

הפקולטה להנדסה

המעבדה לעיבוד אותות

מחולל פרויקטים

במעבדה לעיבוד אותות

אלון רייכר

ליעד גם

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחה: ליאון רביב

מנחה אקדמי: פרופ' אמיר לשם

אוקטובר 2023

תוכן עניינים

[תודות 3](#_Toc143182198)

[presenting the problem 4](#_Toc143182199)

[Theoretical Background 5](#_Toc143182200)

[Alternative Solutions 8](#_Toc143182201)

[Chosen Solution 9](#_Toc143182202)

# תודות

ברצוננו להודות למנחה שלנו ליאון, למנחה האקדמי אמיר ובעיקר לעצמנו שמצאנו את הכוחות לעשות את זה ולא לפרוש בשיא או משהו כזה.

# presenting the problem

In major urban areas, managing traffic efficiently is a major challenge. Our project introduces a car traffic simulator that explores three distinct routing strategies: random, shortest path, and Q-learning. This simulator serves as a tool to allow us to analyze and compare how these strategies influence traffic flow, congestion patterns, and responses to road obstacles in a controlled environment.

We've designed three different approaches for the cars to follow. First, there's the "random" approach, where cars make decisions without a specific plan, much like taking turns by chance. Then there's the "shortest path" approach, where cars always follow the shortest route, like using a GPS. Lastly, we have the "Q-learning" approach, where cars learn from past experiences to figure out the most efficient routes, even when facing unexpected situations.

However, it's not just about smooth driving. Our virtual city throws in challenges like traffic jams and road closures, just like what happens in real life. This is where our simulator becomes valuable. We can use it to see which approach handles these challenges the best.

Our role in this project is similar to that of researchers studying traffic behavior. By observing and comparing how each approach performs, we can determine which strategy is most effective under different circumstances.

# Theoretical Background

Traffic simulation plays a pivotal role in understanding and managing the complex dynamics of vehicular traffic within urban and transportation systems. By employing computational models, traffic simulation allows researchers, urban planners, and policymakers to analyze traffic patterns, assess the impact of infrastructure changes, and formulate effective traffic management strategies.

Dijkstra's Algorithm for Shortest Path:

Dijkstra's algorithm is a well-known method used to find the shortest path between two points in a graph. In the context of our car traffic simulator, Dijkstra's algorithm plays a significant role in the "shortest path" routing strategy. This strategy focuses on guiding vehicles along the most direct route from their starting point to their destination, aiming to minimize the distance traveled.

How Dijkstra's Algorithm Works:

**Initialization**: The algorithm starts by assigning a tentative distance value to each node in the graph, with the initial node's distance set to zero and all other nodes' distances set to infinity.

**Exploration**: The algorithm examines neighboring nodes, updating their tentative distances if a shorter path is discovered.

**Selection**: The node with the smallest tentative distance is chosen as the current node, and its neighbors are explored.

**Termination**: The algorithm continues selecting nodes and exploring their neighbors until the destination node is reached or all nodes have been visited.

**Backtracking**: Once the destination node is reached, the algorithm backtracks from the destination node to the starting node, tracing the shortest path.

In the context of our simulator, Dijkstra's algorithm provides a systematic and deterministic approach to route vehicles through the road network. This strategy ensures that vehicles follow the path that minimizes the distance traveled, which can be particularly beneficial for optimizing travel times and reducing congestion under normal traffic conditions.

Q-learning for Adaptive Routing:

Q-learning is a reinforcement learning technique used to make decisions in dynamic and uncertain environments. In the "Q-learning" routing strategy of our car traffic simulator, vehicles learn and adapt their routes based on past experiences and traffic conditions. This approach enables vehicles to make informed decisions, even in the face of changing circumstances and obstacles.

How Q-learning Work:

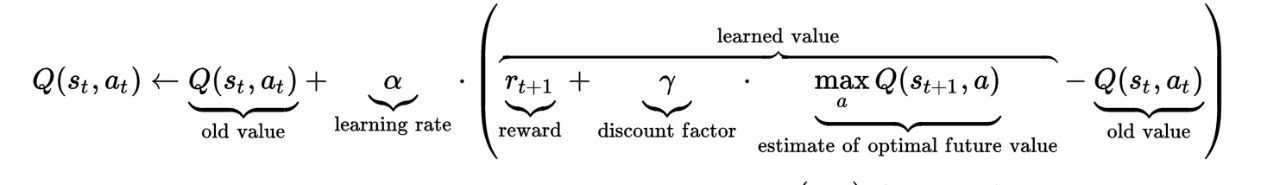
State and Action Space: Q-learning involves defining a state space (representing the environment) and an action space (potential decisions or moves).

Q-Values: Each state-action pair is associated with a Q-value, which represents the expected cumulative reward for taking that action from that state.

Exploration and Exploitation: Vehicles use a balance of exploration (trying new actions) and exploitation (choosing known high-reward actions) to learn optimal strategies.

Updating Q-values: Q-values are updated using a learning rate and discount factor, incorporating rewards obtained from actions and future Q-values.

This is the Q learning update rule:



Learning and Adaptation: Over time, vehicles learn which actions lead to higher rewards in different states, allowing them to adapt their decisions to maximize their cumulative rewards.

In our simulator, the Q-learning strategy empowers vehicles to learn from past experiences, such as traffic congestion and road closures. By adjusting their routes based on the learned Q-values, vehicles can dynamically navigate through the city, optimizing their paths to mitigate obstacles and minimize travel times. This adaptive approach to routing is particularly useful for addressing challenges like congested streets and unexpected road closures, contributing to a more efficient and responsive traffic management strategy.

# Alternative Solutions

For our navigation problem in a congested city, we had 2 alternative solutions:

1. Random navigation:

The first algorithm we implemented is a simple one, when car needs to move to the next road the car simply chooses a random road and moves to the next road. This is a model that imitates a person that doesn’t know the city and just guesses where his destination is.

This is not a good algorithm because a lot of times the car doesn’t even make it to the destination.

1. Shortest path navigation:

The second approach we tried was shortest path navigation. This approach is using Dijkstra algorithm for shortest path finding in a weighted graph.

Our graph's weights are the road's distances from one another, and they can be only positive , Dijkstra algorithm works on a non-negative weighted graph in contrast to Belman Ford, but Dijkstra is faster, that’s why we chose Dijkstra's algorithm.

This approach is faster and more reliable than random routing, in normal conditions its guaranteed to find a path from the source to the destination if one exists, but in the other case this approach does not consider road's speeds and road blockages.

# Chosen Solution

Based on the weakness of the alternative solutions we chose the solution of Reinforcement Learning to be the navigation algorithm of the cars in the simulator.

Reinforcement learning is a machine learning paradigm that involves training agents to make sequential decisions in an environment in order to maximize a cumulative reward. It draws inspiration from behavioral psychology, where individuals learn through trial and error by interacting with their surroundings.

In reinforcement learning, an "agent" interacts with an "environment" to learn how to take actions that lead to the most favorable outcomes over time. The agent doesn't require explicit instructions; instead, it learns by receiving feedback from the environment in the form of rewards or penalties based on its actions. The goal of reinforcement learning is to discover the optimal strategy, known as a policy, that guides the agent's decisions to maximize its long-term rewards.

The Learning Process in Reinforcement Learning:

1. The agent starts with little to no knowledge of the environment and how its actions affect outcomes.
2. It interacts with the environment, taking actions and receiving rewards based on those actions.
3. The agent uses the received rewards to update its knowledge and refine its strategy (policy) to make better decisions in the future.
4. Over time, through repeated interactions and learning, the agent improves its policy to achieve higher cumulative rewards.

More specifically, we chose Q learning which is a specific reinforcement learning algorithm.

Q-learning can be effective for several reasons:

1. Adaptability to Dynamic Environments: Navigation often takes place in complex and dynamic environments, where conditions can change rapidly. Q-learning allows an agent to adapt its navigation strategy based on the feedback it receives from the environment. This adaptability is crucial for handling real-world scenarios such as traffic congestion, road closures, and unexpected obstacles.
2. Trial and Error Learning: Q-learning employs a trial-and-error approach, where the agent explores different actions and learns from the outcomes. This process allows the agent to discover which actions lead to better outcomes and adjust its behavior accordingly. In navigation, the agent can learn to avoid routes that consistently result in traffic jams or delays, leading to more efficient routing decisions over time.
3. Exploration and Exploitation: Q-learning balances exploration of new paths and exploitation of learned knowledge. This balance is vital for navigation, as it enables the agent to discover better routes while also leveraging its existing knowledge of effective paths. For example, the agent might explore new routes during periods of low traffic and exploit well-known routes during peak hours.
4. Long-Term Planning: Q-learning focuses on maximizing long-term cumulative rewards. In navigation, this means that the agent considers not only the immediate consequences of its actions but also the potential benefits or drawbacks of its choices over the entire route. This long-term perspective allows Q-learning to guide the agent towards routes that lead to the best overall outcomes.

# Results

To see how well the Q-learning routing algorithm works, we ran some tests in our simulations. First, we tried navigating from a random starting point to a destination. This helps us compare how the Q-learning method performs against just picking random paths.

For the Q Learning training we set the following parameters:

Number of training episodes to 2000, maximum number of number of steps in each episode to 100, learning rate to 0.05, discount factor to 0.9 and epsilon to 0.2. The parameter's values were decided by trial and error, we tested and adjusted them until got them to a sweet spot where the agent's training won't take too long but the agent will learn the optimal path.

For the simulation we use the map of Tel Aviv, the starting point is from "Rokach Boulevard" to "Dizengof Center".

After the agent's training finished, we got two diagrams that describes the course of the training:

1. תמונה שמכילה צילום מסך, עלילה, טקסט, קו

   התיאור נוצר באופן אוטומטיBar chart of travel time in each episode and an indicator if the agent made it to the destination:

We can see that at first, mostly the agent hasn’t reached the destination but as training continued the agent got to the destination and in more optimized path. We can see that by the number of seconds being lower as the episode numbers grow.

1. Mean rewards over number of episodes:

תמונה שמכילה טקסט, קו, עלילה, צילום מסך

התיאור נוצר באופן אוטומטי

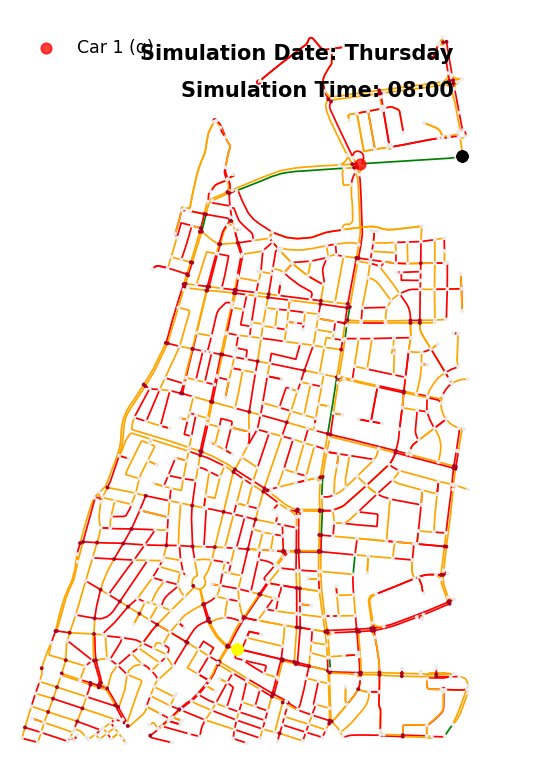
the x axis is the mean rewards over the last 50 episodes, and we can see that the reward of the agent is growing as we train it more and more.

Now let’s look at the result, the agent's route:

תמונה שמכילה תרשים, תוכנית, שרטוט, מפה

התיאור נוצר באופן אוטומטי

The black mark indicates the starting point of the agent, and the yellow mark indicates the ending point.

Additionally, we have an animation of the agent's drive in real time with the current time road's speeds where: red – heavy traffic, yellow - medium traffic, and green – low traffic:

תמונה שמכילה טקסט, תרשים, מפה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, תרשים, מפה

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, תרשים, מפה

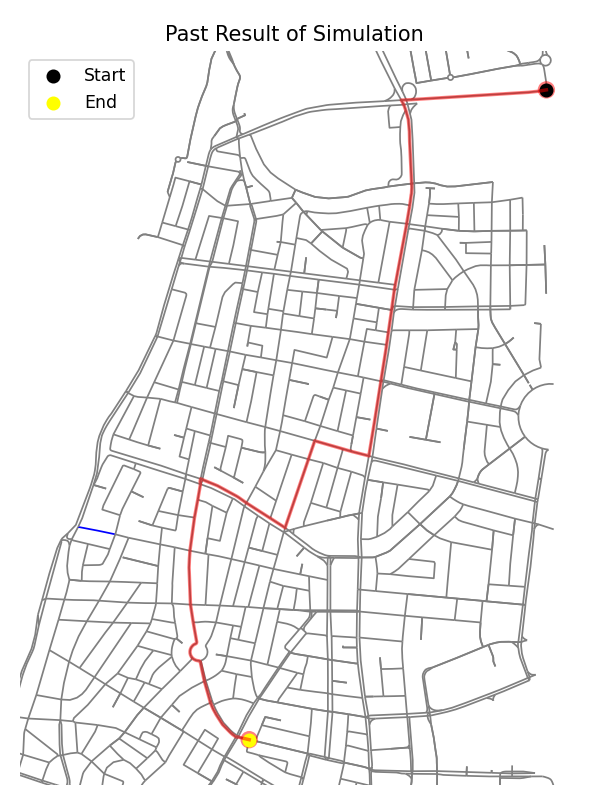
התיאור נוצר באופן אוטומטי

We can see the red dot, representing the q learning agent, navigates from the starting point to the destination. Additionally, we have the day and the time of the simulation. The agent completed this drive in 19 minutes.

תמונה שמכילה קו, עלילה, צילום מסך, תרשים

התיאור נוצר באופן אוטומטיתמונה שמכילה צילום מסך, עלילה, צבעוני, קו

התיאור נוצר באופן אוטומטיNow, to test the q learning algorithm let’s make a scenario, let’s say that in the following Sunday morning the agent wanted to make the same route but this time a road in the original route that the agent took was blocked this morning, let’s see how well the agent adapt:



We can see that there is a road colored in blue in the map, this indicates that the road was blocked. Now we can see that the agent knew the road was blocked and chose a different route this time in order to avoid the closed road. This route took the agent 22 minutes to complete and that’s why this route wasn’t chosen in the first simulation.

# Comparison between the different solutions

# Conclusions

# Appendix

## Appendix A

"Software Requirements Specification" – more specific software descriptions of the program.

## Appendix B

"Simulator HLD" – provides information about configurable parameters, used data and user documentation of all the classes in the project.