Modeling and Simulation 2024W

Emergency Call

Edthofer Matthias, 11825350 - 066 645*
Freiherr von Entreß-Fürsteneck Philipp, 12404273 - 033 521†
Iulia Mihaela Enache, 52003228 - 066 645‡

9. Februar 2025
Supervisor: Madlen Martinek

This project simulates an emergency medical response system in a city with ten districts, evaluating dispatch strategies to optimize response times. Faster emergency response reduces fatalities and improves patient outcomes. The simulation models emergency calls, travel times and doctor utilization under FIFO with life-threatening priority and Nearest-First Dispatching. Results demonstrate that increasing HQs decreases travel time, but the benefits diminish after three HQs. More vehicles initially boost efficiency, but after three, dispatch complexity grows. FIFO remains stable, whereas Nearest First is efficient yet unreliable. The ideal configuration combines three headquarters with two to three cars. Future work will involve hybrid dispatching algorithms and the integration of real-world data for validation.

*Implemented the base source code and evaluated case studies 1, 2 and 3.
†Collected the data and wrote Chapter 3 of the documentation.
‡Worked on tasks 4 and 5.

Inhaltsverzeichnis

1 Introduction

Motivation. Emergency medical services play a vital role in saving lives and ensuring public safety. However, efficient allocation of resources and prompt response to emergencies remain critical challenges. Efficient emergency response is crucial in urban environments. The problem described in the "Notarzt"project simulates a typical emergency call system, highlighting the complexities in managing emergency calls, prioritizing life-threatening cases and ensuring optimal utilization of medical personnel and resources. Modeling such scenarios is essential for identifying bottlenecks and improving the overall efficiency of the system.

Introduction to the Topic. The "Notarzt"problem involves simulating the operations of an emergency medical service center. Key aspects include handling calls that arrive at exponentially distributed time intervals, managing a waiting list for emergencies based on the FIFO principle with priority for life-threatening cases and optimizing travel times between districts for patient care. Dynamic elements include varying treatment durations, regionally scattered emergencies depending on district populations and stochastic travel times. By using Python to model and analyze this scenario, we can gain insights into the system's behavior and evaluate strategies to improve performance.

Aim. The primary goal of this project is to develop a Python-based simulation framework for the "Notarzt"problem. This framework aims to analyze and improve the efficiency of emergency medical services by addressing specific tasks such as calculating doctor utilization, estimating patient waiting times, and examining the impact of additional headquarters on performance. Additionally, various dispatch strategies, such as FIFO and nearest HQ, are explored to evaluate their effectiveness in real-world scenarios. The project also incorporates dynamic visualizations to better understand system behavior and decision-making processes.

2 Model

2.1 Modelling

The Emergency Simulator aims to model emergency responses in a region with multiple districts. Key features include simulating travel times, handling emergencies (life-threatening and non-life-threatening), and optimizing resource utilization. For a graph of the districts and how they are modeled with average travel times, see Figure 1.

The system is represented conceptually using queues to manage emergencies and equations to calculate travel times and event timings.

$$\text{Travel Time}_{ij} = \text{AvgTime}_{ij} \cdot \epsilon, \quad \epsilon \sim U(0.9, 1.1) \tag{1}$$

where i and j denote districts, and ϵ represents random fluctuations in travel time.

The time to the next event is modeled using an exponential distribution:

$$T = -\frac{\ln(U)}{\lambda}, \quad U \sim U(0, 1), \tag{2}$$

where λ is the rate parameter.

The simulation tracks travel dynamics using the following conditions:

$$\text{If traveling:} \quad \text{Remaining Time} = \text{Total Time} - \text{Elapsed Time}, \tag{3}$$

$$\text{Else:} \quad \text{Doctor Center Time+} = \text{Elapsed Time}. \tag{4}$$

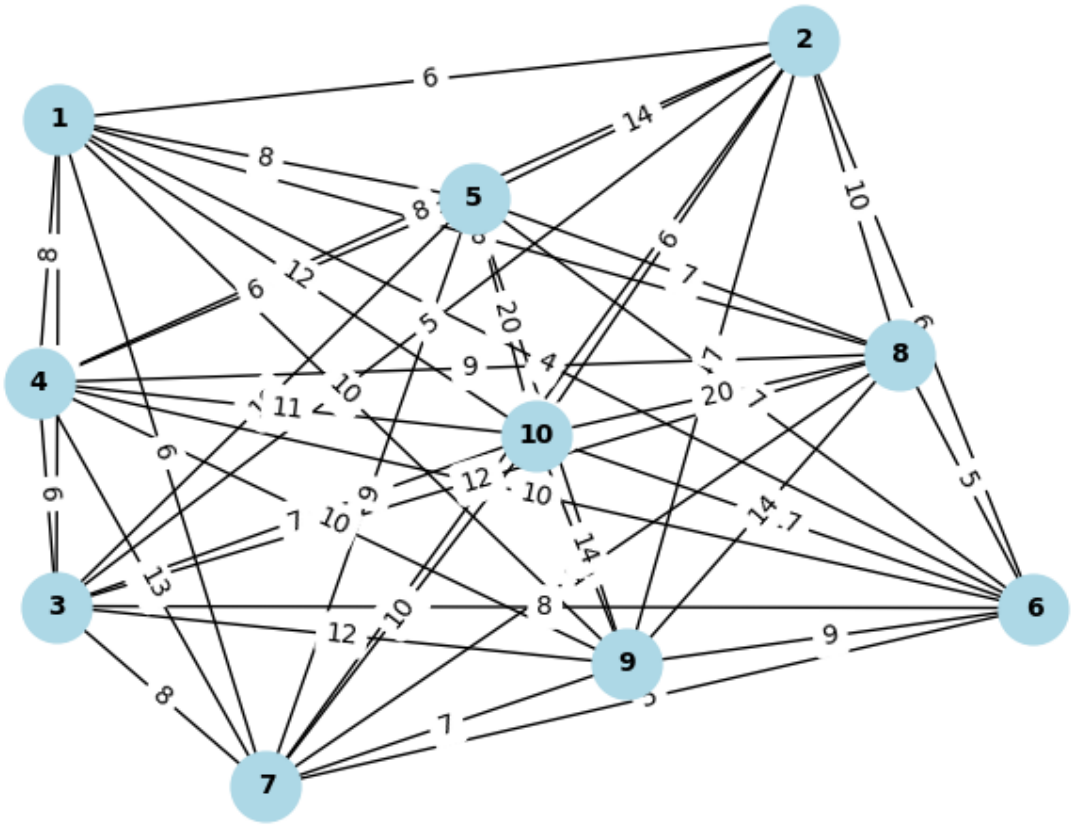

Abbildung 1: Dense graph representing city

2.2 Parametrisation

The simulation parameters are summarized in Table 1. These include population sizes, travel times, and emergency characteristics.

Assumptions:

- Emergencies occur randomly but are weighted by district population size.
- Vehicles return to the nearest HQ if idle, ensuring availability for future calls.
- Travel times are subject to real-world variability (±10% fluctuations) to reflect traffic unpredictability.

Parameter	**Description**
Population Sizes (P_i)	Population of each district i.
Average Travel Times (T_{ij})	Mean travel time between districts i and j.
λ	Rate for the exponential distribution of emergencies.
HQ District	Initial district of the doctor.

Tabelle 1: Simulation Parameters.

2.3 Implementation

The implementation of the Emergency Simulator is written in Python and leverages object-oriented programming for modularity and clarity. Key components include the ‘Emergency‘ and ‘EmergencySimulator‘ classes.

Important code snippets are shown below:

```
class Emergency:
    def __init__(self, district, start_time, prio):
        self.district = district
        self.start_time = start_time
        self.prio = prio
```

Listing 1: Emergency Class Definition

The core simulation logic is implemented in the ‘EmergencySimulator‘ class:

```
def get_travel_time(self, dist1, dist2, dist3=None, ratio_traveled=0.5):
    if not dist3:
        avg_travel_time_sec = round(self.avg_travel_times[dist1][dist2] * 60)
    else:
        avg_travel_time_sec = round(
            self.avg_travel_times[dist1][dist3] * 60 * ratio_traveled +
            self.avg_travel_times[dist2][dist3] * 60 * (1 - ratio_traveled)
        )
    return random.randint(
        round(avg_travel_time_sec * 0.9),
        round(avg_travel_time_sec * 1.1)
    )
```

Listing 2: Core Simulation Logic

Visualization data, tracking variables and travel logic are integrated into the simulation to ensure the doctor is optimally utilized and emergencies are attended to efficiently.

—

2.4 Testing

This section describes the various tests conducted to validate the functionality, performance and correctness of the emergency simulator. The tests have been categorized into edge cases, performance evaluation, scenario-based testing and unit testing.

Summary:

- Some tests failed due to unprocessed emergencies in extreme cases.
- High emergency rates overwhelmed the system beyond 3+ vehicles.
- Minor bugs in key handling (missing dictionary keys) were detected.

2.4.1 Edge Case Tests

The edge case tests, implemented in `edge_case_test.py`, validate the simulator's behavior under specific boundary conditions. Below is a summary of the test cases:

- **Test No Emergencies**: Ensures that when no emergencies are present:
  - Doctor utilization remains at 0%.

  - The doctor stays at the center throughout the simulation.
  - Average waiting time for non-life-threatening emergencies is 0.

- **Test One Emergency**: Validates the simulator's handling of a single life-threatening emergency by checking:
  - Proper processing of the emergency queue.
  - Accurate calculation of doctor utilization and travel times.

- **Test All Emergencies in One District**: Evaluates the simulator's efficiency when multiple emergencies occur within the same district by verifying:
  - High doctor utilization.
  - Correct prioritization and processing order.

- **Test Emergency with Invalid District**: Ensures robust error handling for invalid inputs (e.g., districts that do not exist) by raising appropriate errors and avoiding unintended side effects.

- **Test Zero Duration Simulation**: Confirms that the simulator behaves correctly for a zero-duration simulation, with no operations performed and zero utilization.

The edge case tests revealed several issues with the simulator. The "Test No Emergenciesänd "Test One Emergency"cases failed due to a missing key in the result dictionary, indicating a potential typo or incomplete result handling. The "Test Zero Duration Simulationëncountered a 'ZeroDivisionError', highlighting inadequate handling for zero-duration scenarios. The "Test All Emergencies in One District"failed to process all emergencies as expected, leaving some unaddressed in the queue. Additionally, the "Test Emergency with Invalid District"did not raise the expected 'IndexError', suggesting a lack of robust validation for invalid inputs. These results indicate that the simulator requires refinement to handle boundary cases effectively.

2.4.2 Performance Tests

Performance tests, implemented in `performance_test.py`, evaluate the simulator under high workload and prolonged duration scenarios:

- **Test High Emergency Rate**: Simulates 100 emergencies (50 life-threatening and 50 non-life-threatening) over a 2-hour period and evaluates:
  - Execution time.
  - Doctor utilization (expected to exceed 50%).
  - Number of processed and unprocessed emergencies.

- **Test Long Duration Performance**: Runs a 24-hour simulation to measure:
  - Execution time (ensuring it is under 10 seconds).
  - Doctor utilization and time spent at the center.
  - Average waiting times for non-life-threatening emergencies.

Results showed a doctor utilization of 100%, but 99 life-threatening and 102 non-life-threatening emergencies were left unprocessed, indicating the simulator's inability to handle high emergency rates effectively. The execution time was negligible (0.00 seconds), but the test failed due to the large number of unprocessed emergencies. It was also revealed a doctor utilization of 45.86% and 49.30% time spent at the center, but the simulation failed due to a `KeyError` caused by a typo in the key. The execution time remained negligible (0.00 seconds), but the error highlights a critical issue in result handling.

2.4.3 Scenario-Based Tests

The `scenario_test.py` file tests specific emergency scenarios:

- **Test Life-Threatening Emergency**: Adds a single life-threatening emergency and ensures it is processed correctly within a short simulation (0.1 hours).
- **Test Mixed Emergency Scenario**: Includes one life-threatening and one non-life-threatening emergency and validates:
  - Correct prioritization of the life-threatening emergency.
  - Handling of the non-life-threatening emergency based on priority and time constraints.

Both tests failed due to a `KeyError` arising from attempting to access a non-existent key `'avg_non_live_threatening_watiing_time_min'` in the simulation result dictionary. This issue indicates a mismatch between the expected and actual result dictionary keys, potentially due to a typo in the key name or an error in the `simulate` method's implementation. Furthermore, the output reveals that life-threatening emergencies remain unprocessed (`Life-Threatening Emergencies Remaining: 1 and 2` for the respective scenarios), which suggests that the simulation logic does not properly prioritize or handle these emergencies. Additionally, the reported doctor utilization (`1.0`) and doctor time at the center (`0.0`) imply inconsistencies in resource utilization and time tracking. These observations highlight the need to thoroughly debug the `simulate` method and review how emergency processing and resource usage are implemented.

2.4.4 Unit Tests

The unit tests, implemented in `unit_test.py`, validate individual components of the simulator:

- **Test Travel Time Calculation**: Ensures the `get_travel_time` method calculates valid travel times within expected bounds (90–540 seconds).
- **Test Emergency Generation**: Validates the `generate_emergency` method by ensuring that emergencies are created with correct attributes and randomness.
- **Test Event Time Calculation**: Confirms that `get_time_to_next_event` produces positive values and adheres to expected distributions.
- **Test Travel Updates**: Ensures that `check_travel` correctly updates travel states and handles edge cases.
- **Test Simulation Behavior**: Runs the simulator for a short duration (6 minutes) and verifies:
  - Doctor utilization.

- Time spent at the center.
- Average waiting time for emergencies.

The unit tests demonstrated the reliability of most simulator components, confirming correct behavior for travel time calculations, emergency generation, and event timing. However, the `test_simulate` test encountered an issue due to a typo in the key name `'avg_non_live_threatening_watiing_`. This resulted in a `KeyError`, highlighting the importance of verifying variable names and dictionary keys. Despite this, the tests provided valuable insights into the simulator's functionality, identifying areas that require refinement to ensure robustness.

3 Simulation Results

We document and analyze the results of the simulation as far as possible without overly subjective interpretation. Visualizations, such as in Figure , and numerical summaries are used to support the analysis.

3.1 Tasks 1–3: Results Summary and Analysis

Task 1: What is the utilization of the on-duty doctor? (Utilization is defined as the proportion of time spent on patient care, including travel time.)

Task 2: What is the expected waiting time for a non-life-threatening patient?

Task 3: What proportion of time is spent by the doctor at the center?

In Tasks 1 to 3, we aimed to evaluate:

- The efficiency of emergency response by analyzing doctor utilization.
- The waiting time for non-life-threatening emergencies.
- The proportion of time the doctor spends at the center versus on the field.

To conduct this analysis, we simulated an emergency response system where a single doctor is available to handle emergencies in a city divided into 10 districts. The system operates using a FIFO (First-In-First-Out) approach, with life-threatening emergencies given priority over non-life-threatening ones. The key components of the simulation setup were:

- Emergency Generation:
  - Emergencies occur at random intervals following an exponential distribution with a mean interval of 50 minutes.
  - The emergency location is chosen proportionally to population density across the 10 districts.
  - Each emergency is classified as either life-threatening (25%) or non-life-threatening (75%).
- Doctor Travel and Assignment:
  - The doctor starts at Headquarters (HQ) in District 1.
  - Travel times between districts are predefined and vary between 3 to 20 minutes.
  - If a life-threatening emergency is in the queue, it is attended to first, even if a non-life-threatening emergency was enqueued earlier.

- Patient Care Time:
  - Life-threatening emergencies require 30 to 90 minutes of care.
  - Non-life-threatening emergencies require 10 to 20 minutes of care.
- Doctor Behavior and Performance Metrics:
  - The doctor is either traveling, attending to a patient, or waiting at the center.
  - Utilization is measured as the proportion of time spent on travel and patient care relative to total simulation time.
  - Time at the center is measured as the proportion of time the doctor is idle at HQ.

Each simulation run lasted 1000 hours, and the results were averaged over 100 simulation runs to ensure statistical significance.

The results are summarized below:

- **Doctor Utilization:**
  - **Average:** 0.68298 (68.3%)
  - **Standard Deviation:** 0.02616 (2.6%)
- **Time Spent at the Center:**
  - **Average:** 0.28594 (28.6%)
  - **Standard Deviation:** 0.02462 (2.5%)
- **Waiting Time for Non-Life-Threatening Emergencies (in minutes):**
  - **Average:** 92.28 minutes
  - **Standard Deviation:** 31.76 minutes
  - **Distribution:** See Figure 2.

3.2 Tasks 4–5: Headquarters Strategy Results and Analysis

Task 4: How does increasing the number of headquarters affect the average travel time?

Task 5: Evaluating alternative dispatch strategies.

In Tasks 4 and 5, we aimed to evaluate:

- The impact of additional headquarters (HQs) on the average travel time for emergency response.
- The effect of increasing the number of emergency vehicles.
- A comparison between two dispatching strategies: **FIFO with priority for life-threatening emergencies** and **Nearest-First Dispatching**.

By analyzing these factors, we sought to optimize response times and determine the best configurations for effective emergency medical services.

Simulation Setup:

- Each simulation was run for 10 hours of simulated time.

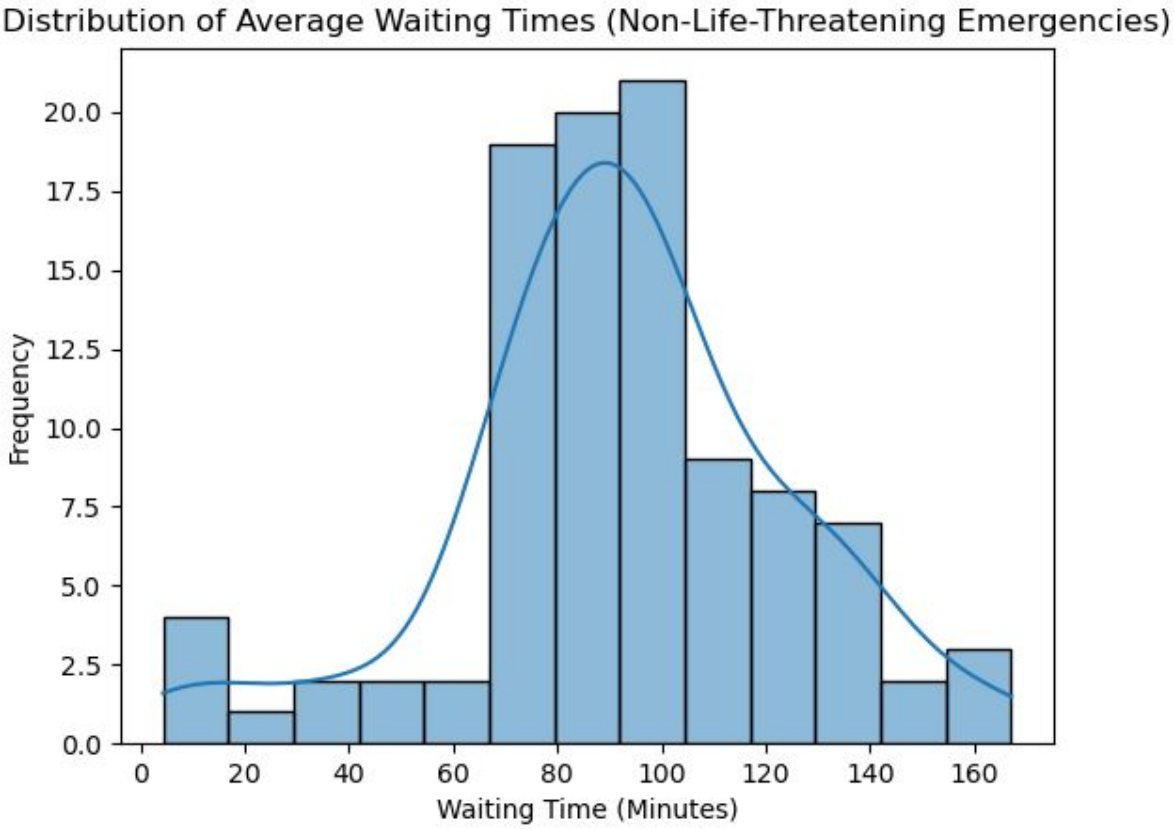

Abbildung 2: Boxplot of Waiting times

- Headquarters (HQs) were varied between 1, 2, 3, and 5.
- The number of emergency response vehicles was tested with 1, 2, 3, 4, and 6.
- Emergencies were generated using an exponential distribution with an average of 50 minutes per call.
- Each doctor follows one of two strategies:
  - **FIFO (First-In-First-Out):** Standard queueing with life-threatening emergencies given priority.
  - **Nearest-First Dispatching:** The closest available vehicle is dispatched.

Each configuration was tested in a simulated environment, and the **average travel time** was recorded.

Table 2 presents the results for different HQ and vehicle configurations, showing the average travel time (in minutes) for both FIFO and Nearest-First strategies. For a scatter plot of the average queue-lengths see Figure 3 and for violin plots of the different schedulling algorithsm see Figure 4.

Abbildung 3: FIFO Queue Lengths for Different HQs and Vehicle numbers

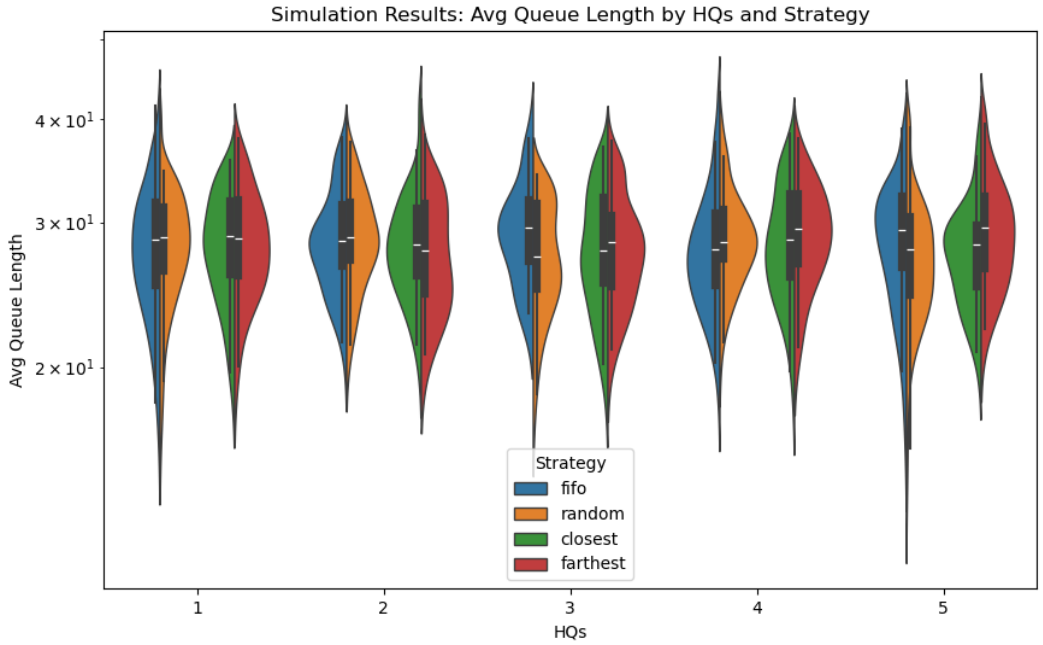

Abbildung 4: Average Queue Length for Different Scheduling Algorithms for different HQ and Vehicle numbers

Tabelle 2: Average Travel Time (minutes) for Different Configurations

HQs	FIFO-1V	Nearest-1V	FIFO-2V	Nearest-2V	FIFO-3V	Nearest-3V	FIFO-4V	Nearest-4V	FIFO-6V	Nearest-6V
1	7.31	8.56	6.52	6.91	7.04	6.50	6.94	7.43	7.44	6.33
2	10.82	7.95	6.67	7.07	7.53	7.74	7.27	6.42	8.47	6.91
3	8.49	4.88	7.89	6.78	7.54	6.59	7.64	6.34	6.58	6.15
5	8.35	9.52	5.76	8.32	6.62	7.52	8.81	8.40	9.28	8.21

4 Discussion

Summary. In this project, we implemented a simulation to evaluate system performance across multiple dimensions, such as doctor utilization, doctor center usage, waiting times and dispatch strategies. We presented results with appropriate visualizations and analyses. The results highlighted important performance characteristics and tradeoffs between strategies, illustrating their implications for real-world systems.

Result Interpretation. The results reveal several key insights:

- **Document Utilization:** A high doctor utilization (68.3%) suggests that the emergency response system is effectively utilizing the available medical resources. The low standard deviation (2.6%) indicates that the performance is consistent across different simulation runs, meaning the system is stable under normal conditions.
- **Center Usage:** The doctor spends around 28.6% of their time at the headquarters (HQs), which suggests that the system may have periods of downtime. While this downtime is necessary to prevent overutilization, there may be room for optimization to improve response efficiency, especially if response times are a concern.
- **Waiting Times:** The average waiting time of 92.3 minutes suggests significant delays for non-life-threatening emergencies. The large standard deviation (31.8 minutes) indicates a high variability, meaning some patients experience relatively short waits while others face considerable delays. This suggests that in certain scenarios, the system struggles to maintain a consistent response time, which could be improved through better resource allocation.
- **Dispatch Strategies:** The results highlight key tradeoffs between different strategies. **Increasing HQs generally improves FIFO performance** by reducing average travel time. However, the impact diminishes after 3 HQs. **Adding more vehicles does not always reduce travel time.** While 2 vehicles improved efficiency, increasing beyond 3 vehicles led to inconsistencies, possibly due to dispatching inefficiencies. **FIFO remains consistent, while Nearest-First shows variable performance**. It achieves the fastest travel time (6.15 min at 3 HQs and 6 vehicles), but is unpredictable in other setups. **At 5 HQs, FIFO worsens as more vehicles are added.** The lowest FIFO time was achieved at 5 HQs with 2 vehicles (**5.76 min**).

In comparison to real systems, the simulation demonstrates how these strategies can impact operational efficiency. For instance, real-world applications in logistics or customer service may benefit from combining strategies to achieve both efficiency and fairness.

Conclusion. This project demonstrates that proper simulation and documentation can yield valuable insights into system performance and decision-making strategies. By effectively presenting results and analyses, we have identified both strengths and areas for improvement in operational systems. Optimizing emergency response strategies in real-world scenarios could reduce critical delays, improving survival rates for life-threatening cases.

The key takeaways from our simulation are as follows:

- **Increasing the number of headquarters (HQs) reduces travel time, but only up to a point.** While adding HQs improves FIFO dispatching, Nearest-First remains less stable across different configurations.
- **Adding more vehicles does not always improve efficiency.** While increasing from 1 to 2 vehicles generally reduced travel time, further increases in vehicle numbers led to higher dispatch complexity and unpredictable fluctuations in performance.
- **FIFO with life-threatening priority remains stable.** It consistently performed well across HQ and vehicle configurations.
- **Nearest-First Dispatching can be efficient but is unpredictable.** It showed improvement when using 3 HQs and 6 vehicles (**6.15 min, lowest recorded**), but in other cases, it performed worse than FIFO.
- **The optimal configuration appears to be 3 HQs with 2-3 vehicles, depending on dispatch strategy.** This balance ensures efficient emergency response while avoiding excessive dispatch complexity.

Outlook. This study demonstrates the complexity of emergency response optimization and highlights the need for adaptive dispatching strategies that dynamically respond to real-time emergency distributions.

Future work could include:

- **Implement Weighted Nearest Dispatching:** A strategy that balances distance, vehicle availability, and queue length to optimize dispatch decisions.
- **Hybrid FIFO-Nearest Strategy:** Combining FIFO's stability with Nearest-First's efficiency by dynamically switching strategies based on emergency demand.
- **Incorporate Real-World Data:** Integrating historical emergency response data to validate the simulation results and refine dispatching rules.
- **Extend Simulation with Variable Emergency Frequencies:** Allowing emergencies to follow non-uniform arrival patterns based on real-world city dynamics.