



PROJECT ANTON

A MEMORY EFFICIENT SOLVER FOR SOKOBAN

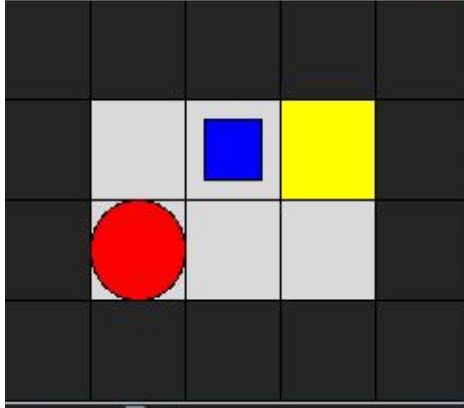
Venkatesh Kotwade (B18CSE023)

Project Report for Artificial Intelligence Course (CS314) by Dr. Yashaswi Verma



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

General rules of the game



1. The Robo can not pass through walls.
2. The boxes can not be pushed through walls.
3. The Robo can only push the box. (never pull)
4. There is no such concept of scoring in the game, as long as you are able to push each of the boxes into one of the goal positions, it is considered as a win.
5. Two boxes can not occupy the same position.
6. The Robo and a box can not be in the same position.

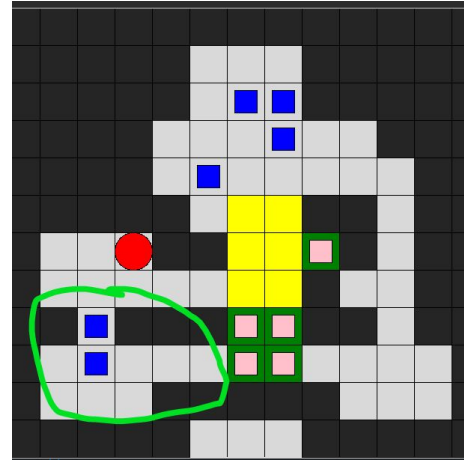
Core Difficulties of Problem

Size of state space


$$S = \alpha \cdot x^A \cdot P_K$$

A: Playable area in unit squares
K: Number of boxes
 α : Accessibility factor [less than 1]
S: Size of Search Space

Dead Locks




Problem background and Description

- 
- Known to be NP-hard and PSPACE COMPLETE
 - Level Environment is a 2D matrix
 - Positions of goals or holes will be common for all states
 - GoalTest pass if all goals have a box placed on them

- Most of the recent advancements use some form of ML to learn about difficult patterns which are prone to deadlock.
- Use Domain Specific Knowledge Database

We will solve it as a pure search problem

Iteration over choosing Search Algorithm

- 
- DFS , BFS - fail due to search space explosion
 - Greedy/UCS - mislead search in wrong direction / number of moves is not important
 - Iterative A* [IDA*] - fails due to higher depth solutions and loosely bound heuristics

Finalised on A*

- Can be generalised over large types of levels
- Modular for different change in strategies

Iteration over choice of Heuristic

1.Sum of Closest Manhattan Distance to any Hole

Naive, Consistent, Inefficient

$$O(|B|^2)$$

2.Minimum Weight Perfect Bipartite Matching among Box Set and Holes Set using Manhattan Distance as Weight of Edge

Novel, has both consistent and overestimating versions, moderately efficient

$$O(|B|^3 \times \beta)$$

Used hashing to avoid redundant calculations

β is generally tending to 10^{-3}

```
long long minBipartiteWithTopologicallyClosestHoleHeuristic(ProblemState& state, Problem & p)
{
    /*
     unordered_map which keep track if the answer was calculated previously
     */
    static std::unordered_map<unsigned long long, long long> memoisedAnswer;

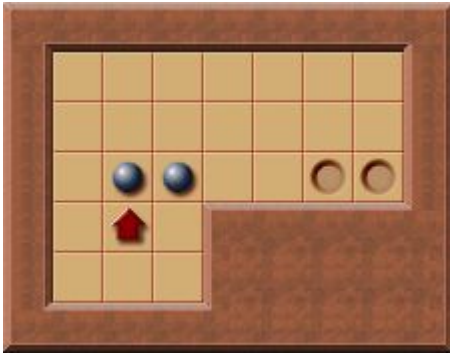
    unsigned long long hashOfSet= hashSetOfPairs(state.bboxes);
    if(memoisedAnswer.count(hashOfSet)) return (memoisedAnswer[hashOfSet]);
```

Iteration over choice of Heuristic

3. Minimum Weight Perfect Bipartite Matching among Box Set and Holes Set using Exact Constrained Distance as Weight of Edge

Novel, has both consistent and overestimating versions, moderately efficient for high depth solutions

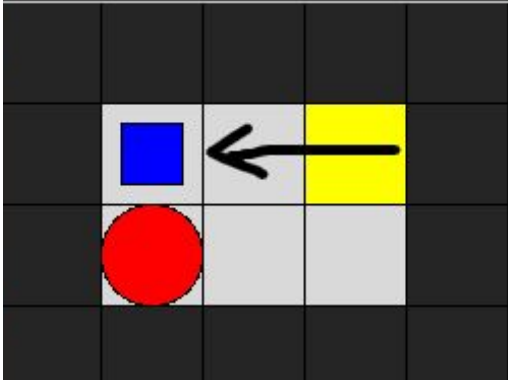
$$O(|B|^2 \times d \times \beta)$$



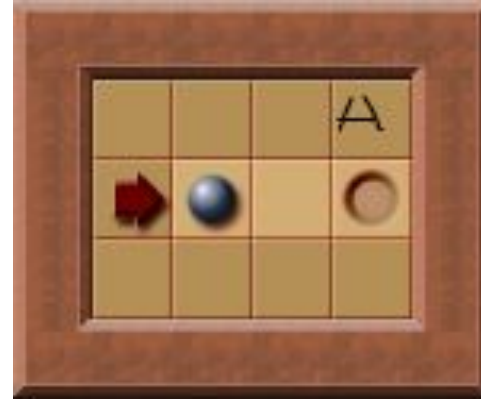
```
for(int i=0; i<4; i++)  
{  
    int nx= x+dx[i];  
    int ny= y+dy[i];  
  
    if(vis.count(nx+(ny<<20))) continue;  
    if(P.level.isWall({nx,ny})) continue;  
    if(state.hasBoxAt({nx,ny})) bfsq.insert({c+6,{nx,ny}}); //+6 because of linear conflict  
    else bfsq.insert({c+1,{nx,ny}});  
}  
std::cerr<<"Control reached invalid end"<<std::endl;
```

Uses linear conflict to reach tighter bound to for optimal heuristic

Types of Deadlocks



First order static

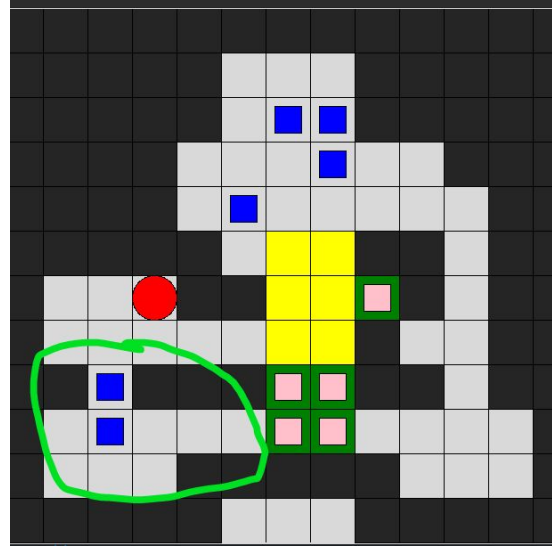


Second-order Static

Types of Deadlocks



First order dynamic



Second Order Dynamic

Algorithm and Results




Used A* with Minimum Weight Perfect Bipartite Matching among Box Set and Holes Set using Exact Constrained Distance as Weight of Edge Heuristic

Strict dead-end checks resulted in an overall reduction of 30% in the run time of the algorithm. It also resulted in a reduction of 63% in the number of nodes expanded.

With strong Deadlock checks
Average time : 12 second
Passed : 149/155
Time Out: 4/155
Unexpected: 2

With weak Deadlock checks
Average time : 8 second
Passed : 141/155
Time Out: 14/155

Theoretical Time Complexity


$$O(|B|^2 \times \bar{d} \times \beta \times (b^*)^m)$$

In our heuristic the b^* reaches close to 1 and m can reach up to 65.

b^* : resultant branching factor

\bar{d} : the average of dimensions of the levels $(h+w)/2$

β : it is the inverse of the average number of repetitions of set B over all instances of Heuristic Function call. It is always less than 1 and tends to 10^{-4} .

$|B|$: a set of boxes

Conclusions



- Any progress further will require pattern detection which can not be implemented programmatically , instead need heavily trained models and domain specific knowledge database.
- Can be combined with most of the human solvable Sokoban Level to provide hints or additional help to players.

Further Possible improvements



- The unexpected result of two sokoban levels was interesting for further investigation
Possible reason
 - 1) Hash collisions - less probable 10^{-11} chance of hash collision
 - 2) False Positive by dead end checker - more probable



Thank you