Mini-Project Final Report

Team members:
- Wilson Thurman Teng (e0697830)
- Zhuang Xinjie (e0202855)

Problem definition:
- Large stochastic environment
- 50 ✕ 10 dimension (10 lanes with each having a width of 50)
- 6 to 8 cars in each lane
- Speed ranges from 1 to 3 cells per time step

Environment:

Most of the state gives reward 0 (except goal with reward 1). This presents us with a sparse reward environment problem. The agent has limited information on the right direction until the goal is reached. Therefore, the agent's exploration is inefficient.

As a result, we want to make use of the known environment indicators such as the goal's position, agent's position and board dimensions to allow for more informative rewards. This results in our agent realising better states more quickly.

Agent implementation:

The function approximation approach was used by implementing the Deep Q-learning agent. This allowed us to scale our solution to handle the dimension requirement for the mini-project. However, some optimization and modification were required to tackle the sparse reward problem.

Optimizations:

Reward shaping

To tackle the sparse reward problem, we shape the reward function with Manhattan distance between agent and the goal.

[1] Raw distance as reward

```
abs(agent_x - goal_x) + abs(agent_y - goal_y)
```

[2] Normalised distance as reward
- Raw distance in [1] normalised with fine-tuned scale

[3] Increase in penalty for more time taken to reach the goal

```
Initial: time = 1
Transition: reward -= time, time += 1
```

[4] Amplified goal reward scale to compensate large negative distance reward

Weighted sampling

Instead of letting the agent explore transitions randomly, we want the agent to prioritise more rewarding states in order to search more effectively.

[1] Weighted sampling based on reward
- High-reward transitions have higher chance to be explored
- Exploration will be more targeted and guided

[2] Maintain ReplayBuffer as a priority queue
- Remove low reward states first when buffer is full
- Training is too slow

Results:

Implementations	Performance

[1] Negative Manhattan distance reward + AtariDQN + weighted sampling	`[t2_tmax50] 300 run(s) avg rewards : 1.4`
	`[t2_tmax40] 300 run(s) avg rewards : 1.0`
	`Point: 1.22`

[2] Extend from [1]: positive reward for being closer to goal + negative reward for being further to goal + penalty on time taken to reach goal	`[t2_tmax50] 300 run(s) avg rewards : 6.0`
	`[t2_tmax40] 300 run(s) avg rewards : 5.1`
	`Point: 5.55`

[3] Optimise from [2]: fine-tuned Manhattan distance reward scale + increasingly larger penalty on time taken to reach goal	`[t2_tmax50] 300 run(s) avg rewards : 7.3`
	`[t2_tmax40] 300 run(s) avg rewards : 6.9`
	`Point: 7.1 (6.816 on AiVLE)`

[4] Include penalty for crashed state	`[t2_tmax50] 300 run(s) avg rewards : 4.3`
	`[t2_tmax40] 300 run(s) avg rewards : 4.2`
	`Point: 4.25`

Conclusion:

By shaping the reward from given knowledge, the agent appears to know better what are good/bad states and become more effective in finding the goal.

From the experiment, we found that penalising on the time taken to reach the goal gives a huge boost on the performance. Also, it is important to note that the scale of both distance reward and time penalty need to be fine-tuned against the goal reward to reach an optimal performance.

Weighted sampling, though theoretically makes sampling much more efficient, provides relatively smaller improvement in agent's behaviour. We have also considered penalising the crashed state. However, it gives an unexpectedly worse score, hence we excluded it from the model.