Busca de custo uniforme

Diferente da busca por largura, que visa expandir o nó mais raso não explorado, a busca de custo uniforme trata do nó ‘n’ com o custo de caminho g(n) mais baixo. Isso é feito através do armazenamento da borda como uma fila de prioridade ordenada por g.

```
função BUSCA-DE-CUSTO-UNIFORME(problema)
  retorna uma solução ou falha

  nó ← um nó com ESTADO = problema.ESTADO-INICIAL, CUSTO-DE-CAMINH

  borda ← uma fila FIFO com nó como elemento único
  explorado ← conjunto vazio

  repita
   se VAZIO?(borda) então
    retorne falha

   nó ← POP(borda)  /* escolhe o nó mais raso na borda */
   se problema.TESTE-DE-OBJETIVO(nó.ESTADO) então
   retorne SOLUÇÃO(nó)
   adicione nó.ESTADO para explorado

   para cada ação em problema.AÇÕES(nó.ESTADO) faça
    filho ← NÓ-FILHO(problema, nó, ação)

    se (filho.ESTADO) não está em explorado ou borda então
          borda ← INSIRA(filho, borda)
        senao se (filho.ESTADO) está na borda com o maior CUSTO-DE-CAMI
           substituir aquele nó borda por filho
```

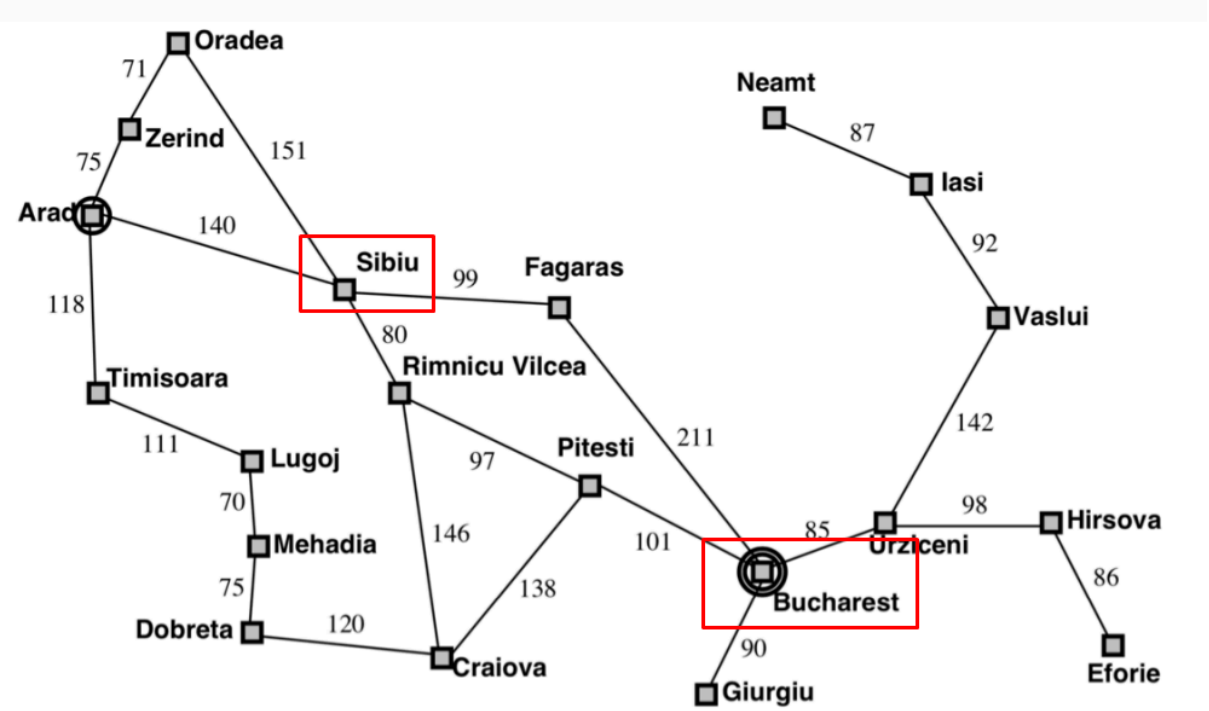
O algoritmo é idêntico ao de busca por largura original, exceto pelo uso de uma fila de prioridade e pela adição de uma verificação extra. A estrutura de dados para a borda deve permitir “os testes eficientes de pertinência em conjunto, por isso deve combinar os recursos de uma fila de prioridade (custo de caminho) e de uma tabela hash”.

Diferenças significativas da busca em largura

Além da ordem da fila por custo de caminho, existem duas outras diferenças:

- o teste de objetivo é aplicado ao nó quando ele é selecionado para a expansão e não quando é gerado pela primeira vez
- é adicionado um teste, caso seja encontrado um caminho melhor para um nó atualmente na borda

Exemplo de ‘Sibiu’ para ‘Bucareste’



- os sucessores de Sibiu são Rimnicu Vilcea e Fagaras (custos 80 e 99, respectivamente)
- o nó de menor custo, Rimnicu Vilcea, será o próximo a ser expandido, acrescentando Pitesti com custo 80 + 97 = 177
- o nó de menor custo agora é Fagaras, por isso será expandido, acrescentando a Bucareste com custo 99 + 211 = 310
- agora foi gerado um nó objetivo, mas a busca de custo uniforme se mantém, escolhendo Pitesti para expansão e adicionando um segundo caminho para Bucareste com custo 80 + 97 + 101 = 278

- o algoritmo verifica que esse novo caminho (278) é melhor que o antigo, isto é, de modo que o antigo seja descartado
- Bucareste, agora com o custo g 278, será selecionada para a expansão e a solução será devolvida

---

Importante

- a busca de custo uniforme é orientada a custos de caminho em vez de profundidade; assim, sua complexidade não pode ser caracterizada em termos de profundidade e ramificação
- considerando custos de caminho não negativos, os caminhos nunca ficam menores à medida que os nós são adicionados. O primeiro nó objetivo que foi selecionado para a expansão deverá ser a solução ótima
- a completeza será garantida se o custo de cada passado exceder uma constante positiva pequena ‘e’
- a busca tem mais trabalho, pois além de encontrar o objetivo, verifica se não existe caminho melhor quando atingido