UNIVERSIDADE FEDERAL DE MATO GROSSO

CURSO DE ENGENHARIA DE COMPUTAÇÃO

DISCIPLINA DE INTELIGÊNCIA ARTIFICIAL

RELATÓRIO DE ATIVIDADE PRÁTICA 1
QUEBRA-CABEÇA DE OITO PEÇAS

BRUNO FERNANDES DE LIMA SANTOS

CUIABÁ, MT

2017

1 RESUMO

1.1 Introdução

Este relatório apresenta os resultados obtidos da atividade prática 1, da disciplina de Inteligência Artificial. A atividade consiste na elaboração de uma aplicação (em MATLAB) de um agente racional capaz de resolver o problema do puzzle-8[1]. Este é um problema que pode ser resolvido através de algoritmos que realizem **busca informada** (ou busca heurística), sendo que o algoritmo **A*** (leia-se "A estrela") foi implementado para tal.

A função que realiza a busca de A* recebe duas entradas: uma determinada configuração do jogo, que pode ser representada em uma matriz 3×3, e deve receber também uma função heurística para tratar o problema. Para isto, temos **três funções heurísticas** distintas aqui, denominadas *Hamming*, *Manhattan* e *Heuristic*.

Para mais detalhes sobre a funcionalidade de cada uma das funções heurísticas (exceto a Heuristic, que será tratada aqui), implementação, bem como funcionalidade do A* etc, consulte o enunciado desta atividade prática.

1.2 Resumo dos resultados obtidos

A primeira observação a ser feita é que o desempenho da busca realizada pelo algoritmo A* depende (e muito) da função heurística escolhida. Por exemplo, temos, em média, que a quantidade de inserções de nós pelo A* utilizando a heurística Hamming é por volta de 8.15 vezes maior do que esta quantidade utilizando-se a heurística Manhattan. Estes resultados serão detalhados nas próximas seções.

2 RESULTADOS E DISCUSSÕES

2.1 Desempenho do A*

Foram executados diferentes testes de execução com o algoritmo A*. O objetivo foi analisar quais os passos (estados) visitados pelo algoritmo, partindo do estado inicial até o estado objetivo, bem como analisar o **desempenho** do algoritmo para resolver o puzzle. O quesito escolhido para medir o desempenho do algoritmo, para um determinado estado inicial e determinada heurística, foi o **número de inserções e remoções na fila de prioridades**. O quesito de tempo de execução poderia, também, ser utilizado, mas este

1 Mais informações sobre o puzzle-8 em: http://www.aiai.ed.ac.uk/~gwickler/eightpuzzle-inf.html

estaria sujeito às condições de máquina (e.g. sistema operacional, quantidade de memória utilizada, quantidade de CPU utilizada etc.) e, portanto, este critério foi desconsiderado.

Cinco testes do algoritmo foram realizadas, utilizando **cada uma das heurísticas** e foi calculado o **número de inserções e remoções** de nós na fila de prioridades utilizada pelo A*. Abaixo são exibidos os cinco casos de teste, correspondendo aos cinco estados iniciais da matriz do jogo, enumerados de 1 a 5.

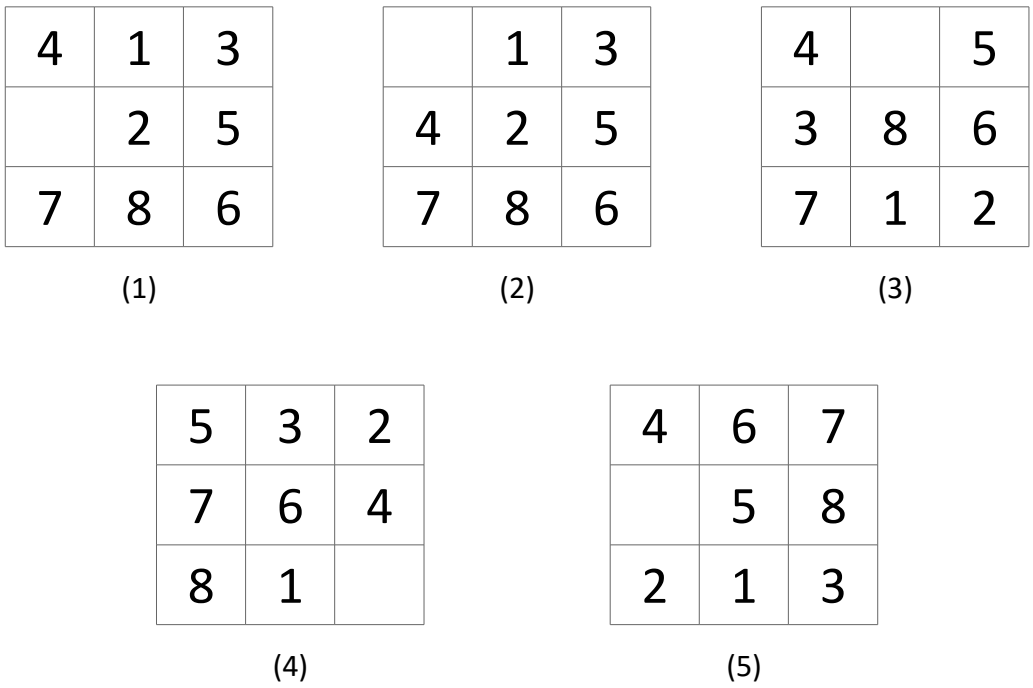

Figura 1: Os cinco estados iniciais para teste.

Como esperado, em todos os testes o desempenho do algoritmo foi **ótimo**, isto é, o número de jogadas para alcançar o estado objetivo foi o mínimo. Entretanto, para cada heurística houve **variação** neste número de inserções e remoções. Na tabela a seguir temos estes resultados exibidos. Note que, para o caso de teste 5, a heurística de Hamming não foi testada devido ao tempo elevado de execução.

Caso de teste	Heurística	Nº de inserções	Nº de remoções
1	Hamming	12	6
	Manhattan	12	6
	Heuristic	12	6
2	Hamming	10	5
	Manhattan	10	5
	Heuristic	10	5
3	Hamming	18306	10843
	Manhattan	1154	692
	Heuristic	922	513
4	Hamming	28696	17061
	Manhattan	2865	1705
	Heuristic	2355	1313
5	Hamming	-	-
	Manhattan	15979	9691
	Heuristic	12750	7086

Tabela 1: Resultados dos casos de teste.

Note que a diferença de desempenho do A* utilizando as funções de Hamming e Manhattan é drástica. O algoritmo possui melhor desempenho com a função Heuristic, sendo esta uma ligeira modificação da função de Manhattan. Mais detalhes sobre esta heurística serão exibidos na seção 2.3.

2.2 Detecção de estados sem solução

Foi implementado ao algoritmo A* uma função que detecta se o estado recebido pelo algoritmo possui solução ou não. A ideia geral da função consiste em analisar cada estado que é removido da fila de prioridades do A*. Se for detectado um estado inválido, a função A* retorna erro.

Um estado inválido é qualquer estado idêntico ao estado objetivo exceto por uma característica: possui um par de células (exceto a célula em branco) trocadas. O estado na figura seguinte é um exemplo de estado inválido.

1	2	3
4	5	6
8	7	

Figura 2: Estado inválido (sem solução).

Note que o algoritmo não detecta de imediato se um determinado estado culminará em um estado inválido ou não. Em vez disso, o algoritmo roda normalmente, buscando pelo estado objetivo. Quando ele seleciona um estado inválido para expandir (como o mostrado na Figura 2), um erro é retornado e o programa encerra. A seguir temos três exemplos de estados iniciais que, posteriormente, resultaram em um estado inválido pela busca A*.

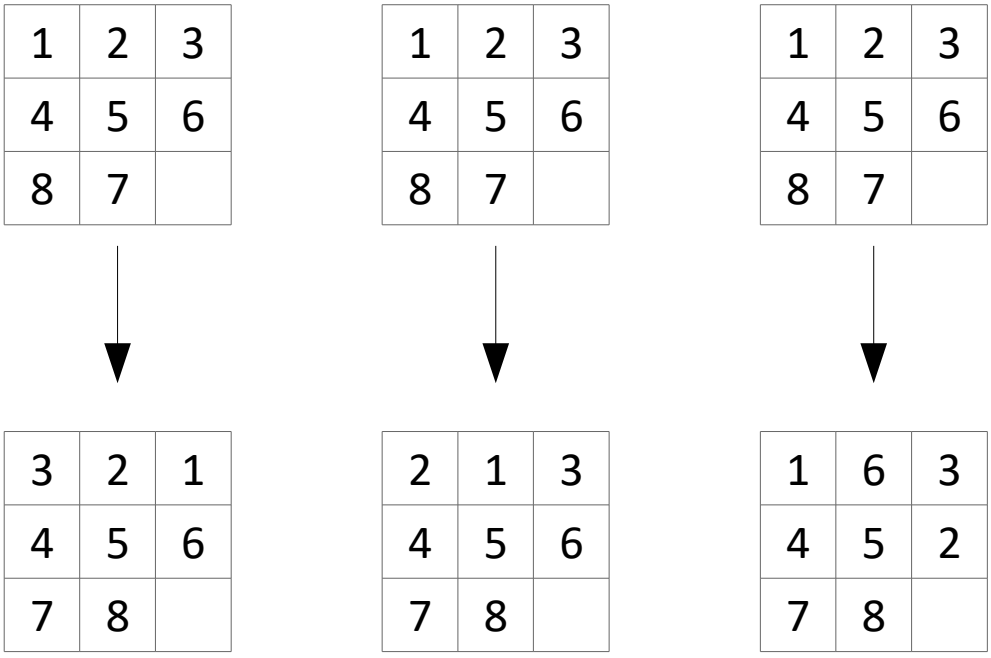

Figura 3: Estados que culminam em estados inválidos.

2.3 Heurísticas implementadas

2.3.1 A função Heuristic

A função realiza um “upgrade” da heurística de Manhattan. O algoritmo consiste em calcular o custo de um estado S utilizando a função de Manhattan. Em seguida, são gerados

os estados filhos de S e calculados os respectivos custos de Manhattan dos mesmos. Caso algum destes estados filhos tenha um custo de Manhattan menor do que o estado pai, a função devolve o valor de custo calculado (isto é, o mesmo custo de Manhattan). Caso contrário, a função devolve o custo de Manhattan acrescentado de um.

O argumento aqui é de que a heurística de Manhattan estima a quantidade de movimentos necessários para alcançar-se o estado objetivo. No entanto, caso nenhum estado filho possui custo menor do que o estado pai, é fácil ver que um movimento a mais será necessário para alcançar o objetivo. Embora intuitivamente esta função pareça ser mais complexa e lenta do que a função de Manhattan, testes executados comprovam a eficácia da função Heuristic, como os exibidos na Tabela 1.

Com a função Heuristic, o algoritmo continua obtendo o caminho ótimo, assim como com a função Manhattan, porém em um tempo médio de execução muito menor. Isto se dá pois, com a função Heuristic, o algoritmo de A-star **seleciona** melhor os estados, inserindo na fila **menos** estados com valores iguais.