2018 BIE-ZUM PROJECT

Sokoban Solver

Berker Katipoglu | katipber

Instructions

Implementation of sokoban solver.

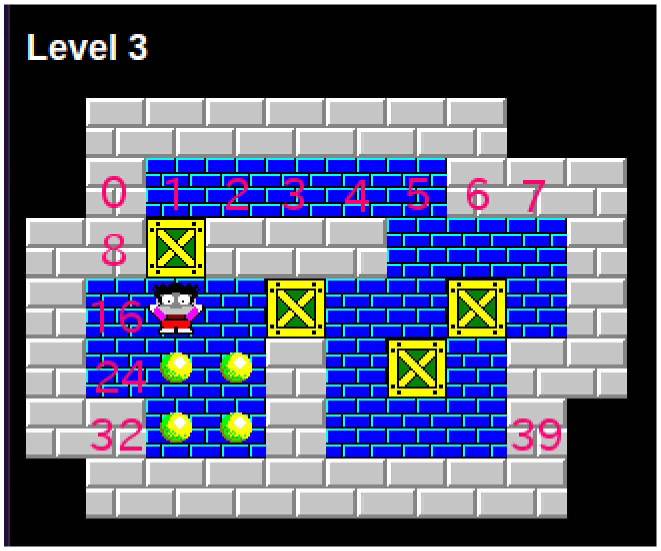
Dataset

Boxword level 3 is used for this task however solution can be applied to any puzzle given in sokoban level format.

Solution

Solution is implemented using Python.

First implementation of the solver was inspired by PDDL language. Each square in the puzzle was numbered and walls, boxes, target points and position of the player were given as the index of the square they are occupying.



For example in this puzzle, values will be:
Walls: 0, 6, 7, 10, 11, 12, 27, 31, 32, 35, 39
Boxes: 9, 19, 22, 29
Target: 25, 26, 33, 34
Player: 17

Algorithm preforms a BFS according according to each possible move of the player. If move is valid, Boxes array and Player position is updated accordingly. Each played position is saved as a tuple of player position and position of boxes, to avoid visiting same position again.

Although this algorithm finds the minimum number of moves to solve the puzzle, for complicated puzzles search tree grows too big and algorithm cannot finish in reasonable time.

Second implementation focuses on some improvements on the sokoban solver, and it uses sokoban level format input.

First improvement is switching to bitmap board representation in order to perform faster evaluations.
Second improvement is to store the positions according to player's reachable squares and box positions. This allows search tree to get smaller and also the number of stored positions to decrease.
Third and final improvement is to allow player only to make push moves. Walking moves are calculated only at the final stage, when solver finds a solution.

Results

Solver results for level 3 is as follows:

	First Solution	**Second Solution**
Time	49 sec	3 sec
Iteration	252,411	13,093
Visited Positions	242,421	13,115
Open Positions	9	21
# of Moves in Solution	115	162

Time of the solution is decreased significantly from 49 seconds to 3 seconds, as well as nodes searched is decreased from 250K to 13K.

Conclusions

Using PDDL like approach allows much more simple implementation, however this kind of solution is far from being efficient.
To be able to build more efficient sokoban solver algorithms, search tree should be narrowed down by implementing calculation of deadzones, or detecting deadlock positions.