DCC028 - Inteligência Artificial
Trabalho Prático 1: Busca em Mapas

10/04/2018

Luiz Chaimowicz[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Caixa Postal 31270-901 - Belo Horizonte - MG - Brasil

chaimo@dcc.ufmg.br

Data da entrega: 07/05/2018

1. Introdução

Neste trabalho, vamos exercitar o básico de alguns algoritmos de busca em espaço de estados [Russell and Norvig 2010] (capítulos 3 e 4). Um conjunto de mapas está disponível no moodle. Estes mapas são bidimensionais, representados por matrizes de caracteres. A formatação dos mapas é simples: células marcadas com um "." estão livres e células marcadas com um "@" estão bloqueadas.

Na implementação dos agentes assumiremos que eles poderão se mover nas seguintes oito direções: cima, baixo, direita, esquerda, todas com o custo de 1, e as quatro diagonais, com o custo de $1,5$ ($\approx \sqrt{2}$). Movimentos diagonais não serão permitidos em cantos que estiverem bloqueados.

2. Arquivo de entrada

Os mundos nos quais os agentes de busca irão interagir são dados como entrada, você deverá ler os mapas que possuem a seguinte estrutura: 1^a linha contém o tipo do mapa[1], a 2^a e 3^a linhas correspondem à altura e largura respectivamente, a 4^a linha contém o nome do mapa e as demais linhas e colunas do arquivo correspondem a representação do mundo em si[2] (caracteres "." e "@").

3. Objetivos

O principal objetivo desse trabalho é entender e implementar os componentes básicos de um algoritmo de busca, fazer uma análise das decisões de projeto tomadas na implementação dos agentes e estudar como elas influenciam os resultados obtidos.

Utilizando os mapas fornecidos no moodle implemente os seguintes algoritmos de busca:

1. Busca de Aprofundamento Iterativa (IDS);
2. Busca de custo uniforme (UCS);
3. Busca gulosa de melhor escolha (*Best First Search*);

[1] Esta característica pode ser desconsiderada pelo algoritmo de vocês.
[2] Abra algum dos mapas com qualquer editor de texto simples e verifique os detalhes do arquivo.

4. Busca A*.

O pseudo-código dos algoritmos de busca listados acima estão presentes nos slides vistos em aula e no livro-texto. Lembre-se que um nó na busca deve conter não só o estado, mas também toda a informação necessária para reconstruir o caminho que leva àquele estado.

Vocês deverão implementar as versões completas (de busca em grafos em espaços finitos) dos métodos listados acima. Em outras palavras o agente deve evitar estados já visitados.

Todas as suas funções de busca devem retornar uma lista de ações que levarão o agente do início ao objetivo. Todas as ações devem ser movimentos legais em direções válidas e sem cruzar espaços proibidos (caracteres "@").

Em todos os métodos acima, além do mapa (que pode ser uma variável global), o seu agente deve receber como parâmetro um par de estados, inicial e meta, e retornar um caminho entre esses dois estados e o custo total encontrado. A escolha das estruturas de dados utilizadas nas implementações da lista **ABERTO** (a borda) e **FECHADO**[3] devem ser feitas com bastante cuidado na criação de cada método, dependendo da estrutura de dados selecionada seu algoritmo ficará demasiadamente custoso. Portanto **justifique** suas escolhas para cada método.

Os algoritmos de busca são muito parecidos. Os algoritmos para IDS, UCS, Busca gulosa e A* diferem apenas em detalhes de como a borda é gerenciada. Portanto, se concentre em implementar o IDS corretamente e o resto será relativamente simples. Repare que é possível implementar um método de busca genérico configurável para diferentes estratégias de gerenciamento de nós. (Sua implementação não precisa ser assim.)

Para a implementação do algoritmo A*, duas funções heurísticas devem ser avaliadas:

1. Função de Distância Manhattan;
2. Função de Distância Octile.

A função de *distância Manhattan* é calculada da seguinte forma:

$$\begin{aligned} dx &= abs(node.x - goal.x) \\ dy &= abs(node.y - goal.y) \\ h(n) &= (dx + dy) \end{aligned} \tag{1}$$

A função de *distância octile*, é calculada da seguinte forma: dada a distância x e a distância y para o objetivo, o valor heurístico é calculado como

$$\begin{aligned} dx &= abs(node.x - goal.x) \\ dy &= abs(node.y - goal.y) \\ h(n) &= max(dx, dy) + 0.5 * min(dx, dy) \end{aligned} \tag{2}$$

[3] DICA: dado que o mundo é uma matriz bidimensional existe uma chave *hash* perfeita.

Para o algoritmo busca gulosa apenas a função de distância Manhattan deve ser analisada.

Uma explicação mais detalhada dessas e de outras heurísticas aplicáveis em grids 2D (que é o caso dos mapas disponibilizados) pode ser encontrado em: `http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html`

Dentre as duas funções heurísticas apresentadas acima qual delas é uma função admissível e consistente? Em seu relatório cite qual deles é a função consistente e admissível e apresente justificativas[4]. Nos experimentos produzidos qual delas é mais rápida? Qual delas expande um menor número de nós? Qual delas apresenta um menor custo? Uma boa referência do impacto que a escolha da heurística aplicada no algoritmo A* causa pode ser encontrado em: `http://movingai.com/astar.html`

4. Observações

Este trabalho deve ser desenvolvido individualmente. Embora a colaboração entre alunos seja permitida, não haverá intercâmbio de código ou resultados empíricos. A colaboração ocorrerá apenas no nível das ideias. Para facilitar tal colaboração será disponibilizado um fórum de discussão no moodle. Ao postar suas dúvidas ou discutir com colegas, tente não dar respostas diretas. O aprendizado é melhor quando entendemos o problema a ponto de elaborar respostas.

Você deverá entregar um arquivo de saída contendo: 1^a linha é o estado inicial, 2^a o estado meta, 3^a linha em branco e a 4^a linha o caminho começando pelo estado inicial e terminando no estado meta sendo cada estado representando por uma 3-upla <valor_x, valor_y, custo>[5] começando pelo estado inicial indo até o estado meta, intercalando-os por um espaço em branco. A Figura 1 mostra o exemplo de como deve ser a resposta dos algoritmos.

```
<190, 249, 0>
<175, 249, 15>

<190, 249, 0> <189, 249, 1> <188, 249, 2> <187, 249, 3> <186, 249, 4> <185, 249, 5>
<184, 249, 6> <183, 249, 7> <182, 249, 8> <181, 249, 9> <180, 249, 10> <179, 249, 11>
<178, 249, 12> <177, 249, 13> <176, 249, 14> <175, 249, 15>
```

Figura 1. Exemplo de saída do algoritmo UCS dado o estado inicial (190, 249, 0) e estado meta (175, 249, 15).

O aluno poderá utilizar qualquer linguagem de programação de sua preferência. No entanto, certifique-se que seu código funciona em uma máquina GNU/Linux. Não imprima **NADA ALÉM** do que foi pedido, pois isso resultará em erro durante a correção. Para cada método, além do programa, você deverá escrever um script em shell (*.sh) simples que execute o seu programa com a entrada desejada. *ids.sh, ucs.sh bg.sh e aestrela.sh*. Todos devem receber como parâmetro o nome do mapa, o estado inicial e estado final. O script *aestrela.sh* deve receber um 4º parâmetro correspondente a qual heurística será utilizada (1 = *distancia Manhattan* e 2 = *distancia octile*). Caso use uma linguagem de programação compilada, escreva um script *compila.sh* que realize a compilação de todo o seu código.

[4] Esta justificativa pode ser feita apresentando contra-exemplos.

[5] Assuma em seu código que x representa as linhas da matriz e y as colunas.

Execute experimentos nos mapas disponibilizados no moodle. Para tal, gere aleatoriamente um conjunto de pares estado inicial e estado meta nos 4 algoritmos implementados. Nos experimentos deve-se observar, analisar e discutir o custo dos caminhos encontrados, assim como o tempo de execução e o número de nós expandidos. Você pode também gerar experimente como o algoritmo WA* (não é necessário fazer w $\neq 1$[6]). O algoritmo WA* é exatamente o mesmo algoritmo A* onde w é normalmente um número natural ($\mathbb{N}^+$) que potencializa o valor da heurística[7], alterando o valor de $f(n)$ (custo estimado da solução de menor custo através de n) fazendo com que o agente tome decisões diferentes. Apresente os resultados de forma sucinta através de tabelas e/ou gráficos devidamente explicados.

5. O que deve ser entregue

. Código fonte do programa e os scripts em shell;
. Documentação do trabalho:.

- Introdução;
- Implementação: descrição sobre a implementação dos agentes, incluindo detalhes da implementação das listas aberto (a borda) e fechado;
- Experimentos: Análise do impacto de um mesmo estado inicial e meta dado os 4 métodos propostos.

A entrega DEVE ser feita pelo Moodle na forma de um único arquivo zipado, contendo o código e a documentação do trabalho. Sendo o formato: tp1-parâmetro1_parametro2_parametro3.zip, onde parâmetros 1º e 2º corresponde ao primeiro e último nome respectivamente e o 3º é o número de matrícula.

Seu código será avaliado automaticamente para verificação de corretude, além de serem submetidos par-a-par em um verificador de plágio. Note, no entanto, que sua pontuação será baseada na avaliação do seu código e relatório, e não no julgamento do avaliador automático. Se necessário, revisaremos sua submissão individualmente para garantir que você receba o devido crédito.

Serão avaliados a implementação dos algoritmos, os experimentos realizados (faça tabelas e gráficos comparativos) e a documentação entregue discutindo os resultados e as heurísticas.

Referências

[Russell and Norvig 2010] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall.

[6] Caso você teste valores diferentes para w, os resultados de wA* ainda são resultados ótimos? Em outras palavras o heurística deixou de ser consistente?

[7] Basicamente a diferença entre A* e wA* é que em wA* a heurística é multiplicada pelo fator w.