**Sokoban Assignment  
Intelligent Search – Motion Planning in a Warehouse**

CAB320 Artificial Intelligence

*April 19, 2024*

**Group 33**

*Kenzie Haigh, n10755012*

*Luke Whitton, n1081425*

*Emma Wu, n1132833*

# Solver Features

## 1.1 State Representation

**States**

A state can be represented by the position of the player and the position of each box on the map and state will change to new state when either box being pushed, or player moves to different position.

represent the state as the objects that are dynamic (moving), i.e., the worker and boxes, in the state representation, and the target squares, walls, and taboo squares, as static (not moving), which remain in the problem instance and referred to when required.

The puzzles and their initial state are coded as follows in the text files,

(space, a free square ’#’, a wall square, ’$’, a box, ’.’, a target square, ’@’, the player '!', the player on a target square, '\*', a box on a target square)

## Taboo cells

\* The taboo cell is identified as a cell inside a warehouse when a box gets pushed on it, the game becomes unsolvable.

\* List to store all possible wall cells according to its movable directions (vertical or horizontal) and used recursive function to identify whether

\* There are two conditions to determine the taboo cells: is a corner cell but is not target, and all cells between non-target corners along a wall if none of these cells are targets.

\* A string representing the warehouse with only the wall cells marked with a '#' and the taboo cells marked with a 'X'.

Implementations:

| Inside warehouse | Identify whether the cell is inside the warehouse  Loop to find cells inside of warehouse loop  Inside cell has four walls around it, find all wall of each cell  Return a list of cell locations inside the warehouse  stops us from having taboo squares outside of the actual warehouse |
| --- | --- |
| Corner | Recursive function to identify whether cell is corner  Input warehouse class with cell location and  Search the non target cell joints one wall cell on both horizontal and vertical directions  Return a tuple contains Boolean whether the cell is corner and absolute location and the relative location |
| Taboo type | X: X corner (neither be moved in horizontal nor vertical direction)  T: T-section/T-corner  C: corner  N: corner/taboo neighbour  #############  #X# N C#  #T# #  # ########  #N ##  # T#  ############# |
| Display | Display taboo cells in the warehouse |

# 2. Methodology and Testing

## 2.1 Search algorithm

## A\* Algorithm with Heuristics

The solution uses two search algorithms to find an optimal solution.

A breadth first search algorithm to find all the cells reachable by the player character

|  |  |
| --- | --- |
| A\* algorithm | best first graph search  to determine the optimal must be consistent, and thus, admissible path for the Sokoban game using  Effective search method to avoid using large memory space and optimise the computational time  Function  f(n) presents the estimated cost of current state as node n to the goal state.  g(n) is the cost of moving from start state to current state node n.  assign an individual pushing cost to each box, whereas for the classical Sokoban, we simply count the number of actions executed. |
| Heuristics | h(n) is the heuristic which presents the estimated cost of moving from current state to the goal statement  Consistency: estimated heuristic costs is less or equal to actual costs  Admissible heuristic  The heuristic uses a ‘Manhattan distance’ formula  Independent |
|  |  |

## 2.2 Validation

Identifying taboo cells

| Rule 1 | Research goal is finding all types of taboo corners  Steps:  check if the cell is in warehouse  if it is a corner cell  loop to search Ts (2 corners) and Xs (4 corners)  if not target, add Ts and Xs in taboo corner list if it is not exist in the list  if the corner cell is target, the cell is safe and not in corner cell list  return a list of all non target taboo corner cells |
| --- | --- |
| Rule 2 | Find non target cells joints same raw/column’s wall cells between corners:  find group of corner neighbours which are between same two corners  (location within deduction of row/column of two corners )  Each group of corner neighbours has no any target cell  Return a list of taboo corner neighbours  Line 345 what is eft? |
| Polish taboo list | Remove:  dupliates from taboo\_straight\_cell\_list  out of bounds taboo cells  all occurances with negitive numbers |

Identifying Sokoban puzzle

|  |  |
| --- | --- |
| Representation of puzzle |  |
| Identify puzzle characters |  |

Identify Solutions

|  |  |
| --- | --- |
| Action sequence |  |
|  |  |
| Cost of solution |  |

# Performance and Limitations

## 3.1 Performance

The programming implementation can be divided into ….steps:

-

-

## 3.2 Limitations

The …… part of output result …… is/is not as expected.

There are limited ……… than the actual value

A further emulation of …. required for optimising the current calculation to achieve