Relatório de projeto - sistemas inteligentes 2017/2018 Sokoban

Ruben Branco 50013 | Raul Silvestre 50023 | Diogo Ribeiro 50042

[Date]

Introdução

Sokoban consiste num puzzle em que um mapa é definido por uma grelha, , no qual existe N caixas, N locais alvo e paredes. O objetivo é encontrar uma serie de movimentos que permitem um agente colocar todas as caixas na posição alvo.

Existem algumas regras e restrições, como o agente só pode empurrar e não puxar as caixas, só pode empurrar uma caixa e não duas adjacentes.

Devido à complexidade que o problema tem, com fatores de ramificação grandes torna o espaço de busca do problema rapidamente grande até para os puzzles simples. É necessário definir uma estratégia, de forma de heurísticas, que consiga restringir este espaço e eliminar os estados que resultam em *deadlocks*.

...

Formulação

## Representação de estados

Os estados foram representados a partir de uma classe *SokobanState*, que faz uso de um grafo para ter o estado visualizado.

Esse grafo é um dicionário em que as chaves são as coordenadas cartesianas de uma certa posição no mapa e o valor é um dos 7 possíveis caracteres que definem os diferentes tipos de entidades: ‘#’, ‘.’, ‘\*’, ‘o’, ‘A’, ‘@’ e ‘B’.

As coordenadas cartesianas são mapeadas a partir de um espaço em que o eixo

dos yy está invertido em relação a um espaço cartesiano normal, ou seja, os yy tem o seu quadrante positivo a baixo do eixo dos xx.

O estado tem também alguns métodos de suporte associados à classe para ajudar na navegação do grafo.

Exemplo:

Para um puzzle

###

#o#

#\*#

#A#

###

Resultaria um grafo da seguinte forma na estrutura de dados nativa de Python, dicionário

{(0,0): ‘#’, (1,0): ‘#’, (2,0): ‘#’, (0,1): ’#’, (1,1): ‘o’, (2,1): ’#’, (0,2): ‘#’, (1,2): ‘\*’, (2,2): ‘#’, (0,3): ‘#’, (1,3): ‘A’, (2,3): ‘#’, (0,4): ‘#’, (1,4): ‘#’, (2,4): ‘#’}

## Operadores

Existem 2 tipos de operadores e 4 sentidos dos mesmos definidos. É possível mover (*Move*) e empurrar (*Push*). Os quatro sentidos para onde se pode fazer estas operações são cima (*Up*), baixo (*Down*), esquerda (*Left*) e direita (*Right*).

A partir da classe de suporte *GraphUtil* e de métodos do *SokobanState*, é restringido as ações ao contexto, ou seja, através da posição do arrumador, é determinado as ações possíveis.

## Visualização de resultados

Para visualizar uma resolução, foi implementada no ficheiro ***input\_output.py*** uma função path\_to\_sequence, que quando passada um resultado.path() de um objeto resultado, faz *print* onde descreve os movimentos do arrumador e uma visualização gráfica do estado do mapa em cada passo.

Heurísticas

Foram definidas 4 heurísticas que visam a formar uma estratégia onde se procura

minimizar a distância de jogador a caixas e das caixas aos objetivos e também evitar estados mortos (explicado na secção de analise).

As heurísticas surgiram incrementalmente, ou seja, cada heurística estende a sua antecessora ao utilizar o seu custo e adicionar sobre ele.

## Heurística 1

A heurística 1 procura minimizar a distância das caixas aos objetivos. Isto é conseguido a partir de distância euclidiana. A heurística é dada por:

Em que c é o número de caixas que **não** se encontram num objetivo, o é o número de objetivos. Isto efetivamente calcula a distância média de todos os pares (caixa, objetivo), caixas que não estão em objetivos.

O valor da heurística é mais baixo conforme a proximidade das caixas aos objetivos, premiando os estados em que estão mais perto ou até no objetivo já.

## Heurística 2

A heurística 2 utiliza a heurística 1 e acrescenta um novo cálculo semelhante, em que pretende aproximar os jogadores às caixas, querendo premiar estados em que o arrumador está mais perto, em média, das caixas.

O valor da heurística é dado por:

Em que c é o número total de caixas que **não** se encontram num objetivo e A é a posição do arrumador.

O valor da heurística é mais baixo conforme a proximidade das caixas aos objetivos e do arrumador às caixas, pois a progressão das caixas aos objetivos só é possível se o arrumador estiver perto delas para as empurrar, necessitando de premiar essa proximidade.

## Heurística 3

A heurística 3 expande, outra vez, a heurística 2 e faz uso dela, mas pretende resolver outro tipo de problemas no jogo do *Sokoban*, que é os estados mortos (*deadlock*). Uma jogada pode tornar uma tentativa de resolução impossível se, por exemplo, encostar um caixa num canto, porque o arrumador não pode puxar caixas, apenas empurra-las, portanto a caixa nunca poderá mover-se do canto, e se o algoritmo não retroceder, poderá nunca convergir para uma solução.

Portanto, esta heurística percorre as caixas que não estão situam-se num objetivo e verifica se estão num canto. Se estiverem, é dado um valor de 1000 como penalidade.

## Heurística 4

A heurística 4 utiliza a heurística 3 e procura outro tipo de situação *deadlock*, que é as paredes. Uma caixa encostada a uma parede pode constituir um estado morto, pois se está encostada a uma parede significa que sem uma abertura onde se pode retirar a caixa da parede, ela permanecerá lá, o que pode tornar o puzzle impossível se não se encontra nenhum objetivo nessa coluna ou perto dela.

Para combater estas situações, se uma caixa estiver encostada a uma parede, é varrida as colunas ou linhas adjacentes com um raio dado (raio utilizado é 3), dependendo se está encostada a uma parede na parte superior ou inferior do mapa ou se está encostada a uma parede numa parte lateral do mapa. Se houver objetivos dentro desse espaço, então o número de objetivos é anotado e o custo da heurística é somado da seguinte maneira:

Em que o alfa é um modificador escolhido, que na implementação tomou um valor de 4.

## Heurística 5, 6, 7, 8

As heurísticas 5, 6, 7 e 8 são variantes iguais à heurística 1, 2, 3 e 4 descritas a cima, mas baseadas em distância de Manhattan em vez de distância euclidiana.

exemplos de execução

Para a execução do programa, existe uma interface por linha de comando, onde os argumentos estão especificados no menu *help* que pode ser acedido ao executar python3 main.py -h .

Exemplo de execução 1: Puzzle 1, Algoritmo A\* H8 com nível verbose a 2 (Estatísticas + Solução gráfica)

python3 main.py data/puzzle1.txt 11 -v 2

\*\*\*\*\*\*\* ESTATISTICAS \*\*\*\*\*\*\*

Algorithm: A Star Search with Heuristic 8

Tempo de resolucao: 0.004849910736083984s

Numero de passos da resolucao: 5

Numero de nos visitados: 9

Resolucao: ['Move up', 'Push left', 'Move down', 'Move left', 'Push up']

Problema inicial

#####

#o..#

#.\*.#

#..A#

#####

Passo 1 - Move up

#####

#o..#

#.\*A#

#...#

#####

Passo 2 - Push left

#####

#o..#

#\*A.#

#...#

#####

Passo 3 - Move down

#####

#o..#

#\*..#

#.A.#

#####

Passo 4 - Move left

#####

#o..#

#\*..#

#A..#

#####

Passo 5 - Push up

#####

#@..#

#A..#

#...#

#####

Exemplo de execução 2: Puzzle 2, Algoritmo Breadth First Search com verbose nível 1 (Estatísticas)

python3 main.py data/puzzle2.txt 2 -v 1

\*\*\*\*\*\*\* ESTATISTICAS \*\*\*\*\*\*\*

Algorithm: Breadth First Search

Tempo de resolucao: 2.238205909729004s

Numero de passos da resolucao: 43

Numero de nos visitados: 2103

Resolucao: ['Move left', 'Move left', 'Move down', 'Move down', 'Move right', 'Push up', 'Move left', 'Move up', 'Push right', 'Push right', 'Move up', 'Move right', 'Move right', 'Move down', 'Move down', 'Move left', 'Push up', 'Move left', 'Move up', 'Move up', 'Move left', 'Push down', 'Push down', 'Move left', 'Move down', 'Move down', 'Move right', 'Push up', 'Move left', 'Move up', 'Push right', 'Push right', 'Move up', 'Move up', 'Move right', 'Move right', 'Move down', 'Push left', 'Push down', 'Move left', 'Move left', 'Move up', 'Push right']

ANÁlise

## Espaço de Procura

É possível estimar o tamanho (D) do espaço de procura dos puzzles encontrados nos testes (Junghanns 2000), da seguinte forma:

Em que N e M são os tamanhos de cada lado do mapa, regularmente N é igual a M e o mapa é um cubo.

Na nossa análise foram utilizados 10 puzzles, cujo mais pequeno tem uma dimensão de 5 por 5 e o maior tendo um tamanho máximo de 9 por 9. Portanto, o espaço de procura encontrado está entre

Em que o *upper bound* é o espaço de procura do puzzle 3 fornecido no projeto.

## Resultados de Execução

Estados visitados

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Puzzle 1 | Puzzle 2 | Puzzle 3 | Puzzle 4 | Puzzle 5 | Puzzle 6 | Puzzle 7 | Puzzle 8 | Média |
| Depth First Graph | 52 | 2098 | N/A | N/A | 35 | 634 | N/A | 867 | 737 |
| Breadth First | 20 | 2103 | N/A | N/A | 18 | 1295 | N/A | 907 | 867 |
| Uniform Cost | 43 | 2191 | N/A | N/A | 44 | 1875 | N/A | 964 | 1023 |
| A Star H1 | 25 | 2080 | N/A | N/A | 25 | 1478 | N/A | 928 | 907 |
| A Star H2 | 21 | 1951 | N/A | N/A | 17 | 1156 | N/A | 899 | 808 |
| A Star H3 | 21 | 910 | N/A | N/A | 17 | 624 | N/A | 278 | 370 |
| A Star H4 | 9 | 943 | N/A | N/A | 9 | 494 | N/A | 271 | 345 |
| A Star H5 | **6** | 2068 | N/A | N/A | 25 | 1425 | N/A | 928 | 890 |
| A Star H6 | 25 | 1920 | N/A | N/A | 16 | 1050 | N/A | 890 | 780 |
| A Star H7 | 20 | 888 | N/A | N/A | 5/25 | 19/1440 | N/A | 29/928 | 353 |
| A Star H8 | 9 | 918 | N/A | N/A | 5/17 | 19/1154 | N/A | 29/899 | 331 |
| Interactive Deepening w/ Depth Limiting | 5/6 | N/A | N/A | N/A | **5/6** | N/A | N/A | N/A | N/W |
| Interactive Deepening w/ A Star H1 | 5/25 | 43/2080 | N/A | N/A | 5/25 | 19/1440 | N/A | 29/928 | 899 |
| Interactive Deepening w/ A Star H2 | 5/21 | 43/1951 | N/A | N/A | 5/17 | 19/1154 | N/A | 29/899 | 808 |
| Interactive Deepening w/ A Star H3 | 5/21 | 43/910 | N/A | N/A | 5/17 | 19/622 | N/A | 29/278 | 369 |
| Interactive Deepening w/ A Star H4 | 5/9 | 43/927 | N/A | N/A | 5/9 | 19/475 | N/A | 29/271 | 338 |
| Interactive Deepening w/ A Star H5 | 5/25 | 43/2068 | N/A | N/A | 5/25 | 19/1387 | N/A | 29/928 | 886 |
| Interactive Deepening w/ A Star H6 | 5/20 | 43/1920 | N/A | N/A | 5/16 | 19/1049 | N/A | 29/890 | 779 |
| Interactive Deepening w/ A Star H7 | 20 | 43/888 | N/A | N/A | 19 | 19/565 | N/A | 29/276 | 353 |
| Interactive Deepening w/ A Star H8 | 9 | 43/918 | N/A | N/A | 9 | 19/437 | N/A | 29/270 | 328 |
| Greedy Best First Graph H1 | 10 | **206** | N/A | N/A | 8 | 142 | N/A | 122 | 97 |
| Greedy Best First Graph H2 | 10 | 437 | N/A | N/A | 7 | 56 | N/A | 154 | 132 |
| Greedy Best First Graph H3 | 10 | 324 | N/A | N/A | 7 | **50** | N/A | **11** | 100 |
| Greedy Best First Graph H4 | 8 | 718 | N/A | N/A | **6** | 143 | N/A | 192 | 213 |
| Greedy Best First Graph H5 | 10 | 337 | N/A | N/A | 8 | 544 | N/A | 163 | 212 |
| Greedy Best First Graph H6 | 10 | 323 | N/A | N/A | 7 | 76 | N/A | 175 | 118 |
| Greedy Best First Graph H7 | 10 | 236 | N/A | N/A | 7 | 62 | N/A | 119 | **86** |
| Greedy Best First Graph H8 | 8 | 670 | N/A | N/A | **6** | 153 | N/A | 192 | 205 |

\*Alguns resultados têm o formato x/y, onde x é a profundidade do resultado e y é o número de estados visitados.

Número de passos da solução

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Puzzle 1 | Puzzle 2 | Puzzle 3 | Puzzle 4 | Puzzle 5 | Puzzle 6 | Puzzle 7 | Puzzle 8 | Média |
| Depth First Graph | **5** | 113 | N/A | N/A | 9 | 39 | N/A | 55 | 44 |
| Breadth First | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Uniform Cost | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H1 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H2 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H3 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H4 | **5** | 47 | N/A | N/A | **5** | **19** | N/A | **29** | 21 |
| A Star H5 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H6 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H7 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| A Star H8 | **5** | 47 | N/A | N/A | **5** | **19** | N/A | **29** | 21 |
| Interactive Deepening w/ Depth Limiting | **5** | N/A | N/A | N/A | **5** | N/A | N/A | N/A | N/W |
| Interactive Deepening w/ A Star H1 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H2 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H3 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H4 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H5 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H6 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H7 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Interactive Deepening w/ A Star H8 | **5** | **43** | N/A | N/A | **5** | **19** | N/A | **29** | **20** |
| Greedy Best First Graph H1 | **5** | 51 | N/A | N/A | **5** | 23 | N/A | **29** | 22 |
| Greedy Best First Graph H2 | **5** | 53 | N/A | N/A | **5** | 25 | N/A | **29** | 23 |
| Greedy Best First Graph H3 | **5** | 53 | N/A | N/A | **5** | 25 | N/A | **29** | 23 |
| Greedy Best First Graph H4 | **5** | 47 | N/A | N/A | **5** | 31 | N/A | **29** | 23 |
| Greedy Best First Graph H5 | **5** | 53 | N/A | N/A | **5** | 23 | N/A | **29** | 23 |
| Greedy Best First Graph H6 | **5** | 45 | N/A | N/A | **5** | 25 | N/A | **29** | 22 |
| Greedy Best First Graph H7 | **5** | 45 | N/A | N/A | **5** | 25 | N/A | **29** | 22 |
| Greedy Best First Graph H8 | **5** | 47 | N/A | N/A | **5** | 31 | N/A | **29** | 23 |

## Análise de resultados

Em média, em termos de espaços visitados até chegar a uma solução e tempo de procura, qualquer dos algoritmos que usam as heurísticas estão em vantagem, que é de esperar devido à habilidade de ordenar a fronteira e puder expandir primeiro os estados que são mais vantajosos e não são estados “mortos”, onde pode haver um *deadlock*.

Em média, o algoritmo uniform cost é o que precisa de visitar mais estados de modo a encontrar uma solução.

O algoritmo que demoram menos tempo é o greedy, pois apenas considera o resultado da função com apenas a heurística, portanto expande o que é imediatamente mais vantajoso.

Para além da esperada vantagem do algoritmo greedy, iterative deepening search com a star em certos puzzles desceu os estados visitados.

Em termos de heurísticas, as heurísticas baseadas em distância euclidiana demoram mais tempo a encontrar a solução do que as heurísticas baseadas em distância manhattan.

Para o tamanho da solução, não existe grande diferença nos puzzles onde a solução foi encontrada de entre os vários algoritmos, embora o algoritmo greedy foi consideravelmente pior em termos de tamanho de solução média devido ao facto de tomar decisões que podem não ser as mais eficientes mas em “*short term*” são as que têm menor custo.

O puzzle 7 é uma reordenação do puzzle 3 e o puzzle 4 é um puzzle obtido do website fornecido no enunciado.

O espaço de estados do puzzle 7 e 3 é e com um fator de ramificação grande torna o tipo de heurísticas usada não eficazes em resolver um puzzle com um espaço de estados tão grande. Seria necessário uma melhor verificação de *deadlock* e uma melhor avaliação da distância a um estado final no geral, pelo que mesmo após deixar o algoritmo correr com a melhor heurística feita durante horas, o resultado não foi encontrado.