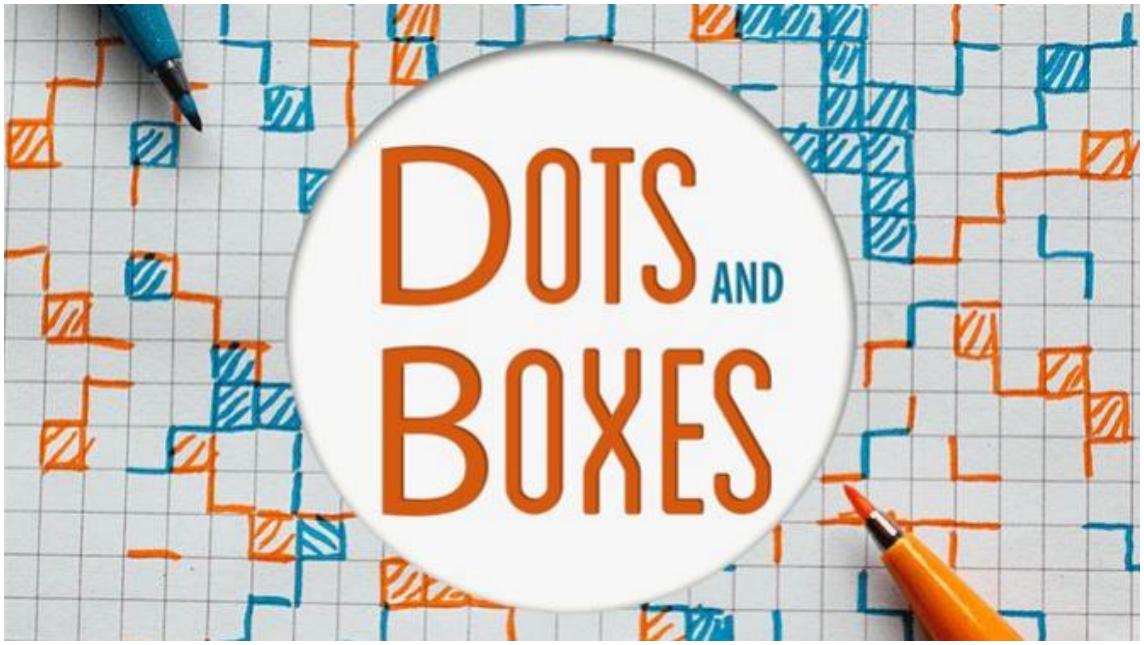
Projeto Nº 1, 1ª fase - Dots and Boxes



Manual Técnico

Inteligência Artificial – 2022/2023

Projeto realizado por:

- Nuno Reis Nº 202000753
- Vítor Nunes Nº 201901148

Arquitetura do sistema

O sistema é composto por nós (listas com vários elementos) e por problemas (tabuleiros do jogo Dots and Boxes) que têm de ser resolvidos recorrendo à utilização dos algoritmos de procura Algoritmo *Breadth-First*, Algoritmo *Depth-First* e Algoritmo *A**.

Entidades e suas Implementações

Os nós são compostos por 5 elementos:

1. Estado do nó (que contém 2 subconjuntos): arcos horizontais e arcos verticais.
2. Número de caixas fechadas que o nó solução (tabuleiro) tem de ter.
3. Profundidade do nó.
4. Valor da heurística do nó.
5. Nó pai

Ao todo existem 6 problemas, tendo em conta que cada problema é um tabuleiro com dimensões variadas e que podem já possuir arcos horizontais ou verticais já posicionados.

Os 6 problemas são:

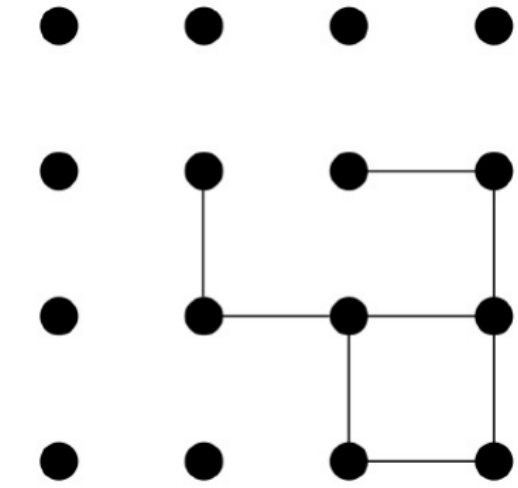


Figura 4: Problema a). Objetivo: 3 caixas fechadas.

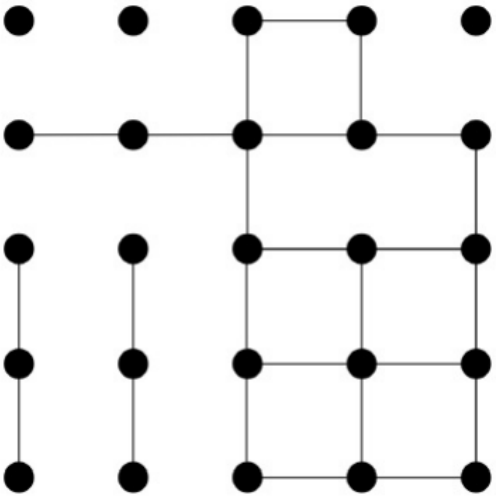


Figura 5: Problema b). Objetivo: 7 caixas fechadas.

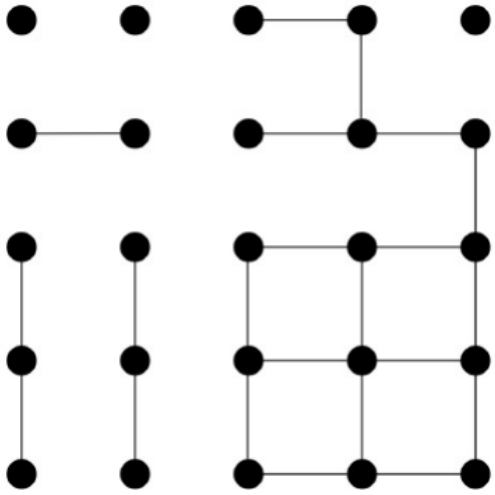


Figura 6: Problema c). Objetivo: 10 caixas fechadas.

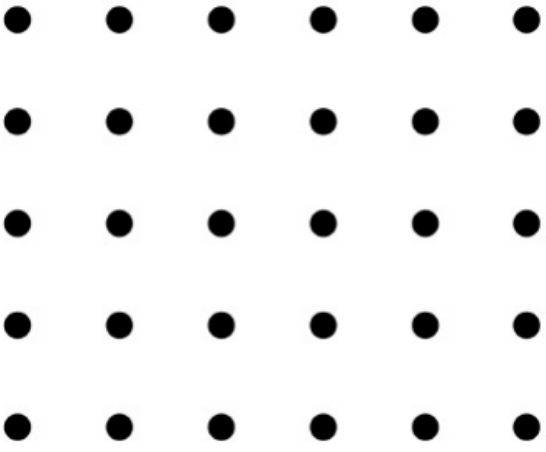


Figura 7: Problema d). Objetivo: 10 caixas fechadas.

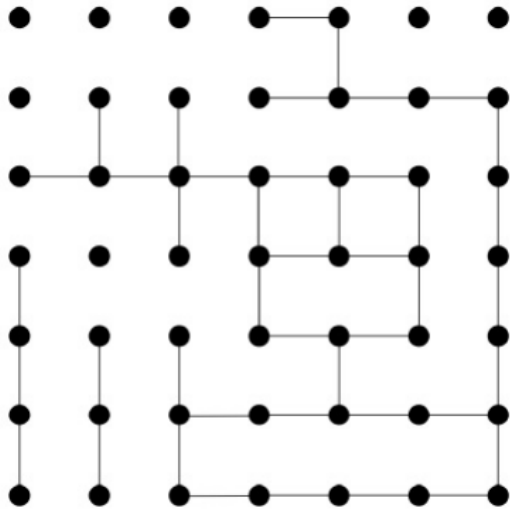


Figura 8: Problema e). Objetivo: 20 caixas fechadas.

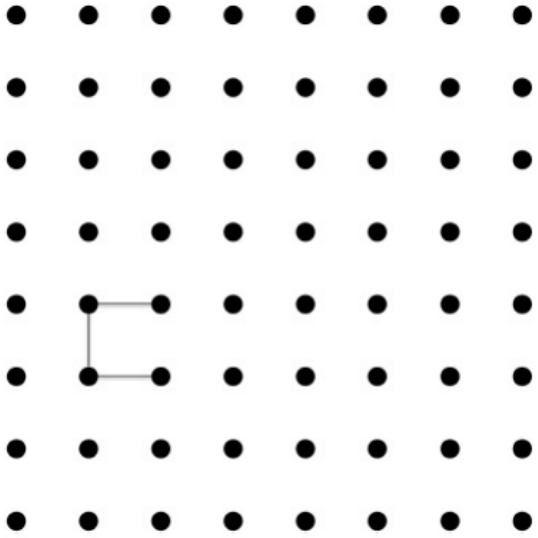


Figura 9: Problema f). Objetivo: 35 caixas fechadas.

Para resolver cada problema, o algoritmo tem de fechar o número de caixas estipulado para cada problema. O número de caixas fechadas varia de problema para problema e esse número está mencionado em cada uma das imagens (exemplo: “Problema e). Objetivo: 20 caixas fechadas.”).

Algoritmos e suas implementações

Os algoritmos de procura presentes no sistema são:

- Algoritmo *Breadth-First;*
- Algoritmo *Depth-First*;
- Algoritmo *A*.*

O algoritmo *Breadth-First* foi implementado seguindo os seguintes critérios:

1. Introduzir o nó inicial na lista de nós Abertos

2. Se a lista de Abertos for vazia falha

3. Remover o primeiro nó da lista de Abertos - nó n - e colocá-lo na lista de nós Fechados

4. Expandir o nó n e colocar os seus sucessores no fim da lista de Abertos

5. Se algum dos nós sucessores do nó n for um nó objetivo, a solução é mostrada. Caso contrário volta ao ponto 2

O Algoritmo *Depth-First* foi implementado seguindo os seguintes critérios:

1. Introduzir o nó inicial na lista de nós Abertos

2. Se a lista de Abertos for vazia falha

3. Remover o primeiro nó de Abertos - nó n - e colocá-lo na lista de nós Fechados

4. Se a profundidade do nó n for maior que a profundidade definida inicialmente para o limite, volta ao ponto 2

5. Expandir o nó n e colocar os seus sucessores no inicio da lista de Abertos

6. Se algum dos nós sucessores do nó n for um nó objetivo, a solução é mostrada. Caso contrário volta ao ponto 2

O Algoritmo *A** foi implementado seguindo os seguintes critérios:

1. Introduzir o nó inicial na lista de nós Abertos

2. Se a lista de Abertos for vazia falha

3. Remover o primeiro nó da lista de Abertos com menor custo - nó n - e coloca-lo na lista de nós Fechados

4. Expandir o nó n e colocar os seus sucessores que não existem nem na lista de Abertos nem na lista de Fechados na lista de Abertos por ordem crescente de menor custo

5. Se o primeiro nó da lista de Abertos, com a lista já ordenada de forma crescente de custo, for um nó objetivo, a solução é mostrada. Caso contrário volta ao ponto 2

Heurística definida pelo grupo de trabalho

Além da heurística sugerida no enunciado do projeto, foi implementada no sistema uma outra heurística que privilegia os tabuleiros com maior número de cantos/vértices, ou seja, um arco horizontal e um arco vertical ligados no mesmo vértice formando um canto de uma caixa.

O cálculo do valor da heurística seria, para um tabuleiro x:

h(x) = o(x) – c(x)

em que:

- *o(x)* é o número de cantos de todas as caixas a fechar consoante o objetivo do tabuleiro x,
- *c(x)* é o número de cantos presentes no tabuleiro x.

O elemento *o(x)* seria fácil de calcular pois cada caixa fechada tem 4 cantos logo seriam 4 cantos por caixa a fechar, então: *o(x)* = 4 × número de caixas a fechar no tabuleiro x (objetivo do problema).

Esta heurística será mais eficiente pois é mais fácil formar um canto do que uma caixa fechada, pois para formar um canto só precisamos de dois arcos e para formar uma caixa precisamos de 4 arcos, logo irá haver sempre uma maior diferenciação entre cada nó no que diz conta à sua progressão rumo ao nó solução.

Podemos observar nas imagens seguintes que esta nova heurística também cria muito menos nós que a heurística sugerida no enunciado, tendo em conta que a heurística criada pelo grupo corresponde à segunda imagem:

```
Heuristicas
Escolha uma opção
0- Voltar atrás
1- Diferença entre o numero de caixas fechadas e o numero esperado de caixas fechadas
2- Diferença entre o numero de cantos do tabuleiro e o numero de cantos esperados(numero esperado de caixas fechadas * 4)
1
Estado: (((0 0 1 0) (1 0 1 1) (0 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (0 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (0 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 0 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (0 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 0 1 1) (1 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 1) (1 0 1 1) (1 0 1 1) (1 0 1 1) (1 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (1 1 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 1) (1 0 1 1) (1 0 1 1) (1 0 1 1) (1 0 1 1)) ((0 0 1 1) (0 0 1 1) (1 1 1 1) (1 1 1 1) (1 1 1 1))), Numero de
Nós Gerados 148039, Nós Expandidos: 11000, Penetrância: 9/148039
```

```
Heuristicas
Escolha uma opção
0- Voltar atrás
1- Diferença entre o numero de caixas fechadas e o numero esperado de caixas fechadas
2- Diferença entre o numero de cantos do tabuleiro e o numero de cantos esperados(numero esperado de caixas fechadas * 4)
2
Estado: (((0 0 1 0) (1 0 1 1) (0 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 0 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (0 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 0 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (0 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (1 0 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (1 1 1 1)) ((0 0 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (1 1 1 1)) ((0 1 1 1) (0 0 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (1 1 1 1)) ((0 1 1 1) (0 1 1 1) (0 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Estado: (((0 0 1 0) (1 0 1 1) (1 1 1 1) (1 1 1 1) (1 1 1 1)) ((0 1 1 1) (0 1 1 1) (1 0 1 1) (1 0 1 1) (0 1 1 1))), Numero de
Nós Gerados 589, Nós Expandidos: 44, Penetrância: 10/589
```

Com esta nova heurística, o algoritmo *A** consegue resolver também o problema E algo que não acontecia com a heurística já dada.

Posto isto, podemos dizer também que esta heurística melhorará o desempenho do algoritmo *A*.*

Descrições das opções tomadas

O sistema foi implementado recorrendo ao uso da recursividade em detrimento do uso de ciclos, as funções são curtas com o objetivo de facilitar a sua compreensão e houve a utilização de funções já feitas dentro de outras funções, reutilizando assim o código já implementado.

Análise comparativa

Devido à criação de uma quantidade imensa de nós durante a execução dos algoritmos fazendo com que o *IDE* chegue ao seu limite, existem alguns problemas que não conseguem ser resolvidos por um algoritmo.