Project part – A (searching) Report

We modeled our agent using a model-based reflex agent, where it commits to one move and checks how the world evolved due to that move and checks what it can do now in the scope of the evolved world.

Some of the characteristics of the game environment we used
The environment is static
The environment is discrete
There is only one agent
The environment is sequential
The environment is deterministic
The environment is fully observable

STATE/ACTION/GOAL TEST/PATH COST:

- State: board placement
- Actions: moving towards a goal, booming a goal, making a stack, breaking a stack
- Goal test: if the board is cleared
- Path costs: number of moves required to reach the goal

Search strategy
We implemented the strategy of finding the points needed for a boom to happen for all blacks to be destroyed such that there is a win for white. After finding these points we allocated a white piece for each goal point.

Algorithms
The algorithm we used for the search was the A-star algorithm. Given the small size of the board, the algorithms O(bd) space complexity (due to storing all expanded nodes) isn't an issue.
Some other facts we considered –

1. It is complete
2. It is optimal
3. O(|E|) = O(b^d) time complexity

The heuristic we used was the Manhattan distance approach. We used this as it is admissible for this problem.

We also used breadth-first search when checking for certain cases.

When searching for the optimum boom locations we used brute force checking all points on the board without a black. This has a worse case space complexity of only O(V) which is not an issue due to the size of the board.

Features and assumptions of the problem

Given the specification that there will be a maximum of three white and twelve black pieces and the assumption that every initial board state will have a solution, there will only be a maximum of three possible goals (boom) locations. Furthermore, the fact that there is a maximum of twelve black pieces on a 8 x 8 board reduces the possibility of extremely computationally difficult cases significantly. The number of black pieces in the input file is not as important as the number of clusters of blacks (can only be a maximum of three).
The need to make stacks increases the space and time complexity of the problem. Furthermore, stacks cause the branching factor at a certain node to increase linearly and given maximum stack size for this problem is only 3 the effect of this is minimized.

Our Approach

The first objective of our search program is to assign goal states for each white piece needed to obtain a winning situation. Then we used BFS to search if each white piece needs stacks to get to the goal (i.e. no direct path). If there is no direct path A star search is used to make a stack with an adjacent white piece. Once no stacks are needed we use A star search but only making one move at a time and checks to see how the move affects the scope of the rest of the board. This is similar to a model-based reflex agent. Therefore, after every move stacks are evaluated to see if they should be de-stacked or not. This approach is repeated until all white pieces are at the goal states they were allocated.

Drawbacks

Due to the approach, we have taken there are certain complex cases that do not find a solution. For example, if a boom is needed to occur in order for a white piece to get to its goal this approach would fail.
This approach also provides a far from optimal solution for certain complex cases due to the mechanism of making stacks.