TÉCNICO LISBOA

# Inteligência Artificial 2021/22 (P4)
# Relatório de projeto: **Takuzu**

**Grupo 30**: Laura Quintas (92759), Raquel Romão (92780)

## 1 Conceito do Projeto

O projeto de Inteligência Artificial do P4 2021/2022 teve como objetivo a implementação de um programa em Python que resolva o problema Takuzu utilizando técnicas de procura de IA. É realizada uma análise crítica dos resultados obtidos executando uma procura em largura primeiro (bfs), uma procura em profundidade primeiro (dfs), uma procura gananciosa e uma procura A*. De acordo com as características do jogo e da forma como foi implementado, todas as procuras são completas, i.e., o número de nós são finitos e não existem repetições de posições ao longo do mesmo ramo da árvore, o que previne *loops*.

### 1.1 Abordagem não informada

Numa primeira fase da realização do projeto, foi considerada uma abordagem *naive*, i.e, foram consideradas todas as ações para cada casa vazia, sem restrições. Dado o peso de considerar todas as ações, apenas foram abertas as ações para uma casa vazia de cada vez, o que melhorou tempos e nós gerados em todos os algoritmos, como se consegue observar comparando os resultados das tabelas nas Figuras 1 e 2, em anexo. Para além disso, foi também criado um dicionário de estados para o qual a chave era uma *string* da board pós-ação e o valor associado o estado correspondente, o que previne a criação de estados já existentes na memória. No entanto, esta abordagem apenas corria em tempo adequado para os primeiros dois inputs disponibilizados.

### 1.2 Abordagem informada - Heurísticas

As heurísticas implementadas foram pensadas com o intuito de guiar a procura no sentido de ações que levavam a estados para os quais não eram quebradas regras, assim como de estados de tabuleiros com um menor número de posições vazias. Além disso, a adição de outras condições à heurística, como a prioridade a ações em linhas e/ou colunas com poucos 2 (para ser dado mais peso a serem completadas linhas e/ou colunas) ou a diferença entre o número de 0's e 1's no tabuleiro, foram testadas, o que resultou numa comparação entre 3 heurísticas (**h1**, **h2** e **h3**, descritas no pseudocódigo 1 em anexo).

Ainda assim, esta abordagem revelou-se igualmente inconcebível quando a dimensão do tabuleiro era superior a 6 - nenhuma das quatro procuras a serem comparadas terminou em menos do *threshold* de espera definido de 3 minutos, sendo que, dos testes fornecidos, apenas o input T01 e T02 correram em tempo útil. Numa primeira instância, foram estes os testes os usados na comparação das técnicas de procura e diferentes heurísticas (Figuras 1, 2 e 3, em anexo). A heurística **h2**, a qual contabiliza a diferença entre 0's e 1's no tabuleiro, revelou ser de uma forma geral a mais benéfica a nível de tempo de execução, tanto para a procura A* como para a procura gananciosa. Mais à frente (Figuras 4 e 5), esta comparação é reiterada e conclui-se que, muito embora a eficiência espacial seja equivalente para as heurísticas comparadas (Figura 4), **h2** foi a mais eficiente a nível temporal para tabuleiros de maiores dimensões (T11, T12 e T13) e para um tabuleiro com um número elevado de casas vazias face à sua dimensão (T03), sendo que fora esses casos apresentou um desempenho semelhante a **h3** (Figura 5).

## 1.3 Abordagem de procura aliada a CSP

Dado que independentemente da heurística utilizada, os testes a partir do input T03 ultrapassavam os 3 segundos objetivo, foi mudada a abordagem. Em vez de serem consideradas para a procura todas as ações para as posições vazias do tabuleiro, estas foram restritas àquelas que apenas respeitam as regras do jogo - deste modo limitando significativamente a árvore de procura. Caso ao longo da árvore uma das posições não tivesse solução, era realizado um corte de modo a otimizar a procura. Foi implementado assim um '*Constraint Satisfaction Problem*' (CSP), sendo que ao ser tomada esta abordagem as posições que apenas tinham uma opção de acordo com as regras eram preenchidas de imediato, tendo estas um efeito dominó sobre o domínio de outras. Dado este conhecimento, quisemos incluir no código uma implementação que beneficiaria deste facto.

### 1.3.1 Implementação de consistência de domínios

Para além de um CSP aliado à procura, foi implementado também um ciclo *while* que apenas parava caso não tivesse sido atualizado nenhum valor do tabuleiro, sendo assim propagadas as restrições de cada mudança efetuada, atribuindo consistência às ações retornadas para a procura.

### 1.3.2 Restrições básicas

As restrições implementadas, numa primeira abordagem, foram as restrições básicas associadas ao Takuzu: linhas e colunas diferentes, número igual de 0's e 1's - no caso dos ímpares, um deles com mais um do que o outro - e a ausência de tripletos de 0's ou 1's, quer verticais, quer horizontais, sendo verificada tanto a adjacência local como a global. Isto permitiu a resolução dos testes disponibilizados em menos dos 3 segundos objetivo, sendo que para apenas 3 deles (T03, T05 e T06) foi necessária iniciar uma procura (Figura 4) e ao avaliarmos os tempos nos gráficos (Figura 6) chegámos à conclusão que o menos eficiente seria o **A*** e o mais eficiente o **DFS**, por isso na submissão do projeto este último seria empregue. No entanto, na plataforma Mooshak apenas 10 testes eram resolvidos abaixo do tempo limite com as **restrições básicas** e **DFS**.

### 1.3.3 Restrições complexas

Numa segunda abordagem, depois de entendido melhor o conceito do jogo, conseguimos deduzir outras regras que seriam subentendidas pela nossa lógica mas que teriam de ser implementadas de forma literal no programa.

Assim sendo acrescentámos, para além das já mencionadas, restrições adicionais: no caso de faltarem ser preenchidas apenas duas posições numa coluna/linha, é feita a verificação se já existe alguma coluna/linha preenchida de igual forma e, se houver, preencher com a forma contrária à já existente; para problemas de número par: onde faltasse apenas acrescentar um único número de 1's ou 0's para perfazer o limite (metade do tamanho do tabuleiro), permitir apenas a adição de um número que impediria a formação de tripletos do número com mais elementos por adicionar nessa linha ou coluna - em anexo encontram-se exemplos (Figura 7).

Depois desta implementação todos os testes disponibilizados foram reduzidos a um problema de restrições que não chegavam a usufruir da procura nos casos das não informadas e da informada gananciosa (Figura 8). Posto isto, a procura informada A*, uma vez que chegava a entrar na procura numa primeira instância - o que resultava num cálculo da heurística - gasta mais tempo do que as restantes, sendo os tempos destas últimas muito semelhantes (Figura 9).

A adição de regras complexas resultou numa resolução da totalidade dos testes dentro do tempo limite, fazendo desta implementação completa e significativamente eficiente.

# 2 Anexo

| Não Informada e Informada | | |
|---|---|---|
| nós expandidos / nós testados / nós gerados | | |
| Algoritmos | 1 | 2 |
| BFS | 64976/464977/106362 | 418560/418561/1063622 |
| DFS | 6420/6421/6466 | 6355/6356/6400 |
| A* | 399/ 400/2210/ | 286/ 287/1716/ |
| GREEDY | 217/ 218/ 824/ | 126/ 127/ 442/ |

(a) Nós resultantes.

| Não Informada e Informada | | |
|---|---|---|
| Times (s) | | |
| Algoritmos | 1 | 2 |
| BFS | 102141.15 | 128496.62 |
| DFS | 1401.83 | 1674 |
| A* | 1406.78 | 942.39 |
| GREEDY | 221.05 | 125.03 |

(b) Tempos.

Figura 1: Abordagem de procura não informada e informada a todas as ações de todas as posições geradas de uma vez.

| Não Informada e Informada | | |
|---|---|---|
| nós expandidos / nós testados / nós gerados | | |
| Algoritmos | 1 | 2 |
| BFS | 212/ 213/ 254/ | 215/ 216/ 254/ |
| DFS | 88/ 89/ 92/ | 81/ 82/ 84/ |
| A* | 10/ 11/ 20/ | 177/ 178/ 184/ |
| GREEDY | 7/ 8/ 14/ | 180/ 181/ 184/ |

(a) Nós resultantes.

| Não Informada e Informada | | |
|---|---|---|
| Times (s) | | |
| Algoritmos | 1 | 2 |
| BFS | 50.45 | 49.05 |
| DFS | 14.93 | 16.19 |
| A* | 4.51 | 70.05 |
| GREEDY | 2.34 | 46.87 |

(b) Tempos.

Figura 2: Abordagem de procura não informada e informada às ações de uma posição de cada vez.

| Comparação de heurísticas | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Inputs | 1 | | | | 2 | | | |
| Algoritmos | A* | | GREEDY | | A* | | GREEDY | |
| Heurísticas | Nós expandidos/gerados | Tempo (ms) | Nós expandidos/gerados | Tempo (ms) | Nós expandidos/gerados | Tempos(ms) | Nós expandidos/gerados | Tempo (ms) |
| h1 | 181/192 | 49.38 | 174/178 | 48.35 | 118/124 | 38.62 | 117/122 | 38.62 |
| h2 | 15/30 | 5.66 | 7/14 | 2.43 | 155/174 | 56.05 | 180/184 | 52.49 |
| h3 | 10/20 | 4.51 | 7/14 | 2.34 | 177/184 | 70.05 | 180/184 | 46.87 |

Figura 3: Nós expandidos, nós gerados e tempo de execução para as 3 diferentes heurísticas testadas.
**h1** - Número de casas vazias + Peso por regra quebrada + Prioridade a ações em linhas/colunas com poucas casas vazias; **h2** - Número de casas vazias + Peso por regra quebrada + Diferença entre número de 0's e 1's no tabuleiro; **h3** - Combinação de h1 e h2.

**Algorithm 1** Combinação das heurísticas (h1 + h2 = h3).*

$f \leftarrow 0$
twos ← número de posições vazias
**if** *parent node* **then**
  *broken rules* ←peso por regra quebrada depois da última ação
  *row* ← número de casas vazias na linha alterada com a última ação (**h1**)
  *col* ← número de casas vazias na coluna alterada com a última ação (**h1**)
  $dif$ ← diferença entre o nº de 0's e 1's no tabuleiro (**h2**)
  $f$ ← twos + dif + brokenrules + row + col
**end if**
**return** $f$

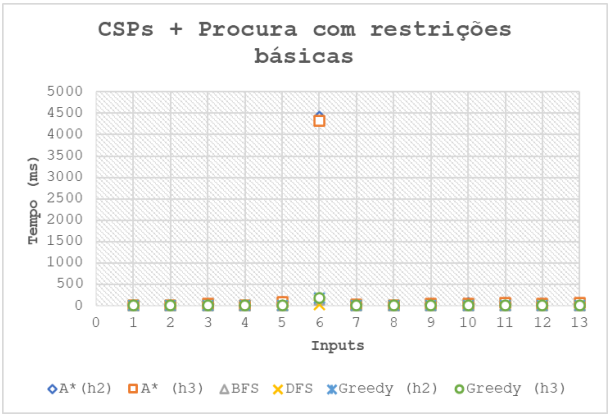
* h1 não apresenta a parcela *dif* e h2 não apresenta as parcelas *row* e *col*

| CSPs | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| nós expandidos / nós testados / nós gerados | | | | | | | | | | | | | |
| Algoritmos | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| BFS | 0/1/0 | 0/1/0 | 1/2/8 | 0/1/0 | 4/5/26 | 18/19/338 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |
| DFS | 0/1/0 | 0/1/0 | 2/3/8 | 0/1/0 | 3/4/26 | 3/4/68 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |
| A* (h2/h3) | 1/2/1 | 1/2/1 | 9/10/12 | 1/2/1 | 35/36/63 | 816/817/4100 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 |
| GREEDY (h2/h3) | 0/1/0 | 0/1/0 | 3/4/8 | 0/1/0 | 7/8/26 | 55/56/278 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |

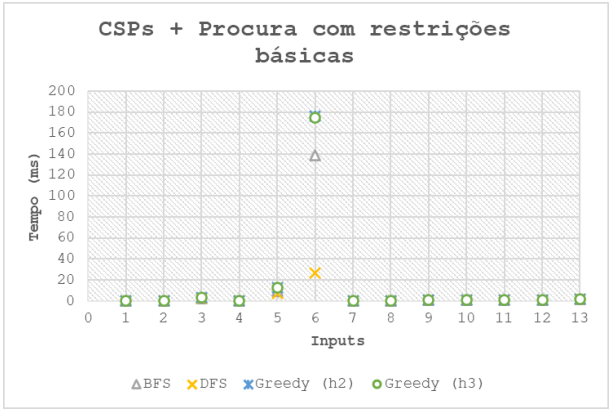
Figura 4: Nós resultantes da abordagem de procura aliada a CSPs com restrições **básicas** para os vários algoritmos.

| CSPs | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Times (ms) | | | | | | | | | | | | | |
| Algoritmos | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **A* (h2)** | 2.03 | 2.18 | 31.42 | 9.88 | 77.31 | 4418.55 | 25.07 | 5.34 | 43.17 | 52.73 | 59.67 | 44.5 | 51.85 |
| **A* (h3)** | 1.92 | 2.04 | 36.52 | 10.09 | 75.37 | 4327.56 | 24.08 | 5.54 | 43.35 | 51.04 | 60.55 | 49.3 | 57.57 |
| **BFS** | 0.19 | 0.24 | 2.43 | 0.32 | 9.38 | 138.61 | 0.5 | 0.53 | 0.68 | 0.79 | 0.85 | 1.11 | 1.62 |
| **DFS** | 0.22 | 0.21 | 2.46 | 0.3 | 7.26 | 26.78 | 0.49 | 0.54 | 0.65 | 0.82 | 0.83 | 1.15 | 1.58 |
| **GREEDY (h2)** | 0.21 | 0.23 | 3.21 | 0.29 | 12.9 | 176.22 | 0.49 | 0.54 | 0.69 | 0.76 | 0.85 | 1.15 | 1.58 |
| **GREEDY (h3)** | 0.19 | 0.26 | 3.31 | 0.33 | 12.66 | 174.63 | 0.49 | 0.51 | 0.69 | 0.92 | 0.84 | 1.13 | 1.57 |

Figura 5: Tempos da abordagem de procura aliada a CSPs com restrições **básicas** para os vários algoritmos.




(a) Todos os algoritmos de procura.




(b) Todos os algoritmos de procura menos A*.

Figura 6: Tempos para todos os inputs com a abordagem de procura aliada a CSPs .

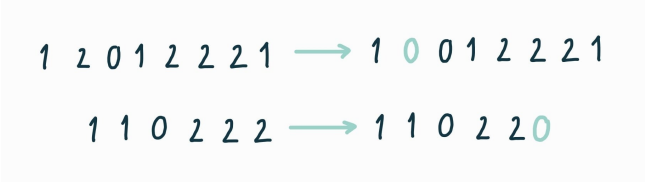



Figura 7: Exemplos de regras complexas.

| CSPs | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| nós expandidos / nós testados / nós gerados | | | | | | | | | | | | | |
| Algoritmos | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **BFS** | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |
| **DFS** | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |
| **A*** | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 | 1/2/1 |
| **GREEDY** | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 | 0/1/0 |

Figura 8: Nós resultantes da abordagem de procura aliada a CSPs com restrições **básicas + complexas** para os vários algoritmos, sendo a heurística h2 a usada.

| CSPs | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Times (ms) | | | | | | | | | | | | | |
| Algoritmos | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **A*** | 2.03 | 2.18 | 42.52 | 16.26 | 37.73 | 66.15 | 25.07 | 5.34 | 43.17 | 52.73 | 59.67 | 44.5 | 51.85 |
| **BFS** | 0.22 | 0.2 | 0.3 | 0.33 | 0.38 | 0.43 | 0.52 | 0.53 | 0.68 | 0.82 | 0.8 | 1.39 | 1.46 |
| **DFS** | 0.22 | 0.2 | 0.29 | 0.34 | 0.33 | 0.41 | 0.5 | 0.53 | 0.65 | 0.8 | 0.82 | 1.4 | 1.44 |
| **GREEDY** | 0.21 | 0.23 | 0.31 | 0.33 | 0.37 | 0.42 | 0.49 | 0.54 | 0.68 | 0.82 | 0.82 | 1.44 | 1.5 |

Figura 9: Tempos da abordagem de procura aliada a CSPs com restrições **básicas + complexas** para os vários algoritmos, sendo a heurística h2 a usada.