**COMP30024 Artificial Intelligence**

**Project 1 Report**

1. **search problem formulation**

*State*: player and obstacle pieces' location on board

*Action*: player can move, jump or exit one player piece per turn defined in specification

*Goal Test*: no player's piece on board

*Path Cost*: 1 cost per action

1. **Search Algorithms**

**Terminology:**

*b*: branching factor for search tree

*d*: length for the solution path in search tree

*δ*: relative error in heuristic = |h\*(s) - h(s)|

*h()*: heuristic function

**A\* search**

This is the search algorithm used in our program. It's a simple but efficient search algorithm. Not only is it complete and optimal, but also optimally efficient, meaning among similar algorithms (ones that expands paths using heuristic as a guide), which use the same heuristic, A\* expands the least (or as least as others) number of nodes (states in this project).

*Time Complexity:*

best case ∈ O(d) if we disregard the complexity of the heuristic calculation

average case ∈ O(bδd) (from lecture)

worst case ∈ O(bd) (because it is uniform cost search now)

*Space Complexity* ∈ O(bδd) (because "keep all nodes in memory")

*Completeness:*

A\* search is guaranteed to find a solution if one exists. As it is guaranteed in the specification that there is at least one solution, A\* search is complete in the project.

*Optimality:*

Yes, as long as h(s) ≤ h\**(s)* ∀ *s* ∈ *state space*

*(A* search is optimal if the heuristic is admissible (required in tree search) and consistent (required in

graph search))

1. **Heuristic Function**

*Admissibility:*

1. **Run time and Space Impact**

*branching factor*

*depth of search tree*

*number of bocks*

*number of player pieces*

*relative error*