CAB 320 – Artificial Intelligence  
Sokoban Puzzle

Braydon Burn & Michael Blair

N9715894 - N9665021

2nd May, 2017

Environment Used

1. Python: Python 3.6.0 was used in the creation of the solver.
2. Python Dependencies:
   1. search.py: A search algorithm library.
   2. sokoban.py: A Sokoban helper class.

Taboo Cells

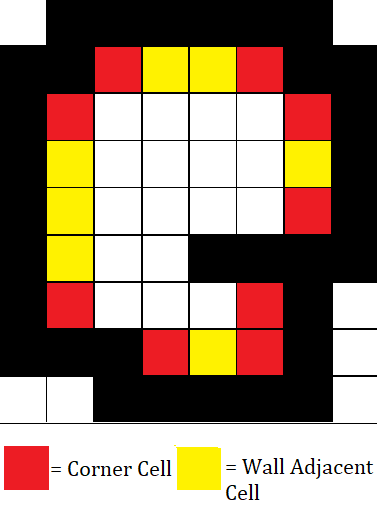
Taboo cells were added to reduce the amount of iterations the search has to go through and in turn increase the overall performance of the solver. Two types of taboo cells were identified when coding the Sokoban solver, the first being corner cells and the second being wall adjacent cells.

**Corner Cells**

If a cell is in a corner and not a target it is deemed to be a taboo cell, this is because once a box is moved into a corner there is no way for it to be moved by the worker. Figure 1 highlights the taboo corner cells in an example warehouse. Corner cells were checked by testing diagonal relationships between wall cells.

**Wall Adjacent Cells**

Wall adjacent cells are another type of taboo cell. If a box moves adjacent to a wall where there is no opening for the worker to get behind the box, it is a taboo cell. Figure 1 highlights the taboo wall adjacent cells in an example warehouse. A helper method called taboo\_along\_wall(warehouse, tup1, tup2) was created to determine wall adjacent cells by checking is there is a direct line between one taboo cell and another.

Figure 1: Corner Cells and Wall Adjacent Cells

Elementary Solver

**Actions**

To make sure that incorrect moves were not given to the search.py class all possible moves for the worker were calculated in “actions”. This method checks if the agent can move a box (Left, Down, Right, Up) without moving it into a taboo cell or pushing two blocks (Invalid move). To do this, a for loop was used to iterate through each possible move, if the move satisfies all the constraints it is then sent to a MovementList.

**State**

To increase the overall performance of the elementary solver, only the necessary elements of the puzzle were stored in state. Other elements such as walls, targets and taboo cells were created locally.

The state is stored as a tuple in the form of worker(x,y) and boxes(x,y).

**Heuristic**

The heuristic function h(self, node) takes the input of a potential node state variable then calculates the manhattan distance of all boxes to their closest goal state. Due to the nature of the Sokoban puzzle, a solution almost never takes the form of moving the boxes to their nearest goal state. In all cases, it is most equal to the actual cost of the solution. Therefore, the manhattan distance heuristic is an admissible heuristic and suitable to find the lowest cost solution to the puzzle.

Macro Solver

Three things were changed between the elementary solver and the macro solver. The action, the result and the heuristic.

**Action**

**Result**

**Heuristic**

**Can Go There**

Can\_go\_there is a method to test if the worker can move to a specific destination without moving any boxes. The method is used in the macro solver to reduce the amount of actions the search must make to get to a specific destination.

Performance

To solve the puzzle an A\* graph search was used. The performance of an A\* search algorithm is much better than a breadth first search or any similar brute force method but could still be improved by using an Iterative Deepening A\* search. An IDA\* search achieves a smaller search depth by constant factor and in turn would undoubtedly reduce the search time by a considerable amount.

**Elementary**

After testing was completed on a range of different warehouses we found that the performance of the elementary solver is dependent on the warehouse. Warehouse’s with simple solutions can be solved almost instantly but larger puzzles take longer (Several minutes to several hours).

**Macro**

In comparison to the elementary solver, the macro solver is slower. When expanded, the actions outputted by the macro solver are the same as the elementary solver, this is an indication that both are admissible heuristics.