Universidade Federal de Minas Gerais

UFMG

Departamento de Ciência da Computação

Inteligência Artificial 2015-2.

Documentação do Trabalho Prático I.

Professor: Luiz Chaimowicz.

Monitor: Anderson Rocha Tavares.

Aluno: Elias Soares.

DCC

1 Introdução:

Este trabalho visa nos introduzir conceitos e algoritmos básicos da Inteligência Artificial. Há uma espécie de jogo do Pac Man mais simplificado, onde usaremos três algoritmos distintos para encontrar o objetivo. Tais algoritmos são: Busca em Largura (BSF), Busca em Profundidade (DSF) e o Algoritmo A*. Vamos executar vários testes e analisar o resultados deles.

O Algoritmo A*, em contraste com os outros dois, pega o nó que tem o menor custo, onde o custo é calculado da seguinte forma: $\mathbf{f(n) = g(n) + h(n)}$, sendo $\mathbf{g(n)}$ o custo real de ter saído do ponto inicial e chegado ao ponto atual e $\mathbf{h(n)}$ é a heurística que têm o custo esperado do ponto atual ao ponto alvo.

2 Desenvolvimento:

2.1 Implementação:

A linguagem de programação utilizada foi **Python 2.7**, nota-se que a função **print** do **Python 2** é diferente da do **Python 3**. Todo o programa foi feito e testado no sistema operacional **Debian 8.2**. O trabalho foi testado em uma máquina do DCC através do SSH, funcionou sem problemas.

Foi criado um arquivo **tp1.py**, responsável por receber os parâmetros dos códigos **Bash** e chamar as funções corretas para cada **.sh**; foi criado também um arquivo **pacMaze.py** que é uma classe que implementa todas as funções exigidas nesse trabalho.

2.2 Ordem da expansão:

Como não foi especificado qual seria a ordem da expansão dos nós; da função **expande**, minha função expande na seguinte ordem: **acima, abaixo, direita e esquerda.** Isso pode resultar em uma resposta ligeiramente diferente.

2.3 Melhoria na implementação:

Inicialmente, implementei o tipo abstrato de dados **explorados** como uma fila que armazenava as coordenadas dos estados já visitados . Assim, toda vez que expandia um nó, era realizado uma busca nessa fila, para ver se os nós expandidos já haviam sido explorados. A complexidade da busca ia crescendo conforme mais nós iam sendo explorados.

Depois de já ter feito a parte 3 do trabalho, percebi que poderia refazer essa busca, por nós já explorados, com complexidade O(1). Criei assim uma matriz do tamanho do mundo PacMaze, onde cada posição representava um estado e teria um valor booleando dizendo se o nó já foi visitado ou não. Mesmo para o exemplo da especificação houve uma notável melhora no desempenho. A mesma solução foi usada no tipo abstrato de dados **caminho**. O **caminho** é usado para recuperar o caminho partindo da solução até estado inicial. Cada posição dessa matriz armazena o pai daquele nó.

2.4 Visualização:

Para poder ver o algoritmo funcionando, criei algumas funções e alguns scripts .sh que vai mostrando passo a passo da solução sendo executada. Para cada algoritmo criei um .sh

que executa e mostra a solução passo a passo. São eles: **visualizacaoAStar.sh, visualizacaoBSF.sh, visualizacaoDSF.sh**. Para executar basta digitar no terminal por exemplo: **./visualizacaoDSF.sh arquivoTeste.txt pontoLinha pontoColuna**.

2.5 Dificuldades:

Inicialmente tive dificuldade de mexer com script **Bash**, nunca tive contato com o mesmo. Mas após ver alguns tutoriais, vi que é bem simples.

3 Resultados:

3.1 Informações importantes:

Para auxiliar em minha análise, criei um programa que gera mais testes. Cria um mundo com tamanho aleatório, preenche ele com paredes e espaços em brancos também aleatórios. Os algoritmos foram executados nos 6 testes que nos foi enviado e em mais 10, criados por mim. Esses 10 testes serão enviados juntos com o trabalho. Para todos os testes, foi adotado como ponto inicial o **(1, 1)**, assim todos os resultados se baseam nesse ponto e não no **(5, 2)** adotado como exemplo.

3.2 Heurísticas:

Ao processar o mundo do PacMaze, salvei a posição onde se enontrava a pastilha, com esse ponto eu criei duas heurísticas para na busca. No código elas têm os seguintes nomes: **heuristica1** e **heuristica2**.

3.2.1 heuristica1:

Essa heurística é baseada na distância euclidiana entre dois pontos. Sendo esses pontos o ponto atual da busca e o ponto onde está o alvo. O cálculo é realizado da seguinte forma:

$$P = \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

P nos fornece a menor distância do ponto atual ao alvo, entretanto, como é a menor distância, essa distância pode ser um caminho na diagonal e movimentos diagonais não são permitidos nesse jogo.

3.2.2 heuristica2:

Uma heurística que considera caminhos diagonais em um jogo que não permite isso, pode trazer problemas. A **heuristica2** foi criada para tentar contornar esses problemas e tentar deixar f(n) mais real.

Como o custo de deslocar de um ponto para o visinho é um, essa heurística simplesmente calcula quantos "passos" terei que dar do ponto atual ao objetivo, desconsiderando qualquer obstáculo, é claro. Seja dois pontos A e B, o custo de sair de A e ir para B é:

$$\begin{gathered} x = |x_a - x_b| \\ y = |y_a - y_b| \\ D = x + y \end{gathered}$$

D nos informa a quantidade mínima de movimentos que teríamos que fazer para chegar ao objetivo se não houver obstáculos.

3.3 Tabelas:

Foram criadas 3 tabelas, as duas primeiras colunas de cada tabela têm a mesma informação, o nome do teste executado e o tamanho do mundo contido no teste, as outras 4 colunas contêm o nome de cada algoritmo onde será armazenado o valor informativo, como quantidade de nós expandidos, custo e tempo. Onde tem 0, quer dizer que o algoritmo não obteve solução para o estado inicial fornecido. A* H1 é o A* usando a **heuristica1** e A* H2 é o A* que utiliza a **heuristica2**.

A **Tabela 1** mostra, para cada teste, a quantidade de nós expandidos pelos algoritmos.

Tabela 1 : Quantidade de Nós Expandidos					
Arquivo de Teste	Tamanho do Mundo	BSF	DSF	A* H1	A* H2
Pacmaze-01	7 X 14	5	5	5	5
Pacmaze-02	7 X 20	59	55	59	59
Pacmaze-03	14 X 20	149	122	149	149
Pacmaze-04	21 X 17	114	40	57	56
Pacmaze-05	37 X 37	291	814	151	142
Pacmaze-06	37 X 37	224	628	120	104
t1	10 X 19	63	43	28	21
t2	28 X 6	0	0	0	0
t3	24 X 8	2	104	2	2
t4	15 X 26	1	1	1	1
t5	28 X 21	266	265	266	266
t6	20 X 25	0	0	0	0
t7	13 X 17	24	111	14	14
t8	20 X 11	3	2	2	2
t9	17 X 14	9	100	6	6
t10	16 X 21	170	164	102	102

A **Tabela 2** mostra, para cada teste, o custo da solução de cada algoritmo.

Tabela 2: Custos					
Arquivo de Teste	Tamanho do Mundo	BSF	DSF	A* H1	A* H2
Pacmaze-01	7 X 14	5	5	5	5
Pacmaze-02	7 X 20	42	42	42	42
Pacmaze-03	14 X 20	62	88	62	62
Pacmaze-04	21 X 17	21	37	21	21
Pacmaze-05	37 X 37	82	116	82	82
Pacmaze-06	37 X 37	50	50	50	50
t1	10 X 19	14	34	14	14
t2	28 X 6	0	0	0	0
t3	24 X 8	2	2	2	2
t4	15 X 26	1	1	1	1
t5	28 X 21	136	148	136	136
t6	20 X 25	0	0	0	0
t7	13 X 17	8	12	8	8
t8	20 X 11	2	2	2	2
t9	17 X 14	5	31	5	5
t10	16 X 21	19	42	19	19

A **Tabela 3** mostra, para cada teste, o tempo que cada algoritmo levou para encontrar uma resposta. O tempo é dado em segundos e foi calculado usando a função do Linux **time**.

Tabela 3: Tempo em Segundos					
Arquivo de Teste	Tamanho do Mundo	BSF	DSF	A* H1	A* H2
Pacmaze-01	7 X 14	0,027	0,028	0,038	0,044
Pacmaze-02	7 X 20	0,026	0,032	0,022	0,022
Pacmaze-03	14 X 20	0,036	0,025	0,026	0,030
Pacmaze-04	21 X 17	0,027	0,035	0,022	0,038
Pacmaze-05	37 X 37	0,038	0,051	0,165	0,025
Pacmaze-06	37 X 37	0,032	0,034	0,064	0,040
t1	10 X 19	0,033	0,033	0,033	0,029
t2	28 X 6	0	0	0	0
t3	24 X 8	0,034	0,032	0,061	0,032
t4	15 X 26	0,033	0,033	0,062	0,033
t5	28 X 21	0,038	0,039	0,041	0,033
t6	20 X 25	0	0	0	0
t7	13 X 17	0,037	0,032	0,060	0,033
t8	20 X 11	0,032	0,031	0,057	0,034
t9	17 X 14	0,026	0,028	0,054	0,027
t10	16 X 21	0,035	0,038	0,064	0,022

3.4 Análise dos Resultados:

Da **Tabela 1** notamos que o BSF e DSF ficaram bem parecidos, o BSF expande todos do mesmo nível, assim, tem uma tendêndia a explorar mais nós, se o objetivo estiver bem fundo na árvore. Para os casos que o DSF expandiu mais que o BSF, a diferença foi muito grande, por exemplo no Pacmaze-05. Já o A* H1 e o A* H2 de 16 testes eles empataram em 12, sendo que a maior diferença entre eles foi 16, o A* H1 explorou 16 nós a mais. O A* H2 foi o mais eficiente em exploração de nós. Essa diferença provavelmente se deve ao fato discutido na **Seção 3.2**. Devido a eficiência do H2, ele foi aderido com o padrão desse trabalho.

Da tabela de custos, **Tabela 2**, comprovamos algo interessante, a Busca em Largura, BSF, tem solução ótima se, que além do custo não diminuir conforme aprofunda na árvore, o custo não aumenta em nós de mesmo nível. Em todos os resultados o custo do BSF foi idêntico aos dos A*s. Em casos bastantes especiais, quando a ordem de expansão "batia" com os movimentos para se chegar no objetivo, o DFS teve resultado ótimo.

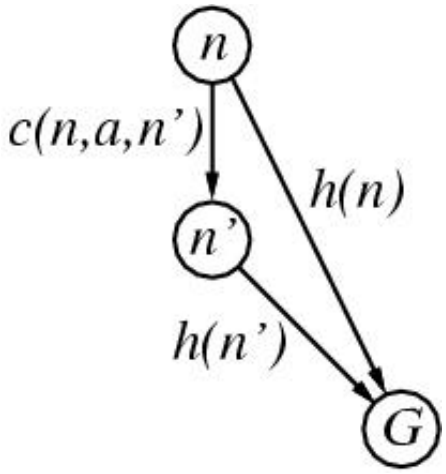
Olhando a **Tabela 3**, notamos que os teste não foram grandes o suficiente para perceber alguma diferença entre os métodos, quanto ao tempo de execução

4 Respostas às perguntas:

4.1 Sua heurística para o A* é admissível? Ela é consistente?

Ambas as heurísticas são admissíveis, os custos de ambas são menores ou iguais ao custo real. A **heuristica1** calcula a distância euclidiana entre os nós, essa distância será no mínimo igual ao custo real, já a **heuristica2**, calcula o que seria a distância real se não houvesse obstáculos, então a **heuristica2** terá no mínimo o valor da distância real.

Definição 1. Uma heurística é consistente se, para um nó n e os seus sucessores n' gerados por uma ação a, o custo estimado de atingir o gol a partir de n não é maior que o custo de chegar a n' somado ao custo estimado de n' para o gol.



$$\mathbf{h(n) <= c(n,a,n') + h(n')}$$

Olhando a definição conclui-se que além de ser admissíveis, essas heurísticas são consistentes.

4.2 A busca em largura e/ou a busca em profundidade apresentam sempre soluções ótimas para o Pac-Maze?

Como já foi dito na **Seção 3.4**, o BSF obteve solução ótima para todos os casos, já que não há acréscimo de custo em nós de mesmo nível. Quanto ao DSF, foram poucos os casos em que ele obteve solução ótima, apenas quando o objetivo estava perto do ponto inicial e a ordem da expansão ajudou ele a encontrar uma solução rápida.

4.3 Soluções encontradas A* H2:

pacmaze-01-tiny: **direita direita direita direita direita.**

pacmaze-02-small: **abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita acima acima direita direita abaixo abaixo abaixo abaixo esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda.**

pacmaze-03-mid-sparse: **abaixo abaixo abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita acima acima acima direita direita abaixo abaixo abaixo abaixo abaixo esquerda esquerda esquerda esquerda esquerda abaixo abaixo esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda abaixo abaixo abaixo direita direita acima direita direita direita direita abaixo direita direita acima.**

pacmaze-04-pacman: **direita direita direita abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo direita direita abaixo abaixo abaixo abaixo direita abaixo abaixo direita.**

pacmaze-05-big-sparse: **abaixo abaixo abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita abaixo abaixo esquerda**

esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda abaixo direita direita direita direita direita direita direita direita direita direita direita direita abaixo abaixo esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda. pacmaze-06-big: **abaixo abaixo direita direita direita direita abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo direita direita abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo direita direita abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo.**

4.4 Soluções encontradas pelo BSF:

pacmaze-01-tiny: **direita direita direita direita direita**

pacmaze-02-small: **abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita acima acima direita direita abaixo abaixo abaixo abaixo esquerda.**

pacmaze-03-mid-sparse: **abaixo abaixo abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita direita acima acima acima direita direita abaixo abaixo abaixo abaixo abaixo abaixo abaixo esquerda abaixo abaixo abaixo direita direita acima direita direita direita abaixo direita direita direita acima.**

pacmaze-04-pacman: **abaixo abaixo abaixo abaixo direita direita direita abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo direita direita direita direita.**

pacmaze-05-big-sparse: **abaixo abaixo abaixo abaixo direita direita direita direita direita direita direita direita direita direita direita direita direita direita abaixo abaixo esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda abaixo direita direita direita direita direita direita direita direita direita direita direita direita abaixo abaixo esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda esquerda.**

pacmaze-06-big: **abaixo abaixo direita direita direita direita abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo abaixo direita direita abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo direita direita abaixo abaixo esquerda esquerda abaixo abaixo abaixo abaixo esquerda esquerda abaixo abaixo.**

5 Conclusão: