INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL
22/23

CAP. 4 PESQUISA NÃO INFORMADA

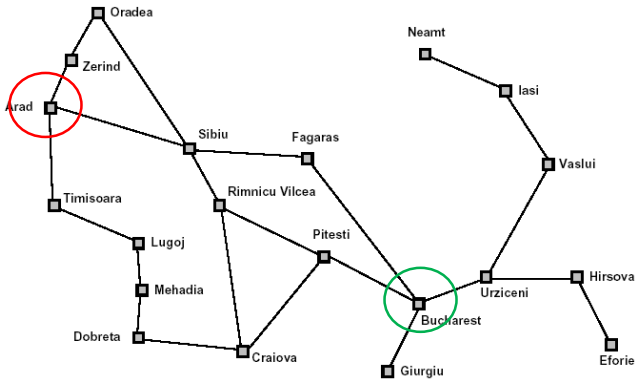
Carlos Pereira
ISEC

Índice

Estratégia de Pesquisa

3

- Encontrar um caminho de Arad para Bucharest:



Estratégia de Pesquisa

4

- …

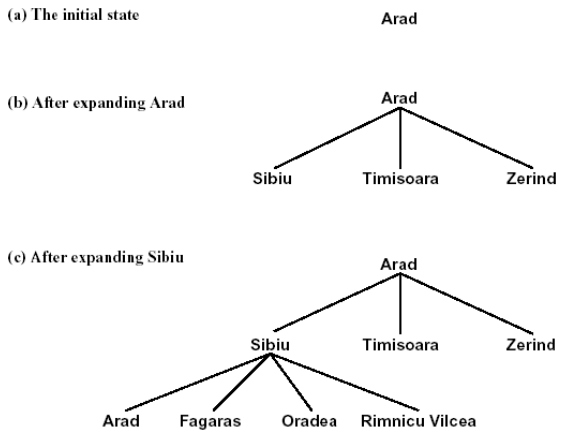
1. Estado Actual = Arad
2. Se o estado actual não é Bucharest
3. Aplicar os operadores ao estado para gerar um novo conjunto de estados (expansão)

   - na primeira iteração, obtêm-se os sucessores de Arad= {Zerind, Sibiu, Timisoara}
4. Escolher um destes estados
5. Voltar ao ponto 2

Estratégia de Pesquisa

5

- …



Estratégia de Pesquisa

6

- …
  - Uma pesquisa constrói uma **Árvore de Pesquisa** cuja raiz é o Estado Inicial.
    - Em cada iteração as folhas são nós sem sucessores, porque não foram ainda expandidos (ou já o foram mas não possuem sucessores).
    - Os nós ainda não expandidos chamam-se Nós Fronteira.
  - A ordem pela qual um conjunto de estados é expandido (ponto 4) define a **Estratégia da Pesquisa**.

Estratégia de Pesquisa

7

- Estratégias:
  - Critérios de Classificação
    - **Completude**: A estratégia garante que se encontra uma solução quando existe de facto uma?
    - **Optimização**: Encontra a melhor solução quando há várias possíveis?
    - **Complexidade Temporal**: Quanto tempo leva a encontrar uma solução?
    - **Complexidade Espacial**: Quais os requerimentos de memória?

Estratégia de Pesquisa

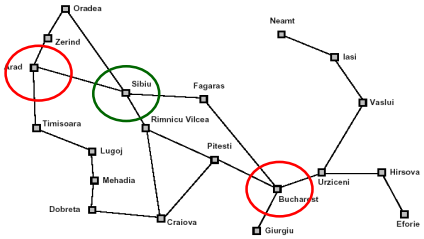
8

- …
  - Definem-se dois tipos de Pesquisa
    - Pesquisa Cega ou Não Informada
      - Procura uma solução sem recorrer a qualquer informação adicional que a guie.
      - Apenas pode comparar o estado actual com o objectivo
      - As variantes deste tipo de pesquisa variam de acordo com a ordem definida para expansão de estados
    - Pesquisa Heurística ou Informada
      - Procura uma solução recorrendo a informação adicional que permite escolher o nó a expandir primeiro

Estratégia de Pesquisa

9

- …
  - Exemplo de Pesquisa Informada:



  - Como Bucharest fica a sudeste de Arad, e dos sucessores de Arad apenas Sibiu se encontra nessa direcção, Sibiu poderia ser seleccionada para expansão em primeiro lugar.

Algoritmo Geral de Pesquisa – AGP

10

- Algoritmo

função pesquisa_geral (*problema, estratégia*):
devolve *solução ou falhanço*
Inicializa a raiz árvore de pesquisa com o estado inicial
Repete as seguintes acções
estado_actual ← Escolhe_estado (estratégia)
Se estado_actual = Ø Então
Devolve pesquisa falhou
Senão Se estado_actual = Objectivo Então
Devolve Solução
Senão
Expande estado_actual
Actualiza árvore de pesquisa
fim_Repete
fim_de_função

Algoritmo Geral de Pesquisa

11

- Implementação…
  - Uma árvore **Ar** de raiz = Estado Inicial regista os caminhos já gerados por aplicação dos operadores do problema aos nós que vão sendo expandidos;
  - Uma lista **NósPorExpandir** contém os nós fronteira (da árvore) a cada momento. A ordem pela qual os sucessores de um nó são inseridos nesta lista determina qual a variante do AGP que se está a considerar.
  - Uma lista **NósExpandido**s contém os nós que já foram expandidos. Esta lista evita que o mesmo nó seja expandido várias vezes, o que poderia originar loops infinitos (o mesmo nó pode ser atingido por vários caminhos)

Algoritmo Geral de Pesquisa

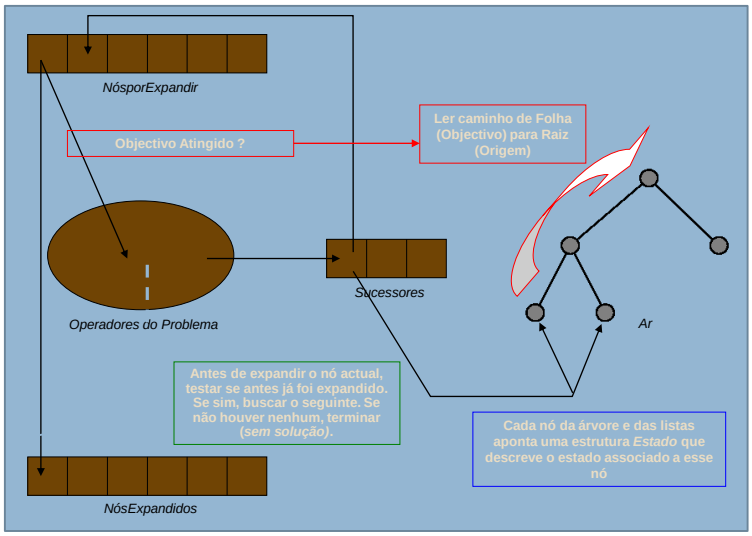
12

- …
  - Cada nó a expandir é removido da 1ª posição de NósPorExpandir e inserido em NósExpandidos. São-lhe aplicados os operadores do problema e gerada uma lista de **Sucessores**. Estes Sucessores são colocados em NósPorExpandir e simultaneamente inseridos em Ar.
  - Antes de expandir um nó, testa-se se esse nó é o objectivo. Se sim, termina.
    - **A solução encontra-se em Ar e pode ser obtida atravessando-a desde esse nó até à raiz.**

Algoritmo Geral de Pesquisa

13

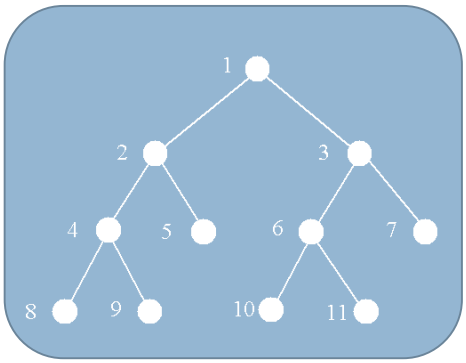
- …



Pesquisa em Largura

14

- Pesquisa em Largura (Breadth-First)
  - Estratégia
    - A partir da raiz, a árvore é expandida por níveis
    - Nós que se encontram a uma profundidade N são expandidos antes dos que se encontram a uma profundidade N+1



Pesquisa em Largura : Sequência de Expansão dos nós

Pesquisa em Largura

15

- ...

função pesquisa_em_largura(*problema, insere_fila):*

devolve *solução ou falhanço*

Fila_nos ← *cria_fila(estado_inicial)*

Repete as seguintes acções

Se fila_vazia(Fila_nos) Então

Devolve pesquisa falhou

estado_actual ← Retira_primeiro_fila(Fila_nos)

Se estado_actual = Objectivo Então

Devolve Solução

Senão

Insere_final(Fila_nos, expansão(estado_actual))

fim_Repete

fim_de_função

Pesquisa em Largura

16

- ...
  - Vantagens
    - Completa
      - porque procura todas as soluções possíveis e portanto encontrará uma caso exista
    - Óptima,
      - desde que o Custo do Caminho seja uma Função Não-Decrescente da profundidade dos nós.
        - A Pesquisa em Largura propõe sempre como solução a que tiver menor número de nós. Portanto, se o custo aumentar uniformemente com a profundidade, as soluções com menos nós representam menor custo.

Pesquisa em Largura

17

- …
  - Desvantagens:
    - O custo da pesquisa é muito elevado
      - Complexidade temporal exponencial
      - Complexidade espacial exponencial
    - Análise da Complexidade da Pesquisa em largura
      - A Pesquisa em Largura tem complexidade temporal e espacial $O(b^d)$, com b=Factor de Ramificação e d=Número de Níveis da Árvore.
        - Se considerarmos um factor de ramificação de 8, o número de nós expandidos é de $1+8+8^2+8^3+8^4+\ldots+8^k$

Pesquisa em Largura

18

- …
  - Para uma Pesquisa em Largura com branching factor (b) = 10 e Processamento executado a 1000 nós/segundo e ocupando 100 bytes/nó:

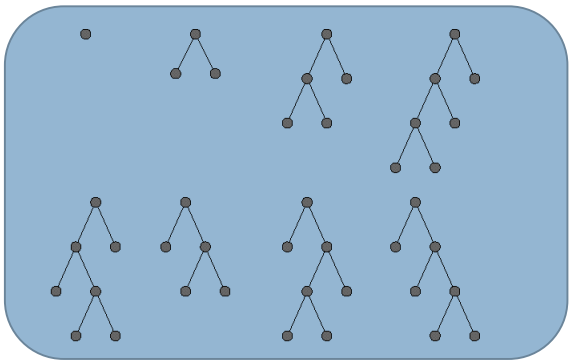
Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

    - Problemas de pesquisa cujos algoritmos têm complexidade exponencial, **apenas podem ser resolvidos para instâncias de pequena dimensão**

Pesquisa em Profundidade

19

- Pesquisa em Profundidade
  - cada nó é expandido até ser atingido o último nível da árvore, a menos que uma solução seja encontrada entretanto:



Pesquisa em Profundidade

20

- Características
  - Incompleta
    - no caso de a profundidade da árvore ser infinita (neste caso tentará atingir o último nível - que nunca alcança - e não retorna nenhuma solução.
  - Não Óptima
    - retorna uma solução qualquer e nenhuma condição pode garantir que seja a melhor.

Pesquisa em Profundidade

21

- …
  - Vantagens da Pesquisa em Profundidade

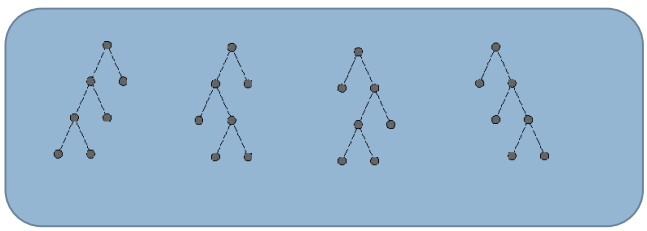
    Considere-se um Espaço de Estados com Factor de Ramificação "b" e profundidade máxima "d":

    - A Complexidade Temporal é $O(b^d)$ (como na Pesquisa em Largura, porque o número total de nós a gerar é o mesmo).
    - A Complexidade Espacial é de apenas O(b.d) porque não há necessidade de ter mais que b.d nós em memória simultaneamente - necessita de pouca memória!

Pesquisa em Profundidade

22

- …
  - Considerando b=2, d=3:
    - No máximo há em memória 2*3=6 nós + raiz



Pesquisa Uniforme

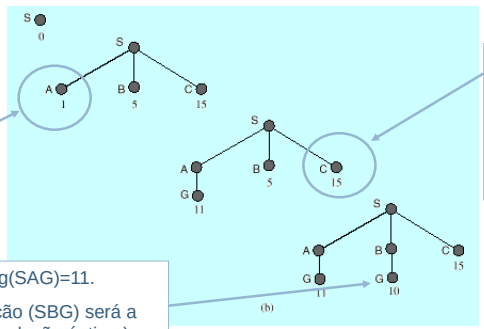
23

- Variante da Pesquisa em Largura
  - Consiste em expandir primeiro os nós que têm um custo associado menor
    - Esta expansão pára quando for encontrada uma solução e o custo acumulado dos caminhos associados aos nós que falta expandir já for superior à solução encontrada.
  - Garante a solução óptima, bastando apenas que o custo aumente com a profundidade
    - Comparativamente com a pesquisa em largura, resolve a limitação da solução óptima só ser garantida para custos que aumentam uniformemente com a profundidade de todos os caminhos.

Pesquisa Uniforme

24

- …
  - Seja "g" o custo do caminho percorrido e "G" o nó objectivo



Expandir primeiro (menor custo)

g(SC)=15 > 11,

Assim, o nó C não será expandido porque fornecerá sempre uma solução de custo maior que (SAG)

g(SBG)=10 < g(SAG)=11.

Assim, a solução (SBG) será a seleccionada (solução óptima)

Pesquisa Uniforme

25

- …
  - Características
    - Completa
    - Óptima
      - desde que o custo aumente com a profundidade: g(Sucessor(n)) ≥ g(n)
      - Se admitirmos que o custo possa diminuir com a profundidade, então seria preciso explorar a árvore toda para determinar qual o caminho óptimo!

Pesquisa Profundidade Limitada

26

- Resolve a limitação da Pesquisa em Profundidade de não retornar resultados em espaços de profundidade muito grande, impondo um limite, "m", à profundidade máxima a atingir.
  - Pode aplicar-se em espaços onde se sabe que a solução terá de existir dentro da profundidade "m"
    - Por exemplo, se um mapa contem 20 cidades, o caminho entre quaisquer duas tem de ser composto, no máximo, por 19. Logo, m=19.

Pesquisa Profundidade Limitada

27

- …
  - Características:
    - Completa
    - Não Óptima
    - Complexidade Temporal $O(b^m)$
    - Complexidade Espacial O(b.m)

IDS - Pesquisa por Aprofundamento Progressivo

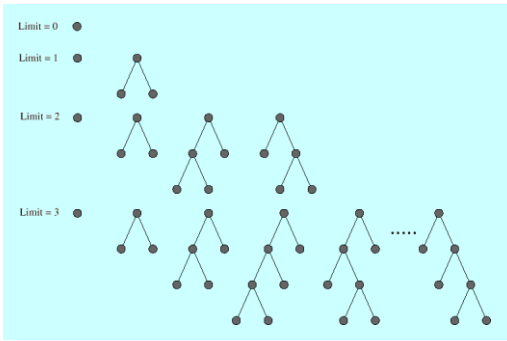
28

- A Pesquisa por Aprofundamento Progressivo (IDS - Iterative Deepening Search)
  - combina as Pesquisas em Largura e em Profundidade
  - evita a necessidade de se definir "m"antecipadamente:
    - Em vez de se estabelecer um só limite geral, começa por se estabelecer um limite inicial de profundidade = 0
    - Este limite vai-se alargando (1,2, 3,...m) para as iterações seguintes (i.e. faz-se uma pesquisa em profundidade de nível 1, depois 2, depois 3... mas para cada pesquisa reinicia-se o algoritmo da pesquisa em profundidade, desde a raíz)

IDS - Pesquisa por Aprofundamento Progressivo

29

- …



IDS - Pesquisa por Aprofundamento Progressivo

30

- …
  - Características
    - Óptima, nas condições da Pesquisa em Largura (custo = função da profundidade)
    - Completa, como a Pesquisa em Largura
    - Complexidade Espacial O(b.m) (como a Pesquisa em Profundidade)
    - Complexidade Temporal $O(b^m)$ (como a Pesquisa em Profundidade)