Inteligência Artificial

Prof. Alexsander da Rosa <alexsand@urcamp.tche.br>
http://www.urcamp.tche.br/~alexsand/

Busca Heurística

- Busca "*best−first*" e Funções heurísticas
- Técnicas de busca com memória limitada
- Algoritmos de aperfeiçoamento iterativo

VERSÃO 1.0

Busca "*best−first*"

Em um algoritmo de busca, o único ponto onde pode ser aplicado o conhecimento é na função de controle da fila, que determina que nodo deve ser expandido primeiro.

Usualmente, esse conhecimento é fornecido por uma **função de avaliação** que retorna um número o interesse (ou a falta dele) em expandir um determinado nodo.

Quando os nodos são ordenados na fila de forma que os com melhor avaliação sejam expandidos primeiro, a estra−tégia é chamada de "*best−first*" (primeiro o melhor). Na realidade, o nodo escolhido é o que *parece* ser o melhor.

Minimizando o custo estimado: *greedy search*

Uma das estratégias de busca *best−first* mais simples é minimizar o custo estimado até o objetivo. Isto é, o nodo cujo estado é avaliado como sendo o mais próximo do objetivo é sempre expandido primeiro. Uma função que calcula tal estimativa de custo é uma **função heurística**.

h(n)=custo estimado do caminho mais curto do estado no nodo n até um nodo associado a um estado final

Uma busca *best−first* que usa apenas *h(n)* para selecionar o próximo nodo a expandir é chamada de **greedy search**.

Em uma função heurística, se *n* é um estado final, *h(n)=0*.

Minimizando o custo total do caminho: *A* search*

A técnica **greedy search** (busca "gananciosa") minimiza o custo estimado até o objetivo, *h(n)*. Infelizmente ela não é ótima nem completa. A busca com custo uniforme, por sua vez, minimiza o custo do caminho, *g(n)*; ela é ótima e completa, mas pode ser muito ineficiente − é busca cega.

$$f(n)=g(n)+h(n)$$

Uma vez que ***g(n)*** é o *custo do caminho do estado inicial até o nodo* ***n***, e ***h(n)*** é o *custo estimado do caminho mais curto entre* ***n*** *e um estado−objetivo*, então a função ***f(n)*** é o *custo estimado da solução mais barata que passa por* ***n***.

Minimizando o custo total do caminho: *A* search*

Se o objetivo é encontrar a solução mais barata, é razoável testar primeiro o nodo com o menor valor de ***f***. Mais do que isso, é possível provar que ***f*** é completa e ótima.

Para isso, a função ***h*** escolhida *nunca deve superestimar* o custo até um estado−objetivo. Esse tipo de função ***h*** é chamado de *heurística admissível*. Heurísticas admissíveis são, por natureza, otimistas, pois pensam que o custo da solução de um problema é menor do que ele realmente é.

Uma busca *best−first* usando ***f*** como função de avaliação e uma função ***h*** *admissível* é conhecida como ***A* search***.

Funções Heurísticas

No problema da viagem pela Romênia, pode−se usar uma heurística de distância em linha reta (*straight line distance*, ou SLD), usando as distâncias em quilômetros do mapa.

O *puzzle* de 8 peças foi um dos primeiros problemas de busca heurística. Ele apresenta algumas características que o tornam ideal para o estudo: soluções típicas têm por volta de 20 passos; e o fator de ramificação é próximo de 3. Isto significa que uma busca exaustiva até a profundidade 20 teria que verificar cerca de $3^{20}=3{,}5\times10^{9}$ estados. Como há apenas 9!=362.880 combinações, o controle de estados repetidos pode diminuir em muito o tamanho da árvore.

Funções Heurísticas

5	4	
6	1	8
7	3	2

Estado Inicial

$h_1=7$

$h_2=18$

1	2	3
8		4
7	6	5

Estado Final

Como o objetivo é achar a solução mais curta (barata), a função ***h*** nunca deve superestimar o número de passos até o objetivo. Duas candidatas são as funções $\boldsymbol{h_1}$ e $\boldsymbol{h_2}$:

- $\boldsymbol{h_1}$ = o número de peças que estão na posição errada;
- $\boldsymbol{h_2}$ = a soma das distâncias das peças até a sua posição final. Também chamada de *distância de Manhattan*.

Técnicas de busca com memória limitada

Os dois principais algoritmos projetados para economizar memória são o IDA* e o SMA*. O primeiro, IDA*, é uma extensão com heurística do algoritmo de *aprofundamento iterativo*. O segundo, SMA*, é similar ao *A* search*, mas restringe o tamanho da fila para que caiba na memória.

A estratégia IDA* usa o aprofundamento iterativo para poupar memória. A busca em profundidade é modificada para usar um limite em ***f*** ao invés de profundidade. Assim, cada iteração expande todos os nós dentro do *contorno* para o limite ***f*** atual, estendendo gradualmente a linha para descobrir o próximo *contorno*. Ela é *completa* e *ótima*.

Técnicas de busca com memória limitada

A estratégia IDA* usa um mínimo de memória, mas como não armazena seu passado é condenada a repetí−lo. Os únicos estados armazenados são os do limite em ***f*** atual.

A técnica SMA* (*Simplified Memory−bounded A* search*) usa toda a memória disponível para conduzir a busca. Suas características são: *i*) utiliza toda a memória disponível ; *ii*) evita estados repetidos enquanto a memória permitir; *iii*) é *completa* e *ótima* se a memória disponível é suficiente para guardar o caminho da solução mais rasa; *iv*) se não puder ser *ótima*, acha a melhor solução possível com a memória disponível; e *v*) tendo memória, a busca é otim. eficiente.

Algoritmos de aperfeiçoamento iterativo

Muitos problemas conhecidos − como o das 8 rainhas ou o projeto de VLSI − têm a propriedade de que a descrição do estado contém toda a informação necessária para a solução, e o caminho pelo qual se chegou nela é irrelevante.

Nestes casos, algoritmos de **aperfeiçoamento iterativo** são às vezes mais adequados. Pode−se, por exemplo, começar com todas as 8 rainhas no tabuleiro, e movê−las tentando reduzir o número de ataques entre elas.

Há duas grandes classes: **subida de encosta** (*hill climbing*) e **têmpera simulada** (*simulated annealing*).

Busca de subida de encosta (*hill climbing*)

O algoritmo é um laço simples que se move continuamente na direção de um valor cada vez maior. O algoritmo não mantém uma árvore de busca, assim a estrutura de dados dos nós só precisa armazenar o estado e sua avaliação.

- Máximo local: é um pico menor que o pico mais alto do espaço de estados; uma vez nele, o algoritmo trava;
- Platô: é uma área do espaço de estados onde a função de avaliação é plana; a busca provavelmente será aleatória;
- Crista: a busca sobe rápido, mas pode oscilar de lado a lado.

Para contornar os problemas, usa−se **recomeço aleatório**.

Têmpera simulada (*simulate annealing*)

Ao invés de recomeçar em um local aleatório cada vez que a busca ficar trancada, pode−se permitir que a busca desça um pouco (*downhill*) para escapar do máximo local.

A idéia da **têmpera simulada** (de quente/frio) é escolher, ao invés do *melhor* movimento, um aleatório. Assim, se o movimento realmente melhorar a situação, ele é sempre executado. Caso contrário, o algoritmo faz o movimento com uma probabilidade que é < 1, e decresce exponencialmente com a *ruindade* ΔE do movimento. O parâmetro T é usado para calcular a probabilidade: valores altos de T tornam mais provável que movimentos *ruins* sejam feitos.