COMP3702 Artificial Intelligence (Semester 2, 2025)

Assignment 1: Search in CheeseHunter - Report

Name: Ngo Thuy Anh Tang

Student ID: 48040811

Student email: anh.tang@student.uq.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** (Complete your full answer to Question 1 on the remainder page 1)

|  |  |  |
| --- | --- | --- |
| **Dimension** | **Value** | **Justification** |
| Planning horizon | Indefinite stage | Game length varies depending on player's actions -> the horizon is finite but unknown. The game does not know how many actions the player will take until they activate all the levers and reach the cheese. |
| Representation | Features | States are described and approximated using features, as each state in the game is represented not just by the player's position, but also by trap statuses and the cost accumulated so far. These compact features capture many combinations, avoiding explicit enumeration of all states. |
| Computational limits | Perfect rationality | Given complete knowledge about the game (action costs, positions of levers and traps, current player position, and accumulated cost), with unlimited time, unlimited steps, and unlimited computational resources, a pathfinding algorithm can always generate the best sequence of actions that achieves the goal at minimal cost. |
| Learning | Knowledge is given | The agent is provided with knowledge about the game state, trap status, action costs, player position, and full schematic data of the environment. |
| Sensing uncertainty | Fully observable | The set of states and features is finite, and the agent knows the exact state of the game environment. |
| Effect uncertainty | Deterministic | The outcomes (player position, accumulated action cost, reaching the goal) are fully determined by the chosen action, prior cost, and the current status/positions of levers and traps. |
| Number of agents | Single agent | There are no other agents in the environment. Only one agent decides the sequence of actions, and no other entities influence the game state. |
| Interactivity | Offline reasoning | The agent uses given knowledge to compute solutions for all possible states it could encounter, then carries out the precomputed plan. |

**Question 2** (Complete your full answer to Question 2 on page 2)

* Action Space is the set of all possible actions the agent can perform.
  + For CheeseHunter, it's A = {Walk Left, Walk Right, Jump, Climb, Sprint Left, Sprint Right, Drop, Activate}. However, not every action is valid in every state.
  + Code reference: game\_env.py, line 54, variable ACTIONS
* State Space is the set of all possible configurations of the world the agent is operating in
  + Since CheeseHunter is fully observable (as justified in Question 1), it's also the Percept Space - the set of all possible things the agent can perceive
  + In CheeseHunter, the state space consists of all possible combinations of the player’s positions (row, column) and the trap status. Not all permutations are valid: some traps are unreachable until certain traps are activated, or some locations may be impossible to reach depending on trap and lever layout in a level.
  + In my code, the GameState class also records additional variables: game environment (self.env), previous state (self.parent), previous action (self.action\_from\_parent), and cost to reach the current state (self.path\_cost). These are mainly used for reconstructing the full path and computing g(n) for frontier priority, they do not represent additional elements of the state space.
  + Code reference: game\_state.py,\_\_init\_\_(), line 19-27
* Transition Function is a function that specifies how the configuration of the world changes when the agent performs actions in it.
  + CheeseHunter's player position:
    - Changes according to valid actions (except ACTIVATE). Example:

, where (x, y) is the player’s row and column in GameState.

* + - Code reference: game\_env.py, in perform\_action(), line 390-409
  + CheeseHunter's GameState's trap status:
    - When the agent performs Activate on a lever, the player’s position does not change, but the trap status updates. This, in turn, changes the set of valid actions available at the locations of related traps.
    - Code reference: game\_state.py, get\_successors(), line 58; game\_env.py, perform\_action(), lines 411–436.
  + Invalid action will not be performed and change any game state or player position.
* Utility Function is a function that maps a state or a sequence of states to a real number, indicating how desirable it is for the agent to occupy that state/sequence of states.
  + In CheeseHunter, the utility is based on reaching all levers, activating them, and reaching the cheese with minimum cost:

as higher values are the more preferred states.

**Question 3** (Complete your full answer to Question 3 on page 3)

**Question 4** (Complete your full answer to Question 4 on pages 4 and 5, and keep page 5 blank if you do not need it)

**Appendix/References**