QXD0037 - Inteligência Artificial

Laboratório 02 - Busca sem Informação

Profa. Dra. Viviane Menezes

Data: 25.02.2021

1 Objetivo

O objetivo desta atividade prática é implementar os algoritmos de busca sem informação (*busca em largura*, *busca de custo uniforme* ou *busca em profundidade*) para solucionar o problema do Mapa Rodoviário da Romênia.)

2 Regras

- A atividade deve ser feita em dupla.
- Cada dupla deve entregar um único arquivo compactado (formato zip), contendo a implementação e a tabela de experimentos.

3 Agentes de Resolução de Problemas

Agentes de resolução de problemas são agentes baseados em objetivos. A resolução de problemas inicia com uma formulação precisa do problema e utilização de algoritmos de busca para solucionar o problema.

Suponha um agente na cidade de Arad, na Romênia, deseje chegar a Bucharest, utilizando como informação o mapa odoviário simplificado da Romênia mostrado na Figura 1.

4 Em Busca de uma Solução

Depois de formular o problema, é preciso resolvê-lo. Uma solução é uma sequência de ações possíveis que começam a partir do estado inicial formam uma **árvore de busca**, enraizada no estado inicial, com nós correspondendo aos estados no espaço de estados do problema e os ramos correspondendo às ações do problema. Um nó na árvore de busca é uma estrutura composta pelos seguintes elementos: o estado no espaço de estados a que o nó corresponde; o

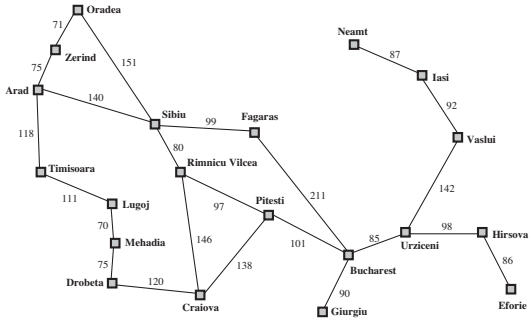

Figura 1: Mapa rodoviário simplificado de parte da Romênia [Russell and Norvig, 2010].

pai na árvore de busca; a ação que foi aplicada para geração deste nó e; o custo de caminho que é o custo de sair do estado inicial e chegar até o nó.

A partir de cada nó, considera-se todas as poss??veis ações aplicáveis a este nó e é feita uma **expansão** gerando nós filhos. O conjunto de todos os nós folhas dispon??veis para a expansão é chamado de **borda**. O processo de expansão dos nós na borda continua até que uma solução seja encontrada ou não esxistam mais estados a expandir. Todos os algoritmos de buscam compartilham essa estrutura básica. Eles variam na escolha do próximo nó a ser expandido.

```
01. BUSCA-EM-LARGURA(problema){
02.   /* borda com nó.estado inicial e explorados vazio */
03.    repita
04.      se borda está vazia
05.           retorne falha
06.      nó ← remover elemento da borda  /*estrutura: fila*/
07.      adicionar nó.estado a explorados
08.      para cada ação aplicável em nó.estado
09.           filho ← criar nó filho
10.           se filho.estado não está em explorados ou borda
11.                se filho.estado é objetivo
12.                     retorne solução
13.                adicionar filho em borda
14. }
```

Figura 2: Pseudocódigo da Busca em Largura, adaptado de [Russell and Norvig, 2010].

4.1 Busca em Largura

O pseudocódigo do algoritmo `Busca-Em-Largura` é apresentado na Figura 2. Ele recebe como entrada um problema e retorna como sa??da a sequência de

ações que leva o agente do estado inicial a um estado objetivo. A busca em largura implementa a borda como uma fila FIFO (*First In, First Out*).

4.2 Busca de Custo Uniforme

O pseudocódigo do algoritmo `BUSCA-DE-CUSTO-UNIFORME` é apresentado na Figura 3. Ele recebe como entrada um problema e retorna como sa??da a sequência de ações que leva o agente do estado inicial a um estado objetivo. A busca de custo uniforme implementa a borda como uma fila de prioridades, ordenada pelo custo de caminho.

```
01. BUSCA-DE-CUSTO-UNIFORME(problema){
02.    /* borda com nó.estado inicial e explorados vazio */
03.     repita
04.       se borda está vazia retorne falha
05.       nó ← remover elemento da borda /*estrutura: fila de prioridades */
06.       se nó.estado é objetivo retorne solução
07.       adicionar nó.estado a explorados
09.       para cada ação aplicável em nó.estado
10.           filho ← criar nó filho
11.           se filho.estado não está em explorados ou borda
12.                 adicionar filho em borda
13.           senão se filho.estado está na borda com maior custo
14.                 substituir nó borda por filho
15. }
```

Figura 3: Pseudocódigo da Busca de Custo Uniforme, adaptado de [Russell and Norvig, 2010].

4.3 Busca em Profundidade

O pseudocódigo do algoritmo `BUSCA-EM-PROFUNDIDADE` é apresentado na Figura 4. Ele recebe como entrada um problema e retorna como sa??da a sequência de ações que leva o agente do estado inicial a um estado objetivo. A busca em profundidade implementa a borda como uma fila LIFO (*Last In, Last Out*), também conhecida como pilha.

5 Implementação

Você deve implementar um dos algoritmos de busca sem informação para resolver o problema do mapa rodoviário da Romênia no qual um agente em uma determinada cidade inicial deseja deslocar-se até *Bucharest*.

- Seu programa deve receber a formulação do problema e devolver a sequência de ações necessárias para que o agente saia da cidade inicial e chegue a Bucharest.

```
01. BUSCA-EM-PROFUNDIDADE(problema){
02.   /* borda com nó.estado inicial e explorados vazio */
03.    repita
04.      se borda está vazia
05.           retorne falha
06.      nó ← remover elemento da borda  /*estrutura: pilha*/
07.      adicionar nó.estado a explorados
08.      para cada ação aplicável em nó.estado
09.           filho ← criar nó filho
10.           se filho.estado não está em explorados ou borda
11.                se filho.estado é objetivo
12.                     retorne solução
13.                adicionar filho em borda
14. }
```

Figura 4: Pseudicódigo da Busca em Profundidade, adaptado de [Russell and Norvig, 2010]..

- Você deve implementar o mapa da romênia como um sistema de transição de estados em que os estados são cidades e as transições são as estradas, *rotuladas* com a distância entre as cidades. Logo, a estrutura de dados que representa este mapa deve ser um grafo cujas arestas contém *pesos* em seus rótulos e são *não-direcionadas*.
- Considerar as representações possíveis para grafos tais como: matriz ou listas de adjacências.

6 Experimentos

Você deve realizar experimentos considerando 10 problemas do mapa rodoviário da Romênia com diferentes cidades de origens (conforme especificado na Tabela 6.2) e com o destino *Bucharest*. Para cada problema, você deve anotar o tempo de execução e o custo a solução para a busca escolhida.

6.1 Medindo o Tempo de Execução

Para medir o tempo de execução, use o comando `time` (do Sistema Operacional Linux) que executa o programa e depois disso mostra o(s) tempo(s) consumido(s). Por exemplo, o comando

```
% time ./RomaniaMap BuscaEmLargura Arad Bucarest

real    0m0.032s
user    0m0.030s
sys     0m0.010s
```

executa o programa `RomaniaMap` que executará a *busca em largura* para encontrar a sequência de ações necessárias para sair de *Arad* para *Bucarest*. O tempo que você deve considerar é o `user time`.

6.2 Construindo a tabela

A tabela deve informar a origem de cada problema, o tempo gasto para resolver o problema e o custo total da solução. Veja modelo na Tabela 6.2.

Problema	**Origem**	**Custo da Solução**	**Tempo Gasto**
01.	Neamt		
02.	Eforie		
03.	Lugoj		
04.	Arad		
05.	Vaslui		
06.	Oradea		
07.	Iasi		
08.	Timisoara		
09.	Zerind		
10.	Hirsova		

Tabela 1: Tabela de Experimentos para problemas do Mapa Rodoviário da Romênia.

Referências

[Russell and Norvig, 2010] Russell, S. and Norvig, P. (2010). *Artificial Intelligence*. Elsevier, 3a edition.