Projeto IA



Grupo: 030 Turno: IArt3L06 Professor: Emanuel Gonçalves

Gustavo Pinto – ist189627

Tomás Cayatte - ist198876

# Introdução

Neste projeto para resolução automática de um tabuleiro incompleto de Takuzu, começamos por definir a classe Board, correspondente à representação interna de um tabuleiro de takuzu, nas condições definidas no problema. Nesta classe, foi decidido guardar os valores presentes no tabuleiro em memória, utilizando uma matriz, ou seja, um vetor duplo. Esta classe, irá também guardar o tamanho do tabuleiro e o número de posições vazias que esse tabuleiro contém nesse momento.

De seguida, para a classe Takuzu, representativa do problema de resolver um jogo de takuzu, sobre a qual se irá proceder aos diversos métodos de procura, o único parâmetro guardado é o estado inicial (classe TakuzuState definida previamente).

# Heurística

Nos casos da procura gananciosa (greedy\_search) e da procura A\* (astar\_search) é necessário definir uma heurística que permita a escolha dos nós ótimos na procura da solução.

Para a definição desta função heurística começamos por reparar que em cada iteração da procura, o problema será sempre preenchido com um valor (entre 0 e 1), numa posição previamente vazia. Asssim, e visto que a função heurística é também uma estimativa de quantos nós ainda são precisos visitar até chegar à solução, a primeira heurística a que chegamos foi:

Que apesar de funcionar para grande parte dos casos, não é suficiente, já que existem casos específicos em que para saber qual dos valores deve ser colocado na posição vazia, é necessário alguma experimentação de valores, para que se chegue a uma resposta válida (ou inválida), nomeadamente a comparar se existiria alguma linha/coluna que fosse igual.

Assim, definimos uma nova função heurística:

Para a heurística final, usamos apenas a agregação das duas heurísticas, dando uma maior importância à primeira:

# Resultados

Após todas as classes e métodos estarem definidos, decidimos comparar o tempo de execução de cada teste público disponibilizado, para os diferentes métodos de procura. Com isso chegamos aos seguintes resultados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | depth\_first\_search | breadth\_first\_search | astar\_search | greedy\_search |
| T01 | 0,001000881 | 0,001000881 | 0,001000643 | 0,001501322 |
| T02 | 0,001501083 | 0,001500368 | 0,001502514 | 0,001502037 |
| T03 | 0,006505251 | 0,009507418 | 0,00950861 | 0,009007454 |
| T04 | 0,005504847 | 0,006506205 | 0,007006645 | 0,007007599 |
| T05 | 0,010509491 | 0,059551001 | 0,01315093 | 0,012510061 |
| T06 | 3,067140579 | Didn’t finish in acceptable time | 0,025021553 | 0,026522875 |
| T07 | 0,019016504 | 0,018516064 | 0,022518873 | 0,022519588 |
| T08 | 0,010008574 | 0,010008574 | 0,011008739 | 0,011010408 |
| T09 | 0,046540022 | 0,046540737 | 0,06055212 | 0,060051441 |
| T10 | 0,07006073 | 0,071561098 | 0,090577841 | 0,089076996 |
| T11 | 0,105591059 | 0,107269764 | 0,125107765 | 0,125108004 |
| T12 | 0,230698347 | 0,234884739 | 0,255719662 | 0,261224985 |
| T13 | 0,454891205 | 0,454391718 | 0,492423773 | 0,494426012 |
| **Média** | 0,309920659 | 0,085103214 | 0,085776898 | 0,086266829 |

O que nos permitiu concluir que o método ideal para o nosso caso seria o A\*, visto que é aquele com uma menor média de tempo (excluindo o breadth\_first\_search, pois não completou o teste T03).