QXD0037 - Inteligência Artificial

Laboratório 02 - Agentes de Resolução de Problemas

Profa. Dra. Viviane Menezes

Data: 20.09.2017

1 Objetivos

Os objetivos desta atividade prática são formular o problema do mapa rodoviário da Romênia e implementar três estratégias de busca (*busca em largura*, *busca de custo uniforme* e *busca em profundidade*) para solucioná-lo.

2 Regras

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

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.

3.1 Formulação do problema

Formule o problema do agente sair da cidade de Arad e chegar até a cidade de Bucharest. Para formular tal problema você deve especificar: o estado inicial do problema, as ações que agente pode executar, o modelo de transição de estados, o teste de objetivo e o custo de caminho.

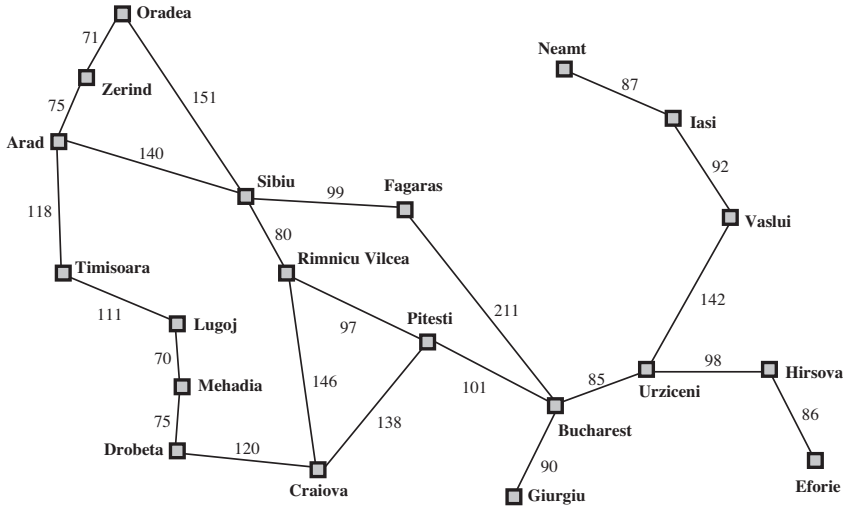

Figura 1: Mapa rodoviário simplificado de parte da Romênia [Russell and Norvig, 2010].

3.2 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 composto pelos seguintes elementos: o estado no espaço de estados a que o nó corresponde; o 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 possíveis ações aplicáveis a este nó e é feita uma **expansão** gerando nós filhos. O conjunto de todos os nós folhas disponíveis 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.

3.2.1 Busca em Largura

O pseudocódigo do algoritmo `Busca-Em-Largura` é apresentado na Figura 3.2.3. Ele recebe como entrada um problema e retorna como saída 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*)

```
01. BUSCA-EM-LARGURA(problema){
02.       nó.estado ← estado inicial do problema
03.       nó.custoDeCaminho ← 0
04.       se nó.estado é um estado objetivo então
05.            retorne solução
06.       borda ← uma fila FIFO contendo apenas nó
07.       explorados ← {}
08.       repita
09.            se borda está vazia então
10.                 retorne falha
11.            nó ← remover(borda)
12.            adicionar nó.estado a explorados
13.            para cada ação aplicável
14.                 filho ← criarNó(problema, nó, ação)
15.                 se filho.estado não está em explorados então
16.                      se filho.estado é objetivo então
17.                           retorne solução
18.                      inserir(filho, borda)
19. }
```

Figura 2: Busca em largura em um grafo, adaptado de [Russell and Norvig, 2010].

3.2.2 Busca de Custo Uniforme

O pseudocódigo do algoritmo `BUSCA-DE-CUSTO-UNIFORME` é apresentado na Figura 3.2.2. Ele recebe como entrada um problema e retorna como saída 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.

3.2.3 Busca em Profundidade

O pseudocódigo do algoritmo `BUSCA-EM-PROFUNDIDADE` é apresentado na Figura **??**. Ele recebe como entrada um problema e retorna como saída 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.

4 Implementação

Você deve implementar os algoritmos de `BUSCA-EM-LARGURA`, `BUSCA-DE-CUSTO-UNIFORME` e `BUSCA-EM-PROFUNDIDADE` para resolver o problema do mapa rodoviário da Romênia no qual um agente inicialmente na cidade de Arad deseja deslocar-se até Bucarest. Siga as seguintes especificações:

```
01. BUSCA-DE-CUSTO-UNIFORME(problema){
02.       nó.estado ← estado inicial do problema
03.       nó.custoDeCaminho ← 0
04.       borda ← uma fila de prioridades, contendo apenas nó
05.       explorados ← {}
06.       repita
07.            se borda está vazia então
08.                 retorne falha
09.            nó ← remover(borda)
10.            se no.estado é objetivo então
11.                 retorne solução
12.            adicionar nó.estado a explorados
13.            para cada ação aplicável em nó
14.                 filho ← criarNó(problema, nó, ação)
15.                 se filho.estado não está na borda ou em explorados
16.                      inserir(filho, borda)
17.                 senão se filho.estado está na borda com maior custo
18.                      substituir nó borda por filho
19. }
```

Figura 3: Busca de custo uniforme em um grafo, adaptado de [Russell and Norvig, 2010].

- Nome do Projeto: `Search`
- O projeto deve conter pelo menos as seguintes classes:
  - `State`: descreve um estado do mundo.
  - `Node`: descreve um nó na árvore de busca.
  - `Action`: descreve uma ação.

Seu programa deve receber a formulação do problema (pense em uma representação possível para o mapa rodoviário na Romênia) e devolver a sequência de ações necessárias para que o agente saia de Arad e chegue a Bucarest.

5 Experimentos

Você deve realizar experimentos considerando pelo menos 10 problemas do mapa rodoviário da Romênia com diferentes origens e destinos. Para cada problema, você deve anotar o tempo de execução para a busca em largura, depois para a busca de custo uniforme e em seguida para a busca em profundidade. Sugere-se que o seu programa receba como entrada o problema a ser resolvido e o tipo de busca a ser executada.

```
01. BUSCA-EM-PROFUNDIDADE(problema){
02.       nó.estado ← estado inicial do problema
03.       nó.custoDeCaminho ← 0
04.       se nó.estado é um estado objetivo então
05.            retorne solução
06.       borda ← uma fila LIFO contendo apenas nó
07.       explorados ← {}
08.       repita
09.            se borda está vazia então
10.                 retorne falha
11.            nó ← remover(borda)
12.            adicionar nó.estado a explorados
13.            para cada ação aplicável
14.                 filho ← criarNó(problema, nó, ação)
15.                 se filho.estado não está em explorados então
16.                      se filho.estado é objetivo então
17.                           retorne solução
18.                      inserir(filho, borda)
19. }
```

Figura 4: Busca em profundidade em um grafo.

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

```
% time ./Search bfs problema1.txt > /dev/null

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

executa o programa `Search` que fará uma busca em largura (`bfs` - *breadth first search*) no problema especificado no arquivo `problema1.txt`, desviando toda a saída para `/dev/null` (ou seja, não mostra nada da saída do programa). No final, o programa time mostra que o programa levou 32ms para executar. Só que, destes, apenas 30ms fora usados pelo programa, (em operação do usuário). O resto foi gasto em ciclos do sistema. O fato que `real` > `user + sys` indica que o programa time não tem uma precisão assim tão grande. O tempo que você deve considerar é o `user time`.

6 Relatório

Pede-se gerar um relatório de no máximo 5 páginas (no formato pdf) contendo os seguintes tópicos:

- o título (*Resolução de Problemas por Meio de Busca*) e os nomes dos autores;
- uma introdução;
- uma descrição da formulação do problema;
- uma descrição da implementação dos algoritmos de busca;
- uma descrição dos experimentos, incluindo as plataformas de hardware e software nais quais os experimentos foram realizados;
- um gráfico de tempo plotado para os 10 problemas e para os três tipos de busca e uma pequena discussão do que pode-se inferir a partir do gráfico e;
- uma conclusão que responda a pergunta: qual algoritmo de busca é mais apropriado para resolver o problema.

Tanto o relatório quanto o programa devem ser inseridos num único arquivo do tipo zip e submetidos por meio do moodle.

Referências

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