Inteligência Artificial

Notas de Aula*
Guilherme Camargo, Priscila T. M. Saito

Tópico 03

1 Resolução de Problemas por meio de Busca

Neste tópico será abordado um tipo de agente baseado em objetivo denominado **agente de resolução de problemas**. Inicialmente, discutimos a resolução de problemas com um definição precisa dos **problemas** e de suas **soluções**, fornecendo alguns exemplos de forma a ilustrar tais definições.

Posteriormente, são apresentados algoritmos de busca de propósito geral, os quais podem ser considerados para resolução desses problemas. São descritos os algoritmos de **busca sem informação**, para os quais não se fornece nenhuma informação sobre o problema a não ser sua definição. Apesar de alguns desses algoritmos resolverem qualquer problema solucionável, nenhum deles oferece soluções de forma eficiente. Já os algoritmos de **busca informada** podem apresentar um melhor desempenho por meio de orientações sobre onde procurar soluções.

Neste tópico será considerado o ambiente de tarefa mais simples, em que a solução para um problema é sempre uma sequência fixa de ações.

1.1 Agentes de Resolução de Problemas

Conforme mencionado, os agentes inteligentes devem maximizar sua medida de desempenho. Tal objetivo pode ser simplificado se o agente adotar um **objetivo** que deseja satisfazer. O primeiro passo para a resolução de problemas é a **formulação de objetivos**, com base na situação atual e na medida de desempenho do agente.

Por exemplo, considerando um agente em uma viagem de férias na Romênia, na cidade de Arad. A medida de desempenho de tal agente pode conter vários fatores (e.g. melhorar o conhecimento do idioma, apreciar as paisagens, aproveitar a vida noturna, entre outros). O problema de decisão é complexo. Considere, então, que o agente tenha que retornar da viagem de férias na manhã seguinte, com partida a partir da cidade de Bucareste. Nesse contexto, o agente deve adotar o **objetivo** de chegar a Bucareste (a partir da cidade atual Arad). As ações que não chegam a Bucareste a tempo podem ser rejeitados. O problema de decisão do agente é simplificado. Os objetivos auxiliam na organização do comportamento, limitando o que o agente tenta alcançar e as ações que devem ser consideradas.

Em seguida, o segundo passo para a resolução de problemas, após a **formulação de objetivos**, consiste na **formulação de problemas**, processo de definir que ações e estados devem ser considerados, dado um objetivo. Nesse contexto, é importante a abstração de detalhes irrelevantes para a definição dos estados e ações – e.g. considerando ações ao nível de "mover o pé esquerdo para a frente uma polegada"ou "girar o volante um grau para a esquerda", dificultaria o agente atingir o objetivo de chegar a Bucareste, dado que nesse nível de detalhe existe bastante incerteza no mundo e muitos passos para se atinigr uma solução. Nesse exemplo, o agente deve então considerar ações no nível de dirigir de uma cidade importante até outra e cada estado corresponde a estar em uma determinada cidade.

Se o agente tem conhecimento do estado inicial e o ambiente é conhecido e determinístico, tal agente terá conhecimento de onde estará exatamente após a primeira ação e o que vai perceber. Após a primeira ação e em seguida o recebimento de uma percepção, a solução pode especificar uma segunda ação possível, e assim por diante.

*Estágio à docência realizado pelo aluno de mestrado Guilherme Camargo do Programa de Pós-Graduação em Bioinformática (PPGBIOINFO) da UTFPR

O processo de procura por tal sequência de ações que atinjam o objetivo é denominado de **busca**. O algoritmo de busca recebe como entrada um problema e retorna uma solução sob a forma de uma sequência de ações. Após encontrar uma solução, as ações sugeridas podem ser executadas (fase de **execução**).

Dessa forma, tem-se um agente simples de resolução de problemas (Figura 1) com o projeto de "formular, buscar, executar". Após a formulação do objetivo e do problema a resolver, o agente realiza a chamada do procedimento de busca para resolvê-lo. Em seguida, o agente considera a solução para definir suas ações, realizando a próxima ação conforme a solução apresentada.

Nas próximas seções, inicialmente, será descrito o processo de formulação de problemas e em seguida alguns algoritmos para a função BUSCA.

função AGENTE-DE RESOLUÇÃO-DE-PROBLEMAS-SIMPLES(percepção) **retorna** uma **ação**
  persistente: *seq,* uma sequência de ações, inicialmente vazia
    estado, alguma descrição do estado atual do mundo
    objetivo, um objetivo, inicialmente nulo
    problema, uma formulação de problema
  estado ← ATUALIZAR-ESTADO(*estado*, *percepção)*
  se *seq* está vazia **então faça**
    objetivo ← FORMULAR-OBJETIVO(*estado*)
    problema ← FORMULAR-PROBLEMA(*estado*, *objetivo*)
    seq ← BUSCA(*problema*)
    se *seq* = falhar **então retorne** uma ação nula
  ação ← PRIMEIRO(*seq*)
  seq ← RESTO(*seq*)
  retornar ***ação***

Figura 1: Algoritmo de um agente simples de resolução de problemas. O agente formula o objetivo e um problema, busca uma sequência de ações que resolvem o problema e executa as ações

1.1.1 Problemas e soluções bem definidos

A definição formal de um problema é dada por cinco componentes:

- **Estado inicial** em que o agente começa. Considerando, por exemplo, o problema do agente na Romênia, o estado inicial poderia ser descrito como *Em(Arad)*.
- **Ações** possíveis que o agente pode executar. A função sucessora *AÇÕES(s)* recebe como entrada um estado s e retornam um conjunto de ações disponíveis que podem ser executadas em s. Considerando, o exemplo do agente na Romênia, as ações aplicáveis são {*Ir(Sibiu)*, *Ir(Timisoara)*, *Ir(Zerind)*}, a partir do estado *Em(Arad)*.
- **Modelo de transição** referente à descrição do resultado de cada ação. A função *RESULTADO(s, a)* recebe como entrada um estado s e uma ação a e retorna o estado resultante. O termo **sucessor** também é utilizado para referir-se a qualquer estado acessível a partir de determinado estado por uma única ação. Por exemplo, considerando o problema do agente na Romênia, RESULTADO(Em(arad), Ir(Zerind)) = Em(Zerind). O estado inicial, as ações e o modelo de transição definem, de forma implícita, o **espaço de estados** do problema, i.e. o conjunto de todos os estados acessíveis a partir do estado inicial, por qualquer sequIencia de ações. O espaço de estados forma um grafo em que nós são estados e as arestas entre os nós são ações. A Figura 2 ilustra o mapa da Romênia, o qual pode ser interpretado como um grafo do espaço de estados se considerarmos cada estrada como duas possíveis ações de dirigir, uma para cada sentido. Um **caminho** no espaço de estados é uma sequência de estados conectados por uma sequência de ações.
- **Teste de objetivo**, o qual determina se um dado estado é estado objetivo. Considerando o problema do agente na Romênia, o objetivo do agente é o estado *Em(Bucareste)*.

- **Custo do caminho** obtida por uma função que atribui um custo numérico a cada caminho. A função de custo é definida de acordo com a medida de desempenho do agente. Por exemplo, considerando o agente no problema da Romênia, para chegar a Bucareste, o tempo é fundamental, sendo assim, o custo de um caminho poderia ser o comprimento em quilômetros. O custo de um caminho pode ser definido como a soma dos custos das ações individuais ao longo do caminho. O **custo do passo** de escolher a ação s para atingir o estado s' é definido por $c(s, a, s')$. A Figura 2 ilustra os custos dos passos para a Romênia como distâncias de rotas.

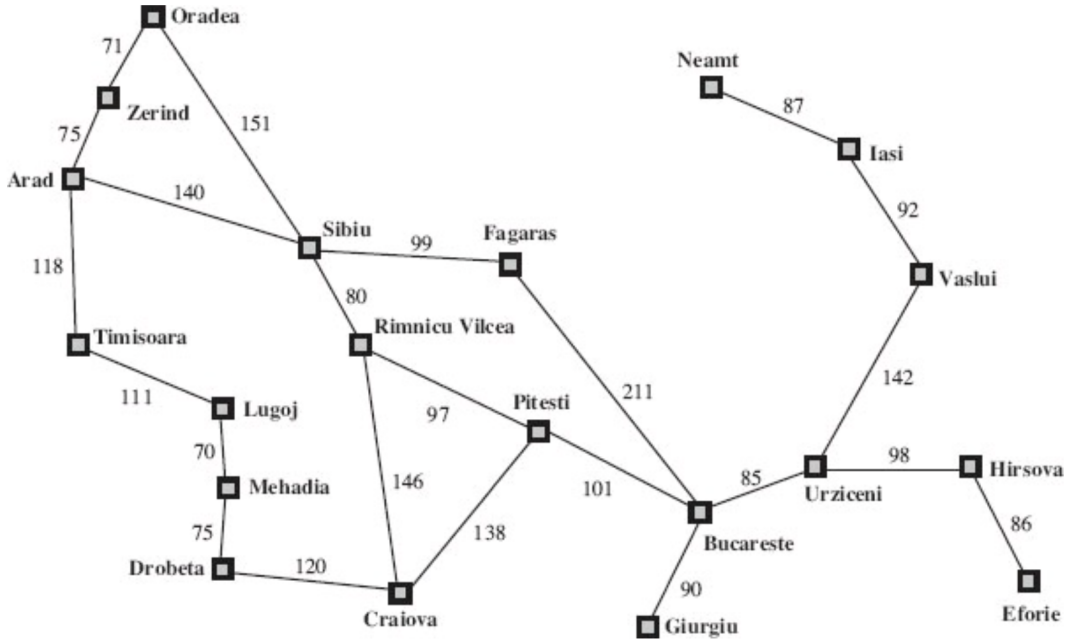


Figura 2: Mapa rodoviário simplificado de parte da Romênia

Os componentes mencionados definem um problema e podem ser reunidos em uma única estrutura de dados fornecida como entrada para um algoritmo de resolução de problemas. Nesse sentido, uma **solução** para um problema é um caminho desde o estado inicial até um estado objetivo. A função de custo de caminho pode mensurar a qualidade da solução, sendo uma **solução ótima** aquela que apresenta o menor custo de caminho entre todas as soluções.

1.1.2 Formulação de problemas

Para a formulação de um problema são especificados o estado inicial, ações, modelo de transição, teste de objetivo e custo de caminho. Considerando o problema do agente de chegar a Bucareste e comparando a descrição do estado definida, e.g. *Em(ARAD)*, a uma viagem real cruzando o país, o estado do mundo compreende muitos itens. Por exemplo, a quantidade de passageiros, o programa de rádio, as paisagens observadas da janela do carro, a proximidade e o número de postos policiais no caminho, a distância até a próxima parada para descanso, as condições da estrada e do tempo, entre outros. Tais itens são desconsiderados das descrições de estados, dado que não são relevantes para encontrar a solução do problema (i.e. encontrar uma rota para Bucareste).

A remoção de detalhes de uma representação é denominada **abstração**. Além da abstração na descrição do estado, as ações do agente também devem ser abstraídas. Por exemplo, a ação de direção envolve muitos itens (e.g. alteração da posição do veículo e dos passageiros, consumo de combustível, gasto de tempo, geração de poluição, mudança do agente, diminuição da velocidade ao aproximar de policiais, entre outros). No caso do problema do agente da Romênia, a formulação definida, com o processo de abstração, leva em consideração apenas a mudança de posição.

A escolha de uma boa abstração envolve a remoção da maior quantidade possível de detalhes. É responsabilidade do projetista estabelecer boas escolhas, de acordo com o problema a ser solucionado.

1.2 Exemplos de Problemas

O problema do **mundo do aspirador de pó** pode ser formulado da seguinte forma:

- **Estados:** determinados pela posição do agente e da sujeira. O agente está em uma entre duas posições, nas quais podem conter sujeira ou não. Sendo assim, existem $2x2^2 = 8$ estados do mundo possíveis
- **Estado inicial:** qualquer estado pode ser consderado estado inicial
- **Ações:** cada estado tem três ações possíveis: esquerda, direita e aspirar
- **Modeo de transição:** as ações apresentam seus efeitos esperados, exceto as ações de mover para a esquerda, no quadrado mais à esquerda; mover para a direita, no quadrado mais à direita, e aspirar, no quadrado limpo, as quais não têm nenhum efeito. O espaço de espaços completo é exibido na Figura 3
- **Teste de objetivo:** verifica se todos os quadrados estão limpos
- **Custo de caminho:** cada passo com custo 1, sendo o custo do caminho a quantidade de passos do caminho

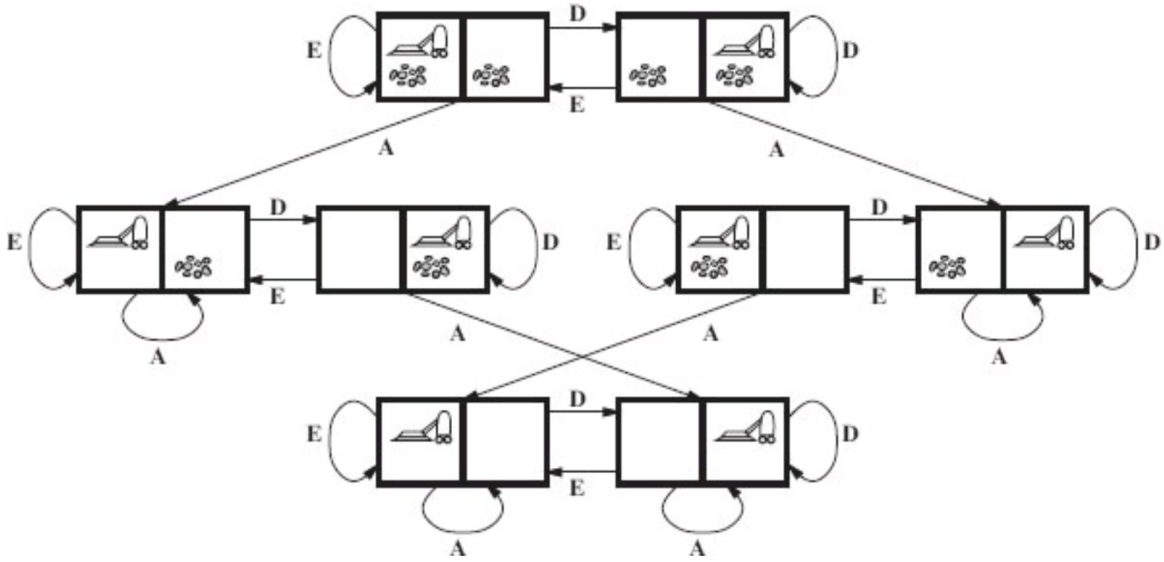
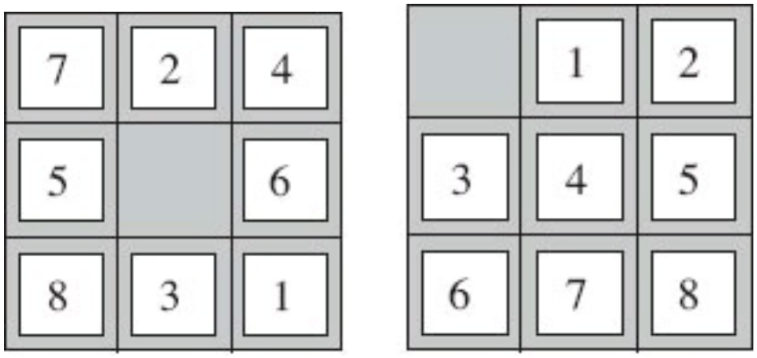


Figura 3: Espaço de estados para o mundo do aspirador de pó. As arestas indicam as ações (E = esqueda, D = direita e A = aspirar)

Outro exemplo de problema refere-se ao **quebra-cabeça de oito peças** (Figura 4), o qual consiste em um tabuleiro $3x3$ com oito peças numeradas e um quadrado vazio. Peças adjacentes ao quadrado vazio podem deslizar para esse quadrado. O objetivo é atingir um determinado estado objetivo, por exemplo, conforme apresentado pelo quebra-cabeça da direita na Figura 4.

- **Estados:** cada um representando a descrição da posição de cada uma das oito peças e do quadrado vazio
- **Estado inicial:** qualquer estado pode ser considerado estado inicial
- **Ações:** movimentos do quadrado vazio para esquerda, direita, para cima e para baixo
- **Modelo de transição:** gera um estado resultante, considerando um determinado estado e ação. Por exemplo, a partir da ação esquerda para o estado inicial na Figura 4, o estado resultante ocasionaria nas movimentações da peça 5 e do quadrado vazio.
- **Teste de objetivo:** verifica se o estado refere-se à configuração do estado objetivo
- **Custo de caminho:** cada passo com custo 1, sendo o custo do caminho a quantidade de passos do caminho



(a) (b)

Figura 4: Problema do quebra-cabeça de oito peças. (a) estado inicial. (b) estado objetivo

O **problema de oito rainhas** consiste em posicionar oito rainhas em um tabuleiro de xadrez, de forma que nenhuma rainha ataque outra. Uma rainha ataca outra situada na mesma linha, coluna ou diagonal. A Figura 5 ilustra uma solução que falhou, dado que a rainha do canto superior esquerdo ataca a rainha na coluna mais à direita.

Para o problema de oito rainhas, existem dois tipos de formulação. Em uma **formulação incremental**, cada ação insere uma rainha ao estado. Em uma formulação de estados completos têm-se todas as rainhas inseridas no tabuleiro e as mesmas são deslocadas no mesmo. Em ambas as formulações, o custo de caminho não tem nenhum interesse, dado que apenas o estado final é importante. Considerando a formulação incremental e a proibição de inserção de uma rainha em qualquer quadrado que já esteja sob ataque, o problema pode ser formulado da seguinte forma:

- **Estados:** a disposição de n rainhas, uma por coluna nas n colunas mais à esquerda, sem que nenhuma rainha ataque outra
- **Estado inicial:** nenhuma rainha no tabuleiro
- **Ações:** inserir uma rainha a qualquer posição na coluna vazia mais à esquerda, de forma que a mesma não seja atacada por qualquer outra rainha
- **Modelo de transição:** uma rainha adicionada em qualquer posição específica no tabuleiro
- **Teste de objetivo:** verifica se as oito rainhas estão no tabuleiro e nenhuma é atacada

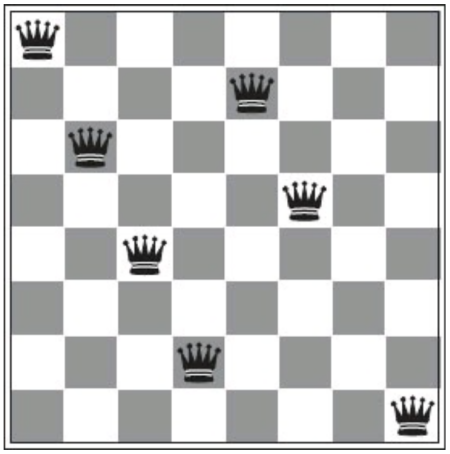


Figura 5: Problema das oito rainhas

1.3 Em Busca de Soluções

Após a formulação dos problemas, é necessário resolvê-los. Para solução uma sequência de ações deve ser executada. Os algoritmos de busca consideram diversas sequências de ações possíveis. As sequências de ações possíveis que iniciam a partir do estado inicial constituem uma **árvore de busca** com estado inicial na **raiz**; as **arestas** são as **ações**, e os **nós** referem-se aos estados no espaço de estados do problema.

Considerando o problema da Romênia, a Figura **??** ilustra árvores de busca parciais para encontrar uma rota desde Arad até Bucareste. O nó raiz árvore refere-se ao estado inicial – $Em(Arad)$. Inicialmente, é realizado o teste para verificar se é um estado objetivo. Em seguida, é necessário considerar a escolha de diversas ações. Nesse caso, o estado atual é expandido (i.e. aplica-se cada ação válida no estado atual), gerando um novo conjunto de estados. A partir do nó pai $Em(Arad)$ são adicionados três novas arestas, gerando três novos nós filhos – $Em(Sibiu)$, $Em(Timisoara)$ e $Em(Zerind)$. Então, deve-se escolher qual das três possibilidades de ações deve ser executada.

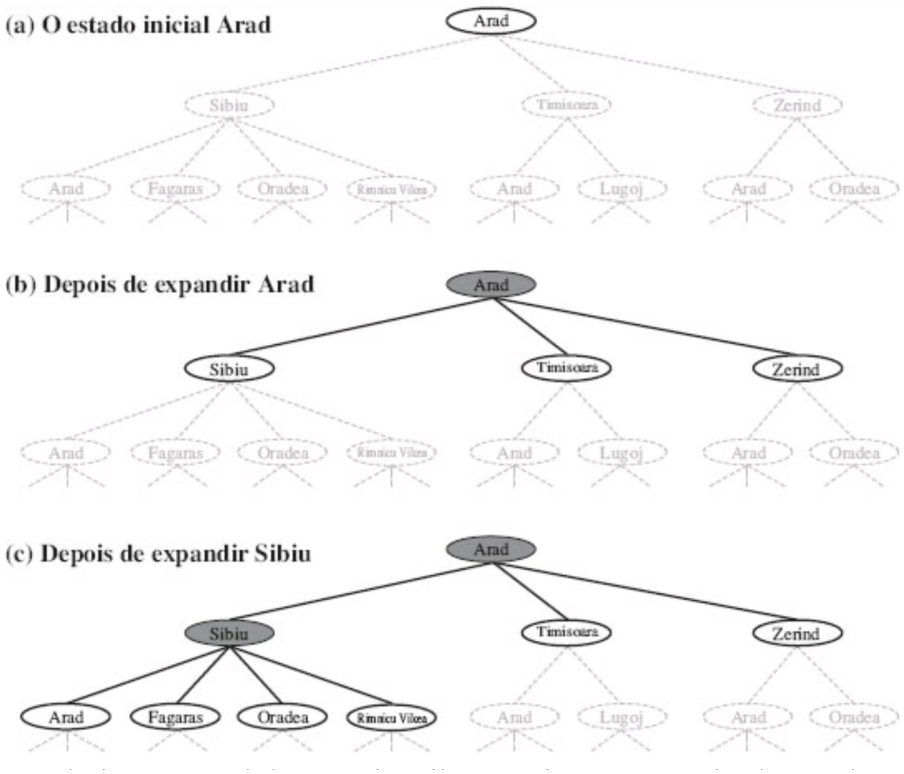


Figura 6: Árvores de busca parciais para encontrar uma rota desde Arad até Bucareste. Nós expandidos encontram-se sombreados; nós gerados mas não expandidos apresentam contorno em negrito; nós ainda não gerados são exibidos em linhas tracejadas

O processo de busca corresponde a seguir uma dada opção e deixar as demais para mais tarde, caso a primeira-opção não tenha levado a uma solução. Considerando o problema da Romênia e que a primeira escolha foi Sibiu, é verificado se o objetivo foi atingido. Nesse caso, não se trata de um estado objetivo, então o mesmo é expandido, gerando $Em(Arad)$, $Em(Fagaras)$, $Em(Oradea)$ e $Em(RimnicuVilcea)$. Qualquer uma dessas opções pode ser escolhida ou então retornar e escolher Timisoara ou Zerind. Cada um desses seis nós (Arad, Fagaras, Oradea, RiminicuVilcea, Timisoara e Zerind) corresponde a um nó folha (i.e. um nó sem filhos na árvore). O conjunto de todos nós folhas disponíveis para expansão em um dado momento é denominado **fronteira** (borda, lista aberta). A fronteira de cada árvore (na Figura 6) é representada pelos nós com contornos em negrito.

O processo de expansão dos nós na fronteira prossegue até que uma solução seja encontrada ou não existam mais estados para expandir. A Figura 7 apresenta o algoritmo geral BUSCA-EM-ÁRVORE. Todos os algoritmos de busca compartilham tal estrutura básica, variando de acordo com a forma escolhida para o próximo estado a expandir, i.e. a **estratégia de busca**.

```
função BUSCA-EM-ÁRVORE(problema) retorna uma solução ou falha
  inicializar a borda utilizando o estado inicial do problema
  repita
    se borda vazia então retornar falha
    escolher um nó folha e o remover da borda
    se o nó contém um estado objetivo então retornar a solução correspondente
    expandir o nó escolhido, adicionando os nós resultantes à borda

função BUSCA-EM-GRAFO(problema) retorna uma solução ou falha
inicializar a borda utilizando o estado inicial do problema
inicializar o conjunto explorado tornando-o vazio
repita
  se borda vazia então retornar falha
  escolher um nó folha e o remover da borda
  se o nó contiver um estado objetivo então retornar a solução correspondente
  adicionar o nó ao conjunto explorado
  expandir o nó escolhido, adicionando os nós resultantes à borda
    apenas se não estiver na borda ou no conjunto explorado
```

Figura 7: Algoritmo geral de busca em árvore e busca em grafo. As partes destacadas em negrito tratam estados repetidos

1.3.1 Infraestrutura para algoritmos de busca

Para controle da árvore de busca que está sendo construída, os algoritmos de busca requerem uma estrutura de dados. Nesse sentido, para cada nó n da árvore, tem-se uma estrutura que contém quatro componentes:

- n.ESTADO: estado no espaço de estado que o nó corresponde
- n.PAI: nó na árvore de busca que gerou tal nó
- n.AÇÃO: ação que foi aplicada ao pai para gerar o nó
- n.CUSTO-DO-CAMINHO: custo, g(n), do caminho do estado inicial até o nó, indicado pelos ponteiros para os pais

A Figura 8 apresenta a estrutura de dados do nó. Os ponteiros do PAI encadeiam os nós juntos em uma estrutura de árvore. Tais ponteiros possibilitam também que o caminho da solução seja extraído quando é encontrado um nó objetivo, utilizando a função SOLUÇÃO para retornar a sequência de ações obtidas seguindo os ponteiros do pai de volta para a raiz.

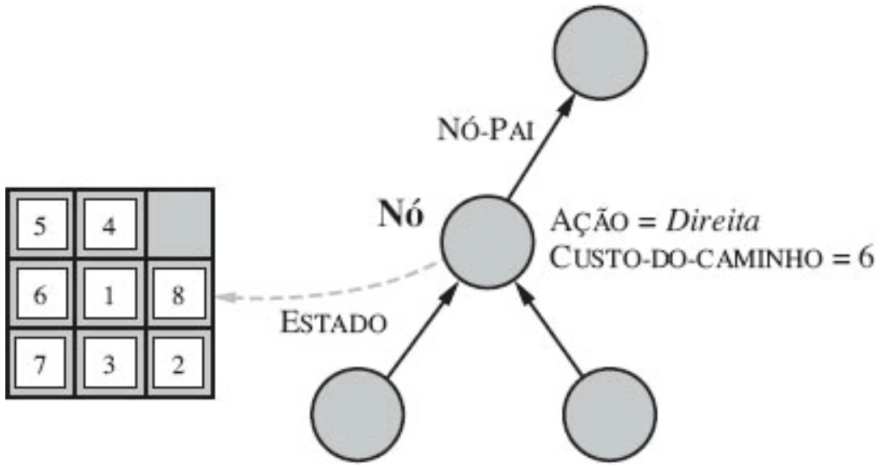


Figura 8: Estruturas de dados de um nó, a partir das quais a árvore de busca é construída. Cada nó apresenta um pai, um estado e outros campos de anotação. As setas (ponteiros) apontam do filho para o pai

É importante realizar a distinção entre nós e estados. Um nó corresponde a uma anotação da estrutura de dados utilizada para representar a árvore de busca. Um estado refere-se a uma configuração do mundo. Além disso, dois

nós diferentes podem representar o mesmo estado do mundo se esta estado for gerado por meio de dois caminhos de busca diferentes.

Definido os nós, é importante definir onde inserí-los. A fronteira precisa ser armazenada, para que o algoritmo de busca escolha o próximo nó para expansão de acordo com a estratégia de busca adotada. A estrutura de dados adequada é uma **fila**, sendo as operações sobre a mesma:

- VAZIA?(fila): retorna verdadeiro se não existir mais nenhum elemento na fila
- POP(fila: remove o primeiro elemento da fila e o retorna
- INSERIR(elemento, fila): insere um elemento na fila e retorna a fila resultante

As filas são caracterizadas pela ordem em que armazenam os nós inseridos, sendo três variantes mais comuns (fila FIFO, fila LIFO – pilha e fila de prioridade). **Fila FIFO** (first in first out), o primeiro a entrar na fila é o primeiro a sair. **Fila LIFO** (last in first out), o último a entrar na fila é o último a sair. **Fila de prioridade** referente ao elemento da fila com a maior prioridade de acordo com uma dada função de ordenação.

1.3.2 Medição de desempenho de resolução de problemas

Antes do projeto de algoritmos de busca específicos, é importante considerar os critérios para realizar comparações e escolha entre os mesmos. O desempenho dos algoritmos podem ser avaliados de acordo com quatro aspectos (completeza, otimização, complexidade de tempo e complexidade de espaço).

- **Completeza:** o algoritmo encontra uma solução quando a mesma existir?
- **Otimização:** o algoritmo encontra a solução ótima?
- **Complexidade de tempo:** quanto tempo o algoritmo leva para encontrar uma solução?
- **Complexidade de espaço:** quanta memória é necessária para execução da busca?

As complexidades de tempo e de espaço de memória são consideradas em relação a uma medida da dificuldade do problema. A medida típida é o tamanho do grafo do espaço de estados. Sendo assim, a complexidade é considerada em termos de três quantidades: b, o **fator de ramificação** (quantidade máxima de sucessores de qualquer nó); d, a **profundidade** do nó objetivo mais raso – menos profundo (quantidade de passos ao longo do caminho da raiz até o estado objetivo mais próximo); e m, o **comprimento** máximo de qualquer caminho no espaço de estados. Em geral, o tempo é medido em relação à quantidade de nós gerados durante a busca, e o espaço é medido em relação à quantidade máxima de nós armazenados na memória.