Royal Institute of Technology

(a) Caroline (b) Paul

(c) Sindri (d) Tobias

Artificial intelligence, dd2380

Final project: sokoban

Caroline Laurène Kéramsi, keramsi@kth.se, XXXXXX-XXXX
Paul Lagrée,lagree@kth.se, XXXXXX-XXXX
Sindri Magnússon, sindrim@kth.se, 871209-7156
Tobias Johannes, Uebbing@kth.se, XXXXXX-XXXX

October 8, 2012

Contents

Abstract

asdf

1 Introduction

Sokoban is an popular puzzel dating back to the eighties. The original setting of the puzzel is a warehouse and the problem is to push boxes around to predefined storage locations. But the underlying problem is of course much more abstract. The rules are simple [1]:

I Only one box can be pushed at a time.

II A box cannot be pulled.

III The player cannot walk through boxes or walls.

IV The puzzle is solved when all boxes are located at storage locations.

Though the rules are simple the problem is quite difficult and has been proven to be NP-hard. This is not only due to the brancing factor but also the enormous deepth of the search tree [1].

2 Design

2.1 Project organisation

We started with a simple implementation which was able to solve a small number of boards. Then we applied different methods to improve it. As it takes time to get feedbacks from the server, we wrote our own bash script to evaluate the number of boards we are able to solve. This script runs our solver on 100 boards (server port 5032). We didn't want to wait too long for the results of the script so we limited the search time to 30 seconds per board.

2.2 States representation

A state can be represented by the position of the player and the position of each boxes on the map. Yet this representation is naive. We want to have a new state when we push a box and not when we only move the player, without touching a box. So instead of considering the player position, we consider the area he can reach. This area can be computed easily by a recursive algortihm. Two states can be compared by looking at the box position and the leftest, upmost case of the reachable area.

2.3 Initial algorithm

To expand a state, we look at each direction of each box. If a direction is inside the area that the player can reach, we look if there is a hindrance (box or wall) in the opposite direction. If this is not the case, the box can be pushed and we can create a new state. The new state is compared with the states which have already been created. If the state existed already, we delete the new state. Otherwise we add it to the list of created states and to the fifo of the states wich must be expanded. Then we can take the next state in the fifo and repeat the procedure. We stop when we reach a final state (ie. a state where all the boxes are on a goal). Once we found a final state, we apply a pathfinding algortihm to build the solution string from the succession of states. This algorithm performs a Breadth-First-Search in the search space.

With this first version we could solve only 1 board out of 100.

2.4 Deadlocks

2.5 Pathfinding algorithm

2.6 Search

sdf [2]

2.7 Pruning

3 Results

4 Conclusions

References

[1] Unknown author. sokoban on wikipedia, Oktober 2012.

[2] Timo Virkkala. Solving sokoban. Master's thesis, UNIVERSITY OF HELSINKI.