Sokoban Solver

Benjamin Huang, Yen-shi Wang, and Zachary Armendariz

Project Proposal

- **What is the application domain/area?** In the domain of Artificial Intelligence, search and game theory are always popular research topics. For this project, we aim to design better kernels according to hardware architecture to solve Sokoban, which is a well-known puzzle since 1982.

- **What is the proposed computation problem?** Sokoban is a puzzle game, involving a character packing boxes in a warehouse. The objective is to move boxes from their starting locations onto the designated positions in the warehouse. The character can only push boxes from adjacent squares in the direction of the box from the character. Often, the warehouse layout is oddly shaped, providing a challenge since pushing a box the wrong way will often result in it being stuck in a corner, and pushing a box the right way requires the character to be positioned correctly.

  Finding a solution is PSPACE-complete. Currently, the best available solver (Takaken) is only able to solve 86 of 90 levels in the standard test suite, known as XSokoban.

  Solving a Sokoban puzzle is essentially a graph search problem. From the initial state, a program branches out to all possible future states. Because of the sheer complexity of the problem, a fair number of pruning techniques are needed to ensure that solvers are not simply doing a brute-force search. This provides many areas for optimization.

  A faster solver would be able to explore more of the search tree, which would enable it to find solutions more quickly. Typically, puzzle solve time is on the order of seconds to hundreds of seconds, depending on the difficulty.

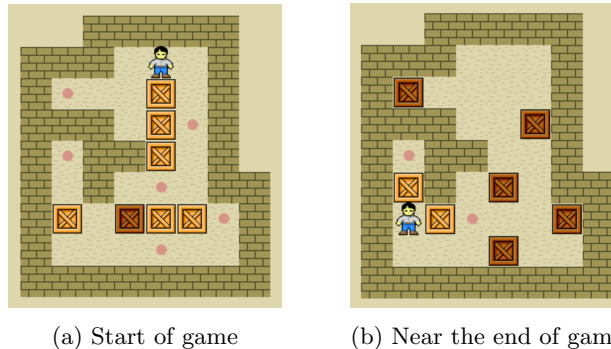
(a) Start of game   (b) Near the end of game

Figure 1: Screenshots of Sokoban

- **What is the state-of-the-art? Is there a standard or algorithm?**

  - Rolling Stone. This is a 2004 solver. It is open source and written in C. It solves 57 of 90 puzzles in the standard test suite.
  - JSoko. This is a current game and solver package written in Java. It is an open source project. It can solve 67 of the 90 puzzles in the standard test suite.
  - YASS. This is a current game and solver package written in Delphi/Kylix. It is an open source project. It can solve 84 of the 90 puzzles in the standard test suite.
  - Sokolution. This is a current solver. It is not open-source and runs on Windows. It can solve 80 out of the 90 standard puzzles.

  The common techniques used to solve Sokoban can be categorized into data structures, search algorithms, and pruning methods.

  - Data structures: hash table for each state, transposition tables, normalized player position.
  - Search algorithms: BFS, DFS, iterative deepening, a-star search, and Bi-directional BFS.
  - Pruning methos: dead lock, PI-Corrals.

- **What is the scale? How much data? About the deadline?** There are about 4000 puzzles that can be used for benchmarking. Most of them can be downloaded from Sokoban Wiki.

- **What is the end product you are expecting to deliver?** The first phrase of this project will focus on choosing a baseline solver, determining which algorithm and data structure to be optimized, and collecting benchmarks we need during the entire project.

  At the end of the project, we expect both algorithm and data structure run faster on ECE workstation and the amount of improvement can be clearly visualized by charts and statistics. Moreover, there should be a version of solver can support parallel computation.

- **What architecture?** ECE Workstation ece030.ece.local.cmu.edu. 2x Intel Xeon E5 2680, 2.7GHz (3.5GHz Turbo), 256GB RAM

Tentative division of labor		Tentative Meeting Times	
Yen-shi Wang	Deepening A* Search, Deadlock Tables, Goal Cuts	1st Preference	Tuesday, 3:00-4:00pm
Benjamin Huang	Minmatching, Tunnel Macros, Goal Macros	2nd Preference	Thursday, 3:00-4:00pm
Zachary Armendariz	Hash Tables, Move Ordering, Pattern Searches	3rd Preference	Friday, 1:00-2:00pm

References

[1] Dorit Dor and Uri Zwick. "SOKOBAN and other motion planning problems". In: *Computational Geometry* 13.4 (Oct. 1999), pp. 215–228. ISSN: 09257721. DOI: 10.1016/S0925-7721(99)00017-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925772199000176.

[2] Nils Froleyks. "Using an Algorithm Portfolio to Solve Sokoban". PhD thesis. Karlsruhe Institute of Technology, 2016, p. 56. URL: https://baldur.iti.kit.edu/theses/SokobanPortfolio.pdf.

[3] Anand Venkatesan, Atishay Jain, and Rakesh Grewal. "AI in Game Playing: Sokoban Solver". In: (2018). arXiv: 1807.00049. URL: http://arxiv.org/abs/1807.00049.

[4] Huaxaun Gao et al. "A Sokoban Solver Using Multiple Search Algorithms and Q-learning". In: ().

[5] James Hyun and Seung Hong. *sokoban-solver*. URL: https://github.com/jameshong92/sokoban-solver (visited on 09/16/2019).