Bubble Blast

TEMÁTICA I: JOGO DO TIPO SOLITÁRIO

INTELIGÊNCIA ARTIFICIAL 2019/2020

INÊS ALVES, JOÃO LEITE, MÁRCIA TEIXEIRA

Especificação do jogo

O Bubble Blast é um jogo do tipo solitário cujo tabuleiro (5x6) é composto por bolhas de diferentes cores, com as quais o utilizador pode interagir clicando nelas. Quando se toca numa bolha, ela cresce, mudando progressivamente de cor (azul – amarelo – verde – vermelho), sendo que uma bolha vermelha irá rebentar e projetar bolhas pequenas em quatro direções: cima, baixo, esquerda e direita. Estas bolhas, por sua vez, ao bater noutra bolha, têm o mesmo resultado do toque do utilizador. O objetivo do jogo é, portanto, rebentar todas as bolhas do tabuleiro, tendo em conta que o utilizador tem um número limitado de toques que pode fazer.

Pesquisa

Para a formulação do problema, baseamo-nos no jogo original “Bubble Blast 2”. Analisamos, também, um solver em Python: https://github.com/zTrix/bubble-blast-solver.

Formulação do Problema

Representação do Estado: através de um vetor bidimensional com as dimensões do tabuleiro (5x6), em que cada célula é representada por um inteiro de 0 a 4, sendo que 0 é um espaço vazio, 1 a bolha vermelha, 2 a bolha verde, 3 a bolha amarela e 4 a bolha azul.

Estado Inicial: tabuleiro inicial (5x6), variável com o nível; deve ter pelo menos uma bolha.

Teste Objetivo: verificar se o tabuleiro fica vazio, sendo o número de toques menor ou igual ao número de toques máximo para o nível.

Formulação do Problema

Operadores:

Sendo B(X,Y) o número da célula com as coordenadas horizontais e verticais X e Y, respetivamente, e T o número de toques disponíveis.

Nome	Pré-condições	Efeitos	Custo
BurstBubble	B(X,Y) = 1 0 <= X < 5 0 <= Y < 6 T > 0	B(X,Y) = 0 X>0,B(X-1,Y) = B(X-1,Y)-1 X<5,B(X+1,Y) = B(X+1,Y)-1 Y>0,B(X,Y-1) = B(X,Y-1)-1 Y<6, B(X,Y+1) = B(X,Y-1)-1	T = T-1
TouchBubble	B(X,Y) > 1 0 <= X < 5 0 <= Y < 6 T>0	B(X,Y) = B(X,Y)–1	T = T-1

Formulação do Problema

Heurísticas:

- Usadas na pesquisa gananciosa e A*:
  - São preferidos nós em que os tabuleiros tenham menor valor da soma total dos valores das suas células e maior percentagem de bolhas vermelhas em relação ao número total de bolhas.

- Usadas no algoritmo A*:
  - É atribuído menor custo a nós em que os tabuleiros tenham uma linha ou coluna de mais que uma bolha vermelha, aqueles em que as bolhas se distribuam em apenas uma única linha ou coluna, uma interseção de linha com coluna (p.e. em formato de "+" ou "T"), ou aqueles que apenas tenham bolhas vermelhas.
  - É também usada a profundidade do nó no grafo para obter o número de passos necessários para chegar ao nó.

Implementação

Optamos por implementar a solução deste problema em C++.

Utilizamos uma árvore para representação do problema, em que cada nó representa o tabuleiro num estado do jogo, e cada aresta as coordenadas da bolha que, quando clicada, leva à transição entre os nós correspondentes.

Foi também utilizada uma fila de prioridade para as implementações do algoritmo A* e custo uniforme, ordenando as bolhas de modo a minimizar ou a soma do custo/heurística (no caso do A*), ou apenas o custo (custo uniforme).

O programa utiliza várias estratégias de pesquisa para resolver o problema: não informada (pesquisa primeiro em largura, primeiro em profundidade com profundidade limitada e custo uniforme) e informada/ heurística (pesquisa gulosa e A*).

Implementação (Pesquisa não informada)

Pesquisa primeiro em profundidade (DFS)
A DFS consiste na expansão dos nós da árvore, começando na sua raiz, e aprofundando progressivamente a cada iteração, até chegar ao nó de profundidade máxima, retrocedendo depois para visitar os restantes. No nosso problema, um nó é explorado em profundidade até se encontrar uma solução, terminando a pesquisa, ou até atingir a profundidade correspondente ao número máximo de movimentos, retrocedendo.

Pesquisa primeiro em largura (BFS)
A BFS, por sua vez, consiste na expansão dos nós em largura, ou seja, nível a nível, apenas passando para uma profundidade d+1 quando todos os nós d tiverem sido visitados. No caso do presente problema, os nós são explorados até ser encontrada uma solução.

Pesquisa de custo uniforme
Esta pesquisa é semelhante à DFS, no sentido em que explora os nós por níveis de profundidade, mas, em vez de os expandir por ordem, estes irão ser ordenados segundo uma função de custo, sendo explorados os nós de um determinado nível por ordem desse custo (de menor para maior).

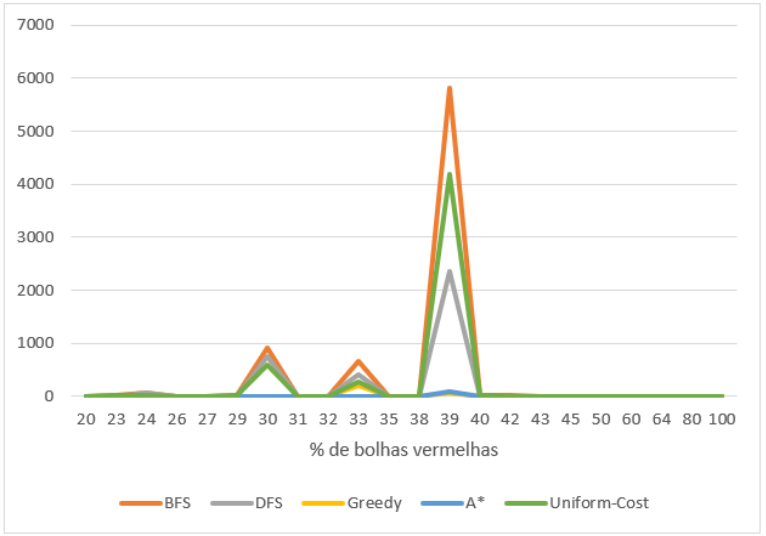
Implementação (Pesquisa informada)

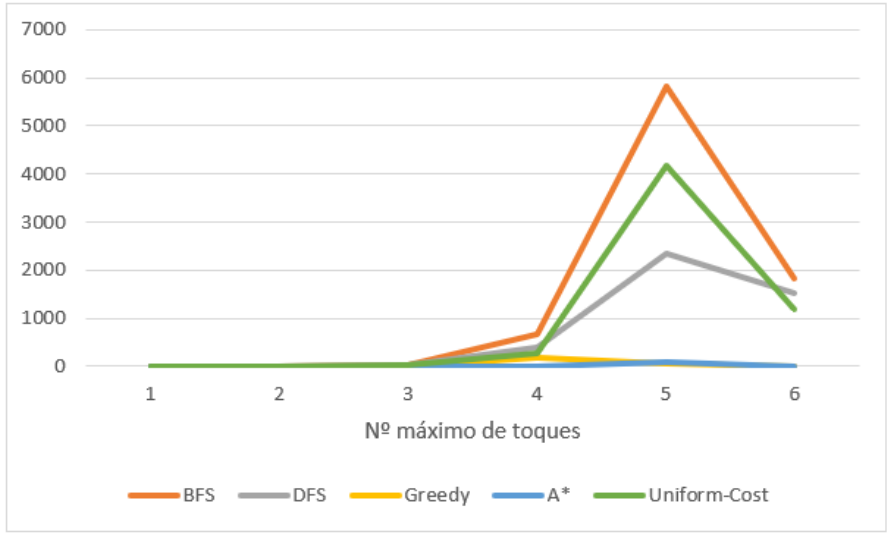
Pesquisa gananciosa
O algoritmo ganancioso explora os nós em profundidade, utilizando uma heurística gananciosa para escolher o próximo nó a explorar. No caso tratado, a heurística atribui um valor a um nó, que será tanto maior quanto maior for a distância estimada à solução. O nó escolhido é aquele que minimiza a distância à solução.

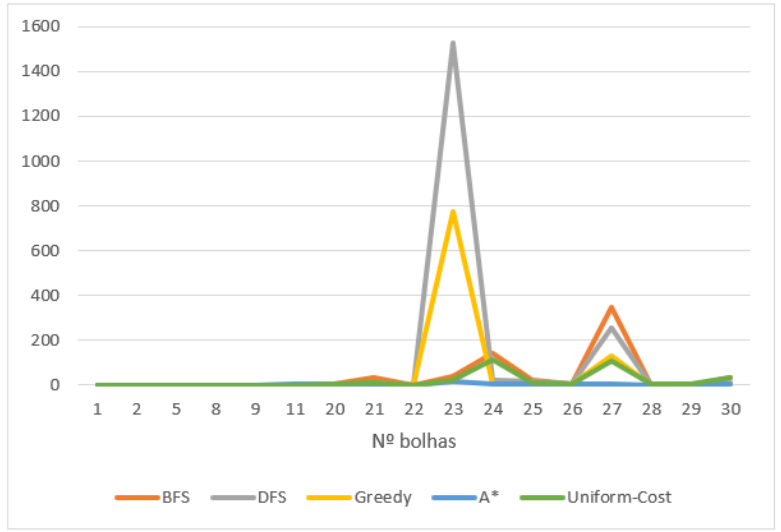
A*
Este combina o custo para chegar até ao nó atual, com a estimativa do custo de avançar para o próximo nó (g(n)), somando isto ao valor retornado pela heurística da pesquisa gananciosa (h(n)), sendo f(n)= g(n)+h(n), seguindo sempre o caminho que minimiza f(n).

Resultados Experimentais - Dados

Mediram-se os tempos de execução para os 35 primeiros níveis do Bubble Blast 2, com os 5 algoritmos implementados. Dado que o tabuleiro tem dimensões fixas e a dificuldade de cada nível depende de vários fatores, organizaram-se os dados obtidos relativos por 3 critérios: número de toques máximo do nível, número total de bolhas do tabuleiro inicial e percentagem de bolhas vermelhas em relação ao número total de bolhas. Os valores no eixo vertical correspondem ao tempo de execução em milissegundos.







Resultados Experimentais - Conclusões

Através da análise dos gráficos pode-se concluir que, apesar de em níveis menos complexos a eficiência dos vários algoritmos ser semelhante, em níveis com maior grau de dificuldade a pesquisa informada, particularmente o algoritmo A*, revela-se notavelmente mais eficiente, e revela também uma eficiência mais consistente entre os vários níveis de dificuldade.

Conclusões

O projeto desenvolvido foi relevante para a consolidação dos conteúdos lecionados na UC de Inteligência Artificial.

Concluiu-se que algoritmos de pesquisa informada (particularmente o A* nos dados obtidos) são geralmente mais eficientes, especialmente para problemas de maior dimensão e que, pela utilização de heurísticas apropriadas, a sua eficiência não diminui em tão grande escala face a problemas de maior complexidade como com algoritmos de pesquisa não informada.

Entendemos também a importância de escolher heurísticas adequadas, sendo que esta foi uma dificuldade com que nos deparamos no desenvolvimento do projeto.

Referências consultadas

- Slides da Unidade Curricular de Inteligência Artificial
- *Artificial Intelligence – A Modern Approach – 3rd Edition*, Stuart J. Russell and Peter Norvig, 2010
- https://en.wikipedia.org/wiki/Depth-first_search
- https://en.wikipedia.org/wiki/A*_search_algorithm