Inteligência Artificial

- Algoritmos -

Marco Martins

Tipos de Procura

- **Procura cega**
- **Procura heurística**

Procura cega

- **Algoritmos:**

- **Procura em Largura Primeiro**
- **Procura em Profundidade Primeiro**
- **Procura de custo uniforme**
- **Procura em Profundidade Limitada**
- **Procura em Profundidade Iterativa**

Procura Heurística

- Algoritmos:

- **Procura Sôfrega**
- **Procura A***
- **Pesquisa IDA***
- **Pesquisa SMA***

Procura cega

Encontram soluções para problemas pela geração *sistemática* de novos estados, que são comparados ao objetivo

Procura em Largura Primeiro

Breadth-First Search (BFS)

```
Função ProcuraLarguraPrimeiro (problema, InsereFila) : solução ou falha
        1. l_nós ← Faz fila ( EstadoInicial(problema) )
        2. Repete
                2.1. Se FilaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraFila(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsererFila(l_nós, Expansão(nó, Operadores(problema)))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

Procura em Profundidade Primeiro

Depth-First Search (DFS)

```
Função ProcuraPrefundidadePrimeiro (problema, InserePilha) : solução ou falha
        1. l_nós ← Faz Pilha ( EstadoInicial(problema) )
        2. Repete
                2.1. Se PilhaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraPilha(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsererPilha(l_nós, Expansão(nó, Operadores(problema)))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

Procura de custo uniforme

uniform-cost search (UCS)

```
Função ProcuraCustoUniforme (problema, InsereOrdem_Fila) : solução ou falha
        1. l_nós ← Faz fila ( EstadoInicial(problema) )
        2. Repete
                2.1. Se FilaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraFila(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsererOrdem_Fila(
                                l_nós, Expansão(nó, Operadores(problema)))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

-Coloca os elementos de menor custo à cabeça da fila

Procura em Profundidade Limitada

depth-limited search

```
Função ProcuraProfundidadeLimitada (problema, InserePilfa, nivel_max) : solução ou falha
        1. l_nós ← Faz Pilha ( EstadoInicial(problema) )
        2. Repete
                2.1. Se PilhaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraPilha(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsererPilha(
                                l_nós, Expansão(nó, OperadoresNmx(problema)))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

-Neste caso só aplica os operadores se o nivel for inferior ao nivel_max

Procura em Profundidade Iterativa

Iterative deepening depth-first search (IDDFS)

Função ProcuraAprofundamentoProgressivo (problema, InserePilha) : solução ou falha
1. **Para** nivel ← 0 **Até** infinito **Faz**
1.1. **Se** ProcuraProfundidadeLimitada (problema, InserePilha, nivel) **Então**
2.1.1. **Devolve** solução
Fim_de_Se
Fim_de_Para
Fim_de_Função

Procura Heurística

Utilizam conhecimento específico do problema na escolha do próximo nó a ser expandido.

Algoritmo de procura sôfrega

greedy algorithm

```
Função ProcuraSôfrega (problema, InsereListaOrdenada, Heuristica) : solução ou falha
        1. l_nós ← FazListaOrdenada ( EstadoInicial(problema) )
        2. Repete
                2.1. Se ListaOrdenadaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraListaOrdenada(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsereListaOrdenada(
                                l_nós, Heuristica(Expansão(nó, Operadores(problema))))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura A*

A*

```
Função A*  (problema, InsereListaOrdenada, g+h) : solução ou falha
        1. l_nós ← FazListaOrdenada ( EstadoInicial(problema) )
        2. Repete
                2.1. Se ListaOrdenadaVazia(l_nós) Então
                        2.1.1. Devolve falha
                   Fim_de_Se
                2.2. nó ← RetiraListaOrdenada(l_nós)
                2.3. Se TesteObjectivo(nó) Então
                        2.3.1. Devolve nó
                   Senão
                        2.3.2. InsereListaOrdenada(
                                l_nós, g+h(Expansão(nó, Operadores(problema))))
                   Fim_de_Se
           Fim_de_Repete
Fim_de_Função
```

Algoritmo de procura IDA*

```
Função IDA*  (problema) : solução ou falha
        1. f_limite ← f ( EstadoInicial(problema) )
        2. Repete
                2.1.  f_limite_ant ← f_limite
                2.2  f_Limite ← PppLimite ( EstadoInicial(problema), f_limite)
          Até sucesso ou f_limite sem alteração
        3. Se f_limite = sucesso Então
                3.1. Devolve nó
          Senão
                3.2. Devolve falha
          Fim_de_Se
Fim_de_Função
Função PppLimite (nó, limite): sucesso ou valor de limite
        1. Se f(nó) > limite Então
                1.1. Devolve f(nó)
          Fim_de_Se
        2. Se TesteObjectivo (nó) Então
                2.1. Devolve sucesso
          Senão
                2.2. Devolve Minimo (PppLimite (n, limite), para n € Sucessores (nó))
          Fim_de_Se
Fim_de_Função
```

Algoritmo de pesquisa SMA*

Função SMA* (problema): solução ou falha
1. I_nós ← FazFilaOrdenada(EstadoInicial(problema))
2. **Repete**
   2.1. **Se** FilaVazia(I_nós) **Então**
      2.1.1. **Devolve** falha
   Fim_de_Se
   2.2. nó ← RetiraFilaMelhor(I_nós)
   2.3. **Se** TesteObjectivo(nó) **Então**
      2.3.1. **Devolve** solução
   Fim_de_Se
   2.4. Sn ← ProximoSucessor(nó)
   2.5. f(Sn) ← max(f(nó), g(Sn)+h(Sn))
      ou infinito se estiver à profundidade maxima
   2.6. **Se** TodosSucessoresGerados(nó) **Então**
      2.6.1. Actualiza(nó)
   Fim_de_Se
   2.7. **Se** TodosSucessoresMemória(nó) **Então**
      2.7.1. RetiraFila(nó)
   Fim_de_Se
   2.8. **Se** MemóriaCheia(I_nó) **Então**
      2.8.1. pior ← RetiraFilaPior(I_nós)
      2.8.2. RetiraListaFilhos(pior)
      2.8.3. InsereFila(Pai(pior))
   Fim_de_Se
   2.9. InsereFila(Sn)
Fim_de_Repete
Fim_de_Função

Função Actualiza(nó): Extrutura actualizada
1. **Se** TodosSucessoresGerados(nó) e TemPai(nó) **Então**
   1.1. f(nó) ← min (f (Sn), para todo o sucessor Sn de nó)
   1.2. **Se** mudou(f(nó)) **Então**
      1.2.1. Actualiza(Pai(nó))
   Fim_de_Se
Fim_de_Se
Fim_de_Função

FIM

Marco Martins
Licenciatura Informática
Universidade Aberta
Novembro 2014
marcopaulomartins@hotmail.com