GRUPO 34
HUGO MANUEL ALVES HENRIQUES E SILVA - 99235
PEDRO CHARNECA FLORINDO - 99299

# RELATÓRIO

## DESCRIÇÃO DO OBJETIVO DO TRABALHO

O propósito do desenvolvimento deste projeto foi aplicar as técnicas de procura dadas em aula e otimizar a resolução de tabuleiros do jogo Takuzu através de CSPs e o uso de heurísticas.

---

## ANÁLISE DA APLICAÇÃO DE CSPS E CRIAÇÃO E ESCOLHA DE AÇÕES

A forma como aplicamos as CSPs e geramos e restringimos as ações foi a seguinte:

- Em primeiro lugar, em cada estado guardamos as posições ainda não preenchidas para não ter de percorrer o tabuleiro todo novamente.
- O principal objetivo deste primeiro passo é jogar sempre numa posição em que só haja uma possível jogada correta da qual temos a certeza. Para isto, verificamos se existem jogadas que criam um 3 em linha de zeros ou 3 em linha de uns, tanto na horizontal como na vertical. A segunda condição é verificar se já existe o número máximo de zeros ou uns aceitável por linha ou por coluna. (Recorremos ao uso de dicionários para não verificar múltiplas vezes as mesmas linhas/colunas). Estas são as nossas jogadas "obrigatórias", ou seja, a jogada oposta infringiria uma regra do jogo.
- Passando agora às jogadas das quais não temos a certeza do seu valor. Inicialmente, verificamos quais as linhas e colunas com o maior número de posições preenchidas. De seguida, passamos às adjacências. Naturalmente, jogar perto de uma posição já preenchida tem a vantagem de criar muitas mais jogadas "obrigatórias" e poupar a eficiência temporal e espacial do algoritmo a longo prazo. Se o número de adjacências for inferior a 2, esta restrição não é posta em causa, devido à sua pouca relevância. Caso contrário, esta restrição prevalece, e escolhemos uma das posições com maior número de adjacências.
- Finalmente, no caso de uma jogada obrigatória, retornamos apenas essa jogada, e para o caso da jogada não obrigatória, retornamos as duas jogadas possíveis (0 e 1) na melhor posição, que respeita de melhor forma as restrições impostas. Desta forma, obrigamos a que o algoritmo descubra o erro o mais rápido possível, caso este exista, levando a uma descoberta mais rápida da solução.

## ANÁLISE DA HEURÍSTICA DESENVOLVIDA

Em relação à heurística, qualquer infração das regras do jogo leva a uma heurística com valor infinito, impossibilitando a escolha deste estado em todos os casos.

Nos restantes casos, foi assumido como valor base da heurística o valor convencional de 10000. Assim, a cada condição a que o novo estado respeite que nos leva mais rapidamente até a solução, é decrementado o valor da heurística em uma unidade. As condições são as seguintes:

- Por cada valor igual, numa posição adjacente (vertical ou horizontal), ao que acabou de ser jogado no novo estado gerado, a heurística base é decrementada em uma unidade.
- Por cada posição adjacente à nova jogada que esteja preenchida é decrementada uma unidade à heurística base.

- Por cada posição preenchida na linha e coluna da jogada que gera o novo estado, é decrementada uma unidade à heurística base.
- Por fim, se a jogada for obrigatória, a heurística é 0, o que implica que esta seja escolhida sempre.

Inicialmente, sem recorrer à implementação desta primeira condição, o programa não era bem sucedido nos testes mais exigentes em nível de tempo e espaço.

## ANÁLISE DOS DADOS EXPERIMENTAIS E OBSERVAÇÕES

Devido ao enorme impacte do uso das CSPs, obtivemos valores muito próximos de execução em todos os algoritmos, na maior parte dos testes, devido à redução de criação de estados pouco úteis. No entanto, nos testes 11, 12 e 13, notou-se uma grande diferença (nomeadamente no teste 13) de tempos de execução, nós gerados e expandidos da BFS para os restantes algoritmos, o que, dentro do contexto do nosso problema, faz muito sentido, uma vez que com o uso de CSPs leva à criação de sequências de melhores jogadas, o que beneficia a procura em profundidade, de forma geral.

Assim sendo, e tal como esperado e comprovado experimentalmente, a DFS é o algoritmo com melhor desempenho na generalidade, em todos os testes, gerando e expandindo o menor número de nós. Os resultados obtidos na procura Greedy e na procura A* foram semelhantes, apresentando em todos os casos o mesmo número de nós gerados e expandidos. Isto deve-se ao facto do custo de cada arco não influenciar na escolha do nó, sendo que a escolha é apenas feita através da heurística.

Decidimos portanto usar a procura greedy, uma vez que a diferença de desempenho para a procura DFS é mínima, e desta forma, conseguimos exibir o uso da heurística.

Antes da implementação das CSPs, o algoritmo gerava demasiado estados, comprometendo a eficiência temporal e espacial, o que impedia que metade dos testes fossem bem sucedidos em tempo aceitável.

| teste1 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0005 | 7 | 7 |
| DFS | 0,0003 | 7 | 7 |
| Greedy | 0,0004 | 7 | 7 |
| A* | 0,0004 | 7 | 7 |

| teste2 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0007 | 7 | 7 |
| DFS | 0,0002 | 7 | 7 |
| Greedy | 0,0004 | 7 | 7 |
| A* | 0,0005 | 7 | 7 |

| teste3 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0045 | 45 | 46 |
| DFS | 0,0034 | 42 | 43 |
| Greedy | 0,0046 | 42 | 43 |
| A* | 0,0051 | 42 | 43 |

| teste4 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0042 | 46 | 57 |
| DFS | 0,0035 | 32 | 36 |
| Greedy | 0,0039 | 36 | 41 |
| A* | 0,0047 | 36 | 41 |

| teste5 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0079 | 64 | 66 |
| DFS | 0,006 | 55 | 56 |
| Greedy | 0,0099 | 58 | 60 |
| A* | 0,0074 | 58 | 60 |

| teste6 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0182 | 133 | 138 |
| DFS | 0,0129 | 79 | 83 |
| Greedy | 0,0195 | 95 | 99 |
| A* | 0,021 | 95 | 99 |

| teste7 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0131 | 69 | 69 |
| DFS | 0,0087 | 69 | 69 |
| Greedy | 0,0101 | 69 | 69 |
| A* | 0,0105 | 69 | 69 |

| teste8 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0022 | 19 | 20 |
| DFS | 0,0014 | 19 | 20 |
| Greedy | 0,0025 | 19 | 20 |
| A* | 0,0019 | 19 | 20 |

| teste9 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0313 | 139 | 139 |
| DFS | 0,0239 | 139 | 139 |
| Greedy | 0,0308 | 139 | 139 |
| A* | 0,0264 | 139 | 139 |

| teste10 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,0463 | 184 | 184 |
| DFS | 0,0339 | 184 | 184 |
| Greedy | 0,0411 | 184 | 184 |
| A* | 0,0407 | 184 | 184 |

| teste11 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 2,7219 | 14799 | 19406 |
| DFS | 0,0415 | 180 | 192 |
| Greedy | 0,0468 | 186 | 199 |
| A* | 0,0472 | 186 | 199 |

| teste12 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 0,4522 | 2280 | 3431 |
| DFS | 0,0344 | 166 | 175 |
| Greedy | 0,0417 | 167 | 177 |
| A* | 0,0408 | 167 | 177 |

| teste13 | Time | Expanded | Generated |
|---|---|---|---|
| BFS | 32,4086 | 70439 | 131878 |
| DFS | 0,0603 | 180 | 196 |
| Greedy | 0,0708 | 184 | 202 |
| A* | 0,076 | 184 | 202 |