Disciplina: Inteligência Artificial
Professora: Inês Dutra
Alunas: Aline Marins Paes
Paula Fernanda M. V. de Carvalho

Algoritmos de busca aplicados ao jogo dos oito

Introdução

Os algoritmos de busca são utilizados para encontrar uma sequencia de ações que, partindo de um estado inicial, levem a uma determinada configuração desejada. Assim, estes algoritmos geram novos estados a partir da aplicação de operadores no estado corrente, até que seja alcançada a solução. Portanto, dado um problema, deve ser definido seu estado inicial, final e os operadores que serão aplicados para gerar novos estados.

Normalmente estes algoritmos são avaliados de acordo com a completude, ou seja, se conseguem chegar a uma solução, otimalidade, que diz respeito a encontrar a solução ótima e complexidades de tempo e de espaço.

Este trabalho se propõe a exibir o funcionamento de alguns métodos de busca conhecidos aplicados ao jogo dos oito. O jogo dos oito, bastante popular, consiste em um tabuleiro de 3 linhas por 3 colunas que, partindo de qualquer estado inicial que contenha algarismos de 1 a 8 e mais um quadrado vazio chegue ao seguinte estado final:

1	2	3
8		4
7	6	5

Os operadores podem ser representados como mover o espaço em branco para cima, para a esquerda, para a direita ou para baixo.

A seguir será explicado como funciona o programa e os métodos de busca implementados.

Algoritmos de busca

➢ Métodos de busca informados

Os algoritmos de busca que utilizam alguma informação específica do problema para gerar um novo estado são chamados de métodos de busca informados. Geralmente é utilizada uma função de avaliação heurística que procura estimar quantos passos são necessários para chegar à solução. No caso específico do jogo dos oito implementado neste trabalho, a heurística utilizada foi a quantidade de peças que

estão fora do lugar em relação ao estado objetivo. Foram implementados os seguintes métodos de busca informados:

- **Busca Gulosa:** este algoritmo dá preferência a expandir o nó que possui a melhor avaliação heurística. Como esta avaliação é uma estimativa e para expandir o nó a busca gulosa não utiliza o custo real da solução, esta estratégia não é completa e portanto não é ótima. Pode acontecer que a busca somente gere estados desnecessários como também não parar nunca e conseqüentemente não achar a solução, no caso de entrar em um caminho infinito e assim não poder voltar para tentar outras possibilidades. Assim sendo, a complexidade de tempo e espaço para esta busca é $O(b^m)$, onde b é o fator de ramificação (quantidade máxima de estados gerados após a expansão de um estado) e m é a profundidade da solução.

- **Busca A*:** o outro método de busca implementado neste trabalho foi a busca A* que para gerar um novo estado dá preferência ao que possui a melhor função de avaliação, considerando nesta função a soma do custo e a avaliação heurística. Este método é ótimo e completo, mas com complexidades de tempo e de espaço exponenciais. Porém geralmente gera menos nós que os outros métodos de busca.

- **Métodos não informados de busca (busca cega)**

Estes métodos não utilizam qualquer conhecimento específico do problema para determinar a prioridade com que os nós serão expandidos, por isso são chamados de busca cega. Apesar de, neste trabalho, calcularmos o custo e a heurística para todos os estados independentemente da estratégia de busca, estas informações não são usadas em nenhum momento para expandir os estados. Foram implementadas as seguintes estratégias:

- **Busca em Largura:** neste método todos os nós que estão no nível d são expandidos antes dos nós que estão no nível seguinte (d+1). Para isso, após cada expansão o algoritmo enfileira os novos nós no fim da fila de abertos, ou seja, no fim da fila dos nós que ainda não foram expandidos. Possui complexidade de tempo e de espaço exponenciais, porém garante encontrar a solução e a solução encontrada é ótima, ou seja, encontra a solução que está no nível mais raso de profundidade.

- **Busca em Profundidade:** a busca em profundidade procura a solução em um caminho específico (ou seja expande os nós do nível seguinte antes do anterior) e só passa a buscar em outro caminho se não houver novos estados a gerar. Portanto este método pode não encontrar a solução, caso a expansão seja infinita e a solução não esteja no caminho que a estratégia resolveu expandir. A complexidade de tempo é

exponencial e a de espaço é polinomial, pois um caminho que não possui a solução não precisará mais ser utilizado e assim pode ser descartado.

- **Busca em profundidade limitada:** segue os padrões da busca em profundidade porém impõe uma profundidade máxima para a expansão, ou seja, o estado que estiver na profundidade limite estabelecida não será expandido. Encontra a solução se esta estiver em uma profundidade menor ou igual ao limite estabelecido. Assim, possui as mesmas complexidades da busca em profundidade, pode encontrar a solução mas esta pode não ser a ótima, já que a ótima pode estar em nível mais raso ainda não expandido.

- **Busca em profundidade limitada iterativa:** funciona como a busca em profundidade limitada, porém ao invés de impor o limite de uma única vez, vai aumentado o limite estabelecido a cada iteração.

Resultados

Configuração 1:

1		2
8	4	3
7	6	5

Busca	Uilização da memória (nº de nós alocados simultaneamente)	Encontrou solução?
Largura	60	Sim (ótima)
Profundidade	Heap overflow [1]	Não
Custo Uniforme	60	Sim (ótima)
Profundidade Limitada	81 (limite = 10)	Sim
Profundidade Iterativa	13	Sim (ótima)
Gulosa	9	Sim (ótima)
A*	13	Sim (ótima)

Conclusões

No caso do jogo dos oito implementado neste trabalho podemos observar que os métodos informados de busca funcionam muito melhor que os demais, gerando cerca de um quarto dos nós das outras estratégias de busca e encontrando a solução no menor número de passos. A busca em largura e custo uniforme para este caso funcionam de maneira idêntica, pois o custo definido foi de 1 para cada nível. Estes

[1] Memória livre insuficiente

geram um número grande de nós para encontrar a solução que está no nível mais raso e que conseqüentemente será a de menor custo. A busca em profundidade iterativa gera menos nós que a busca em largura pois assim que encontra a solução em um caminho não se preocupa em expandir os outros caminhos vindos de outros estados. A busca em profundidade não pára pois a expansão é infinita já que não testamos a geração de estados repetidos, comprovando assim que esta não é completa.