**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.

\* Implementations:

recursive functions to identify corner cells, taboo wall cells between corner

check whether the taboo cell is inside the warehouse

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

# 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 heuristics

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.

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

assign an individual pushing cost to each box, whereas for the classical Sokoban, we simply count the number of actions executed.

## 2.2 Validation

# Performance and Limitations