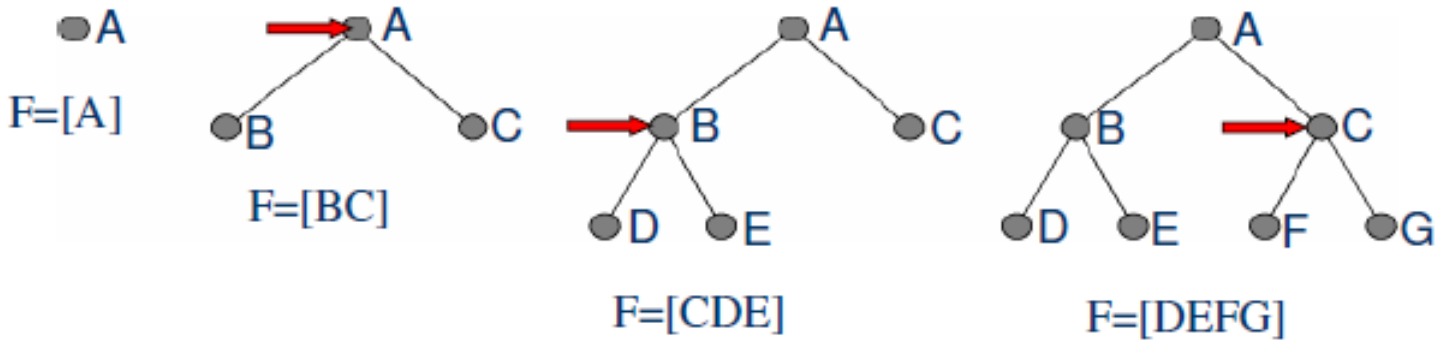
- Para problemas com várias soluções, esta estratégia pode ser bem mais rápida do que busca em largura.
- Esta estratégia deve ser evitada quando as árvores geradas são muito profundas ou geram caminhos infinitos.

Busca em Profundidade Limitada (*DLS -Depth-Limited Search*)

- Evita o problema de arvores não limitadas, limitando a busca a uma profundidade L.
- O conhecimento do problema estabelece o valor de L
- Completa se $L \geq d$
- Ótima apenas no caso em que $L = d$
- Complexidade espacial $O(bL)$
- Complexidade temporal $O(b^L)$

Busca em Largura (*BFS - Breadth-First Search*)

- Também chamada de busca em amplitude
- Faz a busca por cada profundidade do espaço de busca
  - Todos os nós da profundidade 1
  - Todos os nós da profundidade 2
  - …
- Pode-se implementar com uma estrutura FIFO



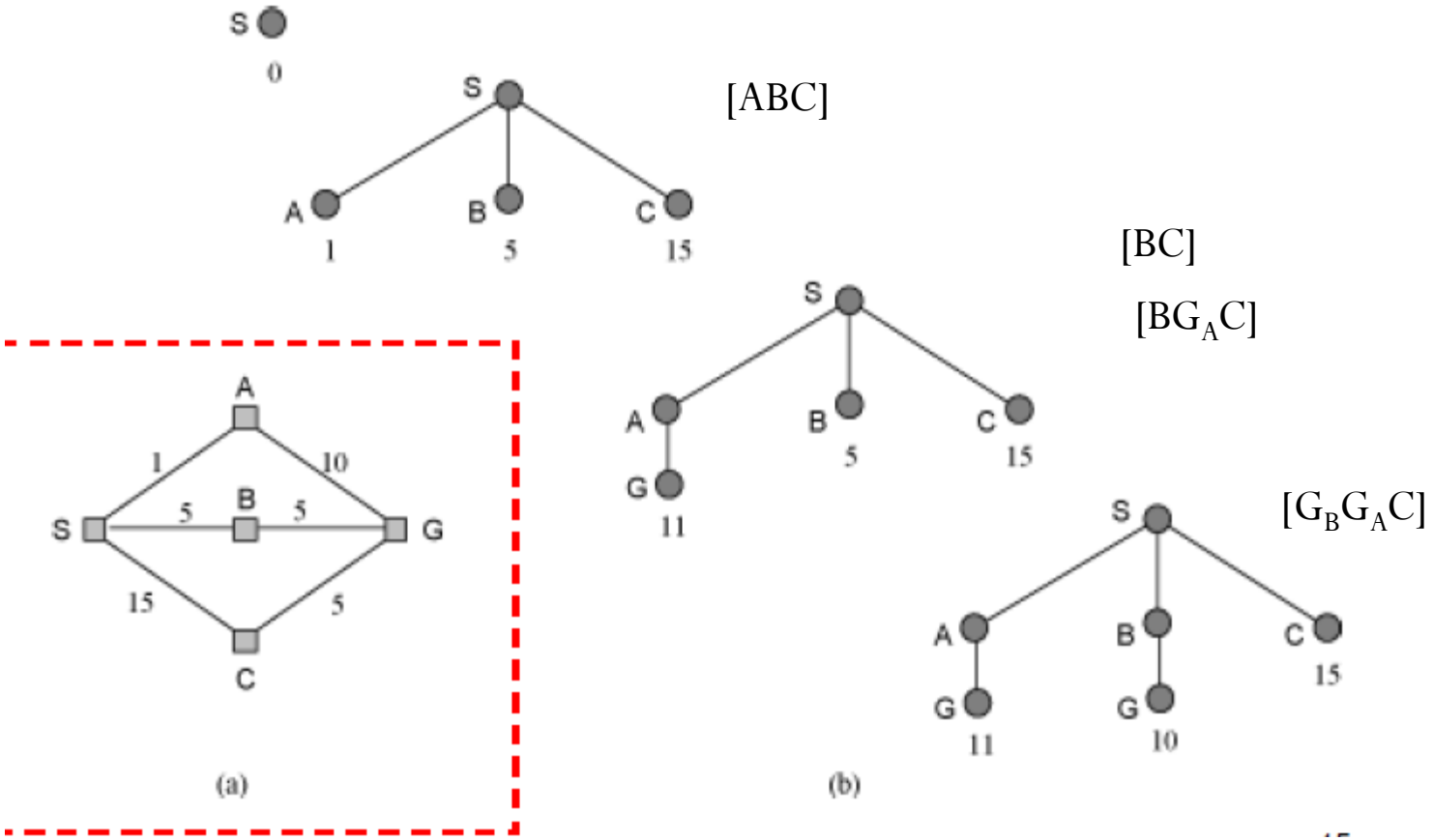
Busca em Largura (*BFS - Breadth-First Search*)

- Se b é finito, esta busca é completa
  - Se existir um nó meta a uma profundidade d o método irá encontrá-lo
- Nem sempre é ótima:caminho mais curto (nó-meta mais próximo da raiz) não é o melhor caminho
- Complexidade espacial: $O(b^d)$
- Complexidade temporal: $O(b^d)$

Busca Uniforme (*UCS - Uniform-Cost Search*)

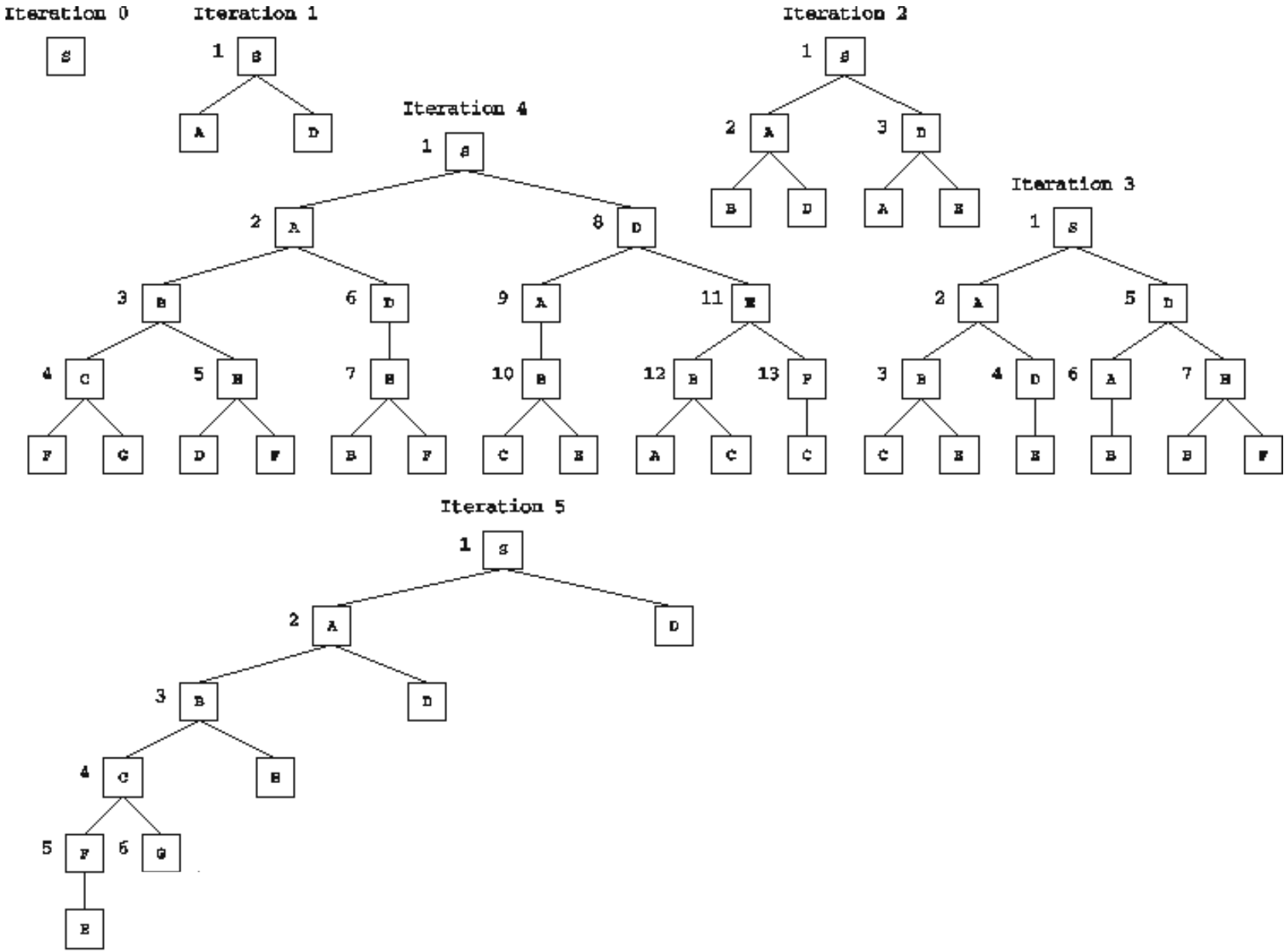
- Faz uso do custo do caminho entre os nós.
- Modificação da busca e largura
- Em vez de expandir o primeiro nó, faz-se a expansão do nó de menor custo de caminho.
- Não leva em consideração a profundidade e sim o custo do caminho
- É completa se cada ação possui um custo positivo e é ótima
- Complexidade espacial: $O(b^d)$
- Complexidade temporal: $O(b^d)$

Busca Uniforme (*UCS - Uniform-Cost Search*)



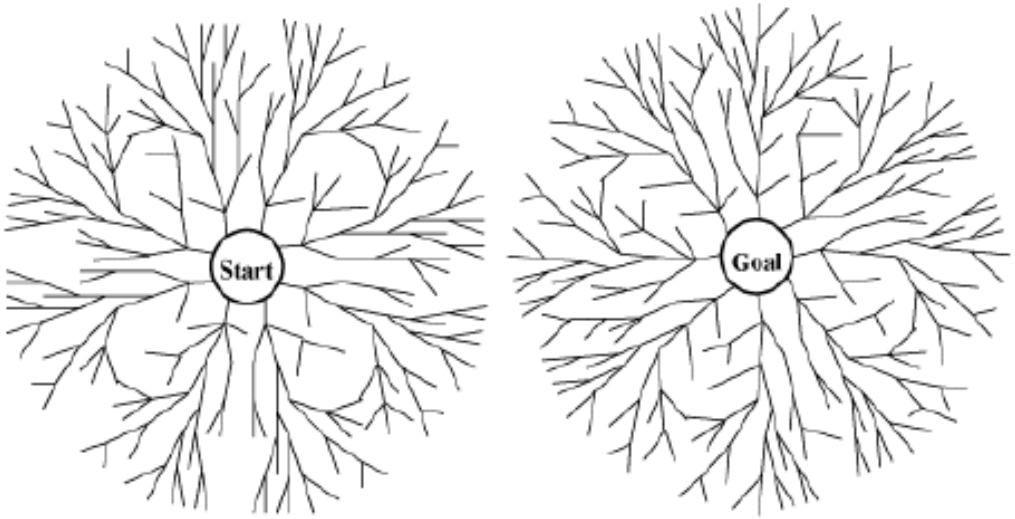
Busca em Profundidade Interativa (*IDS - Iterative Deepening Search*)

- Utiliza limites partindo de zero ate encontrar a primeira solução na profundidade *d*.
- Pode ser vista como uma combinação da busca em largura com a em profundidade
- Em geral é uma boa estratégia quando o espaço de estados é muito grande e a profundidade da solução é desconhecida.
- Complexidade espacial: O(bd)
- Complexidade temporal: $O(b^d)$


Iteration 0
S
Iteration 1
1 S
A
D
Iteration 2
1 S
2 A
3 D
B
D
A
E
Iteration 3
1 S
2 A
5 D
3 B
4 D
6 A
7 H
C
E
E
B
B
F
Iteration 4
1 S
2 A
8 D
3 B
6 D
9 A
11 E
4 C
5 H
7 H
10 B
12 B
13 F
F
G
D
F
B
F
C
E
A
C
C
Iteration 5
1 S
2 A
D
3 B
D
4 C
H
5 F
6 G
E

Busca Bidirecional (*Bidirectional Search*)

- Duas buscas simultâneas
  - Do nó (estado) inicial ao nó objetivo
  - Do nó objetivo ao nó inicial
- A busca para quando os nó a ser criado por uma busca se encontra na profundidade da outra.



Bibliografia

- Capítulo 2
  - Jones , M. Tim. Artificial Intelligence - A Systems Approach. Jones & Bartlett Publishers. 2007.
- Capítulo 3
  - Russell, Stuart. Artificial Intelligence: A Modern Approach. Prentice Hall. 2009.