

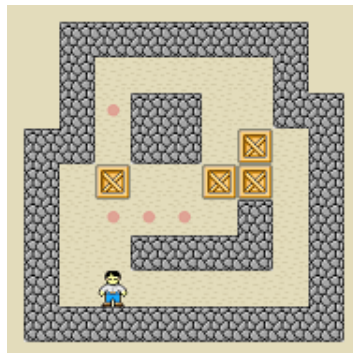
Sistemas Inteligentes 2017/2018

Projecto

v1.1

Resolução de puzzles *Sokoban*

Maio 2018



1	Introdução	2
1.1	Breve descrição dos puzzles <i>Sokoban</i>	2
1.1.1	Alguns exemplos de movimentação	2
1.1.2	Objectivo do puzzle	3
1.2	Implementação	3
2	Trabalho a Realizar	3
2.1	Formulação	4
2.2	Experimentação	4
2.3	Leitura de puzzle de ficheiro	4
3	Recursos disponibilizados	5
4	Entrega	6
5	Avaliação	6
6	Datas	6
7	Créditos	7
8	Versões	7

1 Introdução

Este projecto consiste na implementação de uma forma de resolução de puzzles *Sokoban*, recorrendo a uma formulação de acordo com o paradigma do espaço de estados e à utilização dos algoritmos de procura correspondentes.

1.1 Breve descrição dos puzzles *Sokoban*

Dado um mapa de um armazém, representado por uma grelha ($L \times C$), no qual existem N caixas, N locais alvo, e algumas paredes, um puzzle *Sokoban* consiste em encontrar uma sequência de movimentos de um agente arrumador que permita colocar as caixas nos locais alvo indicados.

As regras de movimentação do agente são as seguintes:

- Pode movimentar-se uma posição de cada vez num dos quatro sentidos principais (cima, baixo, esquerda ou direita), desde que a posição de destino não corresponda a uma parede nem nela esteja uma caixa.
- Pode empurrar uma caixa que lhe esteja adjacente, uma posição, num dos quatro sentidos principais, desde que a posição de destino da caixa esteja livre, passando o agente a ocupar a posição antes ocupada pela caixa.

Note bem:

- As caixas só podem ser empurradas; não podem ser puxadas.
- Só é possível empurrar uma caixa de cada vez.

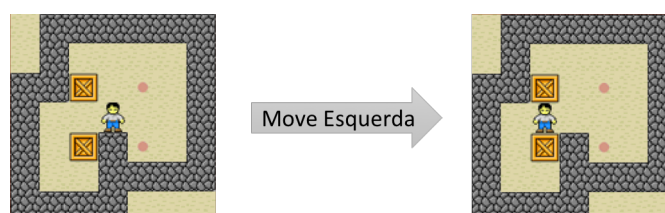
1.1.1 Alguns exemplos de movimentação

Em todos os cenários seguintes, é considerada uma grelha 7×7 .

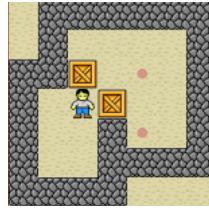
Exemplo 1 – movimentação livre



Neste exemplo, o arrumador tem três movimentos possíveis: deslocar-se para a *esquerda*, para a *direita* ou para *cima*. Não pode ir para baixo, pois tem uma parede. Caso a opção seja a movimentação para a esquerda, obtém-se o seguinte estado:



Exemplo 2 – empurrar caixa



Neste exemplo, o arrumador tem quatro movimentos possíveis: pode *deslocar-se para a esquerda* ou *para baixo*; e pode *empurrar uma caixa para cima* ou *para a direita*. Caso a opção seja *empurrar a caixa para a direita*, obtemos o seguinte cenário:



Note que, neste cenário, a opção de *empurrar a caixa para cima* não seria uma boa escolha, pois, não podendo as caixas ser puxadas, a caixa empurrada, ficando num canto, nunca mais de lá poderia sair.

Exemplo 3 – limitações de movimento

Recorde que o arrumador só consegue empurrar uma caixa de cada vez. Assim, no cenário seguinte, as únicas opções de movimento são: *deslocar-se para cima* ou *para baixo*.



1.1.2 Objectivo do puzzle

O objectivo do puzzle é não só o de colocar as caixas nos locais alvo, como também o de descobrir a sequência de acções do arrumador que o permite fazer no *menor número de movimentos possível*.

1.2 Implementação

O projecto deverá ser implementado em Python recorrendo à plataforma disponibilizada para o efeito.

2 Trabalho a Realizar

O trabalho a realizar tem três componentes principais:

1. Formulação do problema;
2. Experimentação;
3. Leitura de descrição de puzzles de ficheiro.

2.1 Formulação

A modelação, que terá que ser concretizada de acordo com o requerido pela plataforma utilizada nas aulas laboratoriais, deverá ser genérica. Ou seja, deverá permitir a representação de qualquer tipo de puzzle, devendo ser possível variar os seguintes parâmetros:

- Dimensão da grelha (número de linhas, L , e colunas, C). Note que o espaço do armazém é sempre delimitado por células que correspondem a paredes (mas que fazem parte da dimensão da grelha).
- Localização das paredes.
- Número de caixas, N (que tem que ser igual ao número de posições alvo).
- Localização das posições alvo (dentro dos limites do armazém).
- Localização inicial das caixas (algumas poderão coincidir com posições alvo, não necessariamente aquelas em que ficarão colocadas no estado final).
- Localização inicial do arrumador (poderá ser sobre uma posição alvo, mas nunca no mesmo local que uma caixa).

Note que, na implementação, poderá assumir que os valores fornecidos para estes parâmetros estão correctos; não é necessário implementar mecanismos de verificação da sua correcção.

2.2 Experimentação

Esta componente consiste na realização de experiências com a aplicação dos algoritmos disponibilizados pela plataforma. Tenha em atenção os seguintes pontos:

- Deverá utilizar criteriosamente alguns dos algoritmos disponibilizados para resolver os puzzles e comparar os respetivos desempenhos. É possível que nem todos os algoritmos sejam adequados.
- Poderá alterar os algoritmos disponibilizados de forma a coleccionar estatísticas, como por exemplo, o número de estados expandidos numa execução do algoritmo.
- Considere a definição de funções heurísticas e compare também os desempenhos em função das heurísticas que definir.
- Comece por resolver puzzles muito simples (por exemplo o *puzzle1*, apresentado na secção seguinte), avançando, progressivamente, para puzzles mais complexos (grelha maior, mais caixas).
- Não sendo obrigatório, será valorizada a opção de apresentação da resolução do puzzle, passo a passo, após a obtenção da solução.

2.3 Leitura de puzzle de ficheiro

De modo a facilitar a experimentação, é conveniente definir uma função que permita carregar um puzzle a partir de um ficheiro de texto.

A representação dos puzzles em ficheiros de texto deverá adoptar as seguintes convenções:

- o ficheiro deverá ter uma linha para cada linha da grelha;
- cada linha deverá ter tantos caracteres quantos o número de colunas;
- os caracteres possíveis são:

- ‘#’ – para representar as paredes;
- ‘.’ – para representar as posições livres;
- ‘*’ – para representar as caixas;
- ‘o’ – (um ó minúsculo) para representar os alvos;
- ‘A’ – para representar o arrumador.
- ‘@’ – para representar uma caixa numa posição alvo.
- ‘B’ – para representar o arrumador em cima de uma posição alvo.

Por exemplo, o puzzle inicial do exemplo 1 da secção 1.1.1, deverá corresponder a um ficheiro com o seguinte conteúdo:

```
#####
##....#
##*.o.#
#..A..#
#.*#o.#
#..####
#####
```

Já o exemplo que aparece na primeira página deste enunciado será representado em ficheiro do seguinte modo:

```
#####
##.....#
##o##..##
##.##.*.#
#.*...*.#
#.ooo.#.#
#..####.#
#.A.....#
#####
```

Um exemplo muito simples, com o qual poderá fazer as primeiras experiências, é o *puzzle1*, referido acima:

```
#####
#o..#
#.*.#
#..A#
#####
```

3 Recursos disponibilizados

São disponibilizados no moodle alguns ficheiros com puzzles para experimentarem. Estão desde já disponíveis os três referidos acima. Alguns links úteis:

- [Página da wikipedia sobre Sokoban](#)
- [Para jogar online](#) – e ver bastantes exemplos de puzzles de todos os graus de dificuldade.

4 Entrega

Os trabalhos deverão ser realizados em grupos de 2 ou 3 elementos. O registo dos grupos deverá ser feito no *moodle*, no link que vier a ser disponibilizado para o efeito.

Será também disponibilizado um link para a submissão do trabalho. A submissão do trabalho deverá consistir de um único ficheiro, compactado em formato `.zip`, com o nome `1718_sint_NN.zip`, em que *NN* é o número do grupo (por exemplo `1718_sint_01.zip` para o grupo 1) e deverá conter o seguinte:

- os ficheiros Python com todo o código necessário para executar o programa;
- um pequeno relatório `relatorio_NN.pdf`, com um máximo de 10 páginas explicando:
 - Como optaram por formular o problema: como é a representação dos estados, quais os operadores que definiram, etc.
 - Explicação das heurísticas definidas.
 - Ilustração de alguns exemplos de execução.
 - Análise comparativa, e crítica, dos algoritmos experimentados.

5 Avaliação

Para além do que resultar da avaliação do trabalho de grupo entregue, a nota de projeto individual será ponderada pela realização de mini-teste de aferição, de acordo com as seguintes regras. A nota final no projecto (*NFP*) é definida do seguinte modo:

- Sendo *NP* a nota do projecto, considerando o que foi entregue pelo grupo (é uma nota igual para todos os elementos do grupo);
- Sendo *NAf* a nota obtida no mini-teste de aferição;
- *NFP* é igual a:
 - *NP*, se $75\% \leq NAf \leq 100\%$
 - $0,75 * NP$, se $50\% \leq NAf < 75\%$
 - $0,5 * NP$, se $25\% \leq NAf < 50\%$
 - $0,25 * NP$, se $0\% \leq NAf < 25\%$
- Não existe nota mínima como condição de acesso ao exame.

6 Datas

- Publicação do enunciado: 10/5/2018.
- Constituição de grupos no [moodle](#): até 16/5/2018.
- Submissão do projecto: até 29/5/2018, 23:55, hora de Lisboa.
- Teste de aferição: 30/5/2018, 10:00, em sala a anunciar.

7 Créditos

As figuras e alguns puzzles foram extraídos do site: <http://www.abelmartin.com/rj/sokoban.html> de Jordi Doménech.

8 Versões

v1.1 (11/5/2018) Adição de um item na secção 2.1 ([Formulação](#))

v1.0 Publicada em 10/5/2018