Inteligência Artificial

Aula 5

Profª Bianca Zadrozny
http://www.ic.uff.br/~bianca/ia

Busca com informação e exploração

Capítulo 4 – Russell & Norvig

Seção 4.1

Busca com informação (ou heurística)

- Utiliza conhecimento específico sobre o problema para encontrar soluções de forma mais eficiente do que a busca cega.
  - Conhecimento específico além da definição do problema.
- Abordagem geral: busca pela melhor escolha.
  - Utiliza uma função de avaliação para cada nó.
  - Expande o nó que tem a função de avaliação mais baixa.
  - Dependendo da função de avaliação, a estratégia de busca muda.

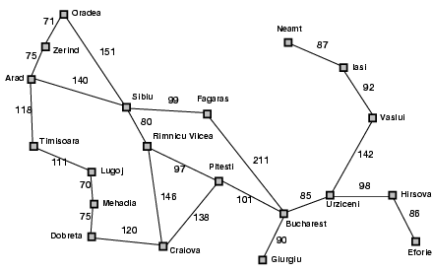
Busca pela melhor escolha

- Idéia: usar uma função de avaliação *f(n)* para cada nó.
  - estimativa do quanto aquele nó é desejável
  - → Expandir nó mais desejável que ainda não foi expandido
- Implementação:
  Ordenar nós na borda em ordem decrescente de acordo com a função de avaliação
- Casos especiais:
  - Busca gulosa pela melhor escolha
  - Busca A*

Busca gulosa pela melhor escolha

- Função de avaliação *f(n) = h(n)* (heurística)
  = estimativa do custo de *n* até o objetivo
  ex., $h_{DLR}(n)$ = distância em linha reta de *n* até Bucareste.
- Busca gulosa pela melhor escolha expande o nó que parece mais próximo ao objetivo de acordo com a função heurística.

Romênia com custos em km



Distância em linha reta para Bucareste

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vasiui	199
Zerind	374

Exemplo de busca gulosa pela melhor escolha

Exemplo de busca gulosa pela melhor escolha

Exemplo de busca gulosa pela melhor escolha

Exemplo de busca gulosa pela melhor escolha

Busca gulosa pela melhor escolha

- Não é ótima, pois segue o melhor passo **considerando somente o estado atual.**
  - Pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca.
- Minimizar h(n) é suscetível a falsos inícios.
  - Ex. Ir de Iasi a Fagaras
    - Heurística sugerirá ir a Neamt, que é um beco sem saída.
    - Se repetições não forem detectadas a busca entrará em loop.

Propriedades da busca gulosa pela melhor escolha

- Completa? Não – pode ficar presa em loops, ex., Iasi → Neamt → Iasi → Neamt
- Tempo? $O(b^m)$ no pior caso, mas uma boa função heurística pode levar a uma redução substancial
- Espaço? $O(b^m)$ – mantém todos os nós na memória
- Ótima? Não

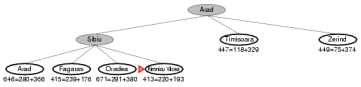
Busca A*

- Idéia: evitar expandir caminhos que já são caros
- Função de avaliação *f(n) = g(n) + h(n)*
  - *g(n)* = custo até o momento para alcançar *n*
  - *h(n)* = custo estimado de *n* até o objetivo
  - *f(n)* = custo total estimado do caminho através de *n* até o objetivo.

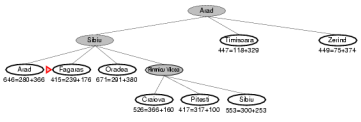
Exemplo de busca A*

Exemplo de busca A*

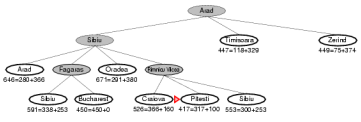
Exemplo de busca A*



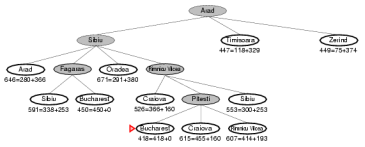
Exemplo de busca A*



Exemplo de busca A*



Exemplo de busca A*



Heurística Admissível

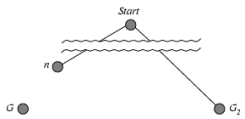
- Uma heurística *h(n)* é admissível se para cada nó *n*, $h(n) \leq h^*(n)$, onde $h^*(n)$ é o custo verdadeiro de alcançar o estado objetivo a partir de *n*.
- Uma heurística admissível nunca superestima o custo de alcançar o objetivo, isto é, ela é otimista.
- Exemplo: $h_{DLR}(n)$ (distância em linha reta nunca é maior que distância pela estrada).
- Teorema: Se *h(n)* é admissível, A* usando algoritmo BUSCA-EM-ARVORE é ótima.

Prova que A* é ótima com heurística admissível

- Assuma um **nó objetivo não-ótimo** G_2, e seja C* o custo da solução ótima. Então, como G_2 não é ótimo e $h(G_2) = 0$, sabemos que:

  $$\mathbf{f(G_2) = g(G_2) + h(G_2)}$$
  $$\mathbf{= g(G_2) > C^*}$$

- Considere qualquer nó de borda **n** que esteja num caminho de solução ótimo. Se h(n) ***não superestimar o custo de completar o caminho de solução***, então: **f(n) = g(n) + h(n) ≤ C*.**



Prova que A* é ótima com heurística admissível (cont.)

- Logo, se $f(n) \leq C^* < f(G_2)$, G_2 não será expandido e A* deve retornar uma solução ótima.
- Isso vale para busca em árvore, para outras estruturas de busca pode não valer.
- Na busca em grafos temos que assegurar que o caminho ótimo para qualquer estado repetido seja o primeiro a ser seguido.
  - Requisito extra para h(n): consistência

Consistência (ou monotonicidade)

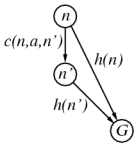
- Uma heurística é consistente (ou monotônica) se para cada nó *n*, cada sucessor *n'* de *n* gerado por qualquer ação *a*,

  $h(n) \leq c(n,a,n') + h(n')$

- Se *h* é consistente, temos

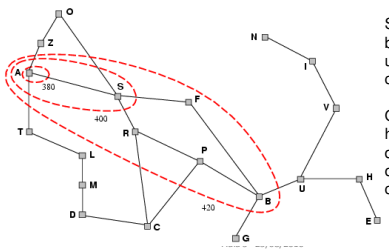
  $f(n') = g(n') + h(n')$
  $= g(n) + c(n,a,n') + h(n')$
  $\geq g(n) + h(n)$
  $= f(n)$

- Isto é, *f(n)* is não-decrescente ao longo de qualquer caminho.
- Teorema: Se *h(n)* is consistente, A* usando BUSCA-EM-GRAFOS é ótima.



A* é ótima com heurística consistente

- A* expande nós em ordem crescente de valores de *f*.
- Gradualmente adiciona "*contornos*" de nós.
- Contorno *i* tem todos os nós com $f=f_i$, onde $f_i < f_{i+1}$



Se h(n)=0 temos uma busca de custo uniforme ⇒ círculos concêntricos.

Quanto melhor a heurística mais direcionados ao objetivo serão os círculos

Propriedades da Busca A*

- Completa? Sim (a não ser que exista uma quantidade infinita de nós com f ≤ *f(G)*)
- Tempo? Exponencial no pior caso
- Espaço? Mantém todos os nós na memória
- Ótima? Sim
- Otimamente eficiente
  - Nenhum outro algoritmo de busca ótimo tem garantia de expandir um número de nós menor que A*. Isso porque qualquer algoritmo que não expande todos os nós com f(n) < C* corre o risco de omitir uma solução ótima.