COMPARAÇÃO ENTRE BUSCAS PARA RESOLUÇÃO DO JOGO RESTA UM

Anderson Ochner, Anderson Pezzini
andersonochner@gmail.com, ander_pezzini@hotmail.com

Resumo
Este trabalho tem por objetivo apresentar a modelagem utilizada para representar o jogo Resta Um e detalhar a diferença de desempenho entre diferentes métodos de busca utilizados para solucionar o jogo. Foram usados os métodos de busca em profundidade, largura, profundidade iterativa, bidirecional, subida da montanha e A*, e constatou-se que devido à grande quantidade de sucessores a busca em profundidade é a mais indicada para este problema. As outras buscas não conseguiram chegar a uma solução em tempo hábil.

Palavras-Chave: Resta Um; Métodos de Busca; Inteligência Artificial.

COMPARAÇÃO ENTRE BUSCAS PARA RESOLUÇÃO DO JOGO RESTA UM

Abstract
*This work has as objective show the modeling used to represent the Peg Solitaire game and detail the performance difference between different search methods used to solve the game.We usedthe depth,width, iterative depth, bidirectional,mountain climbandA * searchmethods,andit was found thatdue to the largeamount ofsuccessorsthe depth searchis the most suitablefor thisproblem.Othersearch methodshave failed toreach atimelysolution.*

Keywords*: Peg Solitaire; Search Methods; Artificial Intelligence.*

1. Introdução

O jogo Resta Um é geralmente jogado em um tabuleiro de 33 (Tabuleiro Inglês) ou 37 buracos (Tabuleiro Europeu), preenchido totalmente por pinos, exceto uma posição que o jogador escolhe.(BELL, 2014).

Um pino pode ser movimentado pulando sobre outros pinos adjacentes (verticalmente ou horizontalmente), similar ao movimento do jogo Damas. O movimento só pode acontecer se a posição onde o pino for parar estiver dentro do tabuleiro e não estiver ocupada por um pino. O pino que foi pulado é removido do tabuleiro, conforme apresentado na figura 1. O objetivo do jogo é remover todos os pinos do tabuleiro, sobrando apenas um (BOGOMOLNY, 2014).

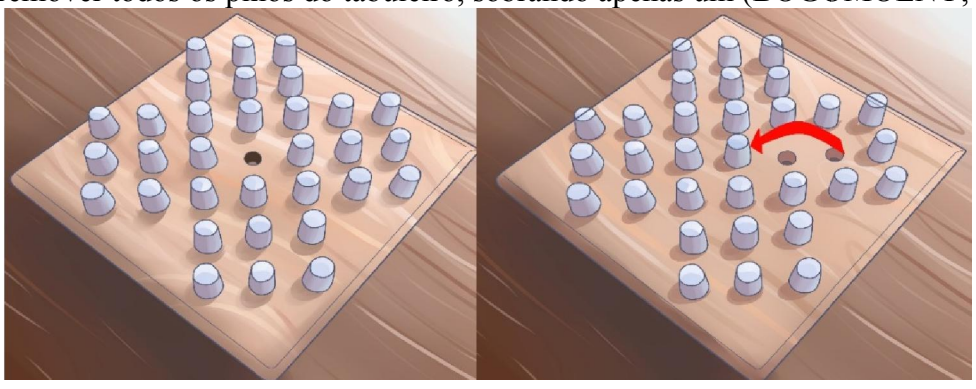

Figura 1 – Demonstração de uma ação possível no jogo (WIKIHOW, 2014)

2. Representação de estado

Os estados do problema são representados por uma matriz quadrada de inteiros M_{ij}, onde *i* é uma linha da matriz (um valor entre 0 e 6) e *j* uma coluna. Cada posição da matriz pode conter-1 (indica uma posição em que não é possível mover um pinodevidoao tabuleiro utilizado), 0 (indica a presença de um buraco onde é possível mover um pino) e 1 (indica a presença de um pino).

O estado inicial do tabuleiro é representado pelas pontas da matriz preenchidas por -1, para dar o formato característico de cruz ao tabuleiro. Todas as outras posições são preenchidas por pinos, exceto uma posição,que é escolhida pelo usuário do sistema e será preenchida com a representação de um buraco sem pino. A Figura 2 mostra o estado inicial de ambos os tipos de tabuleiros com o buraco iniciando na posição 3x3.

	0	1	2	3	4	5	6
0	-1	-1	1	1	1	-1	-1
1	-1	-1	1	1	1	-1	-1
2	1	1	1	1	1	1	1
3	1	1	1	0	1	1	1
4	1	1	1	1	1	1	1
5	-1	-1	1	1	1	-1	-1
6	-1	-1	1	1	1	-1	-1

	0	1	2	3	4	5	6
0	-1	-1	1	1	1	-1	-1
1	-1	1	1	1	1	1	-1
2	1	1	1	1	1	1	1
3	1	1	1	0	1	1	1
4	1	1	1	1	1	1	1
5	-1	1	1	1	1	1	-1
6	-1	-1	1	1	1	-1	-1

Figura 2 – Representação do Estado Inicial nos tabuleiros de 33 buracos e 37 buracos (Produção do autor, 2014)

O número de estados meta é igual ao número de posições do tabuleiro onde é possível haver um pino. Um estado é considerado meta quando resta apenas um pino no tabuleiro, ou seja, apenas uma posição da matriz contendo o número 1. Na figura 3é mostradoum exemplo de estado meta para cada tabuleiro.

	0	1	2	3	4	5	6
0	-1	-1	0	0	0	-1	-1
1	-1	-1	0	0	0	-1	-1
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1
5	-1	-1	0	0	0	-1	-1
6	-1	-1	0	0	0	-1	-1

	0	1	2	3	4	5	6
0	-1	-1	0	0	0	-1	-1
1	-1	0	0	0	0	0	-1
2	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0
5	-1	0	0	0	0	0	-1
6	-1	-1	0	0	0	-1	-1

Figura 3 – Representação de possíveis Estados Meta nos tabuleiros de 33 buracos e 37 buracos (Produção do autor, 2014)

3. Heurística utilizada

A ideia da heurística desenvolvida pelos autores deste trabalho é contar o número de movimentos possíveis no estado. Para cada pino do tabuleiro será somado 1 ao resultado da heurística para cada movimento que este pode realizar.Todos os estados-meta retornamo valor de heurística 0, pois não é possível realizar mais nenhum movimento.

4. Ambiente de simulação utilizado

O programa foi desenvolvido na linguagem de programação Java 8 update 20 utilizando a IDE NetBeans 8.0. Para o desenvolvimento dos testes foi utilizada a biblioteca do professor Jomi Fred Hübner (HÜBNER, 2007). O computador utilizado nos testes é constituído de um processador Intel Core i5-3230M 2.6GHz, com 8GB de memória RAM e sistema operacional Windows 7 (64 bits).

5. Resultados

Os resultados obtidos são uma média de três execuções para cada tipo de busca, utilizando um tabuleiro de 33 buracos e iniciando com a posição vazia em 3x3 (exatamente o meio do tabuleiro). O tempo máximo de execução utilizado foi de 5 minutos. Na tabela 1 são exibidos os dados relativos ao desempenho de cada uma das buscas utilizadas.

Busca	Nós visitados	Profundidade	Tempo decorrido
Profundidade	1066	31	62 ms
Largura	131505	9	5 min
Prof. Iterativa	145504	9	5 min
Bidirecional	40378	6	5 min
Subida da Montanha	11196126	-	5 min
A*	127878	16	5 min

Tabela 1 – Comparativo de desempenho entre buscas (Produção do autor, 2014)

A busca bidirecional obteve o pior desempenho pela quantidade de estados-meta que são possíveis no problema. Como neste problema todas as soluções são igualmente ótimas e existe um número grande de soluções, não seria aconselhável utilizar a busca bidirecional.

As buscas em largura e profundidade iterativa também obtiveram um desempenho baixo, mas por outro motivo. O fator de ramificação da árvore é variável, como apresentado na Figura 3. O fator de ramificação no começo e término da árvore possui um valor baixo, mas no meio da árvore pode chegar a valores muito altos. Como ambas as buscas verificam todos os nós de um nível da árvore, é compreensível que elas não sejam a melhor escolha para este tipo de problema.

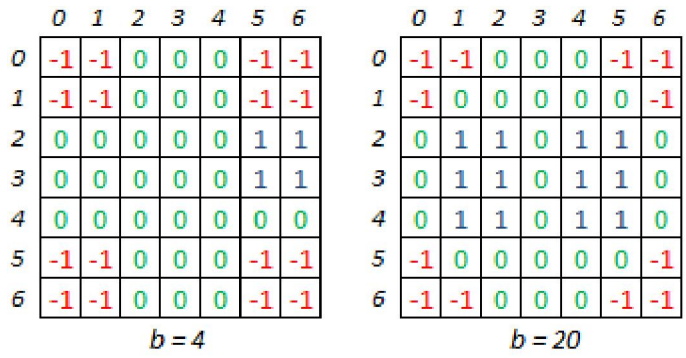

Figura 3 – Comparação entre o fator de ramificação de estados diferentes (Produção do autor, 2014)

Devido à forma como foi desenvolvida a heurística, a busca A* consegui o segundo melhor desempenho. Assim como as buscas em largura e profundidade, o desempenho dela está ligado ao fator de ramificação da árvore. Porém, ela só começa a verificar todos os nós de um nível quando ela chega próximo ao meio da árvore, quando o número de movimentos possíveis começa a diminuir para todos os estados.

A busca subida da montanha não obteve um bom resultado porque ao chegar em um estado que não podia ser evoluído, abusca reiniciava do primeiro nível sem nenhum pino preenchido,

e acabava caindo novamente no mesmo problema.Logo, este algoritmo não é aconselhado em casos que a heurística só tem efeito locale não pode prever um custo do início até o fim do caminho.

O algoritmo de busca em profundidade obteve o melhor desempenho por um motivo inerente ao problema: todos os estados-meta eram igualmente ótimos e estavam na mesma profundidade da árvore de estados. Independente do caminho que o algoritmo utilizasse, quando chegasse ao nível *número de buracos do tabuleiro - 2* ele encontraria uma solução.

Considerações finais

Apesar de não ser uma busca ótima, ou seja, nem sempre encontra a melhor solução, a busca em profundidade aplicada a este problema é tão efetiva quanto as outras técnicas de busca, visto que todas as sequencias de passos que levam ao estado meta são igualmente boas. Além disso, a alocação de memória e a velocidade desta busca são muito mais efetivas quando comparadas às outras utilizadas neste trabalho.

Portanto, mesmo técnicas de busca tidas como melhores podem não sereficientesà determinados problemas, e deve-se a partir de análise selecionar a técnica que melhor se adequa.

Referências

BELL, George. **Peg Solitaire.** 2014. Disponível em: <http://home.comcast.net/~gibell/pegsolitaire/>. Acesso em: 31 ago. 2014.

BOGOMOLNY, Alexander. **Peg Solitaire.** 2014. Disponível em: <http://www.cut-the-knot.org/proofs/pegsolitaire.shtml>. Acesso em: 30 ago. 2014.

HÜBNER, Jomi Fred. **Biblioteca deBusca em Espaço de Estados**. 2007. Disponível em: <http://www.das.ufsc.br/~jomi/ia/busca>. Acesso em: 30 ago. 2014.

WIKIHOW. **How to Win the Peg Solitaire Game (English Board)**.2014. Disponível em: <http://www.wikihow.com/Win-the-Peg-Solitaire-Game-%28English-Board%29>. Acesso em: 31 ago. 2014.