Relatório - Numbrix

Identificações:

- Grupo: Alameda 06
- Diogo Venâncio 95555
- Sofia Morgado 95675

Solução implementada (e versões anteriores):

Versão 1:

Inicialmente, a solução foi recorrer a uma A* cuja função de heurística verificava se a ação encontrava um caminho mais longo desde um valor até ao seguinte valor maior que ela e sempre que o caminho se tornava maior, retornava um valor melhor. Ora, isto fez com que a tivesse uma complexidade de O(N^2) e fosse lenta.

Versão 2:

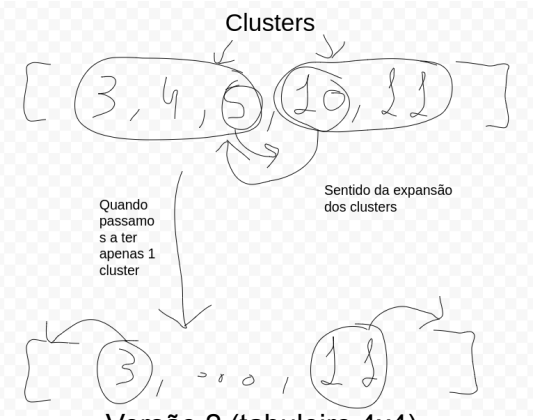
Depois do fracasso da anterior, decidimos avaliar melhor o problema e usar as suas propriedades para o tentar resolver e com isto, desenvolvemos uma A* que usava o conceito de clusters (valores já colocados no tabuleiro que formam subsecções de caminho válidas), que usavam os valores "pivot" (valores nas extremidades dos clusters) para os expandir. Tentava sempre expandir clusters mais pequenos de modo a que as brechas inter-clusters fossem sendo preenchidas aos poucos e dávamos uma maior prioridade a ações cujo resultado da fórmula: abs(X - Y)/abs(Xi-Yi) + abs(Xj-Yj) (sendo X a ação tomada e Y o nó o maior valor acima deste) fosse mais próximo de 1 dado que esta fórmula tem a propriedade de que quando o seu resultado é 1, a ações é válida. No final, quando só restava um cluster, uśamos o número de valores possíveis para cada coordenada para fechar o resto das posições. Esta solução, embora pareça boa, levava na mesma bastante tempo porque gerava-mos imensas ações (até este ponto, na nossa implementação, tínhamos ações para todas as fronteiras de todos os clusters)

Versão 3 (e final):

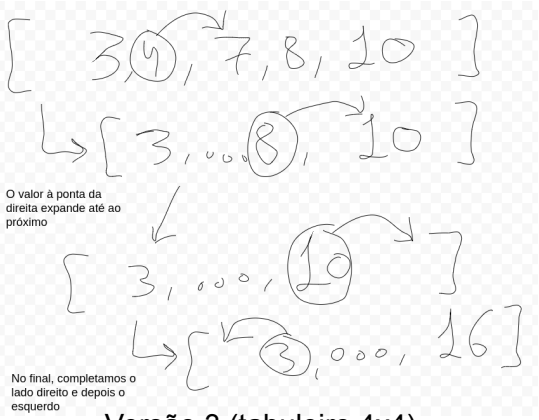
Na última versão, decidimos mudar de estratégia e usámos uma DFS e tornamos mais pequeno o número de ações geradas mas mantendo os conceitos atrás referidos.
Agora, a função actions() vai apenas gerar ações para o cluster que contém o valor mais baixo até atingir o próximo e assim sucessivamente. Quando só resta um cluster, completamos as posições mais à direita do array de valores inseridos e logo a seguir, fazemos o mesmo para a ponta oposta. As ações geradas vão ser ordenadas de acordo com o valor resultante da função descrita atrás através de uma heap. Isto vai fazer com que a DFS comece por usar ações mais próximas do resultado final.

Imagens:

Para melhor explicar a evolução da lista de clusters, foram anexadas as imagens abaixo:



Versão 2 (tabuleiro 4x4)



Versão 3 (tabuleiro 4x4)

Resultados obtidos:

Os resultados obtidos através do uso da função “compare_searchers()” sobre o ficheiro de teste “input10.txt” (Greedy e A* usaram o input1.txt porque não acabavam) disponibilizado pelos professores foram os seguintes:

Algoritmo	Tempo de execução (s)	# Sucessos (expandidos)	# Testes ao Objetivo	# Estados
DFS	0.0987	407	408	420
BFS	0.32734	1617	1618	1617
Greedy	0.02431	5	6	21
A*	0.03087	5	6	21

Análise crítica dos resultados:

Eficácia:

Atendendo a que o algoritmo aplicado ao problema, quando termina, é sempre uma solução ótima, esta solução é eficaz.

Completude:

Atendendo a que, quando o algoritmo é executado num tabuleiro válido, termina sempre, este é completo.

Análise da heurística e comparação com similares:

As análises foram descritas na secção de “Solução implementada (e versões anteriores)” não só porque porque assim permite dar um contexto de como foi usada mas também da evolução e aprimoração, rejeição e razões de descarte das mesmas.