FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

INTELIGÊNCIA ARTIFICIAL

2013-2014

Relatório:

TP1a - Rover Marciano

13 de Outubro de 2013

Docente: Francisco Azevedo

Turno Prático: P4

Realizado por:

Ricardo Cruz, nº 34951

Ricardo Gaspar, nº 42038

Luís Silva, nº 34535

Índice

Introdução

Este trabalho consiste em duas partes de desenvolvimento, a primeira incide sobre os algoritmos de procura leccionados e a segunda sobre a resolução um problema que envolve um Rover em Marte.

Na primeira parte, o objectivo consiste na implementação e comparação do algoritmo de procura A* baseado em outros anteriormente leccionados e implementados.

Na segunda parte, dado um Rover autónomo e um terreno em Marte com uma serie de características especificas, temos por objectivo descobrir a melhor rota entre dois pontos deste mesmo terreno. Para esta exploração pretende-se que seja escolhido o algoritmo de procura que melhor solucione o problema.

Para este efeito teremos que implementar as heurísticas que solucionam ao problema em causa, assim como demonstrar os resultados obtidos e fazer uma análise de modo a compreender qual a heurística mais apropriada.

Testes e Resultados dos Algoritmos

Para ter uma noção prática do desempenho dos vários algoritmos na resolução de problemas foram realizados alguns testes. Estes incidiram sobre dois problemas: VacuumWorld e NPuzzle.

VacuumWorld

Mundo com 5 zonas

Mundo: {true, true, false, false, true}

Estado inicial: Posição 1

Objectivo: Limpar todas as posições

Algoritmo	Solução encontrada	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
BreadthFirst	[LEFT, SUCK, RIGHT, SUCK, RIGHT, RIGHT, RIGHT, SUCK]	5184	15553	8	0,06218
UniformCostSearch	[LEFT, SUCK, RIGHT, SUCK, RIGHT, RIGHT, RIGHT, SUCK]	30	91	8	0,002816
DepthFirstSearch	[LEFT, SUCK, RIGHT, RIGHT, RIGHT, RIGHT, SUCK, LEFT, LEFT, LEFT, SUCK]	12	36	11	0,002101

Tabela 1 – Resultados dos algoritmos de procura no VacuumTest

NPuzzle

Puzzle de 8 peças

Estado inicial: {{7,2,4}, {5,null,6}, {8,3,1}}

Objectivo: Resolver o puzzle de modo a que o estado final seja {{1,2,3}, {4,null,5}, {6,7,8}}.

Algoritmo	Solução encontrada	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
BreadthFirst	java.lang.OutOfMemoryError	-	-	-	-
UniformCostSearch	[RIGHT, DOWN, LEFT, UP, LEFT, UP, RIGHT, RIGHT, DOWN, LEFT, LEFT, UP, RIGHT, RIGHT, DOWN, LEFT, DOWN, LEFT, UP, RIGHT, DOWN, RIGHT, UP, LEFT]	146756	390725	24	115,569716
DepthFirstSearch	java.lang.StackOverflowError	-	-	-	-
AStarSearch	[RIGHT, DOWN, LEFT, UP, LEFT, UP, RIGHT, RIGHT, DOWN, LEFT, LEFT, UP, RIGHT, RIGHT, DOWN, LEFT, DOWN, LEFT, UP, RIGHT, DOWN, RIGHT, UP, LEFT]	442	1260	24	0,045721

Tabela 2 – Resultados dos algoritmos de procura no NPuzzleTest nas condições mencionadas.

Puzzle de 16 peças

Estado inicial: {{7,6,14,4}, {2,3,8,15}, {1,9,10,11}, {5,null,13,12}}

Objectivo: Resolver o puzzle de modo a que o estado final seja {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}, {13,14,15,null}}.

Algoritmo	Solução encontrada	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
BreadthFirst	java.lang.OutOfMemoryError	-	-	-	-
UniformCostSearch	Não termina em tempo útil (>5min)	-	-	-	-
DepthFirstSearch	java.lang.StackOverflowError	-	-	-	-
AStarSearch	[DOWN, LEFT, DOWN, LEFT, UP, RIGHT, DOWN, DOWN, RIGHT, RIGHT, UP, UP, UP, LEFT, LEFT, DOWN, DOWN, DOWN, RIGHT, UP, LEFT, LEFT, UP, UP, RIGHT, DOWN, DOWN, DOWN, RIGHT, RIGHT, UP, UP, UP, LEFT, LEFT, LEFT]	106887	321291	36	7,930651

Tabela 3 – Resultados dos algoritmos de procura no NPuzzleTest nas condições mencionadas.

Problema do Rover Marciano

Características do Rover

Estado

O estado do *Rover* é caracterizado pela posição onde este se encontra no mapa, ou seja as coordenadas x e y desse ponto, bem como um *BitmapTerrain* que representa o mapa no qual o *Rover* se desloca em combinação com as características desse mesmo terreno.

Operadores

Os operadores considerados para este problema são as direcções possíveis de seguir em cada posição (estado) do Rover no mapa. Ou seja, o Rover pode deslocar-se apenas segundo os quatro pontos cardeais (Norte, Este, Sul, Oeste) e quatro pontos colaterais (Nordeste, Noroeste, Sudeste, Sudoeste).

Custo

O custo de cada passo do movimento em linha recta, para qualquer uma das quatro direcções (Norte, Sul, Este , Oeste) é de 10, tendo em conta que cada representa pixel 10 metros. Já o custo do passo quando o movimento é na diagonal é $\sqrt{200}$. Este valor resulta da aplicação do teorema de Pitágoras:

Sendo $a = b = 10$, logo por aplicação do teorema de Pitágoras $c = \sqrt{a^2 + b^2} = \sqrt{10^2 + 10^2} = \sqrt{200}$.

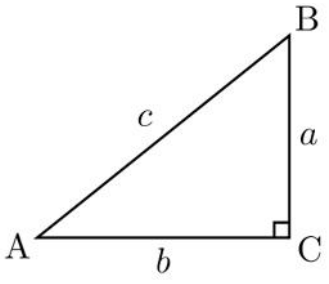

Figura 1 – Aplicação do Teorema de Pitágoras para o cálculo da diagonal.

O custo da deslocação do *Rover* é depois afectado em função do terreno onde nos encontramos. Sendo que como podemos estar perante de 3 tipos de terrenos distintos (*Plain, Sand e Rock*), temos três valores inteiros para serem multiplicados pelo custo, simbolizando a dificuldade da deslocação do *Rover*.

Por fim temos ainda um factor multiplicativo que potencia o custo, conforme a diferença de alturas do terreno em pontos do caminho, regendo-se conforme a equação seguinte:

$$e^{\sqrt{|\Delta h|}}$$

Estudo das Heurísticas

Para a resolução do problema do *Rover Marciano*, fizemos uma selecção de três heurísticas, sendo que em uma delas utilizamos uma variante à original de forma a tentar alcançar a melhor solução.
Assim sendo no total testamos a Manhattan, Distancia Euclidiana em XY, Distancia Euclidiana em XYZ e a Distancia Diagonal.

Distância de Manhattan

Em primeiro lugar temos a Distancia de Manhattan que nada mais é do que a heurística standard para uma grelha quadrada de pontos onde apenas nos podemos mover em quatro direcções.
Podemos à partida dizer que esta heurística não é admissível, ora não é possível garantir que esta escolha um caminho óptimo, ou seja, o cálculo da heurística pode facilmente passar o custo real do caminho a ser percorrido, tornando a não optimista.

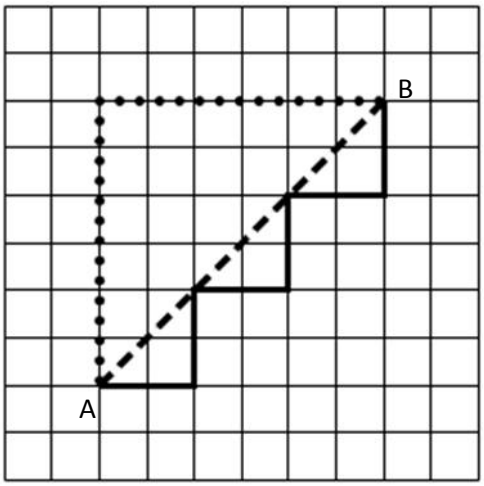

Figura 2 - Exemplo da aplicação do Distancia de Manhattan entre os pontos A e B.

Distância Euclidiana XY (sem altura)

Como o *Rover* pode mover-se em 8 direcções diferentes, podemos utilizar a distância entre pontos para calcular a heurística.
Esta faz o cálculo da distância entre dois pontos no plano de acordo com a repetição do *Teorema de Pitágoras*.
Pode-se ter confiança que esta é admissível, ora a distância entre dois pontos é sempre inferior ou igual á diagonal entre esses mesmos pontos.

Distância Euclidiana XYZ (com altura)

Em suma o calculo desta heurística é muito semelhante à anterior, mas neste caso inclui-se o factor altura dos pontos no cálculo. O que leva a uma melhor estimativa do custo do caminho, transformando o que era um cálculo de apenas a distância entre dois pontos, para um que considera a magnitude do vector entre esses pontos.

Tem a desvantagem de inserir mais complexidade no algoritmo e por consequência torna-lo mais lento na execução.

Distância Diagonal

Neste caso temos que a heurística estima com grande precisão o custo real do movimento do *Rover* entre dois pontos. Como não é factorizada a altura neste cálculo, se os dois pontos estiverem a alturas iguais no mapa a heurística estima o custo real do deslocamento, no caso em que os pontos estejam a alturas diferentes, estima abaixo do custo real o que torna a heurística optimista e consequentemente admissível.

Heurística escolhida

Para o problema do *Rover Marciano* escolhemos a heurística da Distancia Diagonal, ora como mostramos em cima esta é admissível, e segundo os nossos testes práticos expande menos nós e tem um tempo de execução menor que as restantes heurísticas.

Comparação entre heurísticas

Aqui apresentamos as tabelas comparativas entre as quatro heurísticas testadas. Sendo que fizemos o teste para cinco conjuntos de pontos em lugares distintos do mapa.

Nos testes realizados, cujos resultados se encontram em baixo, todos os pontos estão a $\frac{1}{10}$ da escala do mapa. Isto é, as coordenadas situam-se no intervalo [0;1000].

Teste 1

Um trajecto que, em linha recta, faz uma diagonal imaginária desde o meio superior direito até ao meio inferior esquerdo. Tem pelo meio duas grandes zonas de rocha.

Ponto inicial: x = 611, y = 190

Ponto Final: x = 113, y = 714

Heurística	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
EuclideanDistance	537591	4296292	1418978	43,295483667
EuclideanDistanceXYZ	534778	4273794	1418978	68,589031
Manhattan	395927	3165632	1418978	9,083406
DiagonalDistance	515526	4119988	1418978	8,555044

Tabela 4 – Resultados do algoritmo A utilizando cada uma das heurísticas nas condições do teste 1.*

Teste 2

Um trajecto que, em linha recta, faz uma diagonal imaginária desde o canto superior esquerdo até ao canto inferior direito. Tem pelo meio duas grandes zonas de rocha.

Ponto inicial: x = 75, y = 132

Ponto Final: x = 927, y = 866

Heurística	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
EuclideanDistance	953447	7617251	2336215	88,337999
EuclideanDistanceXYZ	950585	7594421	2337710	127,694178
Manhattan	925555	7394901	2336215	28,650027
DiagonalDistance	950428	7593312	2336215	17,535228

Tabela 5 – Resultados do algoritmo A utilizando cada uma das heurísticas nas condições do teste 2.*

Teste 3

Um trajecto que faz uma linha horizontal imaginária próximo do limite superior do mapa. Tem apenas umas pequenas zonas de areia.

Ponto inicial: x = 20, y = 10

Ponto Final: x = 900, y = 10

Heurística	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
EuclideanDistance	77845	620941	961217	2,074535
EuclideanDistanceXYZ	77463	617894	961217	2,89908
Manhattan	45802	364444	969287	0,520945
DiagonalDistance	60886	485425	961217	0,865068

Tabela 6 – Resultados do algoritmo A utilizando cada uma das heurísticas nas condições do teste 3.*

Teste 4

Um trajecto que faz uma linha vertical imaginária próximo do limite direito do mapa. Tem duas zonas com alguma de areia.

Ponto inicial: x =20, y = 10

Ponto Final: x =20, y = 900

Heurística	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
EuclideanDistance	138988	1108291	1237307	6,017751
EuclideanDistanceXYZ	137148	1093583	1237307	10,131572
Manhattan	100675	802141	1237307	1,035455
DiagonalDistance	124547	992913	1237307	1,661944

Tabela 7 – Resultados do algoritmo A utilizando cada uma das heurísticas nas condições do teste 4.*

Teste 5

Um trajecto que faz uma linha horizontal imaginária próximo do limite inferior do mapa. Tem uma zona bastante grande de areia.

Ponto inicial: x =980, y = 10

Ponto Final: x =980, y = 900

Heurística	Nós Expandidos	Nós Gerados	Custo	Tempo de execução (s)
EuclideanDistance	520303	4157503	1977922	41,859711
EuclideanDistanceXYZ	508352	4061940	1977922	59,793179
Manhattan	371778	2970017	1977922	6,919839
DiagonalDistance	505522	4039351	1977922	7,497395

Tabela 8 – Resultados do algoritmo A utilizando cada uma das heurísticas nas condições do teste 5.*

Conclusão

No que diz respeito à comparação de algoritmos e procura cega para o problema do *VacuumWorld* pode-se se constar que os algoritmos *BreadthFirst* e *UniformCostSearch* encontram a solução óptima, apesar de explorarem mais nós. Ao passo que o *DepthFirstSearch* explora menos nós, no entanto encontra uma solução de maior custo.
Seguidamente no problema do *NPuzzle*, foi realizada uma comparação entre os algoritmos de procura cega mencionados anteriormente e outro de procura informada, mais concretamente o *AStarSearch*. Em ambos os testes, para 8 e 16 peças, pode se observar que o *AStarSearch* não só encontra a solução óptima como é o que explora menos nós.

Para resolver o problema do *Rover Marciano*, recorrendo ao *AStarSearch*, foram estudadas quatro heurísticas. Este estudo incidiu sobre a sua admissibilidade, bem como o desempenho nos testes realizados. Destas verificou-se que a melhor é a *Distancia Diagonal* uma vez que encontra a solução óptima e explora menos nós.
Em suma, quanto mais próximo for o valor da heurística do custo do caminho óptimo, menos nós expande e mais rápida é. Contudo há que ter em conta que se a heurística contiver cálculos complexos isso pode prejudicar a velocidade de execução do algoritmo.