RELATÓRIO TP1

(1ª PARTE)

Grupo 53 (P5):

André Fragoso, nº45156
Gonçalo Almeida, nº45353

*O terceiro elemento do grupo não colaborou na realização do projeto.

Pretendeu-se com a realização da primeira parte deste projeto um aprofundamento dos conhecimentos referentes aos algoritmos de procura informada (em Java). Esta família de algoritmos utiliza determinado conhecimento sobre o problema em questão para determinar a ordem pela qual os nós serão expandidos, conhecimento este concretizado através de heurísticas (estimativas).

Esta primeira parte do projeto centra-se na implementação do algoritmo de procura informada A* (AStarSearch), na modelação do problema do Rover Marciano e na definição de heurísticas adequadas/apropriadas para este mesmo problema.

O algoritmo A* é um algoritmo de procura pelo melhor primeiro que combina o custo para chegar ao nó corrente e a sua heurística (custo estimado do caminho mais curto para atingir o objetivo a partir deste). Desta forma, o primeiro nó a ser expandido é aquele que aparenta ser o mais promissor. Para garantir a otimalidade do algoritmo implementado, a heurística tem de ser admissível. Por outras palavras, o valor da heurística em cada um dos nós nunca sobrestima o custo real de alcançar o objetivo.

Seguidamente utilizaram-se dois algoritmos de procura para efeitos de comparação de desempenho:

- **Procura por Custo Uniforme**, em que a função de avaliação é idêntica ao custo total do caminho percorrido;

- **Procura A***, em que a função de avaliação é a soma do custo do caminho já percorrido com o valor da heurística.

Descrição sumária dos resultados obtidos em cada um dos exemplos:

nPuzzle (3x3)
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]
{Node Expansions=141768, Nodes Generated=379535, Nodes Repeated=180445, Runtime (s)=0.889223505}
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]
{Node Expansions=442, Nodes Generated=1260, Nodes Repeated=208, Runtime (s)=0.039727345}

nPuzzle (4x4)
UniformCostSearch
Overhead
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]
{Node Expansions=106887, Nodes Generated=321291, Nodes Repeated=82756, Runtime (s)=7.063437765}

VacuumWorld
UniformCostSearch
[LEFT, SUCK, RIGHT, SUCK, RIGHT, RIGHT, RIGHT, SUCK]
{Node Expansions=29, Nodes Generated=88, Nodes Repeated=48, Runtime (s)=0.007077673}
AStarSearch
[LEFT, SUCK, RIGHT, SUCK, RIGHT, RIGHT, RIGHT, SUCK]
{Node Expansions=23, Nodes Generated=70, Nodes Repeated=35, Runtime (s)=0.005181883}

Como se pôde constatar pelos resultados obtidos, o algoritmo implementado levou a uma solução mais eficiente dos problemas, demonstrando desta forma a vantagem de se introduzir uma estimativa do custo de se ir de um determinado nó até ao nó objetivo, em relação à procura uniforme que, contrariamente ao algoritmo A* apenas assume o custo total do caminho percorrido como função de avaliação.

Atendendo a que o problema central desta primeira parte do projeto é o problema do Rover Marciano (procura informada), apresentar-se-á uma breve descrição deste mesmo problema e demonstrar-se-á de seguida as vantagens a que a escolha de uma boa heurística pode conduzir em termos de performance, e desta forma, determinar a melhor heurística para a resolução do problema referido.

Breve descrição do problema do Rover Marciano

Uma agência espacial prepara-se para enviar um Rover para efetuar a exploração do planeta Marte. O estado de um Rover é descrito na classe RoverState, onde é guardada a sua posição no mapa (x, y e height) e o terreno em si. Tal como descrito no enunciado do problema, ele pode mover-se na direcção dos 4 pontos cardinais assim como os 4 pontos colaterais, sendo que estas direcções são representadas no enum RoverOperator (N, E, S, W, NE, SE, SW, NW). Para o custo do passo, a função retorna 0 se o passo não fizer nada ou então o resultado da fórmula dada (distancia euclidiana multiplicada pelo factor do terreno multiplicado pela expressão $e^{\sqrt{|\Delta h|}}$, tal como explicado no enunciado do problema). O objetivo para este trabalho é obter a melhor rota entre dois pontos no terreno.

Para se obter o resultado pretendido (óptimo) ter-se-ão em consideração heurísticas como:

- ✓ <u>Distância euclidiana no plano</u>
  (Math.sqrt(((goal_x-x)*(goal_x-x))+((goal_y-y)*(goal_y-y)))) - heurística admissível*, pois nunca sobrestima o custo real, pois não tem em conta nem a altura nem o tipo de terreno, considerando apenas a distância pelo chão;

- ✓ <u>Distância euclidiana no espaço</u>
  (Math.sqrt(((goal_x-x)*(goal_x-x))+((goal_y-y)*(goal_y-y))+((goal_height-height)*(goal_height-height)))) - heurística admissível*, pois nunca sobrestima o custo real, pois apesar de ter em conta a altura, não penaliza o tipo de terreno;

*Para encontrar heurísticas admissíveis basta relaxar/esquecer certas restrições do problema tornando-se, assim, mais fácil a determinação de uma heurística que não sobrestime o custo real.

- ✓ Distância de Manhattan (Math.abs(goal_x-x)+Math.abs(goal_y-y)) – heurística não admissível, pois uma vez que não considera os movimentos diagonais, sobrestimaria o custo real. Esta situação é facilmente detetada, como por exemplo, imaginando que quer o ponto de partida quer o ponto de chegada do Rover estão em terreno plano e à mesma altura sendo o caminho óptimo a diagonal por eles formada. O custo estimado sobrestimaria sempre o custo real;

- ✓ (*mult*/2.0) * Distância euclidiana no espaço – onde *mult* é 2, 3 ou 4, se o terreno onde o Rover se encontra for plano, arenoso ou rochoso, respetivamente. *mult* é incrementado em 3 unidades caso a diferença de altura entre o ponto onde o Rover se encontra e o ponto objetivo for superior a 10 unidades. Esta heurística contempla os vários dados do problema: coordenadas x e y, altura e tipo de terreno.

Resultados obtidos:

i=0 j=10 (distância curta)
h(n) = Distância euclidiana no plano
Pathcost = 20637
{Node Expansions=12470, Nodes Generated=99761, Nodes Repeated=82037, Runtime (s)=1.542096762}
h(n) = Distância euclidiana no espaço
Pathcost = 20637
{Node Expansions=12410, Nodes Generated=99281, Nodes Repeated=81663, Runtime (s)=1.333004752}
h(n) = Distância de Manhattan
Pathcost = 20637
{Node Expansions=8375, Nodes Generated=67001, Nodes Repeated=57207, Runtime (s)=1.081788825}
h(n) = (mult/2.0)*Distância euclidiana no espaço
Pathcost = 23392
{Node Expansions=12148, Nodes Generated=97185, Nodes Repeated=79243, Runtime (s)=1.322420316}

i=3 j=9 (distância média)
h(n) = Distância euclidiana no plano
Pathcost = 92662
{Node Expansions=149354, Nodes Generated=1194833, Nodes Repeated=1019559, Runtime (s)=15.879792879}
h(n) = Distância euclidiana no espaço
Pathcost = 92662
{Node Expansions=144074, Nodes Generated=1152593, Nodes Repeated=983391, Runtime (s)=15.191052281}
h(n) = Distância de Manhattan
Pathcost = 92662
{Node Expansions=115884, Nodes Generated=927073, Nodes Repeated=932538, Runtime (s)=14.390427564}
h(n) = (mult/2.0)*Distância euclidiana no espaço
Pathcost = 95555
{Node Expansions=8789, Nodes Generated=70313, Nodes Repeated=65911, Runtime (s)=1.235791303}

i=11 j=2 (distância grande)
h(n) = Distância euclidiana no plano
Pathcost = 158819
{Node Expansions=253911, Nodes Generated=2031289, Nodes Repeated=1734278, Runtime (s)=28.148553738}
h(n) = Distância euclidiana no espaço
Pathcost = 158819
{Node Expansions=252353, Nodes Generated=2018825, Nodes Repeated=1723642, Runtime (s)=27.729233871}
h(n) = Distância de Manhattan
Pathcost = 159247
{Node Expansions=103396, Nodes Generated=827169, Nodes Repeated=2112486, Runtime (s)=41.414513892}
h(n) = (mult/2.0)*Distância euclidiana no espaço
Pathcost = 174254
{Node Expansions=91961, Nodes Generated=735689, Nodes Repeated=3697954, Runtime (s)=64.353106628}

Tal como esperado, ao aplicar uma heurística não admissível ao problema que se pretende resolver, isto é, uma heurística que não garante a otimalidade do algoritmo, verifica-se um agravamento do desempenho/resultado obtido relativamente a heurísticas mais “robustas”.

Decidimos utilizar a distância euclidiana no espaço como a nossa heurística principal, pois é uma heurística admissível e simples de criar. E fácil de provar que é admissível:

Para ser admissível, uma heurística não pode sobrestimar o custo de chegar ao objectivo. Para tal, é necessário que:

$$\forall n, h(n) < h^*(n)$$

Sabemos que a fórmula da distância euclidiana é:

$$\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

Sabemos também que a fórmula do custo real do Rover entre dois pontos é:

$$e^{\sqrt{|\Delta z|}} * fact * \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

Como sabemos que tanto $fact$ como $e^{\sqrt{|\Delta h|}}$ são sempre positivos, é fácil entender que a heurística escolhida nunca sobrestima a fórmula do custo real, ou seja, que a heurística é admissível.