COMP3702/COMP7702 Artificial Intelligence (Semester 2, 2020) Assignment 1: Search in LASERTANK – **Report Template**

Name: Anshuman Mander

Student ID: 45123209

Student email: a.mander@uqconnect.edu.au

Note: Please edit the name, student ID number and student email to reflect your identity and **do not modify the design or the layout in the assignment template**, including changing the paging.

Question 1

- **Modularity: Flat and discrete**
  Since there's only one problem to solve (i.e. reach goal flag), the modularity is flat. Because of change of states after moves (changing grid due to different interactions, like water converting to land), the state is discrete.
- **Planning horizon: Indefinite stage**
  Since the agent doesn't stop till goal is reached, there's no predetermined number of steps. Also, the agent doesn't go on forever (if goal isn't reached) as it stops after every state is visited.
- **Representation: States**
  States (grid layout) are used as depiction of the world.
- **Computational limits: Perfect rationality**
  The agent designed looks for solution without considering computational resources therefore there's no bounded limits.
- **Learning: Knowledge is given**
  Since all data about game rules and interactions is already present, the agent doesn't need to learn anything new.
- **Sensing uncertainty: Fully observable**
  The world is fully observable to agent through LaserTankMap class, which allows agent to explore every element on map.
- **Effect uncertainty: Deterministic**
  The transition from an action to state is clearly defined with no uncertainty.
- **Preference: Goals**
  There's only one goal to reach the flag symbol without any other matters at hand.
- **Number of agents: Single agent**
  Only one agent working i.e. the A* search working towards solving.
- **Interaction: Offline**
  States are reached and reasoned with before defining the solution.

Question 2

Components of agent design →

• **Action Space (A):**

The agent can perform four actions on the player.

- Move the player forward (‘f’)
- Rotate player clockwise (‘r’)
- Rotate player counterclockwise (‘l’)
- Shoot player’s laser (‘s’)

• **Percept Space (P):**

Since the world is fully observable and deterministic with a single agent acting, Percept Space is the same as State Space.

• **State Space (S):**

State Space is the permutations of all possible positions of player on the map and permutations of all possible “positions” after player’s interaction with world elements. As an example, the permutations of player positions after player’s laser shot pushes bridge into water.

• **World Dynamics/Transition Function (T : S × A → S 0):**

Different interactions of agent has different effects on world. Changing position by moving forward, rotating clockwise and counterclockwise changes the player positions whereas other interactions (such as water turning to land) changes the grid structure. Combined, an action by agent changes the grid of LaserTankMap.

• **Perception Function (Z : S → P):**

Since the world is fully observable and deterministic with a single agent acting, the agent doesn’t need to consider Perception Function.

• **Utility Function (U : S → R):**

Cost is 0 where tank is at the flag and 1 at any other state.

Question 3

The following statistics produced when solving map t1_bridgeport.

Nodes generated: UCS – 76117, A* - 67165

Nodes on the fringe when the search terminates: UCS – 395, A* - 494

Nodes on the explored list when the search terminates: UCS – 19425, A* - 17286

Run time of the algorithm: UCS - 11.53 seconds, A* - 9.03 seconds

As can be seen, implementing A* search produced significantly lesser node (around 10,000) although the nodes on fringe at search's end was higher (almost 100). The reason for this might be the A* implementation allowing faster solution and not needing to explore expensive nodes. The estimated goal to cost allows A* search to prioritise better paths. It can also be seen that UCS search explored 2000 more nodes as compared to A*. Overall, less nodes generated and explored allows A* search to run faster and save 2 seconds off algorithm's runtime. A* better prioritisation gives it an edge when choosing the edges to generate and explore.

Question 4

The following heuristics uses the common knowledge of LaserTank environment and works towards making an effective A* search.

Heuristic 1: Minimum of Teleport Distance & Manhattan Distance

```
def heuristic():
        return min(manhattan_distance, dist_between_teleport_and_goal +
        dist_between_player_and_teleport)
```

Teleport tiles significantly reduces the cost of getting to goal. Using their cost as heuristic allows player to find shortcuts towards goal and works as effective heuristic. Additionally, Manhattan distance is used to see if the teleport's closer or the goal. Distance between player and teleport + teleport and goal is calculated collectively to ensure that goal isn't closer to player normally. This ensures that always shortest path is taken to goal.

Heuristic 2: Ice Cells and Manhattan Distance

```
Def heuristic():
        return min(manhattan_distance, dist_between_ice_and_goal +
        distance_between_ice_and_player)
```

Ice tiles allows player to slide and skip over tiles. Using the above heuristic, the player would be able to take path with ice tiles if cheaper and save cost to reach goal. As previously, minimum of manhattan and ice tile is taken to ensure that it isn't cheaper to reach goal directly.

Heuristic 3: Water Cells and Manhattan Distance

```
Def heuristic():
        return  manhattan_distance - distance_from_water_to_goal
```

As water cells blocks the path and adds to cost (re-routing), distance of closest water cell will increase the cost of path. Therefore, to avoid running into water cells, their distance must be included to ensure that path taken has water cells further away from goal.