Resolução de Problemas por Meio de Buscas

Conteudista: Prof. Me. Manuel Fernández Paradela Ledón
Revisão Textual: Prof. Me. Luciano Vieira Francisco

Objetivos da Unidade:

- Conhecer formas de resolução de problemas por meio de buscas;
- Conhecer os tipos de buscas e comparar as suas características.

Material Teórico

Exercício ou Problema?

Nesta Unidade abordaremos a **resolução de problemas por meio de buscas**.

Poderia parecer óbvio, mas, o que significa "problema"? "Problema" e "exercício" têm o mesmo significado? Estes termos possuem uma grande semelhança, mas, alguns autores, como Paul Zeitz (2007), fazem questão de diferenciar, vejamos:

> "*First, what is a problem? We distinguish between problems and exercises. An exercise is a question that you know how to resolve immediately. Whether you get it right or not depends on how expertly you apply specific techniques, but you don't need to puzzle out what techniques to use. In contrast, a problem demands much thought and resourcefulness before the right approach is found.*"
>
> **- ZEITZ, 2007**

Basicamente, esse teórico quis deixar claro que um exercício é utilizado para praticar sobre algo que sabemos como resolver. Já para resolver um problema, cuja solução não é tão óbvia ou fácil,

por conta de sua complexidade, necessitamos de maior reflexão e possivelmente estudar até encontrar uma estratégia adequada para a solução.

Comparando a Eficiência de Algoritmos para Resolver Problemas

Como estudaremos aqui, quanto às estratégias para a resolução de problemas por meio de buscas, chegaremos a conclusões e compararemos, finalmente, tais algoritmos quanto ao seu desempenho e também à utilização de memória auxiliar necessária para a implementação dos mesmos.

O desempenho ou eficiência na solução de problemas está relacionado com vários elementos, mencionados a seguir:

- **Completeza ou completude:** o algoritmo oferece certeza de encontrar uma solução para o problema?
- **Otimização:** o algoritmo encontrará uma solução ótima para o problema?
- **Complexidade de tempo – velocidade:** quanto tempo é necessário para executar a busca?
- **Complexidade de espaço:** quanta memória é necessária para executar a busca?
- **Complexidade algorítmica:** qual é o custo da implementação, dada a sua complexidade?

Na Ciência da Computação é frequentemente utilizada uma notação conhecida como big-O, com esta forma: O(expressão), que permite identificar a eficiência de um algoritmo – quanto à velocidade ou memória – para resolver determinado problema. Tal expressão normalmente considera o algoritmo em uma situação onde a quantidade de dados é grande, utilizando o conceito matemático de limite, ou seja, é efetuada uma análise assintótica para verificar a eficiência do algoritmo em situações extremas. Quando o cálculo de O() se preocupa com a

velocidade, será analisada a quantidade de comparações ou operações necessárias para se chegar à solução final.

Por exemplo, um algoritmo poderia ter eficiência $O(n)$, $O(n^2)$, $O(\log_2 n)$ etc., o que caracterizará a sua complexidade, sendo $O()$ uma representação do número de passos do algoritmo em função do tamanho n da entrada. Por exemplo, um algoritmo que executa em $O(n)$ é melhor que outro com $O(n^2)$, ainda que para $n = 1$ ou valores pequenos de n sejam equivalentes.

Outros dois exemplos mais críticos seriam algoritmos que executam em $O(2^n)$ e $O(n!)$, porque seriam algoritmos muito ineficientes. Observe que uma função exponencial como 2^n cresce muito rápido, à medida que aumenta o valor de n.

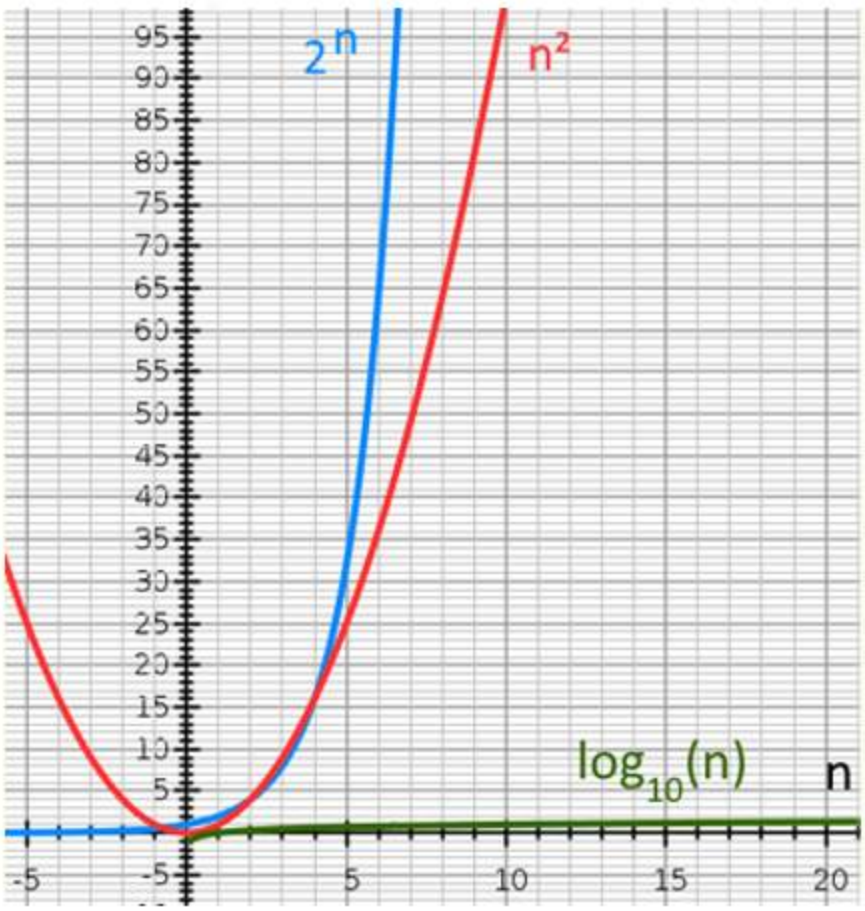

Figura 1 – Funções $O(2^n)$, $O(n^2)$ e $O(\log_{10} n)$.

Fonte: Reprodução

Na Figura anterior, construída com a ferramenta disponibilizada em: <http://www.graphsketch.com>, podemos observar que um algoritmo com eficiência logarítmica, de $O(\log_{10} n)$, ainda que normalmente utilizemos um logaritmo de base dois, $O(\log_2 n)$, será muito eficiente, mesmo para valores grandes de **n**.

Já um algoritmo com eficiência $O(n^2)$ é bem menos eficiente. Ainda que não desenhado na Figura, um algoritmo com eficiência $O(n)$ seria um caso intermediário, uma função que seria uma linha reta, que representa uma eficiência que depende linearmente da quantidade de dados **n**. Um algoritmo com $O(\log_2 n)$ continuaria sendo bem melhor que os restantes.

Na outra função mostrada, um algoritmo que execute em $O(2^n)$, exponencial com base dois, poderia ser considerado muito ineficiente, caracterizando o problema resolvido como intratável, se for esta a melhor solução possível para a implementação.

Métodos de Resolução de Problemas

Além de outros autores, Russell e Norvig (2004) classificam os métodos para a resolução de problemas como mostramos a seguir:

- **Busca sem informação:** não existe nenhuma informação do problema além da sua definição;
- **Busca com informação:** informações conhecidas sobre as especificidades do problema ajudam a encontrar soluções mais eficientes; buscas heurísticas;
- **Problemas de satisfação de restrições:** utilizam alguma representação simples padrão (como um "grafo de restrições") para representar os elementos do problema e suas restrições;

- **Busca competitiva:** a busca de soluções se caracteriza pela participação de agentes que estão em conflito; a **solução** de um agente **compete** com a **solução de outro** que planeja contra ele (típico em jogos);

- **Métodos estocásticos:** utilização da teoria das probabilidades e outros métodos que utilizem conhecimento incerto, aproximado ou probabilístico para a solução de problemas (Adaptada de RUSSELL E NORVIG, 2004).

Especificamente, os métodos de resolução de problemas dentro da **busca sem informação em espaços de estados** costumam ser subdivididos em algumas possibilidades, como mostrado a seguir:

- Busca em extensão;

- Busca de custo uniforme;

- Busca em profundidade;

- Busca em profundidade limitada;

- Busca de aprofundamento iterativo em profundidade;

- Busca bidirecional (Adaptada de RUSSELL E NORVIG, 2004).

Já George Luger (2013) divide o seu estudo sobre métodos para a resolução de problemas em três temas, a saber:

- Busca em espaço de estados;

- Busca heurística;

- Métodos estocásticos.

Resolução de Problemas por Meio de Buscas

Muitos livros e textos de inteligência artificial estudam este assunto, chamado de **resolução de problemas por meio de buscas**. Mas, o que isto significa?

Tecnicamente, isto seria utilizar alguma estratégia para buscar soluções para um problema, encontrando sequências de ações que levem a estados desejáveis e preferivelmente a uma solução ótima do problema.

Dito de outra forma, partimos de um estado inicial e queremos chegar a um estado final, que seria uma solução para o problema que estamos resolvendo; mas queremos que seja garantido que sempre cheguemos a uma solução utilizando esse método – completeza ou completude: que encontremos sempre uma solução – e que a solução seja ótima, a melhor possível quanto ao tempo e à memória utilizada.

Espaço de Estados – *State Space*

Na representação da solução de um problema como um espaço de estados, podemos utilizar um grafo:

- Cujos nós são estados parciais, intermediários ou finais na solução do problema; um desses nós será considerado o estado inicial para o princípio da solução do problema;

- Cujos arcos são passos ou etapas no processo de solução do problema;

- Que costuma definir estados objetivos – metas –, que seriam resultados esperados para a solução do problema.

Uma busca em um espaço de estados será encontrar uma solução do problema que se caracteriza por encontrar um caminho entre o estado inicial e um estado objetivo, preferivelmente ótimo. Russell e Norvig (2004) mencionam que um problema pode ser definido por quatro componentes:

1. O estado inicial em que o agente começa a resolver o problema;
2. Uma descrição das ações possíveis que estão disponíveis ao agente. Uma função sucessora pode determinar dentro de um espaço de estados qual será a próxima ação na sequência;
3. Um teste de objetivo que verifica se determinado estado é objetivo ou final – por exemplo, no xadrez, a condição de xeque-mate seria um estado objetivo ou final;
4. Poderá ser considerado um custo de caminho atribuído a cada percurso, calculado como a soma dos custos de passos, onde cada passo pode ser um valor numérico e será o custo de uma ação para ir de um estado para outro. Um custo de caminho mínimo poderia ser uma solução ótima do problema.

Analisemos o exemplo da Figura a seguir: os nós são cidades, são os estados deste espaço, sendo que Cidade 1 é o estado inicial e a Cidade 7 é o estado final. As outras cidades são estados parciais ou intermediários na solução do problema. Os arcos são caminhos que especificam a distância em quilômetros entre duas cidades. Neste grafo poderíamos acrescentar – nos arcos – os nomes das estradas e as direções ou sentidos.

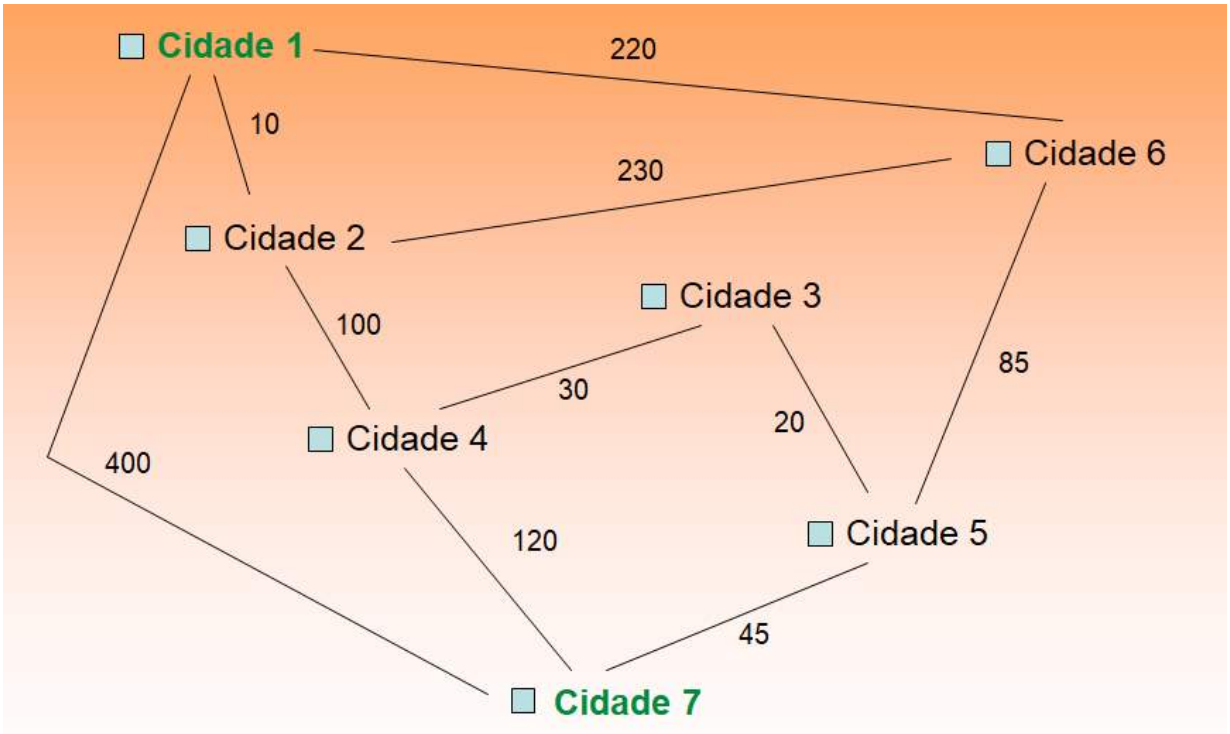

Figura 2 – Exemplo: um espaço de estados, com cidades (nós) e caminhos (arcos)
Fonte: Reproduçã

A solução do problema deste exemplo seria encontrar um caminho eficiente para viajar de uma cidade a outra. Mas teremos várias questões a serem respondidas, seguindo o esquema geral descrito:

- Qual é o estado inicial?
- Qual é o estado final ou objetivo?
- Qual seria a função sucessora?
- Qual seria o teste de objetivo?
- Qual seria uma solução ótima?

Antes, porém, mencionamos que um custo de caminho mínimo poderia ser uma solução ótima do problema. Ao analisar inicialmente a Figura anterior, podemos achar várias possibilidades que, aparentemente, seriam as melhores para ir do estado inicial Cidade 1 até o estado final Cidade 7. Mas, calculando os custos de caminhos – quilometragem neste exemplo –, um caminho menos óbvio é o que possui o custo mínimo: Cidade 1, Cidade 2, Cidade 4, Cidade 3, Cidade 5, Cidade 7 (10 + 100 + 30 + 20 + 45).

Vejamos outro exemplo de "espaço de estados": Você possivelmente lembrará de um jogo cujo nome, em inglês, é 8-puzzle – reproduzido na Figura a seguir. Consiste em, a partir de um estado inicial qualquer, mover os números nos quadros, utilizando o único espaço vazio existente, para chegar a um estado final que mostre os números ordenados em sequência crescente: 1, 2, 3, 4, 5, 6, 7 e 8. O diagrama de estado mostra todas as possibilidades ou estados, dependendo do movimento que será feito. A descrição utilizada pelo *luger* – *up*, *down*, *left*, *right* – refere-se ao movimento do espaço, à operação "mover branco".

Novamente, os conceitos explicados quanto a definir os estados inicial e final do espaço de estados, ou seja, de ter uma função sucessora para decidir o próximo movimento, o teste de objetivo, estabelecer um custo de caminho atribuído a cada passo, para calcular um custo de percurso mínimo e que permita chegar à solução do problema – estado final ou objetivo – o mais rápido possível.

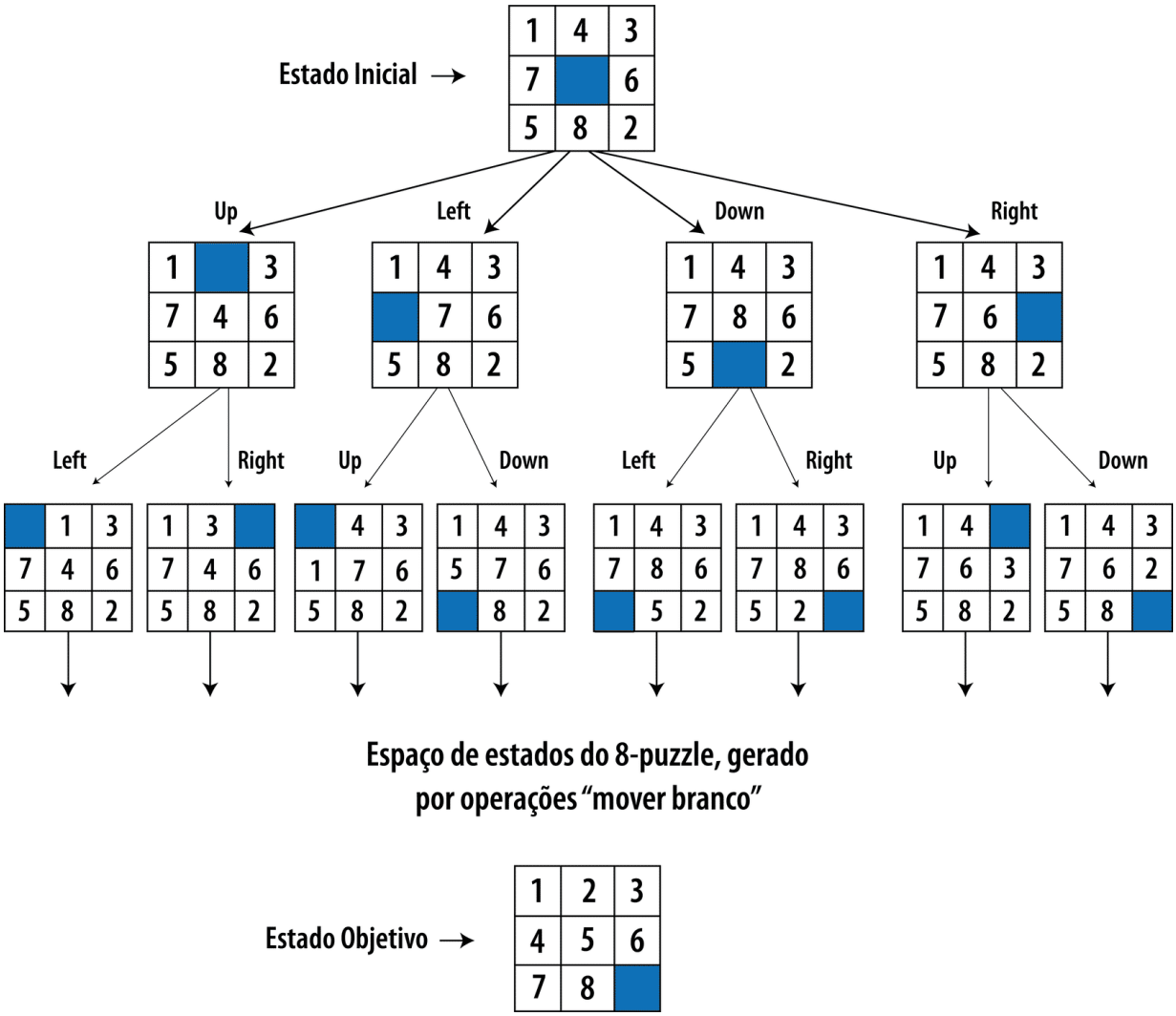

Figura 3 – Espaço (parcial) de estados para o jogo 8-*puzzle*
Fonte: Adaptada de LUGER, 2013

Buscas Sem Informação em Espaços de Estados

Analisaremos três tipos de buscas sem informação – busca não informada, *blind search*, busca cega – em espaços de estados. Observe que isto significa que conhecemos a definição do problema, os estados, mas não temos informações adicionais ou sugestões sobre como encontrar mais rápido a solução ou o objetivo final. Assim, abordaremos brevemente apenas três buscas sem informação em espaços de estados, que diferem na forma em que são gerados/visitados os estados do espaço, tratam-se das buscas:

- Em extensão;

- Em profundidade;
- Bidirecionais.

Busca em Extensão

A busca em extensão ou amplitude – em inglês, *breadth-first search* – caracteriza-se pela seguinte lógica: o nó raiz é expandido primeiro, em seguida serão expandidos todos os sucessores do nó raiz, depois os sucessores desses nós e assim por diante.

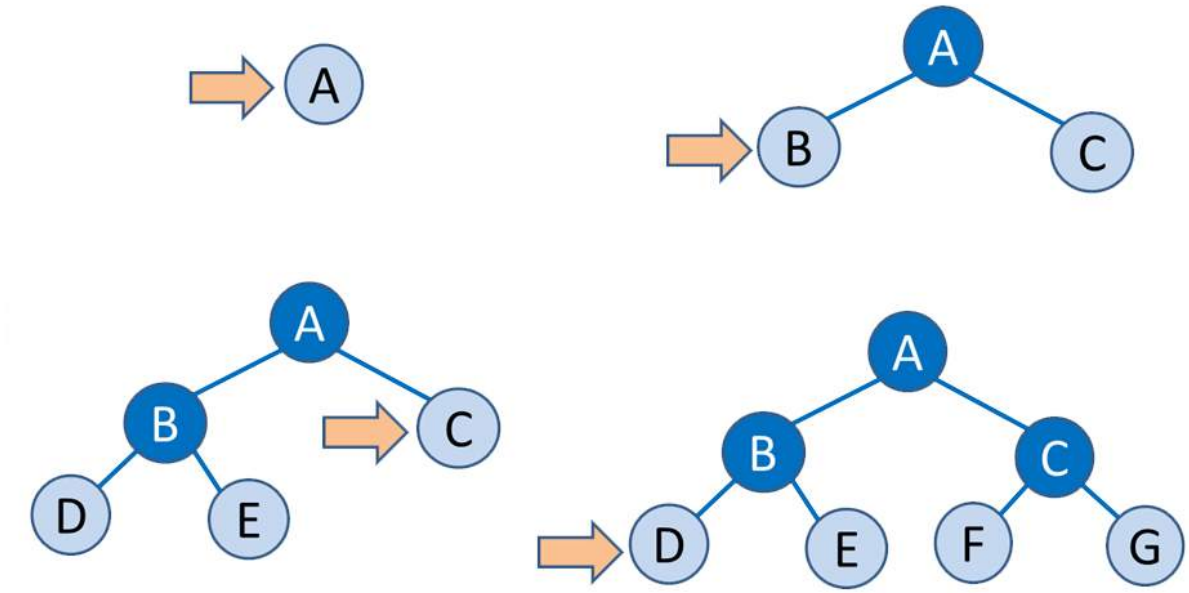

Figura 4 – Busca em extensão

A visita de um estado e expansão do mesmo ocorre completamente em um nível para depois passar a expandir o próximo nível – é uma busca de nível a nível. Na Figura 4, os estados já visitados e expandidos foram coloridos na cor azul escura e os estados ainda não visitados/expandidos na cor azul clara. Ademais, a seta laranja aponta ao estado examinado.

Em uma busca em extensão, como mostrado na Figura 4, os estados são "examinados" nesta ordem: A, B, C, D, E, F, G..., o que mostra a característica de visitar os estados nível a nível, semelhante a uma leitura de cima para baixo e da esquerda para a direita em cada linha, ou seja, visitar o espaço de estados nível a nível.

Vejamos aspectos positivos desse método de busca em um espaço de estados: é uma estratégia relativamente simples, é completa – sempre encontraremos um estado objetivo, uma solução – e podemos chegar a encontrar uma solução ótima – se os custos de todos os passos são iguais.

Já quanto aos problemas: para uma solução encontrada na profundidade d e supondo que os estados tenham b nós sucessores, pode ser demonstrado que a eficiência em armazenamento e de tempo é de $O(b^{d+1})$, ordem exponencial, o que o cataloga como um método ineficiente de busca em um espaço de estados para valores grandes de **b** e **d**.

Busca em Profundidade

Examinemos agora a busca em profundidade, conhecida, em inglês, como *depth-first search*.

A busca em profundidade em um espaço de estado se caracteriza por esta estratégia: quando um estado é examinado, todos os seus filhos e descendentes são averiguados, antes que qualquer um de seus irmãos.

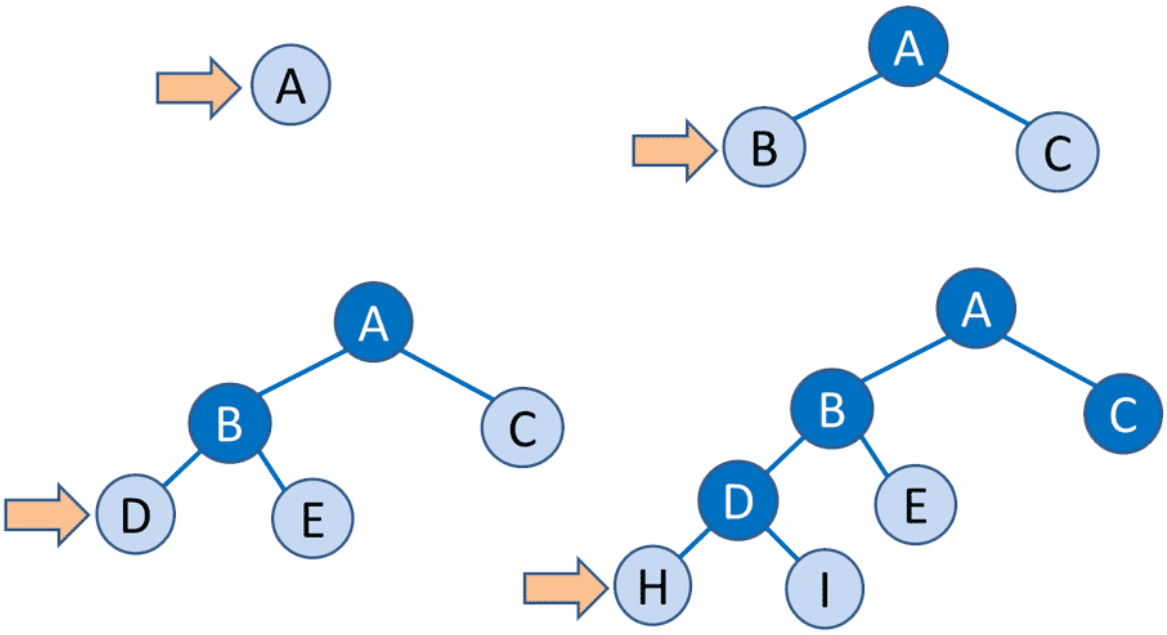

Figura 5 – Busca em profundidade

Analisemos a Figura 5: o primeiro estado examinado é A. Imediatamente, abrimos os seus filhos B e C. Observe que B será, então, examinado, com seus filhos, antes de averiguar os irmãos de B – antes de examinar C, no exemplo.

O mesmo acontece com **D** que, com seus filhos **H** e **I**, serão abertos e examinados antes de averiguar o seu irmão **E**. Esta lógica – que justifica o nome de busca em profundidade – mergulha pelo caminho do primeiro filho esquerdo no espaço de estados, antes de analisar os irmãos.

São aspectos positivos desse método de busca em um espaço de estados: para um espaço de estados com fator de ramificação **b** e profundidade máxima **m** (níveis), exige o armazenamento de, apenas, bm + 1 nós, ou seja, é de ordem O(bm), polinomial, pelo que a **busca em profundidade** seria mais eficiente quanto à memória – armazenamento –, se comparada à busca em amplitude – extensão.

Sobre os problemas: pode acontecer que efetuemos uma escolha de caminho equivocada – em um percurso muito longo ou mesmo infinito –, no lugar de escolher uma solução próxima à raiz da árvore de procura. A busca em profundidade nem sempre é ótima. Se for escolhida uma subárvore de profundidade ilimitada, sem nenhuma solução, essa busca nunca terminaria, então não seria completa. No pior caso, pode chegar a ser de ordem $O(b^m)$, exponencial, quanto ao tempo de processamento.

Exemplo: Analisemos um caso a partir da figura presente em Luger (2013), a fim de demonstrar as sequências das buscas em extensão – amplitude – e profundidade em um espaço de estados:

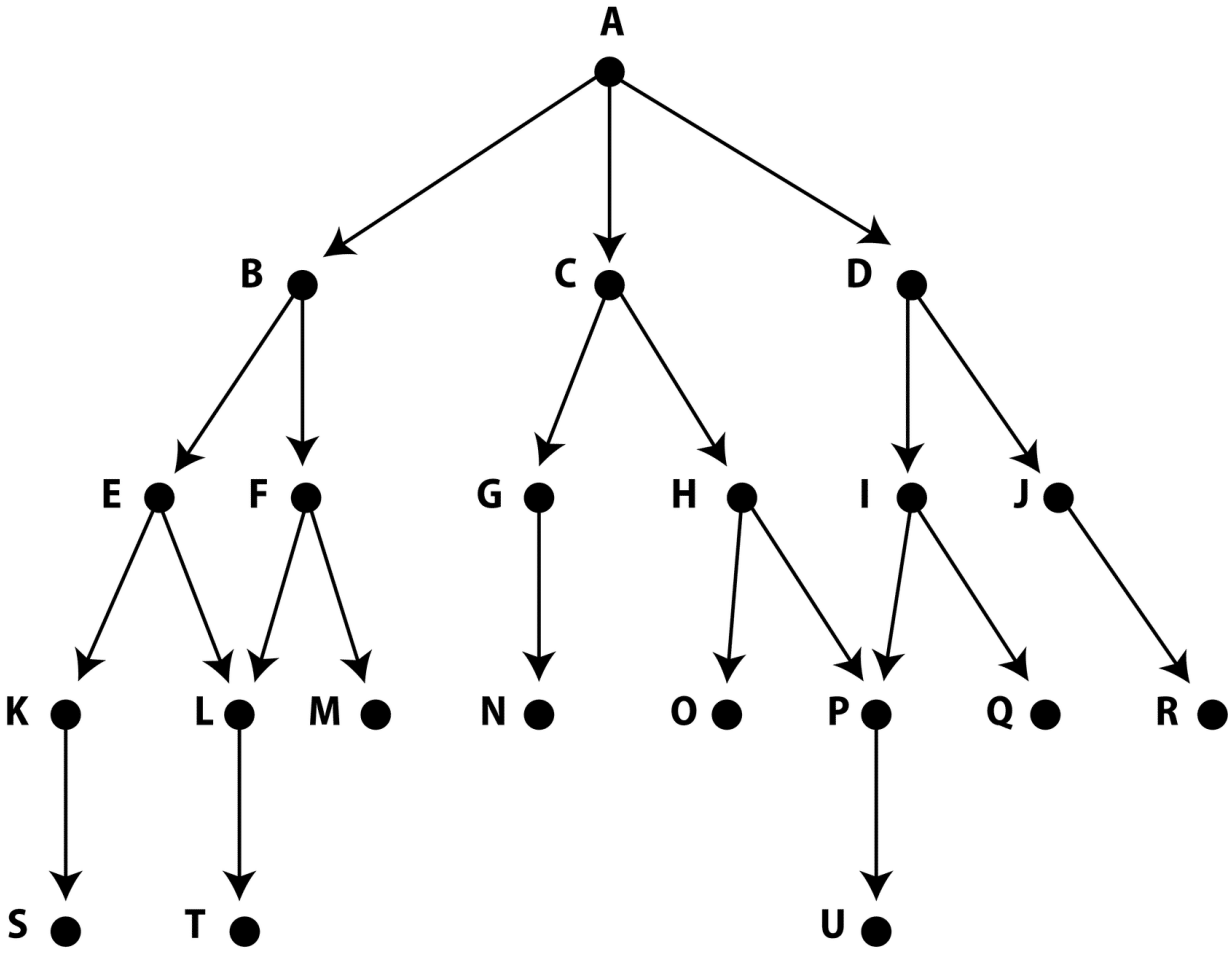

Figura 6 – Um grafo de um espaço de estados
Fonte: LUGER, 2013, p. 82

Uma busca em extensão – amplitude – nesse espaço de estados, examinará os estados nível a nível, nesta ordem:

A

B C D

E F G H I J

K L M N O P Q R

S T U

Já uma busca em profundidade neste mesmo espaço de estados, examinará os presentes nesta ordem:

A B E K S L T F M C G N H O P U D I Q J R

Isto porque a busca se aprofundará nos caminhos esquerdos antes de examinar os estados dos nodos irmãos.

Busca Bidirecional

Em uma busca bidirecional serão feitas procuras em extensão simultaneamente, partindo de um estado inicial e objetivo até se encontrarem em um ponto intermediário.

Este método de busca possui eficiências de armazenamento e de processamento de ordem $O(b^{d/2})$, uma função exponencial, sendo que **d** é a profundidade da solução mais rasa – em menor nível.

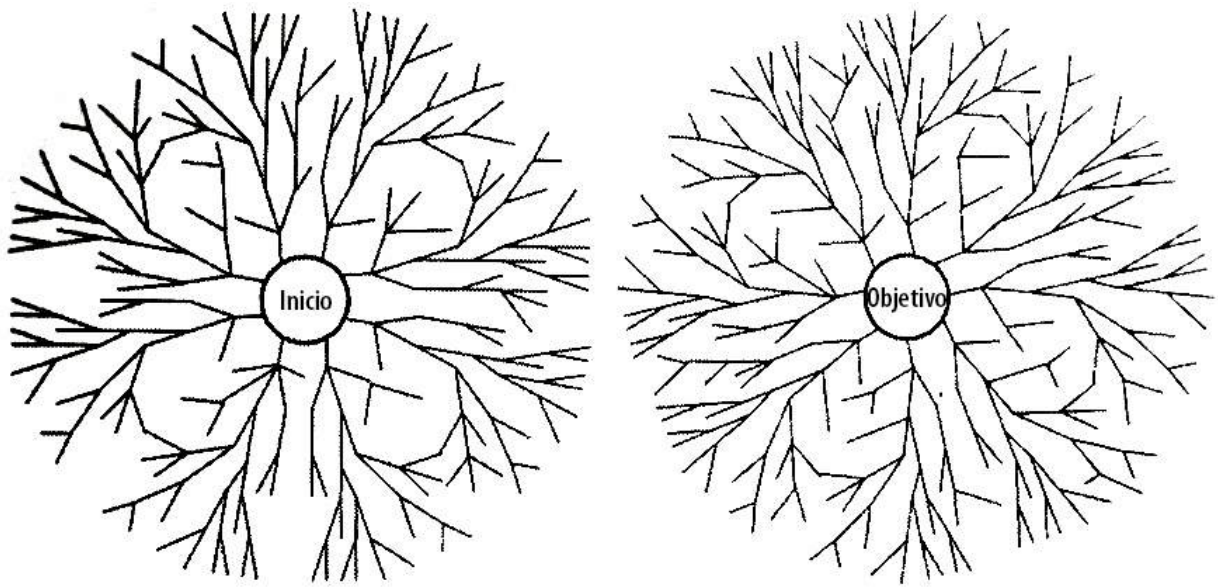

Figura 7 – Busca bidirecional em um espaço de estados
Fonte: Adaptada de RUSSELL E NORVIG, 2004

Busca Heurística – Busca com Informação, Busca Informada

O que significa heurística? Trata-se do: "Conjunto de regras e métodos que visam à descoberta, à invenção ou à resolução de problemas" (AURÉLIO, 2009).

"Eureka!", disse o grego Arquimedes, expressão que significa o **achei!**, **o encontrei!**

Em buscas em espaços de estados, as heurísticas são identificadas como regras ou funções para escolher aqueles caminhos em um espaço de estados que levam a uma solução do problema aceitável. É um caso de busca com informação.

Situações básicas onde os "resolvedores" de problemas utilizam heurísticas:

- Um problema poderia não ter uma solução exata, como consequência de ambiguidades na formulação do problema ou nos dados disponíveis – por exemplo,

situações de diagnóstico médico e visão/reconhecimento;

- Um problema poderia ter uma solução exata, mas o custo computacional – memória/tempo – poderia ser proibitivo.

Determinadas heurísticas poderiam resolver problemas nessas duas situações, propondo soluções baseadas na experiência sobre o problema e oferecer caminhos mais promissórios ou com menores custos de tempo/memória.

Busca Gulosa

Conhecida como busca gulosa, busca gulosa pela melhor escolha, *greedy* e Busca pela Melhor Escolha (BME) – em inglês, *best-first search*. Esta busca poderia ser utilizada para encontrar uma solução em um espaço de estados.

Tal busca, dentro de um espaço de estados, tenta expandir – analisar – o nó mais próximo da meta ou do objetivo – nodo mais próximo, perto da solução, com menor tempo ou qualquer outra função heurística semelhante –, na suposição de que essa estratégia levará a uma solução mais rápida.

É utilizada uma função heurística h(n), que retorna o custo estimado do caminho mais econômico do nó **n** até um nó objetivo – para alguma solução do problema. Pressupõe conhecimento ou informação sobre o problema a resolver, ou seja, é um método de busca com informação.

Exemplo: analisemos um caso de busca gulosa apresentado por Russell e Norvig (2004, p. 65, 95-97), em que utilizaram um modelo com cidades da Romênia:

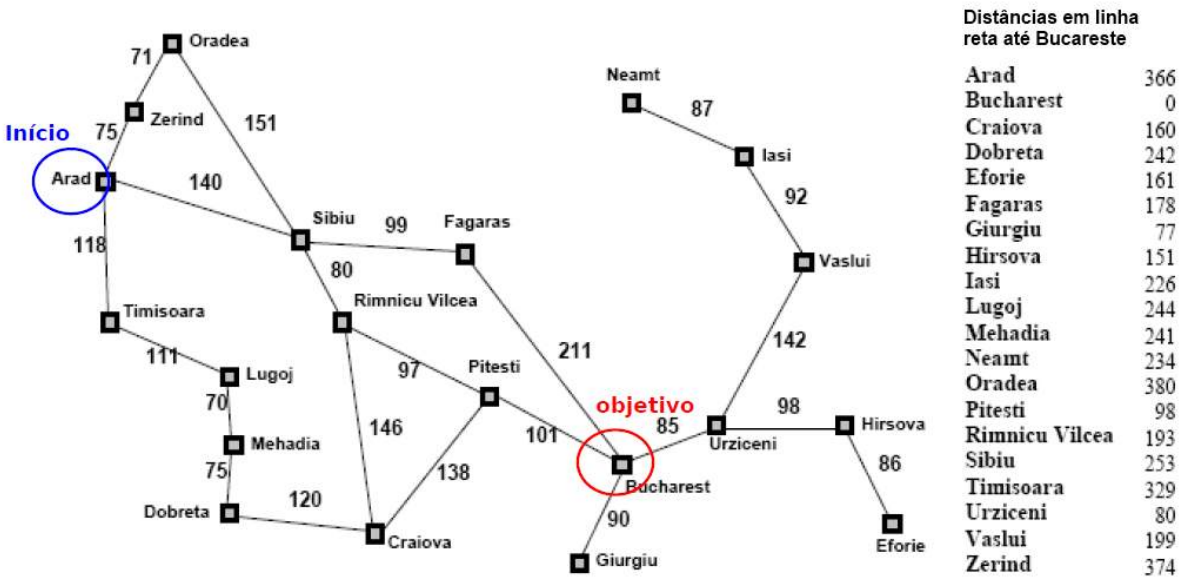

Figura 8 – Exemplo de busca gulosa
Fonte: Adaptada de RUSSELL e NORVIG, 2004, p. 65, 95-97

O problema será encontrar o melhor caminho para ir de **Arad** até **Bucareste**. Será realizada, portanto, uma análise considerando distâncias e ignorando fatores importantes, tais como estado de uso das estradas, trânsito nas mesmas, paisagem, velocidade permitida, lugares históricos para visitar etc.

Na Figura 8, os nós são as cidades – estados do problema – e os números são as distâncias rodoviárias entre duas cidades.

A função heurística que será utilizada, h(n), é a distância em linha reta de uma cidade **n** até Bucareste, que é o destino final.

Observe, no exemplo da Figura 8, que:

- O primeiro nó que será expandido a partir de Arad será Sibiu porque, segundo a função heurística h(n), é a cidade que está mais próxima (253) em linha reta até Bucareste. Os outros nós adjacentes a Arad são Zerind (374) e Timisoara (329);

- Analisando as cidades adjacentes a Sibiu e utilizando a mesma função heurística, será selecionada Fagaras, que é a que fica mais próxima a Bucareste em linha reta.

- Fagaras gera Bucareste, que é o objetivo desejado;

- Se analisarmos detalhadamente, o caminho Arad-Sibiu-Fagaras-Bucareste, recomendado pela função heurística, tem 140 + 99 + 211 = 450 km, de modo que não seria o melhor caminho. Temos 418 km, uma menor distância, se vamos por Arad-Sibiu-RimnicuVilcea-Pitesti-Bucareste;

- **Podemos encontrar determinados problemas, vejamos:** utilizando uma função heurística semelhante para ir de Iasi até Fagaras, sugerir-se-ia tomar o caminho Iasi-Neamt, um beco sem saída, uma vez que não existe estrada entre Neamt e Fagaras. Neste caso, a função heurística propõe um caminho que não é uma solução correta, ou seja, não garante a completude.

Importante!

Uma função heurística aprimorada poderá levar a soluções muito mais eficientes.

Finalmente, compararemos os métodos para "resolução de problemas por meio de buscas em um espaço de estados" que foram aqui estudados. Para tanto, veja o seguinte Quadro:

Quadro 1 – Comparando os métodos estudados para resolução de problemas por meio de buscas

Critério	Em extensão	Em profundidade	Bidirecional	Gulosa
Completa?	Sim	Não	Sim	Não
Tempo	$O(b^{d+1})$	$O(b^{m})$	$O(b^{d/2})$	$O(b^{m})$
Espaço	$O(b^{d+1})$	$O(bm)$	$O(b^{d/2})$	$O(b^{m})$
Ótima?	Sim	Não	Sim	Não

d – é a profundidade da solução mais rasa
b – é o fator de ramificação
m – é a profundidade máxima de árvore ou espaço de busca

Os elementos marcados com a cor vermelha representam situações piores, enquanto que os assinalados com a cor verde são as condições melhores.

Eficiências que possuem funções exponenciais serão críticas para grandes tamanhos dos parâmetros, seja quanto ao tempo ou à memória utilizada pelo método. Eficiências que possuem funções polinomiais ou uma função exponencial menos agressiva serão melhores, ainda para tamanhos grandes dos parâmetros – e, no Quadro 1, foram marcadas na cor verde.

Quanto aos comentários **sim** e **não**, o desejado é que um método seja completo – que sempre permita chegar a uma solução – e ótimo, marcado com o valor **sim** no Quadro 1.

Finalmente, não existe um método perfeito. Por exemplo, a busca em extensão é completa e permite encontrar uma solução ótima para esta sequência de busca, mas as funções de eficiência quanto ao tempo e à memória são exponenciais.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Livros

Inteligência Artificial

WINSTON, P. H. **Inteligência artificial**. Rio de Janeiro: LTC, 1988.

Leitura

Artificial Intelligence

Clique no botão para conferir o conteúdo.

ACESSE

Resolução de Problemas por Meio de Busca

Clique no botão para conferir o conteúdo.

ACESSE

Referências

AURÉLIO. **Novo dicionário eletrônico Aurélio**. Versão 6.0. Curitiba, PR: Positivo, 2009.

LUGER, G. F. **Inteligência artificial**. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

RICH, E.; KNIGHT, K. **Inteligência artificial**. 2. ed. São Paulo: Makron Books, 1994.

ROSA, J. L. G. **Fundamentos da inteligência artificial**. Rio de Janeiro: LTC, 2011.

RUSSELL, S. J.; NORVIG, P. **Inteligência artificial:** referência completa para cursos de Computação. 2. ed. Rio de Janeiro: Elsevier, 2004.

ZEITZ, P. ***The art and craft of problem solving***. 2. ed. New York: Wiley, 2007.