IFN 680 Artificial Intelligence and Machine Learning

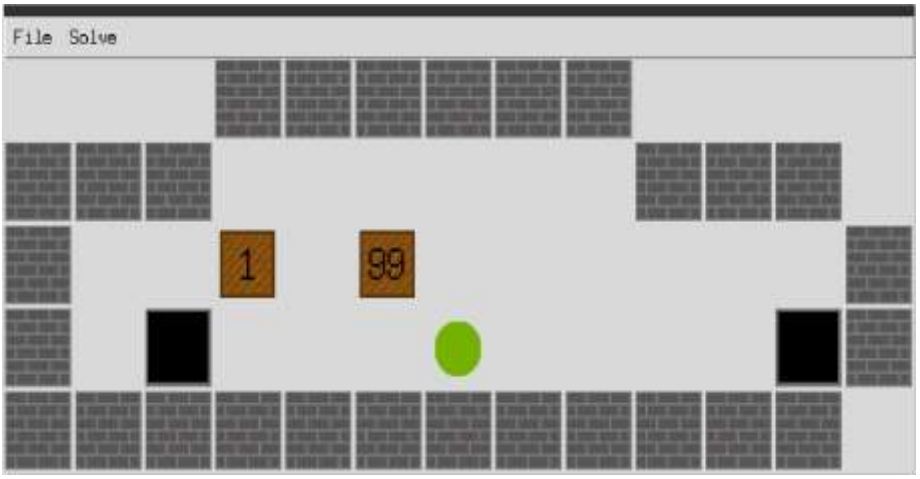
Assignment 1 – Sokoban Report

19/09/2021

Name	ID
Zhi On Tan	N9641556
Pallav Chamoli	N10635700

Introduction

The Sokoban puzzle is game based on payer/worker pushing boxes to goal location. It consists of walls, boxes , player and goal location. The system provide authority to push the boxes around the puzzle to place them correctly in destinated location. The only available directions are allowed are. Right, Left, Up, Down. Despite the fact, the Sokoban puzzle modelled as a computer game. It is considered to be an NP-hard approach. It is related to an automatic planning problem for robots using AI techniques. Below is one of the layout level of sokoban game.



Objective

The task is to construct an agent / solver capable of solving the puzzle with different levels which can be achieved through search algorithms with the heuristics implementation. The expected solution to solve this puzzle could be done with single and macro move

Puzzle representation / important features

- The puzzle layout to understood the puzzle , below is the provided design for the puzzles. Which consist of **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. When a text file contain weights , they would be mention in the first line of the warehouse text file. Otherwise consider zero (0).

```
        #   #   #   #
#   #   #           #   #   #   #
#                       $       #
#       #               #   $   #
#       .       .       #   @   #
#   #   #   #   #   #   #   #   #
```

Heuristic Search

The search algorithm used is the A* heuristic search. The A* search is simply based on a BFS method which results in sum of the path cost and heuristic function. For the Sokoban puzzle, the heuristic which is used is “Manhattan distance” between boxes and target to calculate shortest path

The Manhattan distance simply calculated as the sum of the difference between the initial and final x and y position of the object .

Distance = abs(x1-x2) + abs(y1-y2)

Using above formula of Manhattan we have given our solver to determine which object is close to the goal and results in the shortest path for the problem can be found. In BFS the nodes are explored as a FIFO queue or a Priority queue.

Testing Methodology

For the purpose of testing the code, we have been provided with a python file named sanityCode.py which test our method/code to process and design the solver solution to approach the different level of the Sokoban puzzle. Now validating the code the answer , for that purpose we formulated actions and calculating the deadlocks or situations and actions that can make the puzzle unsolvable. The proximity of the puzzle becoming unsolvable if the worker pushed the box in the taboo cells. Taboo cells are corners of the puzzles other than goal/ target or cells between two corners along the wall. Once we determined the our next step ahead , we will store it in the list and fetch the nodes that can be used with our heuristics search algorithm.

Performance and Limitations

As per the performance of the A*search the total computational time much quicker than other search algorithm due to its heuristic nature. Because A* follows the BFS algorithm , it would not be effective if the goal is far away or cover a long path. If we add more boxes to the puzzle the computational time increases. But as the puzzle gets more complicated by adding more boxes, the larger the Sokoban puzzle gets the more chances it will be for the weakness to be exposed in A* algorithm. for final conclusion , we have seen is A* is fast and effective for low complexity level puzzle with less boxes and small warehouse., however with the increasing path complexity the BFS(Breath first search) algorithm will expose weakness it would be much better to use DFS (Depth first search) for more complex levels.