Agentes de Procura

Procura Cega

Uninformed Search

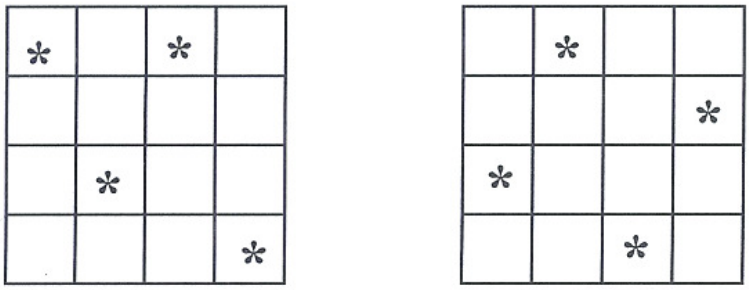
Capítulo 3:
Costa, E. e Simões, A. (2015). Inteligência Artificial – Fundamentos e Aplicações, 3.ª edição, FCA.

Agentes de Procura

- Vamos analisar diferentes estratégias possíveis para a resolução de problemas
- Os ambientes em que ocorrem os problemas são tipicamente acessíveis, geralmente deterministas e estáticos
- As diferentes estratégias vão ser agrupadas em função do grau de conhecimento que o agente tem sobre o domínio do problema
- Será feito um estudo da complexidade das diferentes propostas

Problemas, Estados, Operadores e Procura

Problema das N-Rainhas

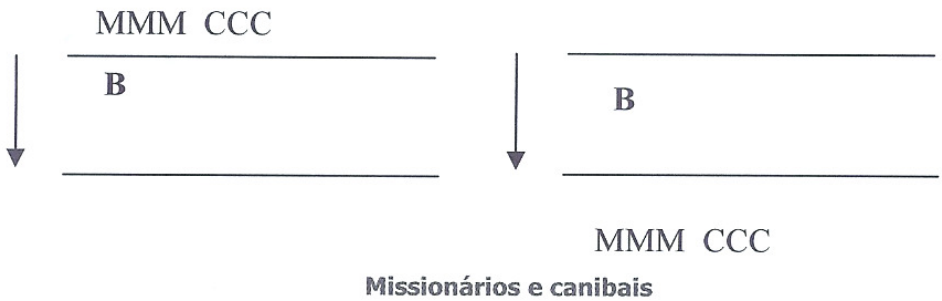


Problema das 4 rainhas

- O problema consiste em colocar N rainhas num tabuleiro de dimensão NxN de modo a que nenhuma possa atacar as restantes – a solução obriga a que apenas possa existir uma rainha em cada linha, coluna ou diagonal

Problemas, Estados, Operadores e Procura

Problema dos Missionários e Canibais



Missionários e canibais

- 3 missionários e 3 canibais querem atravessar com segurança um rio, dispondo para isso de um barco que pode transportar no máximo 2 pessoas
- Nunca pode existir, nas margens ou no barco, um número de canibais superior aos missionários, sob pena de serem comidos

Problemas, Estados, Operadores e Procura

Problema das Torres de Hanói

Torres de Hanói

- N discos diferentes estão colocados numa vara de acordo com o seu diâmetro
- Como transportar os discos, um a um, para uma de outras duas varas, de modo a que fiquem na mesma posição relativa, nunca podendo em momento algum existir um disco em cima de outro de diâmetro menor ?

Problemas, Estados, Operadores e Procura

Problema da Charada de 15

1	5	7	13
12	6	3	4
8		2	9
10	15	14	11

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Charada de 15

- Num quadro de dimensão 4x4 encontram-se colocadas peças quadradas numeradas de 1 a 15
- Existe 1 posição vazia
- Como movimentar os números, inicialmente dispostos aleatoriamente, de modo a que fiquem ordenados de acordo com o seu valor ?

Problemas, Estados, Operadores e Procura

Aspetos comuns a estes problemas:

- O agente é confrontado com uma situação, configuração ou estado inicial e uma configuração, situação ou estado final
- Existe um critério de reconhecimento de que o problema foi resolvido ou que o objetivo foi alcançado

Problemas, Estados, Operadores e Procura

Aspetos comuns a estes problemas:

- Associado à solução ou ao processo de gerar a solução existem restrições à *"ação do agente"*:

N-Rainhas:

- para estarmos perante uma solução, estas não se podem atacar mutuamente;

Missionários e canibais:

- o barco tem um limite de passageiros que pode transportar;
- nunca se pode criar uma situação na qual o número de canibais seja superior ao número de missionários

Problemas, Estados, Operadores e Procura

Aspetos em que estes problemas são diferentes:

- No problema das N-Rainhas estamos apenas interessados em encontrar a solução;
- No problema dos Missionários e canibais estamos interessados na sequência de ações que respeite as restrições e permita alcançar o estado objetivo;

Problemas, Estados, Operadores e Procura

Aspetos em que estes problemas são diferentes:

- Nalguns problemas apenas interessa encontrar uma solução, enquanto que noutros interessa encontrar a melhor solução;

- Se a tarefa for demonstrar teoremas, o importante é conseguir fazer a prova;

- Se a tarefa for encontrar uma solução para um caixeiro viajante que tem de passar por um conjunto de cidades e regressar à cidade de origem, interessa encontrar a solução mais económica.

Problemas, Estados, Operadores e Procura

Aspetos em que estes problemas são diferentes:

- Os problemas que descrevemos confrontam-se todos com um ambiente determinista. No entanto, os problemas de jogos como o xadrez ou as cartas, envolvendo adversários, introduzem uma componente de incerteza.

Problemas, Estados, Operadores e Procura

Como caracterizar um problema ?

- Um problema pode ser definido pelo conjunto possível das suas configurações ou estados
- Um desses estados é o estado inicial

  Em alguns problemas a situação inicial é sempre a mesma, enquanto que noutros casos pode ser qualquer estado admissível
- Outros são estados finais: aqueles que satisfazem o objetivo estipulado no enunciado do problema

Problemas, Estados, Operadores e Procura

Como caracterizar um problema ?

- Ao conjunto de todos os estados chamaremos espaço de estados ou espaço de procura
- Além dos estados, pode existir um conjunto de restrições que devem ser satisfeitas, quer para uma solução ser válida, quer para o processo de procura da solução ser válido
- Existe um conjunto de operadores que permitem atuar sobre os estados, alterando-os

Problemas, Estados, Operadores e Procura

Como caracterizar um problema ?

- As restrições e os operadores definem implicitamente ligações entre alguns estados

1	5	7	13
12	6	3	4
8		2	9
10	15	14	11

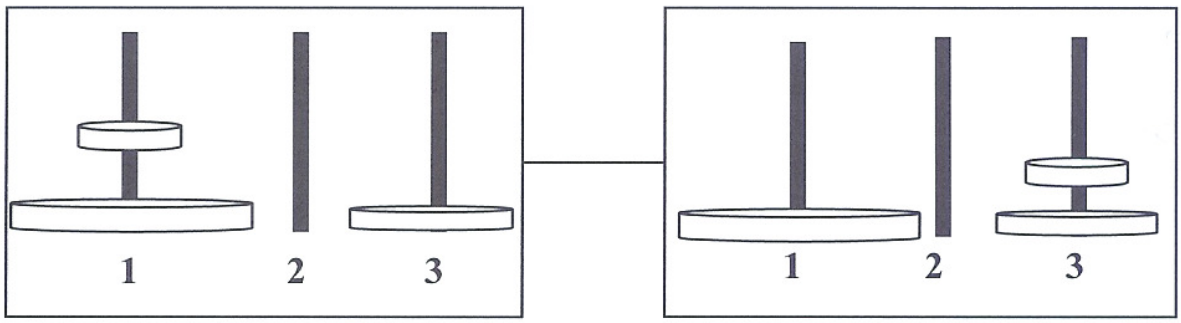
1	5	7	13
12	6	3	4
8	2		9
10	15	14	11

Estados associados na charada de 15

Problemas, Estados, Operadores e Procura

Como caracterizar um problema ?

- As restrições e os operadores definem implicitamente ligações entre alguns estados



Estados associados nas torres de Hanói

Problemas, Estados, Operadores e Procura

- O espaço de estados com as respetivas ligações pode ser visto como um grafo, por vezes chamado grafo do espaço de estados

Problemas, Estados, Operadores e Procura

Grafos – Linguagem e designações

- Um grafo é formado por um conjunto de nós (ou vértices) e por um conjunto de ligações entre nós;
- Ao conjunto de nós ligados a um dado nó chamaremos vizinhos desse nó;
- Quando as ligações têm uma orientação recebem o nome de arcos e os vizinhos são chamados de sucessores

Problemas, Estados, Operadores e Procura

Grafos – Linguagem e designações

- Os sucessores de um nó referem-se a ele como o seu pai;
- Um caminho é uma sequência de nós ligados entre si;
- Uma árvore é um grafo no qual existe um caminho único entre quaisquer par de nós;

Problemas, Estados, Operadores e Procura

Grafos – Linguagem e designações

- As árvores podem ver designado qualquer dos seus nós como raiz. Se as ligações tiverem uma orientação, então todos os caminhos da árvore têm origem na raiz;

Problemas, Estados, Operadores e Procura

Para que o agente possa resolver este tipo de problemas, tem de ter um conjunto de capacidades:

- Tem de ser capaz de percecionar os estados e construir uma representação interna (um modelo) desses estados;
- Tem de ter a capacidade de atuar sobre eles, de acordo com as regras do problema, provocando transições entre estados, isto é, tem de ter operadores de mudança de estado;
- Tem de ter a capacidade de reconhecer que alcançou um estado final.

Problemas, Estados, Operadores e Procura

Estratégia:

- A estratégia genérica que o agente adotará basear-se-á na navegação pelo espaço de estados, procurando um caminho que ligue o estado inicial a um estado final;
- Esta estratégia é conhecida pelo paradigma da procura no espaço de estados;
- Pode ser usada nos mais variados problemas.

Problemas, Estados, Operadores e Procura

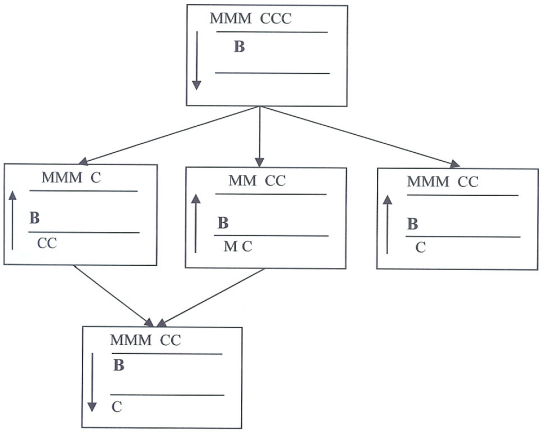
O agente vai procurar usar uma estratégia que satisfaça algumas características:

- Completa: se existir uma solução para o problema, ela será encontrada em tempo finito.
- Discriminadora (ou ótima): caso existam várias soluções, que seja encontrada a melhor.
- Económica: a solução é encontrada o mais rapidamente possível, ocupando o menor espaço de memória possível.

Problemas, Estados, Operadores e Procura

- Na maioria dos problemas não é possível ao agente adotar uma estratégia que envolva uma representação de todo o grafo dos estados e suas ligações.

Problemas, Estados, Operadores e Procura



Missionários e canibais: grafo parcial do espaço de estados

- Na figura encontra-se o grafo parcial do espaço de estados do problema dos Missionários e canibais, a partir do estado inicial.
- As setas indicam o sentido da procura

Problemas, Estados, Operadores e Procura

Problema: Procura de caminho entre 2 cidades.

- Problema de referência;
- Neste problema um estado coincide com uma cidade;
- A associação entre estados será dada pela existência ou não de uma ligação direta por estrada entre duas cidades.

As estradas entre as capitais de distrito

- Existe apenas um operador que corresponde à travessia de uma estrada entre duas cidades vizinhas

Problemas, Estados, Operadores e Procura

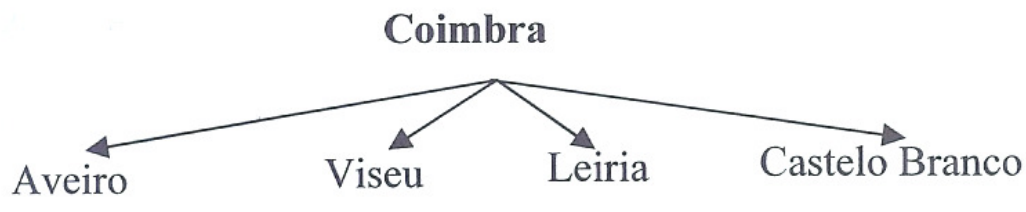
- O modelo abstrato será um grafo em que cada nó representa uma cidade enquanto que uma ligação coincide com uma estrada

AVEIRO	Porto (68)	Viseu (95)	Coimbra (68)	Leiria (115)
BRAGA	Viana C.(48)	V. Real (106)	Porto (53)	
BRAGANÇA	V. Real (137)	Guarda (202)		
BEJA	Évora (78)	Faro (152)	Setúbal (142)	
C. BRANCO	Coimbra (159)	Guarda (106)	Portalegre (80)	Évora (203)
COIMBRA	Viseu (96)	Leiria (67)		
ÉVORA	Lisboa (150)	Santarém (117)	Portalegre (131)	Setúbal (103)
FARO	Setúbal (249)			
GUARDA	V. Real (157)	Viseu (85)		
LEIRIA	Lisboa (129)	Santarém (70)		
LISBOA	Santarém (78)	Setúbal (50)		
PORTO	V. Castelo (71)	V. Real (116)	Viseu (133)	
V. REAL	Viseu (110)			

Distâncias quilométricas entre cidades portuguesas

Problemas, Estados, Operadores e Procura

- Considerar que problema consiste em encontrar um caminho entre Coimbra e Faro
- Utilizando uma estratégia baseada no paradigma da procura num espaço de estados, o agente irá construir um grafo parcial
- Inicialmente poderá seguir em direção a Aveiro, Viseu, Leiria ou Castelo Branco



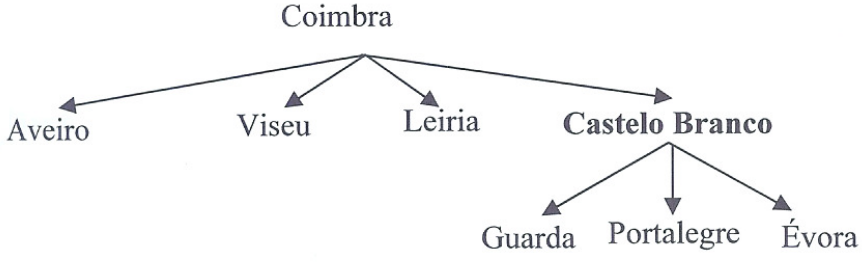
Primeira expansão a partir de Coimbra

Problemas, Estados, Operadores e Procura

- Este processo de determinar os vizinhos, ou sucessores, de uma cidade (nó) ainda não considerados designa-se por expansão (do nó)

Problemas, Estados, Operadores e Procura

- Faz parte da estratégia do agente o modo como escolhe o caminho a seguir entre as várias alternativas possíveis
- Admitamos que opta por Castelo Branco. As novas direções a tomar a partir daí são Guarda, Portalegre e Évora (e Coimbra ?)



Expansão a partir de Castelo Branco

Problemas, Estados, Operadores e Procura

- O processo pode ser iterado até que a estratégia encontre um caminho que ligue Coimbra a Faro, passando por Castelo Branco, Évora e Beja

Problemas, Estados, Operadores e Procura

Comentários

- A escolha *"primeiro por Castelo Branco e depois por Portalegre"* não foi determinada por nada;
- Se o objetivo fosse percorrer o menor número de quilómetros, uma possibilidade seria considerar a cada momento a cidade mais próxima, de entre as vizinhas;
- Mas, em geral, esta forma de proceder não garante que se encontre a melhor solução;
- Podia acontecer que Portalegre apenas estivesse ligada por estrada a Castelo Branco. Nesse caso, seria necessário voltar atrás e escolher, por exemplo, Guarda;

Problemas, Estados, Operadores e Procura

Comentários

- Mas qualquer uma das outras cidades (Aveiro, Viseu e Leiria) devem ser consideradas como alternativas;
- No caso de a estratégia apenas considerar como escolha uma cidade vizinha, dizemos que se trata de uma estratégia local;
- Caso todas as cidades ainda não visitadas sejam uma possibilidade de escolha, dizemos que a nossa estratégia é global.

Problemas, Estados, Operadores e Procura

Comentários

- Em cada instante haverá um conjunto de cidades já visitadas e outras cuja ligação a Coimbra (neste exemplo) – direta ou indiretamente – é conhecida mas que ainda não foram exploradas;
- A estratégia é que indica a escolha a fazer;

Problemas, Estados, Operadores e Procura

Comentários

- Se não existir a possibilidade de, a partir de uma cidade, chegar a outra por caminhos distintos, então em vez de um grafo de procura, teremos uma árvore de procura;
- Mesmo quando tal não acontece, o grafo pode ser transformado numa árvore cuja raiz é a cidade de partida a que se associam os diferentes caminhos possíveis;
- Aparecerão na árvore cidades duplicadas;

Problemas, Estados, Operadores e Procura

Comentários

- Em cada momento as cidades conhecidas mas ainda não visitadas formam a fronteira do espaço de estados;
- A construção de uma estratégia que seja simultaneamente completa, económica e discriminadora é complexa;

Problemas, Estados, Operadores e Procura

- Estrutura geral do algoritmo que implementa qualquer estratégia baseada no paradigma da procura num espaço de estados:

Função Procura Geral (problema, *estratégia*): solução ou falha
1. Inicializa a árvore de procura com o estado (nó) inicial do problema
2. Repete
2.1. Se não há candidatos para expandir, **Então**
2.1.1. Devolve falha
Fim_de_Se
2.2. Escolhe um nó na fronteira para expansão, de acordo com a *estratégia*
2.3. Se o nó contém o objectivo **Então**
2.3.1. Devolve a solução correspondente
Senão
2.3.2. Expande o nó e acrescenta à árvore de procura os seus sucessores, de acordo com a *estratégia*
Fim_de_Se
Fim_de_Repete
Fim_de_Função

Algoritmo geral de procura

Procura Cega

- Quando não existe informação sobre o problema que nos permita ajudar no processo de geração e teste dos nós, teremos de nos socorrer de uma estratégia dita cega

- Estas estratégias definem um modo sistemático de navegar na árvore de procura

Procura Cega

Existem fundamentalmente 2 possibilidades, cada uma com variantes:

- Analisar repetidamente um sucessor do último nó analisado (profundidade primeiro);
- Analisar todos os sucessores de um dado nó antes de passar para a análise de todos os sucessores dos sucessores (largura primeiro).

Procura em Largura Primeiro

Breadth-First Search

- Numa árvore de procura cada nó tem um nível
- A raiz tem nível 0, os seus sucessores têm nível 1, os sucessores dos sucessores têm nível 2, e assim sucessivamente.
- A estratégia de procura em largura primeiro caracteriza-se pelo facto de os nós do nível n serem todos analisados antes dos nós do nível n+1

Procura em Largura Primeiro

Para o problema de encontrar o caminho de Coimbra para Faro:

- A árvore de procura irá passando sucessivamente pelas etapas seguintes

Coimbra

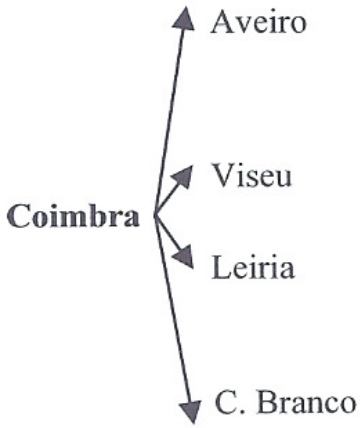
Árvore de procura contendo apenas o nível 0

Procura em Largura Primeiro

- Inicialmente a árvore de procura terá apenas um nó correspondente ao estado inicial (Coimbra)

- Como Coimbra não corresponde ao objetivo (Faro), o nó correspondente irá ser expandido, aparecendo todos os nós do nível seguinte (e que são os vizinhos de Coimbra)

Procura em Largura Primeiro

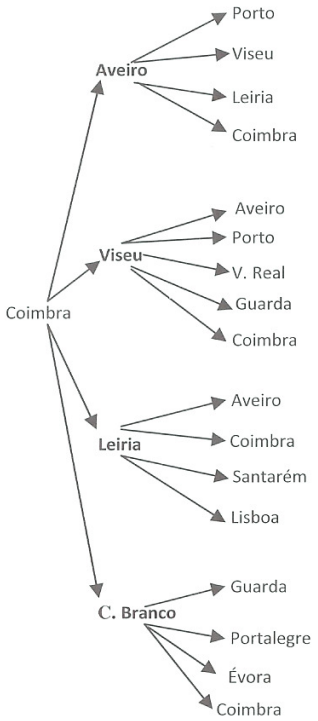


Árvore de procura após a expansão do nó do nível 0

Procura em Largura Primeiro

- O algoritmo em largura primeiro irá analisar sucessivamente estes nós
- Como nenhum deles é a solução, o algoritmo passa a analisar os nós do nível 2

Procura em Largura Primeiro



Árvore de procura com todos os nós até ao nível 2

Procura em Largura Primeiro

- O processo de expansão e análise continuaria até se alcançar o objetivo Faro
- Do ponto de vista algorítmico, se estivermos a resolver um problema em que apenas interessa a solução, então não será necessário manter guardada toda a árvore de procura, mas sim a sua fronteira
- Caso nos interesse o caminho, como neste exemplo, teremos de manter todos os caminhos que ligam a raiz da árvore aos nós na fronteira

Procura em Largura Primeiro

- A fronteira deverá ser guardada numa fila (estrutura de dados na qual os primeiros elementos a serem armazenados são os primeiros a sair – FIFO)
- Os elementos à cabeça da fila serão os primeiros a ser analisados
- Os sucessores de um nó, gerados pela estratégia, serão introduzidos no fim da fila

Procura em Largura Primeiro

Algoritmo de procura em largura primeiro

```
Função ProcuraLarguraPrimeiro (problema, InsereFila): solução ou falha
   1. l_nós ← Faz_Fila (EstadoInicial (problema))
   2. Repete
         2.1. Se VaziaFila(l_nós) Então
                 2.1.1. Devolve falha
              Fim_de_Se
         2.2. nó ← RetiraFila(l_nós)
         2.3. Se TesteObjectivo(nó) Então
                 2.3.1. Devolve nó
              Senão
                 2.3.2. InsereFila(l_nós, Expansão(nó,Operadores(problema)))
              Fim_de_Se
      Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura em largura primeiro

Procura em Largura Primeiro

- Vejamos como a variável l_nós evolui ao longo do tempo (contém em cada momento os nós expandidos mas ainda não analisados)

ITERAÇÃO	L_NÓS
0	[Coimbra]
1	[Aveiro, Viseu, Leiria, C. Branco]
2	[Viseu, Leiria, C. Branco, Porto, Viseu, Leiria, Coimbra]
3	[Leiria, C. Branco, Porto, Viseu, Leiria, Coimbra, Aveiro, Porto, V. Real, Guarda, Coimbra]

A evolução do conteúdo de l-nós

Procura em Largura Primeiro

- Os nós de um dado nível não são gerados todos na mesma iteração;
- Aparecem nós já anteriormente expandidos e analisados (possibilidade de existirem ciclos)
- O crescimento do número de nós na lista de nós não é linear

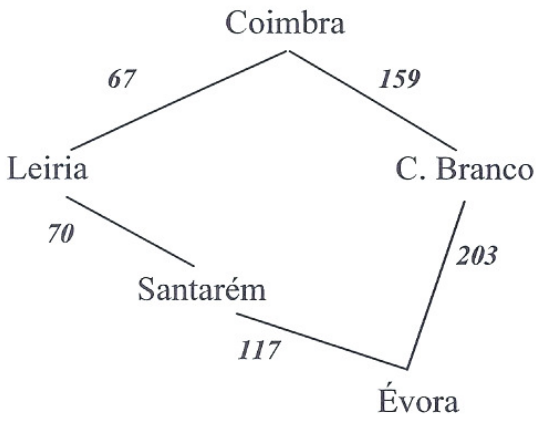
Procura em Largura Primeiro

Este Algoritmo:

- É Completo: se existir uma solução, mais tarde ou mais cedo ela será encontrada;
- É Discriminador ?
- Se todas as ligações tiverem o mesmo custo (neste exemplo, se a distância entre 2 quaisquer cidades for a mesma), então a solução encontrada será a melhor (menor quilometragem percorrida) pois o factor determinante será o nível a que foi encontrada a solução (será impossível existir uma solução melhor a um nível mais baixo)

Procura em Largura Primeiro

- Mas, no caso geral, esta estratégia não garante que se encontre sempre a melhor solução
- Exemplo:



De Coimbra a Évora por caminhos de comprimento diferente

Procura em Largura Primeiro

- Existe uma solução no nível 2 que envolve o caminho

  Coimbra – Castelo Branco – Évora

  de custo 362.

- No entanto, existe uma solução no nível 3

  Coimbra – Leiria – Santarém – Évora

  cujo custo é apenas 254

Procura em Largura Primeiro

- É Económico ?

Esta análise desdobra-se em 2 componentes:

- uma análise do tempo de execução do algoritmo, e
- Uma análise do espaço de memória que é necessário.

Procura em Largura Primeiro

- No 1.º caso (tempo de execução) utilizamos como referência o número de nós analisados até encontrar a solução;
- No 2.º (espaço de memória), vamos considerar o número de nós armazenados em cada instante até se encontrar a solução.

Procura em Largura Primeiro

Vamos considerar que:

- Todos os nós têm o mesmo número de sucessores: fator de ramificação, r;
- n representa o nível de um nó.

Procura em Largura Primeiro

Análise temporal

- Admitamos que a solução se encontra no nível n;
- Todos os nós dos níveis anteriores tiveram de ser analisados;
- No nível n a solução pode estar entre as "posições" 1 e r. No pior caso a solução encontrar-se-á no último nó do nível;
- No nível k existem r^k nós.

Procura em Largura Primeiro

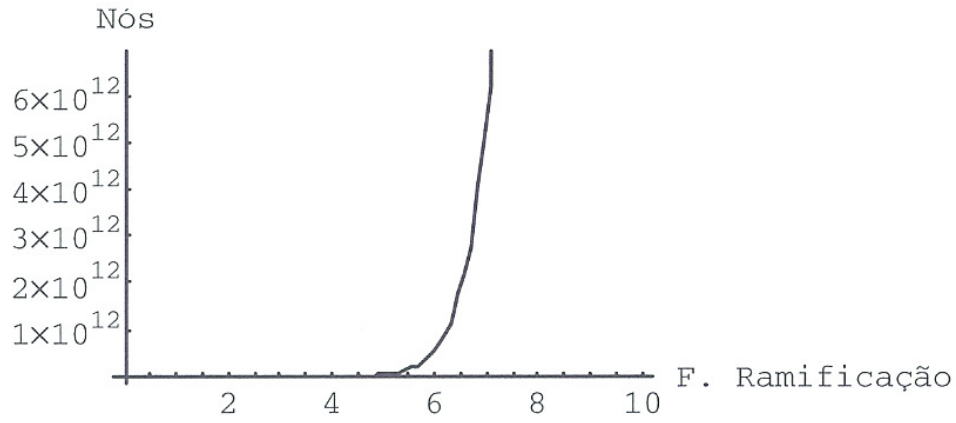
- Assim,

$$1 + r + r^2 + r^3 + \ldots + r^n = \sum_{k=0}^{n} r^k = \frac{r^{n+1} - 1}{r - 1}$$

Número de nós analisados na procura em largura primeiro

Procura em Largura Primeiro

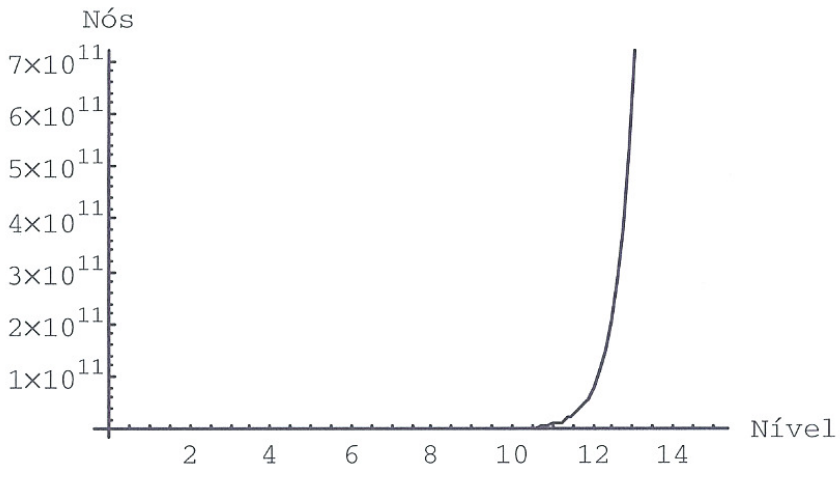
- O número de nós analisados cresce exponencialmente com o fator de ramificação, para um dado nível



Número de nós analisados em função do factor de ramificação (n = 15)

Procura em Largura Primeiro

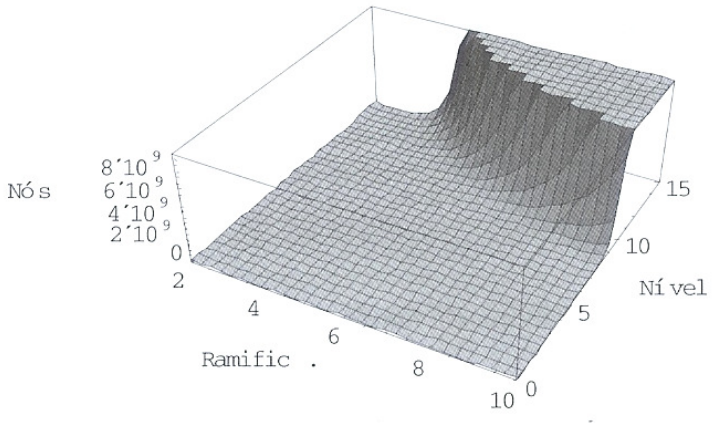
- O mesmo comportamento (forte crescimento exponencial) pode ser observado analisando a variação do n.º de nós visitados em função do nível a que se encontra a solução (fator de ramificação fixado no valor 8)



Variação do número de nós visitados em função do nível (r = 8)

Procura em Largura Primeiro

- Se considerarmos o comportamento do n.º de nós analisados para diferentes valores de r e n conjugados, uma vez mais se verifica o efeito de forte crescimento exponencial



Número de nós visitados em função do factor de ramificação e do nível

Procura em Largura Primeiro

- Podiamos ter chegado a estas conclusões através de uma análise da complexidade assimptótica.
- Para valores elevados de r e n, o n.º de nós analisados pode ser aproximado por

$$\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \approx r^n = \mathrm{O}(r^n)$$

Complexidade temporal assimptótica

Procura em Largura Primeiro

- Análise da complexidade espacial (memória ocupada)
- Para analisar os nós de um dado nível, o algoritmo terá de os manter todos em memória;
- Como no nível n existem r^n nós, a complexidade assimptótica será também $O(r^n)$.

Procura em Largura Primeiro

- Os problemas que apresentam uma complexidade exponencial podem ser resolvidos por métodos convencionais, em particular os métodos cegos, apenas para pequenos valores de r e n.

Procura em Largura Primeiro

- As dificuldades deste método estão patentes no seguinte quadro
- Fator de ramificação: 8
- 1 nó é processado num milissegundo (10^{-3}s)
- Espaço em memória ocupado por 1 nó: 10 Bytes

Nível	Nós	Tempo	Espaço
0	1	1 ms	10 *bytes*
5	4681	4,68 s	45 *Kbytes*
10	$153*10^6$	1,9 dias	1,5 *Gbytes*
15	$5*10^{12}$	175 anos	50 *Tbytes*

Tempo e espaço de memória em função do nível (r = 8, 1 ms por nó, 10 *bytes* por nó)

Procura de Custo Uniforme

Uniform-Cost Search

- Pode existir um custo associado às transições de estados
- No problema das cidades, esse custo será dado pela distância quilométrica entre duas cidades
- Ao aplicar uma sequência de operadores desde o nó inicial até um dado nó k, o custo do caminho correspondente será definido pela soma dos custos parciais
- Designemos g(k) a função que nos dá o custo associado ao caminho que nos leva ao nó k

Procura de Custo Uniforme

- O algoritmo de procura de custo uniforme é uma variante do algoritmo de procura em largura primeiro, que escolhe para expansão a cada momento o nó pertencente à fronteira cujo custo é o mais pequeno
- Se g(k)=nível(k) então o algoritmo de custo uniforme reduz-se ao algoritmo de procura em largura primeiro

Procura de Custo Uniforme

Algoritmo de procura de custo uniforme

```
Função ProcuraCustoUniforme (problema, InsereOrdem_Fila): solução ou falha
  1. l_nós ← FazFila (EstadoInicial (problema))
  2. Repete
       2.1. Se VaziaFila(l_nós) Então
              2.1.1. Devolve falha
            Fim_de_Se
       2.2. nó ← RetiraFila(l_nós)
       2.3. Se TesteObjectivo(nó) Então
              2.3.1. Devolve nó
            Senão
              2.3.2. InsereOrdemFila(l_nós, Expansão(nó, Operadores(problema)))
            Fim_de_Se
     Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura custo uniforme

Procura de Custo Uniforme

- A única diferença neste algoritmo reside no facto de a inserção na fila ser feita de forma ordenada
- Os elementos de menor custo são agora colocados à cabeça da fila

Procura de Custo Uniforme

- Evolução do conteúdo da variável l_nós

ITERAÇÃO	L_NÓS
0	**[[Coimbra, 0]]**
1	**[[Leiria, 67], [Aveiro, 68], [Viseu, 96], [C. Branco, 159]]**
2	[[Aveiro, 68], [Viseu, 96], **[Coimbra, 134]**, **[Santarém, 137]**, [C. Branco, 159], **[Aveiro, 182]**, **[Lisboa, 196]**]
3	[[Viseu, 96], [Coimbra, 134], **[Porto, 136]**, **[Coimbra, 136]**, [Santarém, 137], [C. Branco, 159], **[Viseu, 163]**, [Aveiro, 182], **[Leiria, 183]**, [Lisboa, 196]]

A evolução do conteúdo de l_nós

Os valores indicados correspondem ao custo dado pela função g(k) – a negrito indicam-se os novos caminhos introduzidos em cada etapa

Procura de Custo Uniforme

- Mantendo o teste e expansão do elemento à cabeça da fila, vemos que a ordem de análise irá ser distinta da procura em largura primeiro
- Existe uma condição importante da função de custo g(k) que permite ao algoritmo apresentar boas propriedades:

$$g(sucessor(k)) >= g(k)$$

O custo de um caminho não pode diminuir à medida que se desce de nível

Procura de Custo Uniforme

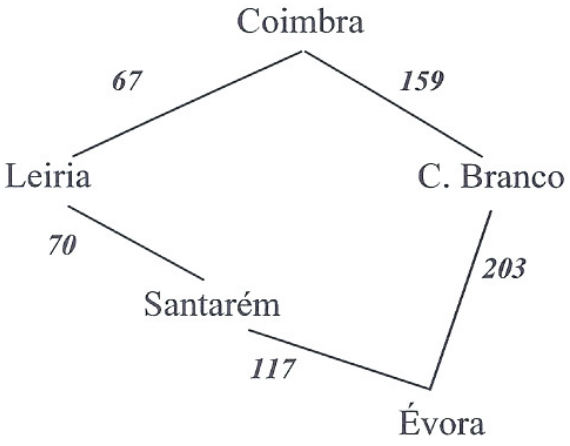
Assim,

- O algoritmo é completo: todos os nós irão ser analisados de forma sistemática até ser encontrada a solução;

- O algoritmo é discriminador: encontra sempre a solução de menor custo.

Procura de Custo Uniforme

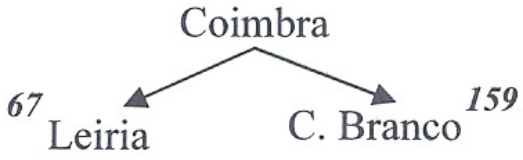
- Exemplo: Ir de Coimbra para Évora



De Coimbra a Évora por caminhos de comprimento diferente

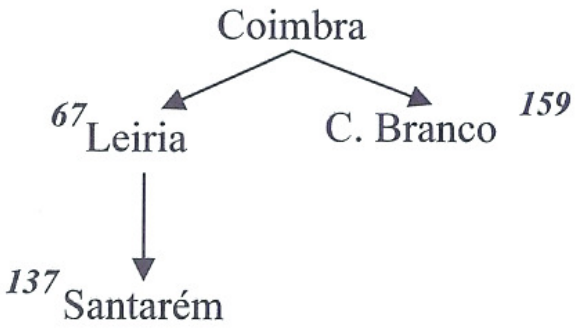
Procura de Custo Uniforme

- Evolução da árvore de procura (simplificada)



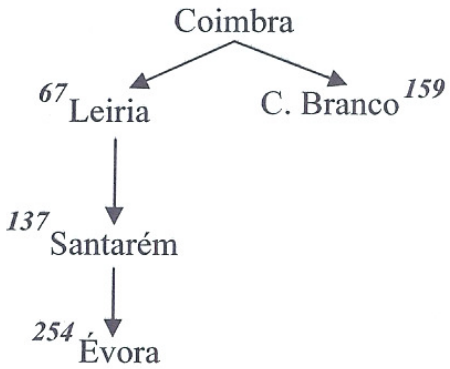
- Numa 1.ª fase são expandidos Leiria e Castelo Branco, optando-se por Leiria por ter o menor custo

Procura de Custo Uniforme



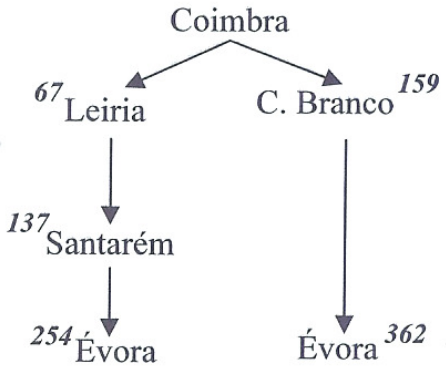
- Como Leiria não é solução, é expandido, dando origem a Santarém
- Apesar de estar num nível mais baixo do que Castelo Branco, Santarém apresenta um custo menor pelo que vai ser testado primeiro

Procura de Custo Uniforme



- Como Santarém não é solução, é expandido, dando origem a Évora, o nosso objetivo, com um custo 254
- Como existe um caminho de custo menor (Castelo Branco, 159), é esse a ser testado primeiro, e não Évora!

Procura de Custo Uniforme



- Como Castelo Branco não é solução, é expandido, dando origem a Évora, agora com um custo 362
- A fila é composta por 2 caminhos que levam a Évora. O 1.º a ser testado é o de menor custo

Procura de Custo Uniforme

Vertente económica do algoritmo:

- Este algoritmo tem características semelhantes à procura em largura primeiro, pelo que apresentará complexidade temporal e espacial exponencial, $O(r^n)$.

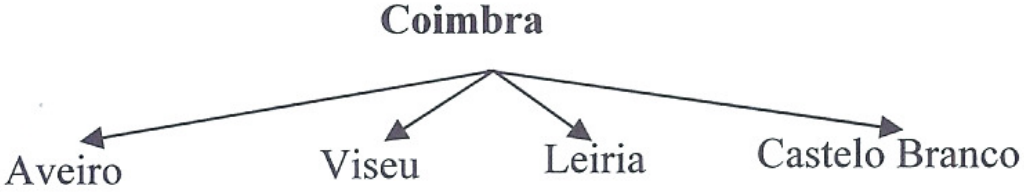
Procura em Profundidade Primeiro

Depth-First Search

- Partindo da raiz, o algoritmo vai expandindo um nó, escolher um dos seus sucessores, expandi-lo por sua vez, continuando o processo até que ou encontra a solução ou o nó não possa ser mais expandido.
- Neste último caso, o algoritmo continua o processo com um irmão do último nó analisado, caso exista, ou regressa ao nível anterior para continuar o processo.

Procura em Profundidade Primeiro

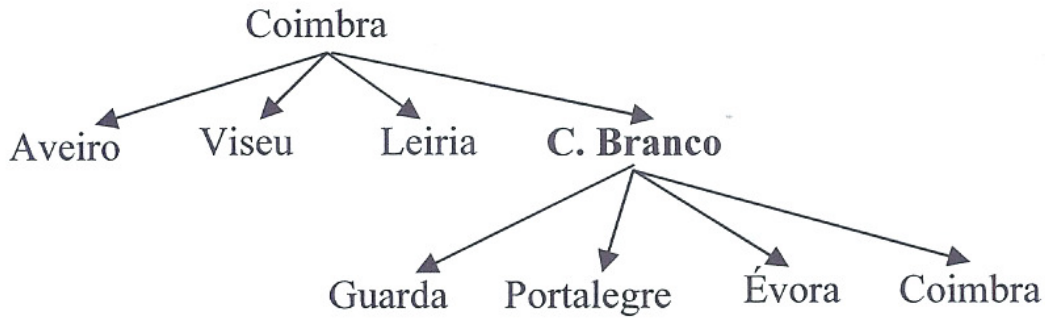
- Exemplo: Viagem de Coimbra a Faro
- Começamos por Coimbra, que não sendo a solução, vai originar a expansão do nó:



Árvore de procura após expandir Coimbra

Procura em Profundidade Primeiro

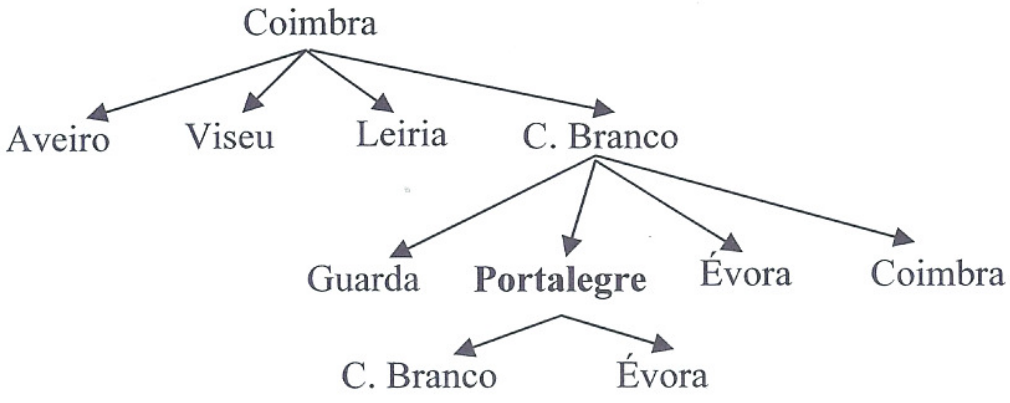
- Consideremos que o algoritmo opta por Castelo Branco.
- O facto de não ser solução vai originar a sua expansão



Árvore de procura depois de expandir C. Branco

Procura em Profundidade Primeiro

- Suponhamos que a análise se faz com Portalegre, que não sendo solução, origina a sua expansão, crescendo a árvore mais um nível



Árvore de procura depois da escolha e expansão de Portalegre

Procura em Profundidade Primeiro

- Para garantir que o algoritmo se comporta conforme indicado, basta que a fronteira da árvore de procura seja guardada numa pilha (estrutura de dados na qual os últimos elementos a serem armazenados são os primeiros a serem processados – LIFO)

```
Função ProcuraProfundidadePrimeiro (problema, InserePilha): solução ou falha
    1. l_nós ← FazPilha (EstadoInicial (problema))
    2. Repete
        2.1. Se VaziaPilha(l_nós) Então
            2.1.1. Devolve falha
        Fim_de_Se
        2.2. nó ← RetiraPilha(l_nós)
        2.3. Se TesteObjectivo(nó) Então
            2.3.1. Devolve nó
        Senão
            2.3.2. InserePilha(l_nós, Expansão(nó, Operadores(problema)))
        Fim_de_Se
    Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura em profundidade primeiro

Procura em Profundidade Primeiro

- Vejamos como evolui o conteúdo da variável l_nós ao longo das primeiras iterações do algoritmo:

ITERAÇÃO	L_NÓS
0	[*Coimbra*]
1	[*C. Branco*, Aveiro, Viseu, Leiria,]
2	[*Portalegre*, Évora, Guarda, Coimbra, Aveiro, Viseu, Leiria]
3	[*Évora*, C. Branco, Évora, Guarda, Coimbra, Aveiro, Viseu, Leiria]

O conteúdo da pilha ao longo das primeiras iterações do algoritmo

Procura em Profundidade Primeiro

- Os nós dos níveis mais pequenos que não são escolhidos vão ficando para trás na pilha;
- Por isso, são analisados posteriormente àqueles que se encontram mais fundo na árvore de procura;
- Existe a possibilidade de existência de ciclos;
- O número de nós na pilha aparenta não crescer tão rapidamente como no caso das estratégias em largura primeiro.

Procura em Profundidade Primeiro

- Esta estratégia não é completa
- Não é completa mesmo que o espaço de procura seja finito
- Basta que exista um ciclo no caminho de aprofundamento escolhido, para que não mais termine.
- Exemplo: se o sucessor de Castelo Branco escolhido fosse Coimbra, entraríamos num ciclo infinito que gerava o caminho

Coimbra – Castelo Branco – Coimbra – Castelo Branco - …

Procura em Profundidade Primeiro

- Esta estratégia não é discriminadora
- Podem existir 2 soluções, uma num nível alto e outra num nível baixo, podendo o algoritmo seguir pelo caminho que leva à primeira (que não é a ótima);
- Mesmo que os custos associados aos caminhos sejam não negativos, a solução de menor custo também não será encontrada;
- Exemplo: Para encontrar o caminho entre Coimbra e Castelo Branco, a opção pode ser o caminho

  Coimbra – Viseu – Guarda – Castelo Branco (287 Km)

  em vez de se optar pelo caminho direto (159 Km)

Procura em Profundidade Primeiro

Vertente Económica

- Qual a complexidade temporal do algoritmo ?
- Consideremos um fator de ramificação constante e igual a r;
- Quantos nós são analisados quando estamos no nível n ?

Procura em Profundidade Primeiro

- No melhor dos casos, se a solução estiver no ramo mais à esquerda no nível n, então serão analisados n+1 nós;
- Se a solução estiver no ramo mais à direita e no último nível, teremos o caso menos favorável, semelhante à procura em largura primeiro;
- O valor concreto será um valor entre estes dois extremos. Consideremos o valor médio:

$$\frac{\frac{r^{n+1}-1}{r-1}+(n+1)}{2}=\frac{r^{n+1}+rn-n+r-2}{2(r-1)}$$

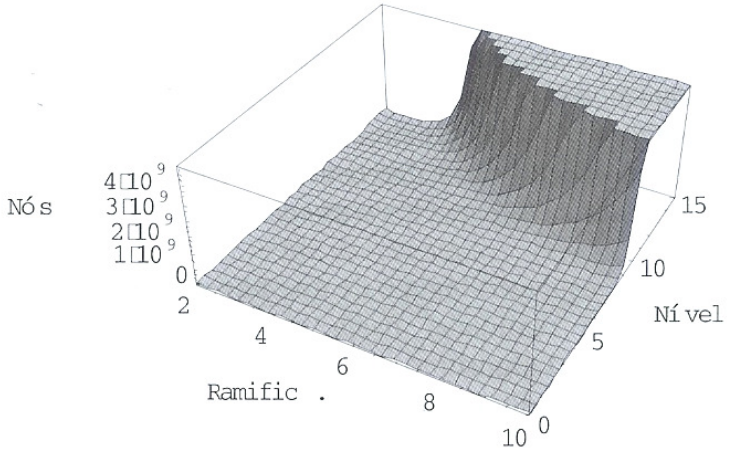
Complexidade temporal: valor médio

Procura em Profundidade Primeiro

- Se considerarmos a complexidade assimptótica, ou seja, para valores de r e n bastante grandes, teremos $O(r^n)$ como valor para a complexidade

Procura em Profundidade Primeiro

- O comportamento exponencial da complexidade temporal é idêntico ao da procura em largura primeiro, embora os valores sejam um pouco menores:

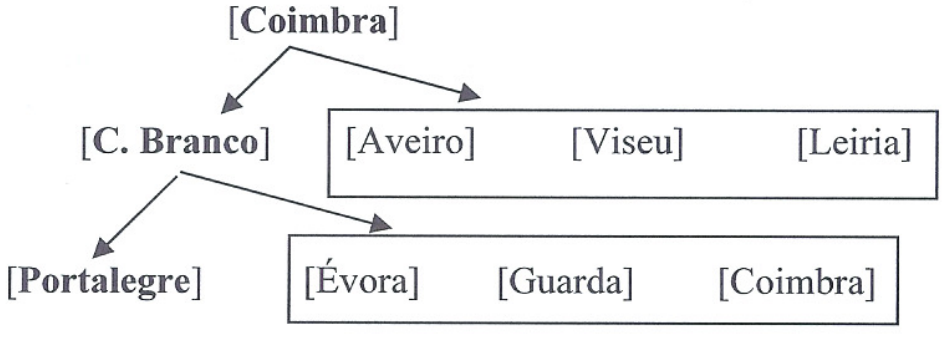


Complexidade temporal

Procura em Profundidade Primeiro

Complexidade espacial (memória ocupada)

- Cálculo do máximo de memória necessária quando a solução se encontra no nível n. Vejamos o exemplo da figura



Árvore de procura em profundidade primeiro

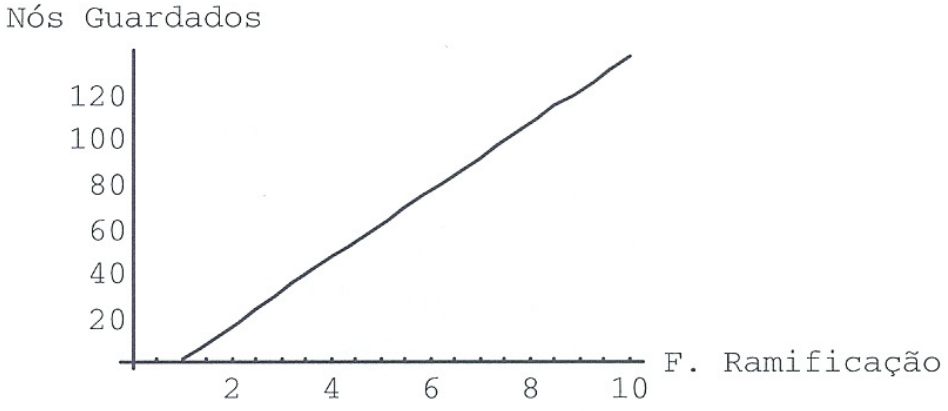
Procura em Profundidade Primeiro

- Quando estamos para analisar Portalegre, temos guardado em memória os 3 sucessores de Coimbra, mais os 3 sucessores de Castelo Branco, mais Portalegre
- Generalizando, para um fator de ramificação constante, teremos, no caso de a solução se encontrar no nível n, um valor máximo de nós guardados de

$$n\,(r-1) + 1$$

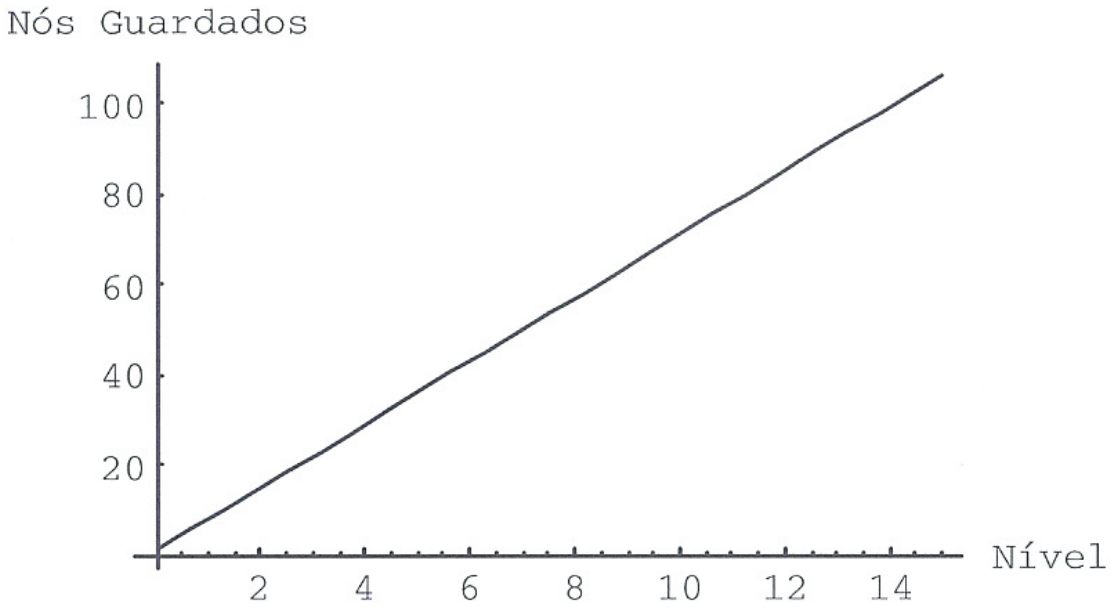
Procura em Profundidade Primeiro

- Para valores elevados de n e r a complexidade assimptótica será da ordem de $O(n \times r)$, ou seja, linear com o nível.



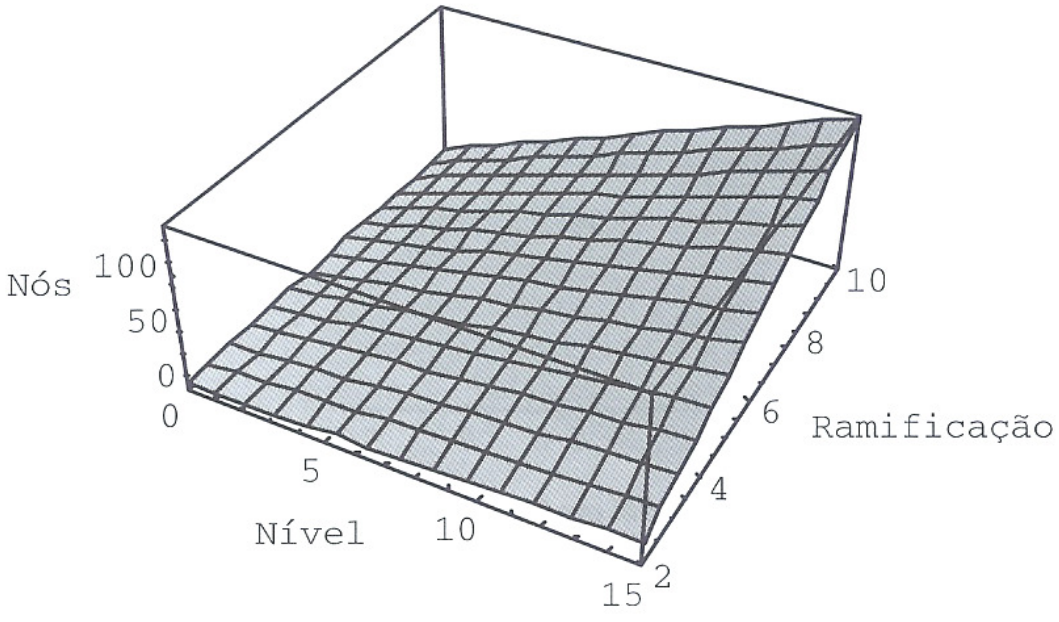
Complexidade espacial em função do factor de ramificação (n = 15)

Procura em Profundidade Primeiro



Complexidade espacial em função do nível (r = 8)

Procura em Profundidade Primeiro



Complexidade espacial

Procura em Profundidade Primeiro

Vejamos alguns valores concretos

- Fator de ramificação: r = 8
- 1 nó é analisado num milissegundo (10^{-3}s)
- Espaço para armazenar 1 nó: 10 Bytes

Nível	Nós	Tempo	Nós	Espaço
0	1	1 ms	1	10 *bytes*
5	9807	9,807 ms	36	360 *bytes*
10	$164*10^6$	1.89 dias	71	710 *bytes*
15	$2,7*10^{12}$	85,6 anos	106	1 *Kbytes*

Valores para a complexidade temporal e espacial, para r = 8.

Procura em Profundidade Primeiro

- Notar que, neste caso, os nós para efeitos do cálculo do tempo são em número diferente dos nós para efeitos de cálculo do espaço.
- Comparando os resultados com os obtidos com a procura em Largura Primeiro, o ganho na quantidade de memória necessária é fantástico e há também uma redução do ponto de vista temporal

Pontos fracos:

- Impossibilidade de lidar com ciclos;
- Não há garantia de encontrar a melhor solução.

Desafio: como alterar o algoritmo por forma a que o problema dos ciclos seja evitado ?

Procura em Profundidade Limitada

Depth-Limited Search

- Este algoritmo procura evitar o problema da incapacidade em lidar com caminhos infinitos do algoritmo anterior, fixando o nível máximo da procura.
- O algoritmo exige apenas uma pequena modificação:
- Supõe-se que faz parte da descrição do estado o nível a que se encontra, podendo os operadores ser aplicados apenas caso o nível do nó seja inferior ao máximo escolhido.

Procura em Profundidade Limitada

```
Função ProcuraProfundidadeLimitada (problema, InserePilha, nível_max): solução ou falha
    1. l_nós ← FazPilha (EstadoInicial (problema))
    2. Repete
        2.1. Se VaziaPilha(l_nós) Então
            2.1.1. Devolve falha
        Fim_de_Se
        2.2. nó ← RetiraPilha(l_nós)
        2.3. Se TesteObjectivo(nó) Então
            2.3.1. Devolve nó
        Senão
            2.3.2. InserePilha(l_nós, Expansão(nó, OperadoresNmx(problema)))
        Fim_de_Se
    Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura em profundidade limitada

Procura em Profundidade Limitada

- O problema que se coloca é o de saber qual o valor máximo para o nível que deve ser usado;
- Nem sempre é possível determinar esse valor;
- E quando é possível, depende do problema;
- No exemplo que temos vindo a considerar existem 18 cidades: o comprimento máximo de uma solução, um caminho entre 2 cidades, será 17.

Procura em Profundidade Limitada

- Admitindo que é possível definir um limite máximo então o algoritmo de procura em profundidade limitada será completo;
- Continua, pelos mesmos motivos, a não ser discriminador;

Procura em Profundidade Limitada

Se l for o valor do limite máximo, teremos que:

- A complexidade temporal assimptótica será da ordem de $O(r^l)$
- A complexidade espacial assimptótica será da ordem de $O(l x r)$

Será, assim, económico do ponto de vista do espaço mas não do ponto de vista temporal.

Procura de Aprofundamento Progressivo

Iterative Deepening Depth-First Search

- No caso de não se saber antecipadamente o valor do limite máximo a que se pode encontrar a solução, faz-se variar esse limite entre o nível 0 e infinito

```
Função ProcuraAprofundamentoProgressivo (problema, InserePilha): solução ou falha
    1. Para nível ← 0 Até infinito Faz
            1.1. Se ProcuraProfundidadeLimitada(problema,InserePilha, nível) Então
                    1.1.1. Devolve solução
            Fim_de_Se
    Fim_de_Para
Fim_de_Função
```

Algoritmo de procura em aprofundamento progressivo

Procura de Aprofundamento Progressivo

- Este algoritmo consiste na chamada repetida do algoritmo de procura limitada para valores crescentes do limite máximo;
- Combina aspetos positivos da procura em largura e da procura em profundidade;
- É completo: o problema dos caminhos infinitos desaparece;
- Não é discriminador: não é necessariamente encontrada a solução mais próxima da raiz.

Procura de Aprofundamento Progressivo

- A complexidade espacial é a da procura em profundidade primeiro, ou seja, da ordem $O(r \times n)$, com n o nível da solução.

- Põe-se a questão, no entanto, da sobrecarga temporal, pelo facto de vários nós serem analisados mais do que uma vez.

Procura de Aprofundamento Progressivo

De facto, se a solução estiver no nível k:

- a raiz será analisada k+1 vezes;
- os nós do nível 1 serão analisados k vezes;
- os nós do nível k-1 serão analisados 2 vezes;
- Finalmente, os nós do nível k serão analisados 1 vez.

(esta situação corresponde ao caso mais desfavorável da solução se encontrar no ramo mais "à direita")

Procura de Aprofundamento Progressivo

$$N_{ap} = \sum_{k=0}^{n} \frac{r^{k+1}-1}{r-1} = \frac{1}{r-1}\left[r\left(\sum_{k=0}^{n} r^{k}\right) - \sum_{k=0}^{n} 1 \right] = \frac{1}{r-1}\left[r\left(\frac{r^{n+1}-1}{r-1}\right) - (n+1) \right]$$

Número de nós analisados em aprofundamento progressivo

- Esta expressão pode ser simplificada para

$$N_{ap} = \frac{r^{n+2} - 2r - rn + n + 1}{(r-1)^2}$$

Nós analisados pelo algoritmo de aprofundamento progressivo

Procura de Aprofundamento Progressivo

- Podemos ter uma ideia rigorosa da sobrecarga temporal dividindo este valor pelo valor obtido com a procura em largura primeiro, para diferentes valores do fator de ramificação e do nível

$$1 + r + r^2 + r^3 + \ldots + r^n = \sum_{k=0}^{n} r^k = \frac{r^{n+1} - 1}{r - 1}$$

Número de nós analisados na procura em largura primeiro

Procura de Aprofundamento Progressivo

R	N	Eq,.3.6 / Eq. 3.1
2	5	1.90
	10	1.99
5	5	1.25
	10	1.25
10	5	1.11
	10	1.11

Sobrecarga temporal no aprofundamento progressivo

- Para um fator de ramificação de 10, a sobrecarga é de apenas 11%
- Mesmo no caso simples de um fator de ramificação de apenas 5, a sobrecarga é da ordem dos 25%
- Claro que para valores mais elevados estes resultados serão ainda mais favoráveis.

Procura de Aprofundamento Progressivo

- Para valores elevados do fator de ramificação podemos aproximar aquele valor por

$$\frac{r^{n+2}}{(r-1)^2} \quad \text{e} \quad \frac{r^{n+1}}{r-1}$$

nós analisados por aprofundamento progressivo e por largura primeiro

Procura de Aprofundamento Progressivo

- O quociente das duas expressões é dado por:

$$\frac{r}{r-1}$$

Sobrecarga da complexidade temporal

podendo ser usado para determinar a sobrecarga da complexidade temporal motivada pelo algoritmo de procura por aprofundamento progressivo – notar que não depende do nível.

Procura de Aprofundamento Progressivo

- O algoritmo de procura por aprofundamento progressivo é uma excelente opção para problemas em que é necessário recorrer a um método cego, o espaço de procura é grande, mas não se sabe qual é o nível máximo em que pode estar uma solução.