

Sokoban: Search in a complex domain

Yann Chazallon, Nicolas Dossou-Gbété, Tony Chan Ki Hong
and Michal Staniaszek

October 21, 2013

Table of Contents

① Introduction

② Development Process

③ Evaluation

④ Conclusions

What is Sokoban?

Sokoban is a puzzle game first published in 1982

- The goal is to push boxes onto goal locations in a map
- Player can move up, down, left or right
- Boxes can only be pushed into empty spaces
- Can only move one box at a time

Why is it interesting?

- AI in games—investigate techniques in simple environments
- High branching factor comparable to chess, based on possible box moves
- Solution depth much deeper than any chess game
- Solutions can be arbitrarily long due to repeated motions

Table of Contents

① Introduction

② Development Process

③ Evaluation

④ Conclusions

Board Representation

- Two layers
 - Static (walls, goals): singleton
 - Dynamic (player, boxes): search space
- Static cost map calculated at launch
 - Cost from each point to each goal
 - Cost from each point to each initial box position

Static Lock and Cost Map

```
#####  
#X#####X210X#  
#X#####6321X#  
#Xcba98765432X#  
#####XXXXX#  
XXXXXXXXX#####
```

Board hashing and equality

- Hash: array of chars: blank, \$ and @
- 2 versions: with and without player position
- Used for `getHash()`, `getHashCode()`, `equals()`
- 3 types of equality checks used:
 - Without player position: check if state is goal
 - With top leftmost player position: repeated state checks
 - Exact player position: during the actual path reconstitution

Heuristics

- First heuristic: Manhattan distances
- Other unsuccessful attempts:
 - Real cost
 - Pseudo MinMatching

Locked State Detection

- Not using any pattern dictionary
- First implementation was building a graph of dependencies
- Then improved to explore implicit graph
- Should never return false positives
- Side-effect: able to detect corner locks (redundancy with static check)

Dynamic Lock Test Map

```
#####
#$      $$      #
#  $$      #
#$  $$      #  #
#          $$$  #
#      $#      $$  #
#  #$      ###
#   $      .+#
#####
```

Player Space Search

Our first approach

- Successors of states based on the motions of the player
- Very slow—useless moves, even deeper solutions
- Applied A* search to find solutions
- Only trivial maps solved

Board Space Search

Improvement on the player space search

- Successors of states based on possible moves of accessible boxes
- Search changed to best-first search—don't need to find optimal solution
- BFS is complete, since we are using a closed list
- A* used to rebuild player path when a solution is found
- Managed to solve some nontrivial maps

Bi-directional Search

Our final version, improved search method rather than heuristics

- Previous attempts at improving heuristics failed
- Improving search seemed to be a better option
- Reduces the complexity from $\mathcal{O}(b^d)$ to $\mathcal{O}(b^{d/2})$
- Need to use multiple initial states for the reverse search

Table of Contents

① Introduction

② Development Process

③ Evaluation

④ Conclusions

Method Comparison

Search Method	Time limit		
	5 sec	11 sec	15 sec
A*	12	15	16
Best First	56	60	64
Bi-directional Best First	76	81	82
Bi-directional A*	39	41	43

- No significant difference in number of maps solved with different limits
- Is the search going in the right direction?

Map Performance

- Can be solved within 15 sec, but not 11
- Requires a box to be positioned (at x) and not moved until the end.
- Problem is caused by heuristic preferring boxes on goals

Map 54

```

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

Map Performance

- Solved very quickly
- All but one box require only a single move
- Heuristic gives accurate estimate to the goal

Map 66

```

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

```


Map Performance

- Unsolved within 15 sec
- Intermediate goal area causes issues with heuristic
- Requires making specific move sequences to get boxes on goals

Map 93

```

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

Table of Contents

① Introduction

② Development Process

③ Evaluation

④ Conclusions

Conclusions

- Focus on search more than heuristic
- Took a long time to get a simple solver
- A* is good, but application dependent

Questions

?