Relatório projeto IART 2021/2022

Grupo: al070 | Alunos: Valentim Santos (99343), Tiago Santos (99333)

Descrição da solução

Para resolver o problema Takuzu foram definidas restrições básicas:

- Não podem existir **trios** (tanto de 0s ou de 1s).
- Uma linha/coluna apenas pode ter no máximo (**board.size//2)** 0s ou 1s, ou, no caso de tabuleiros de tamanho ímpar, (**board.size//2 + 1**).
- Não podem existir linhas/colunas **iguais**.

Análise dos resultados (Gráficos no final do relatório)

Tendo em conta a natureza dos testes utilizados para a criação dos gráficos, é de esperar que sejam todos semelhantes, contudo podemos concluir que utilizar **DFS** é a forma mais eficiente de resolver problemas deste tipo (levando em consideração a solução apresentada, visto que pode não ser o caso para outras). Mesmo que **BFS** tenha tempos semelhantes (para os testes usados), esta gera muitos mais nós do que uma **DFS**, tornando desta forma a última mais eficiente.

Todos os algoritmos de procura são completos, para o problema apresentado, pois as restrições definidas não permitem a existência de **loops**, e desta forma, o programa apenas para a sua execução quando encontra uma solução.

Podemos reparar que para as procuras **Greedy search** e **A*** os gráficos são muito semelhantes, tal deve-se ao facto de que, ao retornar o **conjunto de actions** para uma certa posição, já fazemos uma "escolha greedy" destas mesmas, escolhendo apenas a primeira jogada certa para a primeira posição vazia, ou no caso de não existir nenhuma jogada certa, escolhemos as duas possíveis jogadas para a primeira posição vazia que encontramos, tornando assim as procuras **Greedy search** e **A*** bastante semelhantes.

Análise da heurística

A heurística usada é a seguinte (para cada posição vazia):

h = a * (nº de pos vazias na linha e coluna da pos atual) + b * (4 – nº de adj)

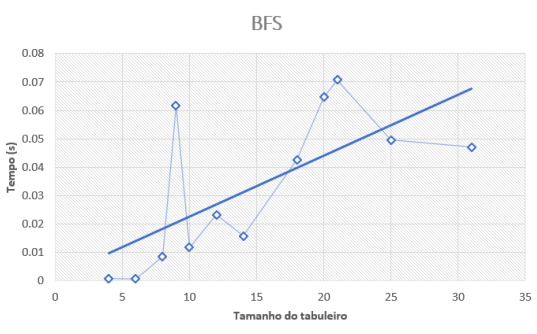
A heurística de cada estado (do tabuleiro) é calculada somando as heurísticas de todas as posições vazias do mesmo.

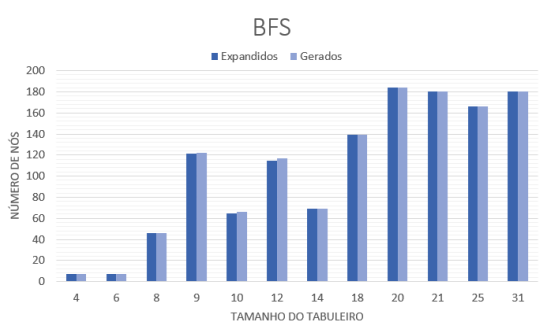
Quanto menor a heurística de cada posição melhor. Quantas mais adjacências preenchidas, assim como número de posições preenchidas na mesma linha e coluna, uma posição tiver menor será a sua heurística, sendo por isso melhor, assim o é porque o preenchimento destas posições, mesmo que não leve de imediato a uma solução, tem maior chance de obrigar outras posições a terem apenas uma jogada possível, sendo assim mais rápido chegar a uma solução.

Foi experimentado apenas considerar o **número de posições vazias por linha e coluna** para a heurística, contudo, tal criaria bastantes "empates", e por isso, foi adicionado o segundo fator, o **número de adjacências.**

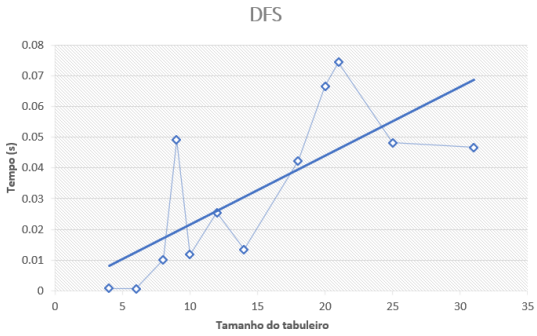
São também utilizadas duas variáveis, **alfa (a)** e **beta (b)**, para dar mais enfase a um ou a outro dos dois fatores falados acima, para a nossa solução, decidimos considerar **a=1** e **b=2**, dando desta forma mais "força" ao número de adjacências, com o objetivo de **diminuir os empates**.

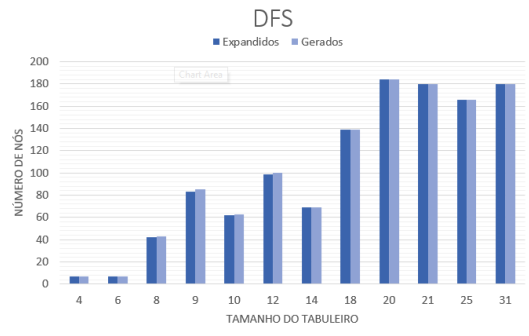
BFS:



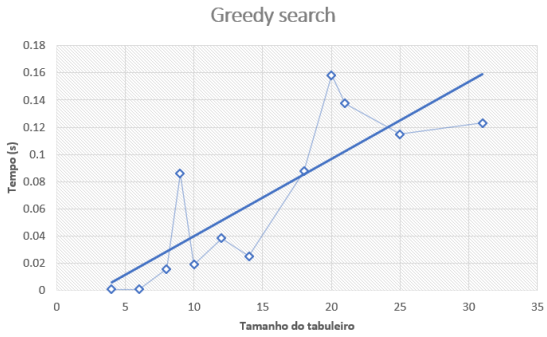


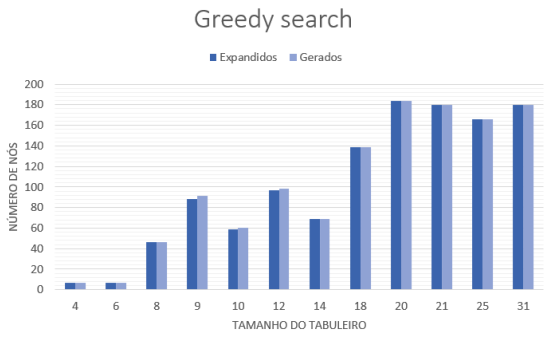
DFS:





GREED SEARCH:





A*: