CSD311 Course Project: Sokoban

5-Sep-2022

1 Sokoban

Sokoban is a seemingly simple board game where a player has to push all stones (also called crates/boxes) available on the board to final/goal positions on a board that is a maze. An example game is shown below in figure 1. The game rules are:

1. The player can move one square at a time in the four directions N,E,W,S.
2. The player can only push (no pulling) one stone one square at a time in the four directions N,E,W,S (no diagonal push).
3. No stone or the player can go through a wall or other stones in a move.
4. To push a stone in the relevant direction the player has to be in the appropriate immediately neighbouring square.
5. The game ends when all stones are in goal positions. A stone can occupy any goal position.

The objective of the game is to push all the stones to their final positions. A solver may choose to optimize i) the total number of moves or ii) the number of push moves. A push-move is one where a stone is moved.

Certain board positions are *deadlocks* - that is the puzzle cannot be solved from that position. For example, if a stone is pushed into a non-goal corner square we have a deadlock state.

The game has been proved to be PSPACE-complete so no simple strategies are likely to be available.

A good solver will require search, planning, learning and using knowledge.

Multiple sets of puzzle instances (often called levels) are available online. (see the references).

2 Sokoban: representation

The standard pictorial text file representation is called the xsb format. A single character represents the different entities and their state.

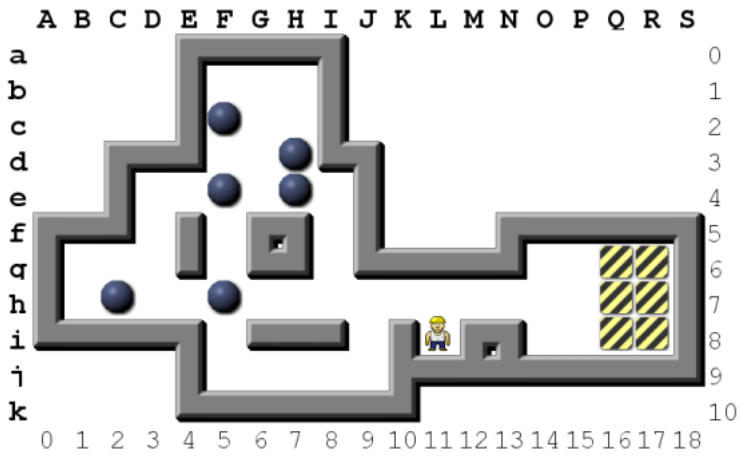

Figure 1: Sokoban game instance. Initial state: Stone positions: Ch,Fc,Fe,Fh,Hd,He; Goal positions: Qg,Qh,Qi,Rg,Rh,Ri; Player position: Li

```
' ' (white space) - Floor
@ - Player
+ - Player on goal
# - Wall
$ - Stone/Crate/Box
. - Goal
* - Stone/Box/Crate on Goal
```

The semicolon ';' is used as a comment character before the puzzle itself starts. If is used to give the name of the instance etc. The instance itself is flanked by a blank line at the beginning and one after the instance ends. See example below:

```
; 1

   ###
  ## # ####
 ##  ###  #
## $      #
#   @$ #  #
### $###  #
  #  #..  #
 ## ##.# ##
 #      ##
 #     ##
 #######
```

A move by the player is represented by `u, d, l, r` standing for up, down, left, right. If a move results in a push of a stone the relevant letter is capitalized i.e. `U, D, L, R`.

So, a solution to an instance is just a string of letters from the set {`u, d, l, r, U, D, L, R`}.

For graphical viewing we can visualize an instance as follows:
Each instance/ level has a board size and a closed playing area surrounded by walls. The board can be thought to be made up of squares. Each square has a coordinate labelled by a two letter string `<X-cord><Y-coord>` where the X-coordinate is represented by a capital letter from A to Z and the Y-coordinate is represented by a small letter from a to z. So, for us that means the board will never be larger than 26×26. In practice there is no limitation on the size of the board and squares can be labelled with two digit string combinations. The top left square is Aa (see the board in figure 1). Moves can be written as a succession of square coordinates separated by spaces. Any game state can be completely specified by writing the relevant character from the xsb format in a square. If the square is blank it is a floor square. The position in figure 1 is reproduced below.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
a					#	#	#	#	#										
b					#				#										
c					#	$			#										
d			#	#	#			$	#	#									
e			#			$		$		#									
f	#	#	#		#		#	#		#				#	#	#	#	#	#
g	#				#		#	#		#	#	#	#	#			.	.	#
h	#		$			$											.	.	#
i	#	#	#	#	#		#	#	#		#	@	#	#			.	.	#
j					#						#	#	#	#	#	#	#	#	#
k					#	#	#	#	#	#	#								

3 Project guidelines

The project is to be done in groups with at most 5 students in a group. A Google form has been put up so that you can create your group.

The game is well studied and there are numerous implementations that perform at different levels. But the game has not been 'solved'. Write your own solution and do not borrow code from other implementations (see the honesty policy on the course website). Your solver should be written in Python. No other implementation platform is allowed.

The project will be evaluated based on a) how good your solver is on a test set created specially for the project b) the quality of the project report that you submit towards the end of the semester due on 26 Nov. 2022 on BB. You will have to demomstrateoutput your solver on the test set during the week of 19th-25th Nov. 2022.Your solver should use the xsb format for input and use the standard output format mentioned in the previous section for the sequence of moves. Each test instance will be automatically evaluated for correctness and the moves and push-moves statistics for your solver will be accumulated for each instance to yield a test score out of 100 for the test suite. The report will be evaluated out of 30.

4 References

1. http://sokobano.de/wiki/index.php?title=Solver_Statistics (Main source of information for game sets and solvers. It has many more links.)

2. https://www.cs.cornell.edu/andru/xsokoban.html (The XSokoban homepage.)

3. http://www.game-sokoban.com/index.php?mode=catalog (An online Sokoban page. You can play a graphical version of the game to get some insight.)