Artificial Intelligence

Omid Naeej Nejad
610301189
Assignment 1

1 Searching Algorithms

Solution

Edges:

$$S \to A(1),\ S \to B(2),\ A \to C(1),\ A \to D(1),\ B \to D(2),\ C \to D(2),\ C \to G(5),\ D \to G(5)$$

1. BFS (Breadth-First Search)

Layer expansion:

$$\{S\} \to \{A, B\} \to \{C, D\} \to \{G\}$$

Visited order:

$$S,\ A,\ B,\ C,\ D,\ G.$$

Returned path:

$$\boxed{S - A - C - G}$$

2. UCS (Uniform-Cost Search)

Node expansions:

$$S(0),\ A(1),\ B(2),\ C(2),\ D(2),\ D(4),\ G(7)$$

Returned path:

$$\boxed{S - A - C - G}$$

3. DFS (Depth-First Search)

Alphabetical DFS:

$$S \to A \to C \to D \to G$$

Visited order:

$$S,\ A,\ C,\ D,\ G$$

Returned path:

$$\boxed{S - A - C - D - G}$$

2 Heuristic Function

Solution

We denote by $h^*(n)$ the true (optimal) cost from node n to the goal G. From the graph we obtain:

$$\begin{aligned} h^*(G) &= 0,\\ h^*(E) &= 2,\\ h^*(F) &= 5,\\ h^*(D) &= 8,\\ h^*(C) &= 11,\\ h^*(B) &= 13,\\ h^*(A) &= 14. \end{aligned}$$

The given heuristic values are $H(A) = 10$, $H(C) = 9$, $H(D) = 7$, $H(E) = 1.5$, $H(F) = 4.5$, $H(G) = 0$, and $H(B)$ is unknown.

1. **Admissibility.**

   A heuristic is admissible iff $H(n) \le h^*(n)$ for all nodes n. All fixed values already satisfy this, so we only need

   $$H(B) \le h^*(B) = 13.$$

   (If we also require non-negative heuristics, then $0 \le H(B) \le 13$.)

2. **Consistency.**

   Consistency requires, for every edge (x, y) with cost $c(x, y)$,

   $$H(x) \le c(x, y) + H(y).$$

   Only edges involving B give constraints on $H(B)$.

   From edge $A \leftrightarrow B$ (cost 1):

   $$H(A) \le 1 + H(B) \ \Rightarrow\ 10 \le 1 + H(B) \Rightarrow H(B) \ge 9,$$

   $$H(B) \le 1 + H(A) \ \Rightarrow\ H(B) \le 11.$$

   From edge $B \leftrightarrow D$ (cost 5):

   $$H(D) \le 5 + H(B) \ \Rightarrow\ 7 \le 5 + H(B) \Rightarrow H(B) \ge 2,$$

   $$H(B) \le 5 + H(D) \ \Rightarrow\ H(B) \le 12.$$

   Combining all these inequalities gives

   $$9 \le H(B) \le 11.$$

   So H is consistent iff $H(B) \in [9, 11]$.

3. **A* expansion order $A \to C \to B \to D$.**

   In A* we use $f(n) = g(n) + H(n)$.

   After expanding A (with $g(A) = 0$), we have:

$$g(B) = 1, \quad f(B) = 1 + H(B), \qquad g(C) = 4, \quad f(C) = 4 + 9 = 13.$$

   For C to be expanded before B we need

$$f(C) < f(B) \quad \Rightarrow \quad 13 < 1 + H(B) \Rightarrow H(B) > 12.$$

   After expanding C and generating D from C:

$$g(D) = 4 + 3 = 7, \quad f(D) = 7 + 7 = 14.$$

   Now the open list contains B and D. For B to be expanded before D we need

$$f(B) \leq f(D) \quad \Rightarrow \quad 1 + H(B) \leq 14 \Rightarrow H(B) \leq 13.$$

   Combining both conditions:

$$12 < H(B) \leq 13.$$

   Therefore, A* expands the nodes in the order $A \to C \to B \to D$ exactly when

$$H(B) \in (12, 13].$$

3 Intelligent Agents, Multi Agent Systems

Solution

I PEAS for S1 (automatic wiper agent)

S1: an intelligent wiper controller that keeps the front windshield clear under varying rain conditions.

- Performance measure (P):
  - Good forward visibility for the driver (few water drops or streaks).
  - Safety: avoid situations where the driver cannot see because wiping is too slow, which potentially increases accident risk.
  - Comfort: avoid unnecessarily high speed, noise, or excessive wiping on a dry windshield to declining energy waste.
  - Efficiency and wear: minimise washer–fluid use and wiper wear while keeping visibility acceptable.
- Environment (E):
  - Outside weather: rain intensity, snow, fog, spray from other cars.

- State of windshield: amount and distribution of water/dirt, presence of ice.
- Vehicle state: speed, acceleration, turning, stopping.
- Driving Mode: In city, desert highways, mountain roads, night/day

- Actuators (A):
  - Wiper motor (off / intermittent / low / high speed, continuous or variable duty cycle).
  - Washer pump (spray washer fluid).
- Sensors (S):
  - Rain sensor on windshield (optical/infrared).
  - Camera looking at the windshield, light sensor, humidity sensor.
  - Vehicle sensors (speedometer, steering angle, etc).
  - Feedback from driver controls (manual override).

II Environment classification for S2 and S3

We classify the environments using: observable / partially observable, deterministic / stochastic, episodic / sequential, static / dynamic, discrete / continuous, single / multi–agent, known / unknown.

S2: autonomous driving on a highway with CAA and LKA

- Partially observable: sensors cannot see all hidden vehicles, driver intentions, road conditions behind hills, etc.
- Stochastic: behaviour of other drivers, sudden braking, lane changes, and small disturbances are not exactly predictable.
- Sequential: each action (accelerate, brake, steer) affects later states and future decisions.
- Dynamic: the world changes continuously while the agent is deliberating (other cars move, traffic lights change).
- Largely continuous: positions, velocities, distances, steering angles and time evolve on continuous scales, although some decisions (change lane / keep lane) are discrete.
- Multi–agent: many other vehicles and possibly infrastructure systems act independently with their own goals.
- Mostly known: physics, car dynamics and traffic rules are known; the exact behaviour model of each driver is uncertain, but we usually assume a known model class learned in advance.

S3: Chess (assume rules are fully known), both with a clock and without a clock

- Fully observable: The entire board position is visible; there is no hidden information.
- Deterministic: Next state is uniquely determined by the current board and chosen move. There is no randomness, assuming the opponent is a master.
- Sequential: Each move affects all future possibilities; planning requires looking ahead.
- Static vs Dynamic:
  - **Without a clock:** the environment is **static**. Nothing changes while the agent is thinking; the board remains the same.
  - **With a clock:** the environment is **dynamic** because time continues to shrink even if the agent does not move. The state (time remaining) changes during deliberation.
- Discrete: A finite number of board states and a finite set of legal moves.
- Multi–agent: Two players (opponents) act alternately; the opponent is an agent with its own goals.
- Known: Rules, transitions, and legal move generation are fully known to the agent.

III Choosing an appropriate agent type for CAA and LKA in S2

We consider the standard types: simple reflex, model–based reflex, goal–based, utility–based, learning agent.

CAA: Collision avoidance must reason about different possible future motions of surrounding vehicles, distances, and relative speeds, and then choose actions that maximise safety while keeping driving reasonably smooth.
A utility–based agent is most appropriate:

- It can define a utility function that trades off safety, comfort and speed (very high penalty for any collision, smaller penalties for hard braking or large deviations from the desired speed).
- It can evaluate alternative actions (brake gently, brake hard, change lane, keep speed) by estimating their expected utility given the current state and a model of other vehicles.

So CAA is best modelled as a utility–based agent.

LKA: Lane keeping mainly depends on the current lane markings and the current lateral position and orientation of the car. The correct steering correction at each instant can often be computed from the current percepts alone:

- If the car is too far left, steer slightly right;
- If the lane curves right, apply a right steering angle following a control law.

Thus a simple reflex agent with a small internal state is appropriate:
Condition–action rules (or a controller) directly map visual lane features and current offset to a steering command. No high level goal reasoning is required beyond "stay centred in the current lane". Therefore, LKA is naturally implemented as a (simple or slightly model–based) reflex agent.

IV Designing a learning agent for improving LKA

We design a learning agent whose task is to improve lane keeping over time (e.g., smoother steering, better behaviour on curves, rain, worn lane markings).
The standard components are: performance element, learning element, critic, and problem generator.

- Performance element:
  - Inputs (percepts): camera image or lane–feature vector (lane centre line, curvature, distance to left/right markings), vehicle speed, yaw rate, steering angle.
  - Outputs (actions): steering commands and possibly small braking/acceleration adjustments for stabilisation.
  - Internally this might be a controller with adjustable gains or a neural network that maps features to steering angle.
- Critic:
  - Observes the actual driving trajectory and computes a performance signal.
  - Example metrics: average squared lateral offset from lane centre, number and size of lane departures, steering smoothness (penalise very frequent or large steering changes), passenger comfort.
  - Produces a scalar evaluation (reward/cost) for each episode or time window.
- Learning element:
  - Uses the feedback from the critic to adjust the parameters of the performance element.
  - For example, it may apply gradient–descent on controller parameters, train a neural network from logged data (supervised learning) or reinforcement learning using the critic's reward.

  - Its goal is to modify the mapping from percepts to actions to increase performance.

- Problem generator:
  - Suggests informative experiences that help learning, while keeping safety.
  - Examples: choose to practise on different road types (straight, curves, uphill/downhill), different speeds, different weather (day/night, rain), and purposefully explore slightly different steering strategies in a simulator or with strong safety constraints.
  - Generates training scenarios or triggers exploration so that the agent can discover weaknesses of its current policy and improve them.

In summary, the learning agent repeatedly observes driving data, the critic evaluates how well the car stayed in the lane, and the learning element updates the LKA controller parameters so that future steering commands keep the vehicle closer to the lane centre under a wide variety of conditions.

V Why use a MAS for S4 (travel planning)?

S4: a travel–planning system that considers multiple flights, routes, hotels and user preferences, and returns the best itinerary.
Using a multi–agent system (MAS) instead of a single monolithic agent has several advantages.

Advantages

- Specialisation and modularity:
  - Each agent can handle a specific subtask: a flight–search agent for airlines, a hotel–booking agent, a ground–transport agent, a pricing/budget agent, and a user–preference agent.
  - This makes the system easier to design, maintain, and extend (e.g., adding a new agent for local events or visas).
- Parallelism, scalability and robustness:
  - Different agents can query different providers (airlines, hotel sites, train companies) in parallel, reducing response time.
  - If one agent or data source fails, others can still function, so the overall system is more robust.
  - The system can scale by adding more agents for new regions or services without redesigning the whole planner.

One challenge of MAS

- Coordination and conflict resolution:
  - Agents may propose incompatible partial plans (for example, a flight that arrives after the last train, or a cheap hotel far from the selected airport).
  - The system needs explicit communication protocols, negotiation or a supervisor agent to combine their suggestions into a globally coherent itinerary that satisfies constraints and user preferences.
  - Designing such coordination mechanisms (who has authority, how to resolve conflicts, how to share information) is non–trivial and is a main challenge of MAS architectures.