Room Forming in Sokoban

Jacob Yaskowich, Nicholas Westbury, and Harm van Seijen

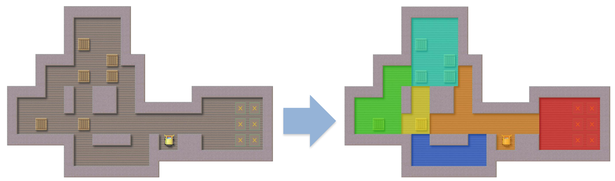
University Of Alberta, Computer Science Department, RLAI Lab

Introduction

Sokoban is a Japanese puzzle game in which you control a character capable of moving in four directions (Up, Down, Left, and Right), pushing crates to storage locations. The character cannot pull crates. To solve a given puzzle, every crate must be at a storage location. **Making use of Harm's solver program we created an algorithm that finds rooms, which the solver program uses as input.**

Purpose of a Room

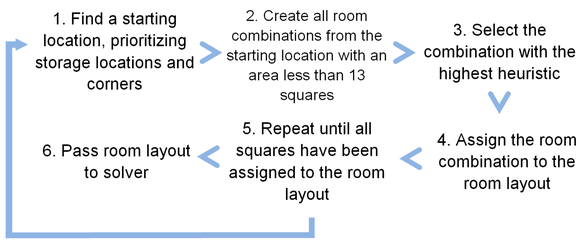
The computation time of the solver depends on how well levels are divided into rooms. A bad room division can increase exponentially the computation time compared to a good room division. The purpose of a room is to split a complex level into simpler pieces so that the solution can be more easily computed. An example of a room division is pictured below, with every color representing a different room.



What did we do?

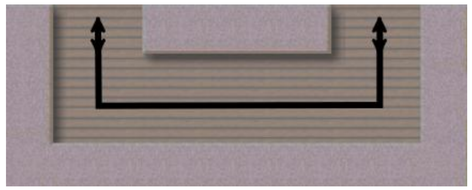
- By analyzing the behavior of the solver, we determined what makes a good and a bad room division.
- We created a heuristic function, a general rule used to evaluate rooms, based on these intuitions.
- We used the heuristic that evaluates single rooms, to divide a level into multiple rooms.

How does our Algorithm Work?



What makes a good room?

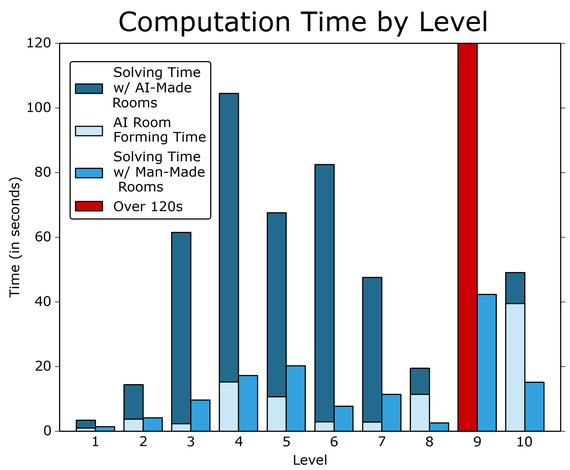
- **The size of the room –** The room should not be too large or too small.
- **The number of ports** – Ports are the entrances/exits to a room. Generally, favoring a small amount of ports is optimal.
- **The number of Storage locations and/or crates in the room**
- **Port positioning** – Is the port in a good position or a bad one?
- **Room Purpose:**
  - Corridors - Can only the character pass through this room?
  - Storage Area – Does the room contain storage locations?
- **Balance** – Are the factors weighted correctly? How does one variable affect another? Some factors are more important than others.



Above: A block cannot be pushed from one port to the other as it would be stuck or "deadlocked" in its path. A character on the other hand, can pass through the room. This attribute makes this room "good" as it has a well-defined purpose and only two ports.

Results

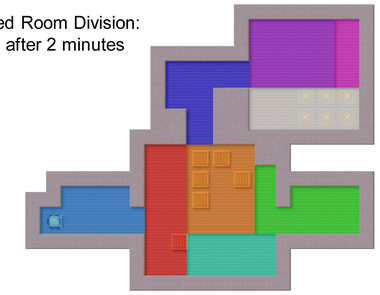
The results below show the room-forming algorithm (left bars) versus the best human-found room division (right bars). Level 9 was stopped after running over two minutes using the current room-division algorithm. Level 10, an open-space level, on the other hand was able to be solved faster than the best human-divided room could.



Results (Cont.)

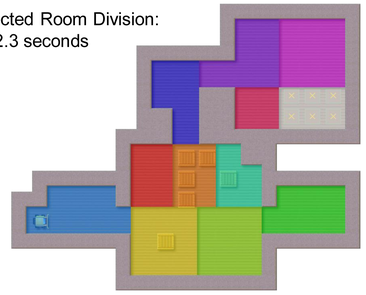
The current algorithm excels in levels that have clearly-defined large square boundaries and preforms worst on levels with multiple non-rectangular small rooms and corridors.
Level 9 is a demonstration of this phenomenon:

AI-selected Room Division:
Unsolved after 2 minutes



VS

Human-selected Room Division:
Solved in 42.3 seconds



Future Directions

There are many directions the project could take to improve. For instance:

- Adding a "global" heuristic that would evaluate how good the overall room setup is to stop rooms from interfering with each other
- Find more factors that make a good room to construct better rooms
- Create a learning or evolving room process to select optimal weights
- Optimizing the code by improving the algorithm or porting it to a language that can be compiled such as C (currently in Python).

Acknowledgements

We would like to thank the RLAI lab and the HIP program for this excellent opportunity to explore the fascinating world of computer science!