AI Assisted Design of Sokoban Puzzles using Automated Planning

Tomáš Balyo[1] Nils Froleyks[2]

[1]CAS Software, Karlsruhe, Germany
tomas.balyo@cas.de

[2]Johannes Kepler University, Linz, Austria
nils.froleyks@jku.at

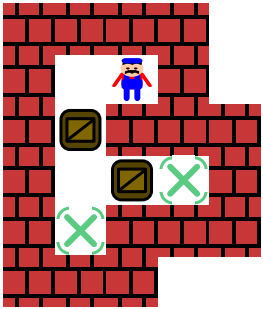
EAI ArtsIT 2021 – 10th EAI International Conference: ArtsIT, Interactivity & Game Creation

Contents of the Talk

- What is Sokoban
- Designing levels for Sokoban
- How does our new level design tool work
- Usage demonstration
- Evaluation and Conclusion

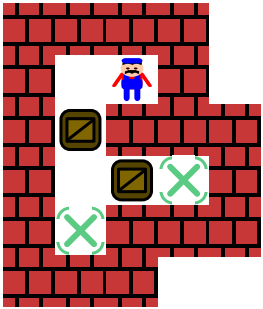
What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



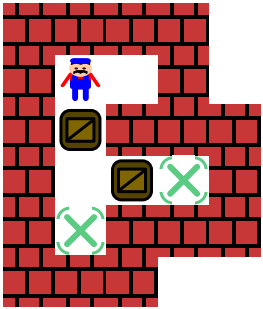
What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



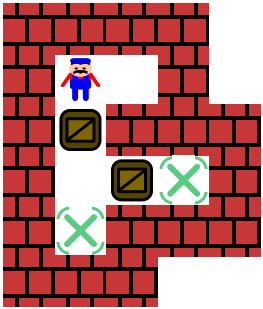
What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



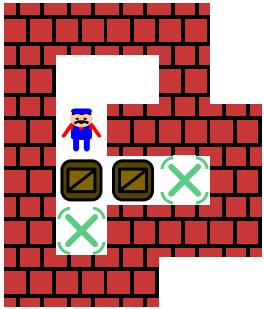
What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



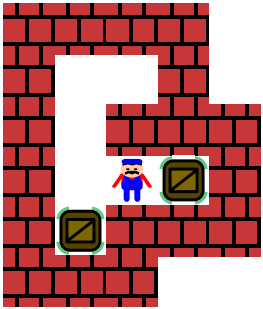
What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



What is Sokoban

- A puzzle game originally from Japan in 1982
- A level (warehouse) consists of a single worker and multiple walls, boxes and goal positions
- The player can move the worker in the 4 cardinal directions (up, down, left, right)
- The worker cannot walk through walls
- The worker can push a box if the tile behind it is empty (no wall or other box there)
- The goal is to push the boxes onto goal positions



Why Sokoban?

- Sokoban is well known in Video Games Communities
  - Multiple implementations on almost all existing platforms
  - Thousands of levels created by designers and fans
- Sokoban is also well known in Computer Science Research
  - In Complexity Theory – it was proven to be P-SPACE complete
  - In Artificial Intelligence – numerous papers on solving and generating Sokoban levels have been published

What is a good Sokoban Level?

- A level should be solvable
  - unsolvable levels are interesting in research but frustrating for human players
  - easy to guarantee in automatic generation
- A level should be challenging
  - avoid levels that are solved with very few steps
  - avoid levels with obvious solutions
  - challenging to ensure in automatic generation
- A level should be visually attractive for human players
  - very difficult to achieve without a human designer

How our Generator Works

- basic idea – translate the task of designing a level into a standard planning problem and use a state-of-the-art automated planner to solve it
- what is a planning problem?
  - given a set of possible actions, initial state and goal conditions
  - find a sequence of actions that can get us from the initial state to a state where all goal conditions are met
  - a well researched problem with very efficient solver programs available

How our Generator Works

- basic idea – translate the task of designing a level into a standard planning problem and use a state-of-the-art automated planner to solve it
- what is a planning problem?
  - given a set of possible actions, initial state and goal conditions
  - find a sequence of actions that can get us from the initial state to a state where all goal conditions are met
  - a well researched problem with very efficient solver programs available

How our Generator Works

- how to express level design as a planning problem?
  - the initial state is an unfinished level where additional boxes, walls, and the worker can be added at certain locations – specified by the human designer
  - the goal conditions are that the specified number of boxes, walls, and the worker is added and the level is solvable
  - the possible actions are placing items and playing the game
  - the final plan has two parts, the first part is placing items to finish the level, the second is solving the level

How our Generator Works

- how to express level design as a planning problem?
  - the initial state is an unfinished level where additional boxes, walls, and the worker can be added at certain locations – specified by the human designer
  - the goal conditions are that the specified number of boxes, walls, and the worker is added and the level is solvable
  - the possible actions are placing items and playing the game
  - the final plan has two parts, the first part is placing items to finish the level, the second is solving the level

How our Generator Works

- how to express level design as a planning problem?
  - the initial state is an unfinished level where additional boxes, walls, and the worker can be added at certain locations – specified by the human designer
  - the goal conditions are that the specified number of boxes, walls, and the worker is added and the level is solvable
  - the possible actions are placing items and playing the game
  - the final plan has two parts, the first part is placing items to finish the level, the second is solving the level

How our Generator Works

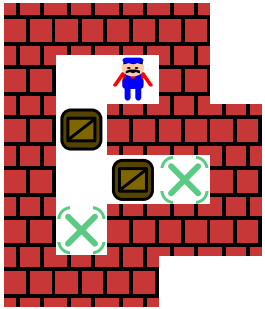
- how to express level design as a planning problem?
  - the initial state is an unfinished level where additional boxes, walls, and the worker can be added at certain locations – specified by the human designer
  - the goal conditions are that the specified number of boxes, walls, and the worker is added and the level is solvable
  - the possible actions are placing items and playing the game
  - the final plan has two parts, the first part is placing items to finish the level, the second is solving the level

How our Generator Works

- how to express level design as a planning problem?
  - the initial state is an unfinished level where additional boxes, walls, and the worker can be added at certain locations – specified by the human designer
  - the goal conditions are that the specified number of boxes, walls, and the worker is added and the level is solvable
  - the possible actions are placing items and playing the game
  - the final plan has two parts, the first part is placing items to finish the level, the second is solving the level

How to use our Generator

There is a standard text format for Sokoban levels



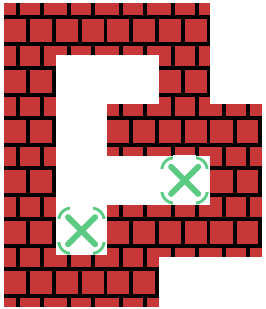
```
####
# @#
#$###
# $.#
#.###
###
```

symbols:

- `#` - wall
- `@` - worker
- `+` - worker on goal
- `$` - box
- `.` - goal
- `*` - box on goal

How to use our Generator

We extend the format with additional symbols to specify possible locations of additional walls, boxes, and the worker.



```
####    symbols:              what can be added:
# 1#     # - wall              0 - anything
#0###   @ - worker            1 - worker
#00.#   + - worker on goal     2 - wall
#.###   $ - box               3 - box
###     . - goal              4 - worker or wall
        * - box on goal       5 - worker or box
                              6 - wall or box
```

How our Generator Works

1. use any text editor to create level template as shown above
2. save to a text file, for example *level.txt*
3. download our tool from `https://github.com/biotomas/sokoplan` and build it following the instructions
4. run the tool with the command `./run.sh level.txt`
5. the generated level will be saved to a file *level.txt.solution*
6. you can see and test your level using the included *levelTester.html*

Demonstration

Previous vs Our Generator

- Previous work
  - fully automatic level generators
  - based on the principle of generating plenty of levels and then filter the good ones
  - the search algorithm is fully integrated, the generator does not improve over time without extensive maintenance
- Our Generator
  - the tool utilizes human input
  - based on the principle of searching for the good level directly
  - the search is outsourced to Automated Planners, which get better over time improving the generator tool automatically

Conclusion

- We developed a new tool to assist a human level designer in developing Sokoban puzzle levels.
- It has some limitations
  - the outline of the level must be provided
  - all the goals must be placed by the designer
  - performance problems with large levels
  - very technical user interface
- Future work
  - fix the limitations
  - use the approach to develop tools for other similar games