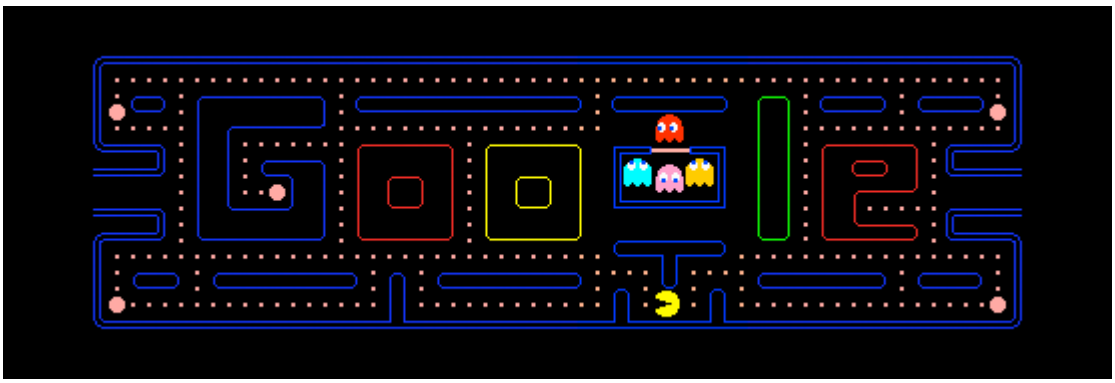
Sistemas Inteligentes

ANO LECTIVO DE 2018/2019 – 2º SEMESTRE

À Procura do PacInv

(Modelação do Problema)



Grupo Nº 33

Colin Flaherty Nº 53931

António Ajuda Nº 53589

Conteúdo

1. RECOLHENDO DADOS SOBRE A PROCURA

1.1 OS MÉTODOS AO MODO STATS

A estrutura dos métodos ao modo stats é bastante similar à estrutura dos métodos das procuras normais. Tanto depth_first_graph_search e breadth_first_graph_search usam o método graph_search com argumentos de fronteira difererentes, assim depth_first_graph_search_stats e breadth_first_graph_search_stats usam o método graph_search_stats. A solução e o custo são atributos da classe Node, assim como também colocamos a estatística do numero de nós visitados, do número de nós expandidos e do maior tamanho da fronteira como atributos da classe Node, chamado Node.num_explored, Node.num_expanded, e Node.max_frontier respectivamente.

1.2 OS MÉTODOS DE STATS_TEST

Para recolher o tempo que demorou uma procura, usamos os métodos *stats_test_pac* e *stats_test_grapho*. Embora ter métodos seperados para problemas de PacInvisiv e problemas de grafos não seja a melhor solução, dado o tempo limitado foi para nos a melhor maneira de conseguir recolher as estatísticas. As *stats_test_pac* e *stats_test_grapho* usam o modulo *time* para medir o tempo que demorou uma procura, e tem a estrutura seguinte, onde *heurística* é a heurística para ser usada nas procuras A* e sôfrega, e *depth* é o limite para uma procura em profundidade iterativa.

```
def stats_test_pac(algo, prob, heuristica, depth):
"""dado um algoritmo e problema, volta os estatisticos pedidos no enuciado"""
```

1.3 As heurísticas

Criámos duas heurísticas, h1 e h2, colocadas no search_pacman. h1 retorna o número de pastilhas acima do pacman. Como o custo de mover-se para cima é maior do que o custo de mover-se abaixo, é preferível um estado com menos pastilhas por cima do pacman. h2 retorna a distancia manhattan da pastilha mas perto do pacman.

h1 não é consistente nem admissível. Seja 'e' um estado com 1000 pastilhas acima do pacman e 8 abaixo, numa configuração onde é possível anular todas as casas adjencentes ao pacman com estas pastilhas.

h1(e) = 1000, mas c(e) < 80. Portanto h1(e) > c(e). Porque h1 não é admissível, não é consistente.

h2 não é consistente nem admissível. Seja 'e' um estado com oito pastilhas, uma em cada casa adjacente ao pacman. Então $h2(e) = 1$, mas como as pastilhas já tocaram cada casa adjacente, $c(e) = 0$. Portanto $h2(e) > c(e)$. Porque *h2* não é admissível e não é consistente.

Estas heurísticas não são consistentes nem admissíveis, então não garantem encontrar a solução ótima. Apesar disso, consiguem muitas vezes encontrar a solução ótima, como são exemplos os casos 3a e 3b. Também demoram muito tempo na procura. Serão discutidas novamente na secção 8.

2. COMPARAÇÃO DO DESEMPENHO DOS ALGORITMOS DE PROCURA

2.1 A GERAÇÃO DOS CASOS BENCHMARK

Os casos benchmark para grafos abstratos foram dados no moodle e só foi necessário criar classes análogas à classe *ProblemaGrafoAbs*. Os casos benchmark para PavInvisiv também foram fáceis gerar, usando a classe *EstadoPacman*. Geramos um estado aleatório para a comparação usando o método *GerarEstado*. É possível desenvolver esta secção com mais estados aleatórios, mas o run-time para fazer todas as procuras para todas os problemas é muito longo. Então optamos para analizar os resultados de só um estado aleatório.

2.2 ESCREVER OS RESULTADOS NUM FICHEIRO .CSV

Usamos os métodos *collect_write_stats_Pacinvisiv* e *collect_write_stats_graph* para escrever os resultados nos ficheiros *.csv*. Estes métodos aceitam listas dos algoritmos e heuristicas, um problema, um limite, e o nome dum ficheiro para ser gerado que vai conter as estatísticas. Porque no caso duma procura em profundidade é necessário tentar a procura muitas vezes, é melhor ter um método assim, embora possa parecer indireto fazer a procura e processar os resultados antes de escrever, do que fazer a procura e escrever no mesmo método. Também esta estrutura permite que para cada problema temos um ficheiro que contém as estatísticas de cada algoritmo, que torna mais fácil e conveniente a análise dos algoritmos.

2.3 O MÉTODO GERAR_FICHEIROS

O método Gerar_Ficheiros não aceita um argumento nem retorna nenhum valor. Apenas serve para simplificar a geração de resultados para os problemas benchmark.

3. PARTE IV: MODO PEDAGOGICO

3.1 OS MÉTODOS PEDAGÓGICOS

É pedido para ilustrar o funcionamento passo a passo da procura A*, procura de Custo Uniforme, e procura Sôfregra. Para isso criamos as funções astar_search_pedagogica, uniform_cost_search_pedagogica, e greedy_best_first_graph_search_pedagogica. Todas fazem uma chamada a best_first_graph_search_pedagogica, só com argumentos da heurística diferentes. O best_first_graph_search_pedagogica faz um best first graph search cada vez que usamos um nó da fronteira na consola gráfica do estado que representa este nó e nota que este nó está a ser expandido. Também dá os gráficos dos estados dos nós sucessores, e para cada sucessor refere que foi posto na fronteira, ou que já esteve na fronteira, ou que o estado que representa o sucessor já foi explorado.

4. O FICHEIRO

Este ficheiro não contem README. Esta secção funciona como o README.

O programa *pacman_1.py* não gera nenhuns resultados. Os resultados são pre-gerados e incluídos no ficheiro .zip, junto com os módulos modificados necessários para que corra o programa. Estes módulos são *Grapho.py* e *search_pacman.py*. Todos os métodos tem comentários para explicar o funcionamento e como chama-os.

5. ANÁLISE DOS RESULTADOS

O algoritmo DFS é o que apresenta custo mais elevado, e os algoritmos Custo Uniforme e A* os que apresentam menor custo.

Os algoritmos Custo Uniforme e A* apresenta maior número de estados visitados e estados expandidos, assim como tamanho de fronteira e duração de tempo. Nos vários casos experimentados esta relação manteve-se, embora com alguma oscilação.

Para cada caso a procura de custo uniforme dá melhores resultados. A procura A* consegue encontrar a mesma solução, mas demora um pouco mais e explora mais estados. Por exemplo no caso 2, usando a heurística h1 (dados disponíveis no ficheiro resultados_pac_caso2_h1.csv), a procura A* e a procura de custo uniforme econtram a mesma solução de custo 15 (o que é a solução ótima, porque a procura de custo uniforme é um algoritmo ótimo), mas A* demorou 2407 segundos (40 minutos!) e explorou 100829 estados. A procura de custo uniforme demorou 1742 segundos (30 minutos) e explorou 90733 estados. Nota-se que a diferença de tempo não é proporcional à diferença de estados explorados, ou seja, o tempo para computar h1 tornou-se importante durante a procura. Vemos uma diferença semelhante com a heurística h2.

Embora esta estatística indique uma formulação pior das heurísticas, quando olhamos a procura sôfrega que utiliza só as nossas heurísticas, vemos que é muito mais rápido nas procuras. Para caso 3a usando a heurística h2(dados disponíveis no ficheiro resultados_pac_caso2_h2.csv) a procura sôfrega identifica uma solução com custo 39 (onde a solução ótima identificada pela procura de custo uniforme tem custo 37) em só 20 segundos, quando a procura A* e de custo uniforme ambas demoram 2000 segundos, ou seja tem um runtime 100 vezes menor! A heurística h1 neste problema tem um runtime 10 vezes menor que A* e a procura de custo uniforme.

6. ERROS

Caso 1 tem uma solução, mas nenhum algoritmo encontra-a. Parece que o estado está formulado corretamente, porque se jogamos por mão a sequencia [E,E,E,NE,NE] da um estado goal. Ou seja, o código seguinte da "True"

```
e1 = prob_caso1.result(prob_caso1.initial, "E")

e2 = prob_caso1.result(e1, "E")

e3 = prob_caso1.result(e2, "E")

e4 = prob_caso1.result(e3, "NE")

e5 = prob_caso1.result(e4, "NE")

prob_caso1.goal_test(e5)
```

Por isso, os ficheiros *resultados_caso1_h1.csv* e *resultados_caso1_h2* não contem uma solução nem um custo.

Quando o algoritmo não encontra uma solução, o método ao modo *stats* volta o número de nós expandidos como 0. A medida do numero de nós expandidos funciona nos outros casos. Isso afeta os ficheiros *resultados_caso1_h1.csv, resultados_caso1_h2, resultados_caso4a_h1.csv, resultados_caso4a_h2, resultados_caso4b_h1.csv* e *resultados_caso4b_h2.*