# AI Final Project

## Introduction

After the three of us moved to Tel Aviv, we discovered that one of the most frustrating aspects of daily life here is the constant struggle to find a parking spot near home .This issue is not just a minor inconvenience; it represents a significant problem for residents, leading to wasted time, increased stress, and even environmental concerns due to the additional fuel consumption and emissions from circling the streets in search of parking. Addressing this problem is not only interesting but also crucial for improving the quality of life in urban areas.

The challenge we are facing is to develop a systematic approach to finding the optimal parking spot. Optimality is determined by the proximity to the desired destination, typically one's home, and time spent on searching for a parking spot. To solve this problem, we propose a model that treats the urban environment as a grid-based matrix representing roads, buildings, and parking spots. Each parking spot in the matrix is associated with a probability of being available, reflecting the uncertainty and variability of parking availability in a real-world scenario.

Given the probabilistic nature of parking availability, we believe that the most appropriate framework to model this problem is a Markov Decision Process (MDP). An MDP is a mathematical model that captures the decision-making process in situations where outcomes are partly random and partly under the control of a decision-maker. By modeling the parking problem as an MDP, we can effectively capture the underlying dynamics of the urban environment, including the stochastic nature of parking spot availability and the sequential decision-making process involved in searching for parking.

To better capture the real-world scenario, where we lack precise knowledge of the probabilities for each parking spot, we proposed using Reinforcement Learning (RL) to learn these underlying probabilities. RL is well-suited for this problem because it enables the model to learn the MDP dynamics without prior knowledge and to develop optimal strategies through experience. We like to picture this scenario as a person moving to a new city and gradually learning the best areas to search for parking spots.

Our baseline for comparison is a simple strategy: parking in the first available parking spot found by depth-first search (DFS) algorithm, without considering any probabilistic information. This baseline provides a straightforward, yet naive solution to the problem. We aim to demonstrate that MDP and RL approaches significantly outperform this simplistic strategy, by leveraging the knowledge of the parking environment to make more informed decisions.

In this work we used OpenAI's ChatGPT LLM model to brainstorm ideas and to implement some of the low-level code. In addition, as a code base, we used and adjusted some of the graphics implementation provided by the course staff on the different course exercises, and implementations of algorithms we wrote during the course as part of the exercises.

## Previous Work

The problem of optimizing parking space search has been studied with different algorithmic approaches proposed to address parking-allocation in both static and dynamic urban environments. These approaches focus on improving search efficiency, reducing traffic congestion, and minimizing environmental impacts.

Most prior work assumes smart environments with real-time parking-availability data collected via sensors, making the problem deterministic. In contrast, our work addresses a more complex scenario where parking-availability is stochastic, changing unpredictably, reflecting the reality in cities lacking smart-infrastructure. Another common assumption is the existence of traffic congestion to consider when choosing an optimal parking and path, unlike our solution which assumes equal cost of travel. In addition, most research focuses on multi-user parking optimization, while we center on single-user view, targeting individual parking search.

Different studies solved similar but slightly different problems with varied approaches.

One study implemented an improved version of Dijkstra’s algorithm to allocate parking in multistorey lots, optimizing computing resources[[1]](#endnote-1). It uses parking availability sensors to allocate the closest free spot to each arriving car and find the optimal path to that parking.

Several studies address city-wide parking allocation, focusing on reducing traffic congestion and balancing the load on parking lots by efficiently distributing vehicles. Solutions include IoT-driven systems[[2]](#endnote-2), hierarchical algorithms[[3]](#endnote-3), and multi-objective decision-making[[4]](#endnote-4).

Another study used Multi-Agent Systems combined with Genetic Algorithms, where agents communicate and negotiate to find optimal parking based on proximity, wait time, and costs[[5]](#endnote-5).

Two other studies used predictive modeling to forecast parking availability and optimize routes[[6]](#endnote-6)[[7]](#endnote-7). These models predict available spots based on current occupancy, arrival rates, and estimated travel times.

Across these studies, the primary performance benchmarks are consistent:

* **Efficiency and Search Time Reduction**: Minimizing time spent searching for parking.
* **User Satisfaction**: Proximity to destination, shorter wait times, and lower parking costs.
* **Computational Efficiency**: Reducing overhead for scalability, especially in structured environments.

Due to the differences mentioned above we struggled to benchmark our work directly against findings of past work.

## Methodology

### Modeling the Environment

To address the challenge of optimizing parking in an urban setting, we represent the city as a grid-based matrix. Each cell in this matrix corresponds to a specific element of the urban environment, allowing us to capture the complexity of city navigation and parking availability. The matrix is designed with the following possible cell types:

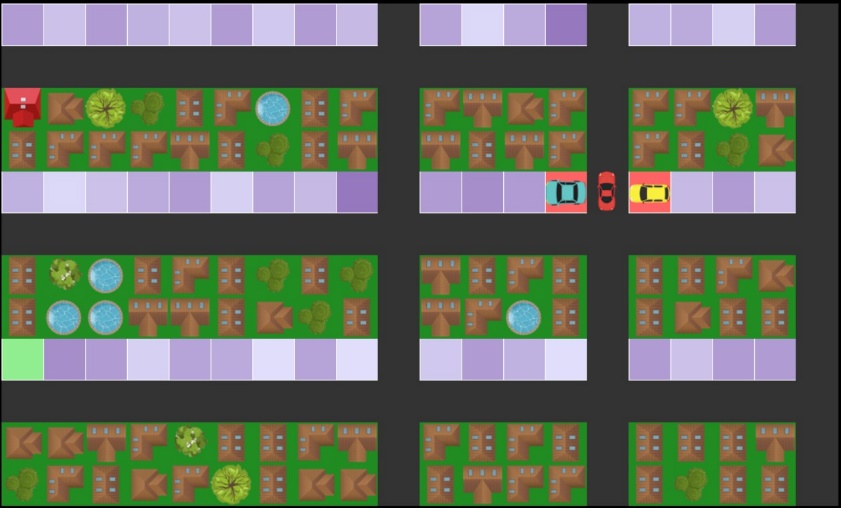


Figure 1 – General Map

1. Home: Represents the driver’s destination (Figure 1: red house).
2. Blocked: Cells that represent areas inaccessible to vehicles and pedestrians, such as buildings or other structures that are not roads (Figure 1: houses, trees, pools).
3. Road: Cells that denote paths navigable by vehicles or pedestrians (Figure 1: black block).
4. ParkingSpot:

Each parking spot has an underlying probability of it being available. At each step, the car observes the status of its four neighboring cells. If adjacent to a parking spot, the environment determines its availability based on the spot’s probability. When the probability of a parking spot is one or zero, we color it as available or occupied respectively, even it’s not adjacent to the car.

* Available: A parking spot that is currently free (Figure 1: green block).
* Occupied: A parking spot that is currently taken (Figure 1: red block with car).
* Unknown: A parking spot with uncertain availability, represented by a probability of being free (Figure 1: purple block), the higher probability the darker purple.

### Operational Model

The model operates in a sequence of states and actions, reflecting the decision-making process of a driver in search of parking:

* State: Defined by the car's position and the Boolean values indicating the availability of parking spots in the four adjacent cells.
  + Initial State: The car starts on a road cell at a designated starting point.
  + Goal State: The car reaches any available parking spot.
* Actions: The car can move to any adjacent cell (up, down, left, or right) that is either a road or an available parking spot.

### Assumptions underlie the Model

1. Static Map: The city layout, including the locations of blocked areas, roads, parking spots, and the home is constant.
2. Probabilistic Parking: We assume that parking spot availability is probabilistic rather than deterministic. Each observation of a parking spot is independent, and its availability may change with each observation according to its probability. This simulates real-world scenarios where a parking spot can become free or occupied at any time.
3. Limited Knowledge: The agent has a restricted view of its surroundings, knowing only the exact status of its current cell and the four adjacent cells. This mirrors a driver’s real-world experience, where they can fully see nearby spots, but have only a general awareness of the area’s layout.
4. Single Way Home: Once the agent finds a parking spot, the proximity to home is calculated using a Breadth-First Search (BFS) algorithm (as detailed in the following section). This represents the driver walking home from the parked location.
5. Cost of Travel – There are no traffic congestions, each step of travel across the grid has an equal cost or travel penalty.
6. Optimal Solution: The optimal solution maximizes the reward function, which is based on the proximity of the parking spot to home and the number of steps required to reach it (as detailed in the following section).

### Reward Function

The reward function balances the trade-off between the proximity of a parking spot to home and the number of steps taken to reach it.

The reward function is defined as:

* Discount Factor, equals 0.99. To ensure that the models can be compared effectively and are incentivized to explore rather than hastily conclude their search, the discount factor was used across all models.
* :

: Calculated using breadth-first search (BFS), navigating only through roads. This represents the shortest walking path from the parking spot to home. The closer the parking spot is to home, the greater the reward.

* : To discourage prolonged searches, each step the car takes results in a reward penalty of . By punishing on each step the car takes, the model balances the trade-off between proximity to the home and the time spent searching.

In this model, each step the car takes is valued 10 times lower than a pedestrian’s step, emphasizing that walking is slower and less desirable. This choice reflects the efficiency of driving compared to walking, encouraging the model to prioritize closer parking spots over the first available ones.

The reward is structured so that entering a parking spot yields a positive value, allowing the MDP and Q-learning algorithms to effectively learn.

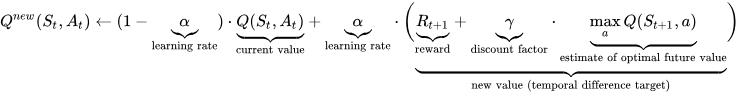
### Solutions

As mentioned in the introduction we choose three different approaches to solve the parking problem: baseline (DFS-like), MDP and RL.

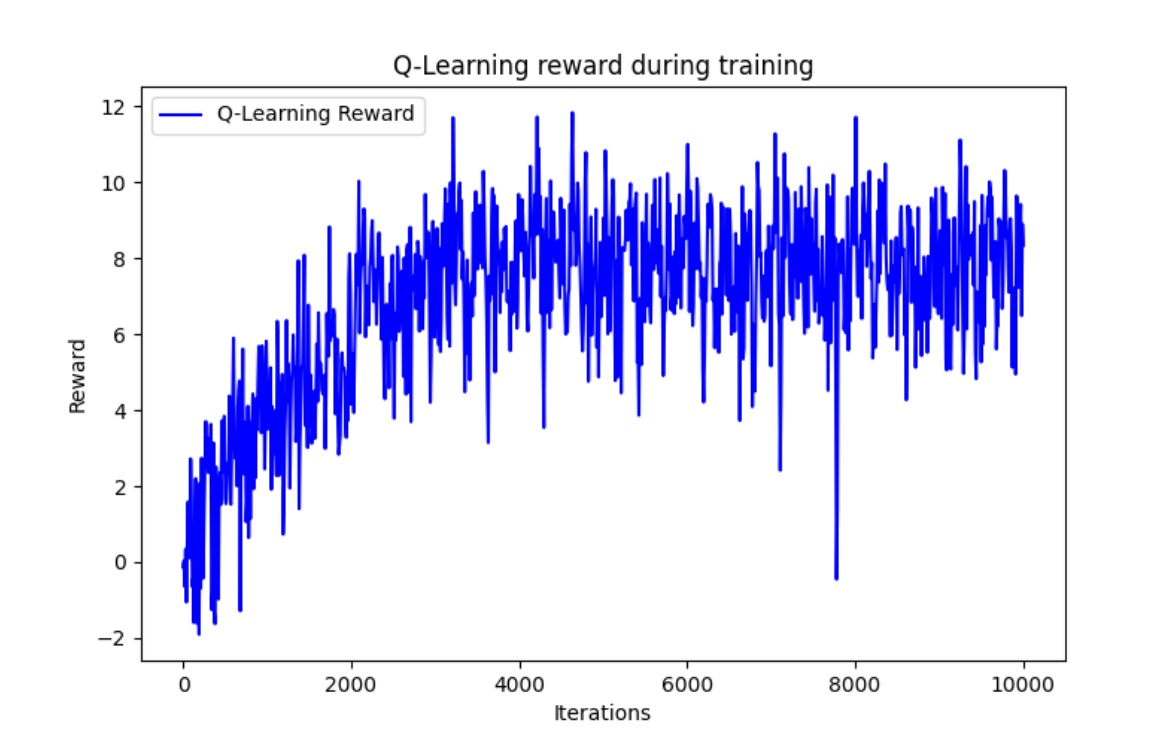
**Baseline Approach:** The baseline for our problem is a DFS-like solution that explores the city grid in a manner similar to running DFS on a graph. The car navigates through the map, parking in the first available spot it encounters. However, unlike a traditional DFS, if the agent encounters a dead-end with no further moves, it backtracks to its previous position. Notably, when backtracking, the car may encounter different states due to the dynamic nature of parking spot availability. Additionally, during this backtracking process, the reward for each action taken is accumulated, differing from the standard DFS algorithm, which does not account for rewards in this manner. When the car encounters multiple possible actions, it chooses one randomly. If the car visited the entire map without finding a parking spot it will start searching again from the current location, cleaning the visited list. The justification for continue the search is the dynamic nature of the parking spots availability – one will become free eventually.

**MDP Approach:** The MDP method enables the agent to consider the probabilistic nature of parking spot availability and make reward-based decisions. We implemented the MDP solution using value iteration, running it for 50 iterations.

**Reinforced Learning Approach:** In the real world, the availability of parking spots is uncertain, and learning where to find them takes time and experience. To create a more realistic model and effectively address this problem, we chose a reinforcement learning approach using Q-learning.

 Q-learning is a model-free technique, meaning it doesn't rely on knowing the environment's dynamics, such as the exact probabilities of parking spot availability. Instead, the agent learns an optimal policy through exploration and experience, making decisions based on the rewards it receives as it interacts with the environment.  
We trained the model for 10,000 iterations using the epsilon-greedy approach. This approach balances exploration and exploitation, allowing the agent to learn an optimal policy over time while still considering alternative paths. For training, we used a learning rate (α) value of 0.5 and epsilon (ϵ) value of 0.3 - meaning there is a 30% chance of exploring a random action over exploiting the current best-known action. Afterward, we tested the model under the same conditions as the previous algorithms, using the data it had collected, without the exploration component (ϵ = 0), and the learning rate.

Learning Curve of Q-Learning training:



This graph showcases the model's improvement in policy effectiveness with increased experience, reflecting its ability to adapt and optimize its strategy through ongoing learning.

All the algorithms used to determine the agent's policy always returns a valid policy according to the environment limitation. We'll discuss the soundness and completeness of the different algorithms in regard of the different policies each algorithm results.

**Soundness** – An algorithm is sound if, anytime it returns an answer, that answer is true. Each of the algorithms and agents will return an answer only if it is a valid parking spot, therefore all of them are sound.

**Completeness** – An algorithm is complete if it guarantees to return a correct answer for any arbitrary input. All the algorithms are not complete because in a map were all the parking spot have probability of zero, all the algorithms will run forever searching for one and will not return an answer of no parking spot available.

## Results

In this section, we present the findings from our experiments, comparing different approaches to solving the parking optimization problem. Our primary focuses were:

1. Evaluating the performance of the MDP compared to the baseline DFS.
2. Evaluating the performance of RL by comparing its results to those of the MDP, as RL aims to learn the MDP's dynamics.

### Evaluation Metrics

We measured the success of each model based on the following metrics:

1. Reward: The total reward accumulated by each model, factoring in both the proximity to home and the penalties for travel distance.
2. Travel Distance: The total number of steps the car took to find an available parking spot. This metric reflects the efficiency of each model in minimizing search time.
3. Proximity to Home: The distance from the chosen parking spot to home. This distance is calculated using BFS as explained in the previous section.

Extracting these two components from the reward help us to intuitively understand how the algorithms behave.

### Experiments

The experiments are structured in a step-by-step manner, starting with basic sanity checks to ensure that the algorithms behave as expected in straightforward scenarios. Throughout all experiments, we use the same map configuration shown in Figure 1. The variations across scenarios are based solely on the probabilities assigned to the parking spots, while the starting (right-bottom corner) and home (left-top corner) locations remain consistent throughout all the experiments.

All comparisons are made after the MDP has completed the value iterations and the Q-learning model has finished its training.

#### Sanity Checks

We begin by testing the algorithm in several simple scenarios to verify that it behaves as expected. In these initial tests, our expectation is that the MDP will outperform the Baseline approach. Ideally, the Q-learning algorithm should learn to replicate the policy of the MDP.

Furthermore, in deterministic map settings, we anticipate that both the MDP and Q-learning algorithms will exhibit a Reward standard deviation of zero. This is because, with deterministic dynamics, each action from a given position leads to a single, predictable outcome, resulting in consistent behavior without variability.

##### Scenario 1 - Single Available Parking Spot with Probability 1

###### Objective

Verify that the MDP, Q-Learning, and DFS algorithms can find the parking spot.

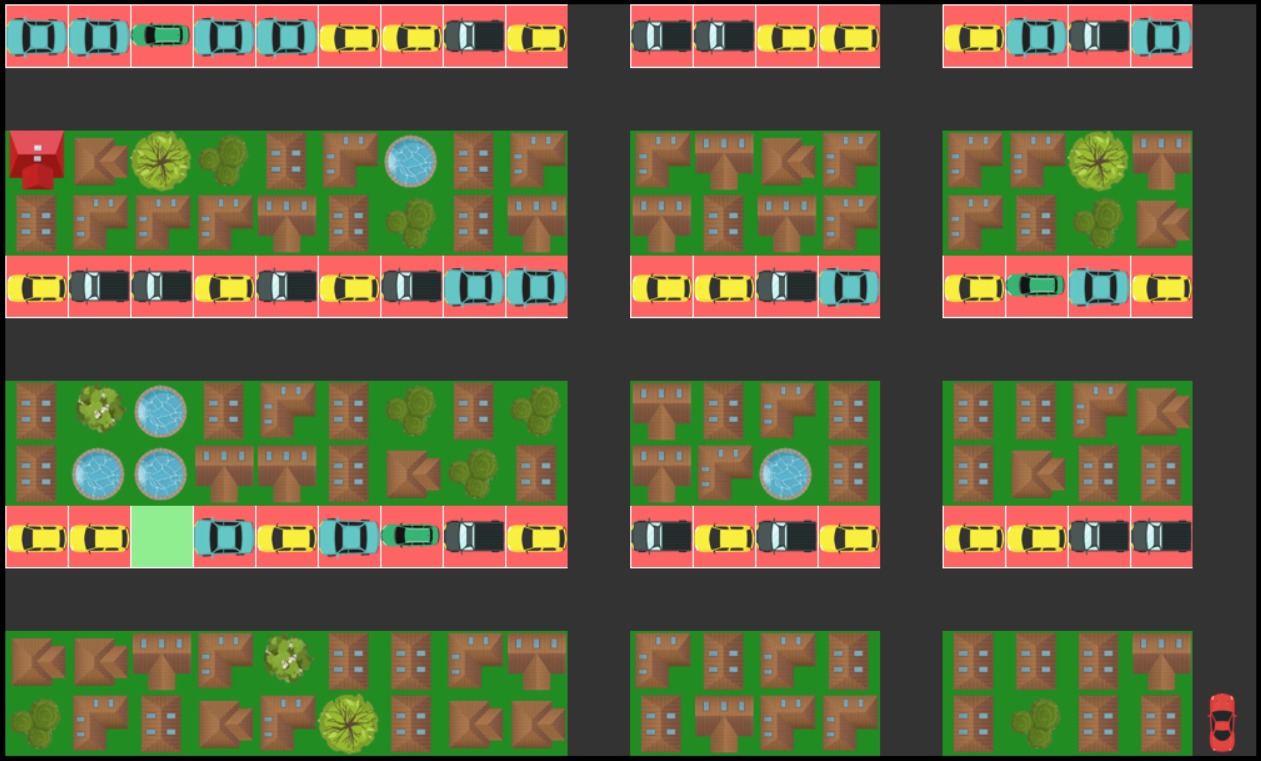


Figure 2 – Scenario 1, Single Available Parking Spot with Probability 1

###### Expectations

* All algorithms (MDP, Q-Learning, and DFS) should successfully navigate to the parking spot.
* MDP and Q-Learning are expected to have the same average reward, both higher than DFS.
* Given the simplicity of the scenario, Q-Learning should effectively learn the optimal policy identical to the MDP policy.
* In this deterministic environment, MDP and Q-Learning should show zero standard deviation, reflecting consistent behavior.

###### Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Average Reward** | **STD Reward** | **Average Travel Distance** |
| Baseline (DFS-like) | -4.99 | 2.1 | 80.88 |
| MDP | -0.19 | 0 | 21 |
| Q-Learning | -0.19 | 0 | 21 |

* **MDP and Q-Learning:** Both achieved the expected higher rewards, navigating directly to the parking spot with no variability in their paths (standard deviation = 0).
* **DFS:** While eventually it reached the parking spot, its random path selection sometimes led to longer routes, resulting in a lower average reward and a non-zero standard deviation.

##### Scenario 2 - Obvious Optimal Parking: Two Parking Spots with Probability 1

###### Objective

Evaluate algorithms performances when presented with two parking options: one near home and another near the starting point, both with certainty of availability. The optimal policy is going directly to the parking near home, we will call it optimal parking from now on.A screenshot of a video game

Description automatically generated

Expectations

* **MDP**: Expected to consistently park near the home, as it’s the optimal parking spot.
* **DFS**: Likely to park near the starting point most of the time due to its first-come, first-served approach, though randomness may occasionally lead it to the optimal spot.
* **Q-Learning**: Might initially be biased towards parking near the start due to immediate rewards, potentially overlooking the greater reward of parking near home. With sufficient training iterations, Q-Learning should ideally learn to favor the more optimal, further parking spot.
* Both MDP and Q-Learning should display a standard deviation of zero, given the deterministic map.

###### Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Average Reward** | **STD Reward** | **Average Travel Distance** | **% Optimal Parking** |
| Baseline (DFS-like) | 2.33 | 5.72 | 27.16 | 24 |
| MDP | 16.63 | 0 | 31 | 100 |
| Q-Learning | 0.46 | 0 | 6 | 0 |

* **MDP:** successfully identified the optimal parking spot near the home every time, as expected.
* **DFS:** mainly parked in the less optimal spot, near the start, but occasionally found the optimal spot by chance, leading to a high standard deviation and slightly better average reward compared to Q-Learning.
* **Q-Learning:** Unlike we expected, it consistently chose the non-optimal parking spot near the start, failing to adjust even after we tried to increase the training iterations by tenfold.

##### Scenario 3 - High Probability Optimal Parking: Two Parking Spots with Varying Probabilities

###### Objective

Assess whether MDP and Q-Learning can learn to prioritize the parking spot near home, which has a higher probability of availability (0.5) compared to the spot near the start (0.15). The goal is to determine if the algorithms can adopt a risk-taking strategy by aiming for the optimal parking with the best expected outcome. To check whether the algorithms adopt risk-taking strategies we will test them on a deterministic version of this map – where both parking spots are available.

A screenshot of a video game

Description automatically generated

Expectations

* **MDP:** Expected to consistently reach the optimal parking near home, recognizing the combination of higher reward and availability probability.
* **Q-Learning:** Anticipated to improve from the previous scenario by learning to target the optimal parking spot near home, avoiding the less rewarding spot near the start.
* **DFS:** Since the test map is identical to the previous scenario map and the DFS doesn’t take into account the probabilistic map, we expect the same result as the previous scenario.

###### Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Average Reward** | **STD Reward** | **Average Travel Distance** | **% Optimal Parking** |
| Baseline (DFS-like) | 1.91 | 5.94 | 30.63 | 23 |
| MDP | 16.63 | 0 | 31 | 100 |
| Q-Learning | 16.63 | 0 | 31 | 100 |

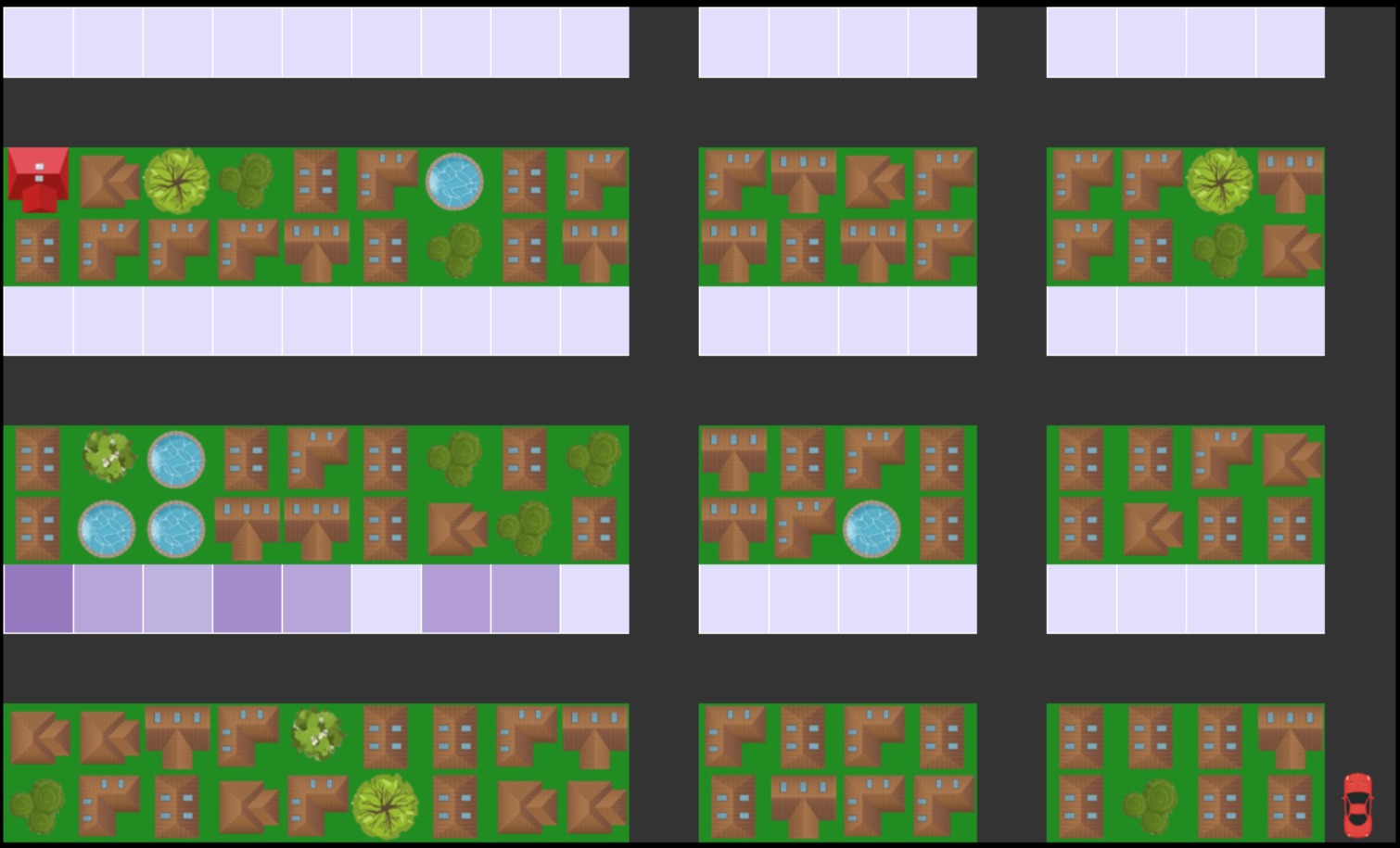
As anticipated, both MDP and Q-Learning consistently identified the optimal parking spot, achieving perfect performance in this deterministic evaluation. From the Optimal parking percentage, we can see that the DFS, as anticipated usually parks in the non-optimal parking spot which is closer to the stating position.

##### Scenario 4 - High Probability Parking Spots Far from Home and Start

###### Objective

Evaluate if MDP and Q-Learning can learn to prioritize optimal parking spots, even if they are far from home and the starting point.

The high-probability parking spots have probabilities ranging from 0.1 to 0.2, while all other spots have a probability of 0.01.



###### Expectations

* **MDP:** Expected to consistently identify and reach the optimal parking area by effectively balancing higher rewards and availability probabilities.
* **Q-Learning:** Anticipated to also consistently identify and reach the optimal parking area, as the reward structure will guide the learning process toward this outcome.
* **DFS:** Likely to park in the optimal parking area, as it will most likely encounter an available parking spot there first. However, it may take a longer route compared to MDP and Q-Learning before arriving to the area.

###### Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Average Reward** | **STD Reward** | **Average Proximity to Home** | **Average Travel Distance** |
| Baseline (DFS-like) | -0.48 | 5.27 | 20.15 | 76.57 |
| MDP | 3.29 | 1.07 | 21.47 | 22.99 |
| Q-Learning | 3.18 | 2.2 | 20.71 | 29.53 |

As expected, all the three consistently identified the optimal parking area. Additionally, the DFS preformed highest number of steps as we expected.

##### Scenario 5 - Low Probability Parking: All Spots with Low Availability

###### Objective

The optimal policy is to drive home and search for parking in that area. At some point, depending on the probability, the car should park if it encounters an available parking spot.

Expectations

* **MDP:** Expected to wonder near home, and opportunistically park if a spot becomes available on its way there, reflecting a cautious but reward-maximizing approach.
* **DFS:** Will continue searching until it finds the first available spot, potentially resulting in a quicker but suboptimal parking choice.
* **Q-Learning:** Likely to struggle, especially at very low probabilities, potentially circling near the start due to difficulty in learning the higher reward associated with parking near home.

###### Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Probability** | **Average Reward** | **STD Reward** | **Average Proximity to Home** | **Average Travel Distance** |
| Baseline (DFS-like) | 0.05 | 3.16 | 4.13 | 19.11 | 38.11 |
| 0.01 | -0.57 | 6.74 | 16.52 | 137.88 |
| MDP | 0.05 | 12.6 | 3.6 | 3.85 | 43.31 |
| 0.01 | 3.56 | 7.67 | 3.83 | 120.27 |
| Q-Learning | 0.05 | 12.51 | 3.51 | 3.84 | 43.88 |
| 0.01 | -1.28 | 5.47 | 17.56 | 124.12 |

When using probability of 0.05 both MDP and Q-Learning successfully reaching the optimal parking area near home, waiting there until a spot became available (marked in green).   
The DFS algorithm, on the other hand, parked as soon as a spot became available, leading to fewer steps on average.

When decreasing the probability to 0.01 MDP's average distance to home remained constant, but the number of steps increased as it waited for a nearby spot to become available, resulting in a lower reward (marked in blue). Q-Learning, on the other hand, waits next to a more distant spot, taking a similar number of steps but yielding a lower reward compared to MDP (marked in red).

Additionally, when observing the MDP run in the GUI, we noticed that with the 0.05 probability the car skips available parking spots next to the start, but with 0.01 probability it parks. . This behavior is also reflected in the standard deviation differences. In other words, the decision point shifted closer to the start, indicating a reduced willingness to take risks in extremely low-probability scenarios.

#### General Test: Analyzing Model Performance in Uncertain Environments

With confidence in our models, we aim to explore their behavior in complex scenarios where the optimal decision isn't always clear. This section examines how our algorithms perform in a general urban environment.

###### Experiment Setup

We tested our models on the street map with random parking spot probabilities ranging from 0 to 0.2 across all parking locations. This was repeated across 100 maps with different probabilities to evaluate the consistency of the results. On each map we preformed 100 iterations.

A screenshot of a video game

Description automatically generated

Figure 3, Map Example

###### Results Overview

As shown in the graph, the MDP consistently achieves the highest rewards. The DFS baseline generally produces the lowest reward, while Q-learning falls in between with noticeable fluctuations. At times, Q-learning matches the performance of the MDP, earning similar rewards, but in other instances, it performs worse than DFS or requires early termination after exceeding 1,000 steps (reflecting in missing parts of the graph).

The standard deviation for DFS is typically higher due to its inherent randomness, leading to more variability in performance. The proximity to home metric is generally higher for DFS, as it often parks in the first available spot, minimizing the distance traveled. This also explains why the number of steps is the lowest for DFS, as it doesn’t prioritize optimality beyond simply finding any available parking quickly. However, this approach results in lower overall rewards, demonstrating that while DFS is efficient in terms of travel time, it lacks strategic planning compared to MDP and Q-learning.

#### Summary of Model Performance and Insights –

1. MDP Model
   1. Consistently balanced minimizing travel distance with finding parking near home.
   2. Outperformed the DFS baseline in stochastic environments by leveraging probabilistic information.
   3. Achieved higher overall rewards and a closer average proximity to home compared to other models.
2. DFS Baseline
   1. Parked in the first available spot encountered, minimizing travel distance but often ending up far from home.
   2. Lack strategic depth needed for more complex environments, resulting in lower overall rewards.
3. Q-Learning (RL Approach)
   1. Showed significant improvement over time, learning to outperform the DFS baseline and often come close to the MDPs results.
   2. In some cases, the policy it learned in training made it go back and forth even if there was an available parking next to it, leading to an infinite loop. Therefore, this algorithm underperformed our expectations.

## Summary

This project focused on the problem of **parking optimization** in urban environments, particularly in densely populated cities like Tel Aviv, where drivers frequently struggle to find parking near their homes. We modeled the environment as a **grid-based matrix** representing roads, buildings, and parking spots. Parking availability was treated as **probabilistic** to mirror real-world uncertainties.

The primary challenge was to develop an approach to finding the optimal parking spot, defined by its proximity to home and the time spent searching. To address this, we implemented agents using **Depth-First Search (DFS)**, **Markov Decision Process (MDP)**, and **Q-Learning**. Each algorithm was evaluated based on a reward system that penalized long searches and rewarded parking spots closer to home.

### *Model and Results*

We modeled the environment using an **MDP**, allowing the agent to account for the stochastic nature of parking availability and make sequential decisions. The **MDP** enabled the agent to learn optimal decisions over time. To enhance the project’s learning capacity, we also used **Q-Learning**, a reinforcement learning technique that allowed the agent to learn optimal policies through exploration and experience, without knowing the exact environment dynamics.

The **DFS-like baseline** algorithm parked at the first available spot it encountered, leading to suboptimal results since it lacked optimization beyond immediate availability. This often resulted in parking spots far from home, yielding lower overall rewards.

Both the **MDP** and **Q-Learning** models significantly outperformed DFS. The MDP, using **value iteration**, consistently selected parking spots closer to home while minimizing travel time. **Q-Learning** also performed well, learning the environment’s dynamics over time and often producing results comparable to the MDP, though it occasionally underperformed when insufficient exploration led to suboptimal policies.

### *Critique of the Work*

Although the MDP and Q-Learning approaches outperform the baseline strategy, several assumptions and limitations should be further examined:

1. **Real-World Solution**: The solution gets a general simulation map as an input. A way to upgrade the project would be to enable choosing an area, start and destination points on a real Tel-Aviv map, and getting a parking policy.
2. **Assumption of Uniform Travel Costs**: Our model assumes uniform movement costs on the grid, ignoring traffic. If this assumption is removed, the optimality of policies from Value-Iteration and Q-Learning could change. Future models could include traffic data in the reward function to better reflect real-world travel costs and adjust the optimal policy accordingly.
3. **Single-User world**: The model optimizes parking for a single driver, ignoring interactions between multiple users competing for the same spots. This is partially considered by the probabilities underlying the MDP. This could be addressed by extending the model to a **multi-agent system**.
4. **Performance of Q-Learning**: While **Q-Learning** performed well in many scenarios, it occasionally struggled with convergence, as described in the results section. This highlights the need for **better exploration-exploitation strategies** such as SoftMax and temperature. With infinite time we could add more iterations to improve the results.

### *Alternative Approaches*

While the MDP and Q-Learning approaches worked well for the problem at hand, other techniques could also be considered:

* **Monte Carlo Tree Search (MCTS)**: This method could be useful for navigating the probabilistic search space of parking. MCTS could evaluate possible sequences of moves by simulating various outcomes.
* **Multi-Agent Systems**: If the goal is to extend the model to multiple drivers, **multi-agent systems (MAS)**, combined with **reinforcement learning** could provide a more scalable solution for optimizing parking in competitive environments.
* **Expectimax**: This was one of our early approaches, fitting the problem’s probabilistic parking availability. We modeled the driver as the player and the adversary as the parking-availability probability. However, we set it aside because the search tree depth was insufficient to solve the problem.

### *Conclusion*

Overall, this project successfully demonstrates how MDPs and Q-Learning can outperform traditional search methods in optimizing parking in a probabilistic environment. The results indicate that using probabilistic models can significantly reduce search time and improve driver satisfaction by finding parking spots closer to home. However, the assumptions regarding uniform travel costs and single-user focus could be reconsidered in future work to improve model robustness and applicability to more complex urban scenarios.

## Link to Code

<https://github.cs.huji.ac.il/talsharon/huji-intro2ai-final-project.git>

1. Improved Optimal Path Finding Algorithm of Multistorey Car Park Based on MCP Protocol Zhendong Liu, Dongyan Li, Xi Chen, Xinrong Lv, Yurong Yang, Ke Bai, Mengying Qin, Zhiqiang He, Xiaofeng Li, and Qionghai Dai, 9 March 2022 [↑](#endnote-ref-1)
2. A Balanced Algorithm for In-City Parking Allocation: A Case Study of Al Madinah City, Mohammad A. R. Abdeen, Ibrahim A. Nemer and Tarek R. Sheltami, 1 may 2021 [↑](#endnote-ref-2)
3. Smart‑parking management algorithms in smart city, Mahdi Jemmal, Loai Kayed B. Melhim, MafawezT.Alharbi, Abdullah Bajahzar & Mohamed Nazih Omri, [↑](#endnote-ref-3)
4. A Hierarchical Algorithm for In-city Parking Allocation Based on Open Street Map and AnyLogic Software, Mohammad A. R. Abdeen, Ibrahim A. Nemer, Tarek R. Sheltami, Mohamed H. Ahmed, Mustafa Elnainay, 19 January 2023 [↑](#endnote-ref-4)
5. A Conceptual Framework for Modeling Smart Parking, Brahim Lejdel, 2020 [↑](#endnote-ref-5)
6. Finding available parking spaces made easy, Andreas Klappenecker, Hyunyoung Lee, Jennifer L. Welch, 17 March 2012 [↑](#endnote-ref-6)
7. Improving parking availability prediction in Smart Cities with IoT and ensemble-based model, Cédric Stéphane Koumetio Tekouabou, El Arbi ABDELLAOUI ALAOUI, Walid Cherif, Silkan Hassan, January 2020 [↑](#endnote-ref-7)