Université Gustave Eiffel

Multi-Agent Systems Project

Master's in Intelligent Systems and Applications

Project 1 : Public Transport Mobility Simulation

Academic year :2024-2025
Course : Multi-agent systems

Team Members :

Mahmoud Aziz Ammar

Karim Damak

Siwar Najjar

Supervised by :

Mrs Linda Belkessa

Acknowledgment

We would like to express our heartfelt gratitude to Mrs. Linda Belkessa for her invaluable guidance, encouragement, and support throughout this project. Her expertise in the field of multi-agent systems and her constant availability for discussions greatly enriched the development of our work.

We are deeply appreciative of the time and effort she invested in mentoring us, and we hope this project reflects her passion for advancing intelligent systems. Thank you, Mrs. Belkessa, for believing in our potential and for being a guiding light throughout this journey.

Contents

List of Figures

Chapter 1

Introduction

Imagine a bustling city where passengers move through a web of streets, buses glide along their routes, and unexpected events like roadblocks or breakdowns add a layer of unpredictability. This report delves into the creation of a virtual urban environment that mirrors these real-world dynamics. Using a multi-agent system built with Python and Mesa, we simulate a miniature city complete with roads, bus stops, passengers, and public transport vehicles.

Our goal is not just to replicate the typical flow of urban transit but to explore how disruptions and unpredictable factors shape the movement of people and vehicles. Through this simulation, we create a vivid scenario where passengers navigate from their starting points to destinations, some walking while others rely on buses adhering to predefined schedules. Along the way, challenges such as sudden road closures or vehicle delays make the system feel alive, capturing the complexity of urban mobility.

This project is more than just an academic exercise—it's a playful and insightful dive into the interplay between public transportation systems and their users. By building this city model, we aim to uncover patterns, test solutions, and better understand the intricacies of urban transit, all within a controlled and dynamic virtual environment.

Chapter 2

System Design

2.1 City Class

The CityModel class serves as the central framework, managing the grid-based environment, agent interactions, and overall simulation. Key responsibilities include:

- Initializing a grid of specified dimensions to represent the city.
- Scheduling agent actions using the SimultaneousActivation mechanism to allow for simultaneous updates.
- Introducing disturbances (e.g., roadblocks or vehicle breakdowns) randomly during simulation steps to mimic real-world unpredictability.

The environment is represented by a MultiGrid, which allows multiple agents to occupy the same cell, facilitating interactions and realistic agent behavior.

2.2 Vehicle Class

The Vehicle class models public transport vehicles that operate along predefined routes and timetables. Key features include:

- Dynamic Movement: Vehicles move along a route represented as a list of grid coordinates, simulating bus routes in a city.
- Passenger Interaction: Vehicles detect and pick up passengers located in adjacent grid cells, ensuring realistic public transport functionality.
- Handling Random Perturbations: Vehicles respond to random events such as breakdowns, delays, or route changes. These disturbances are implemented to test the system's robustness under real-world-like conditions.

Vehicles are synchronized with their positions on the grid and their states within the simulation to ensure consistency.

2.3 Passenger Class

The Passenger class models individuals traveling from a random origin to a specified destination. Key features include:

- Autonomous Navigation: Passengers move step-by-step towards their destinations using a simple pathfinding approach.
- Interaction with Vehicles: Passengers board vehicles when they are in proximity and continue their journey if possible.
- State Tracking: Passengers track their travel time and notify the system upon reaching their destinations, enhancing data collection for analysis.

Passenger movement respects grid boundaries and considers obstacles such as roadblocks or occupied cells.

Chapter 3

Technologies and Tools Used

3.1 Python

Python was chosen for its simplicity, readability, and extensive library support. It served as the primary language for implementing the agents and the simulation logic.

3.2 Mesa

Mesa provided the core framework for developing and visualizing the multi-agent system. Its modular structure and built-in tools for agent-based modeling enabled efficient implementation of the simulation components and real-time visualization.

3.3 Random Library

The random module in Python was used extensively to introduce unpredictability in agent behaviors and disturbances, such as random breakdowns, delays, or roadblocks.

Chapter 4

Implementation and Development

4.1 City Initialization

The city is represented as a **grid-based environment** using the `MultiGrid` class from the Mesa framework. This grid serves as the spatial context for the simulation, where agents (passengers, vehicles, and roadblocks) interact.

- **Grid Setup:**
  - The city grid is initialized with customizable dimensions (`width` and `height`), allowing for simulations of various city scales.
  - The grid is non-toroidal, meaning agents cannot wrap around edges.
- **Dynamic Elements:**
  - Perturbations, such as roadblocks or vehicle breakdowns, can be introduced randomly. These disturbances simulate real-world challenges in urban mobility.
- **Customizable Parameters:**
  - The user can set parameters like grid size, number of passengers, and vehicles through the `ModularServer` interface.

This initialization allows for flexible simulations with varying levels of complexity, reflecting the dynamics of real-world cities.

Figure 4.1: City initialization

4.2 Vehicle Initialization

Vehicles in the simulation represent public or private transportation systems operating along predefined routes.

- **Route Definition:**
  - Each vehicle is assigned a route consisting of a series of grid cells, representing roads or paths.
  - The routes are randomly generated but constrained within the grid dimensions.
- **Attributes:**
  - Vehicles are initialized with a `timetable` to manage stops along their route.
  - They have states such as `broken_down` to account for disturbances.
- **Interaction:**
  - Vehicles can board passengers if they are at the same grid cell.
  - In case of breakdowns, vehicles stop operating until disturbances are resolved.

This initialization ensures realistic modeling of public transport systems, including challenges like breakdowns.

4.3 Passenger Initialization

Passengers are modeled as individual agents with unique IDs and specific travel goals.

- **Journey Assignment:**
  - Each passenger is assigned a random origin and destination on the grid. These locations simulate the diverse travel demands within a city.
- **Behavior:**
  - Passengers start their journey at their origin and attempt to move towards their destination. They either walk or board vehicles when available.
  - Once passengers reach their destination, they are marked as `reached_destination`.
- **Interactions:**
  - Passengers interact with vehicles by boarding or alighting when conditions are met (e.g., the vehicle stops at their destination).

This initialization captures the complexity of urban travel, with passengers dynamically adjusting to available transportation modes.

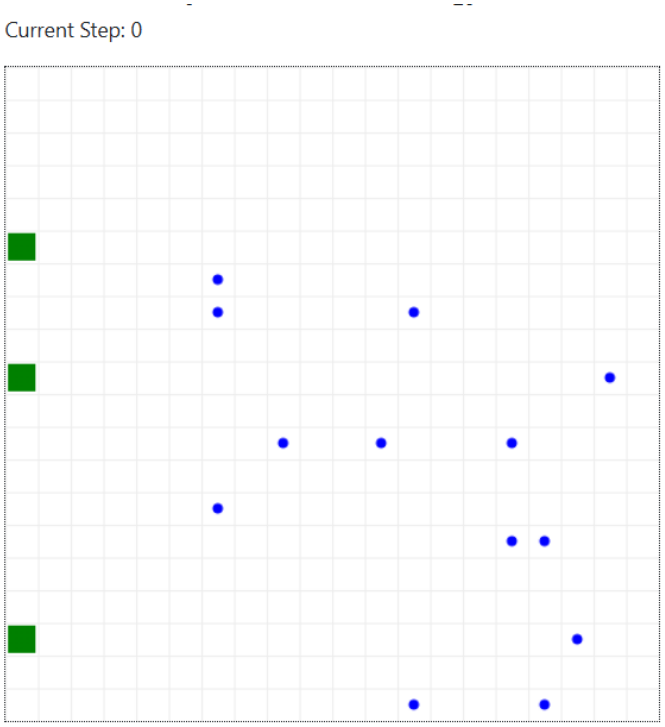

Figure 4.2: Agents Initialization Example

4.3.1 Simulation Steps

The simulation progresses in discrete steps, where all agents update their states simultaneously.

- **Passenger Actions:**
  - Passengers move towards their assigned vehicles or directly to their destination if no vehicle is nearby.
  - Once a passenger reaches their destination, their journey ends, and their total travel time is recorded.
- **Vehicle Actions:**
  - Vehicles move along their predefined routes, stopping to pick up or drop off passengers.
  - Vehicles may stop operating temporarily due to breakdowns.
- **Disturbances:**
  - At random intervals, perturbations such as roadblocks or vehicle breakdowns are introduced, dynamically affecting agent behavior.

This stepwise simulation captures the interactions between passengers, vehicles, and external disturbances, providing a realistic depiction of city dynamics.

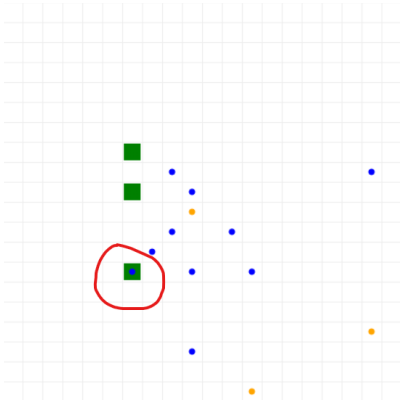

Figure 4.3: Vehicles picking up passengers

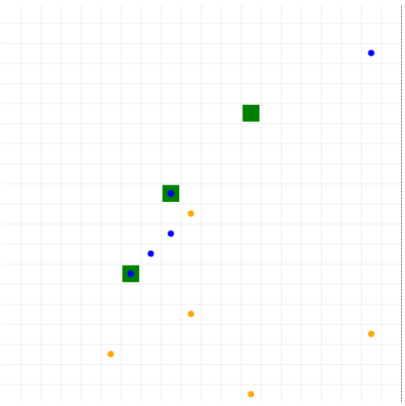

Figure 4.4: Passengers reaching their destination (becoming yellow)

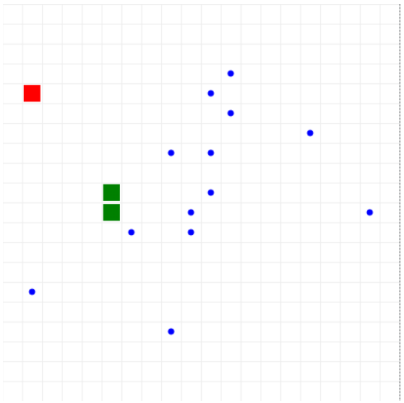

Figure 4.5: Roadblocks (becoming red)

4.3.2 Visualization

Visualization is a critical part of the simulation, allowing users to observe agent behaviors and interactions in real-time.

- **Framework:**
  - The visualization is implemented using `CanvasGrid` from Mesa's visualization module.
  - Each agent type (passengers, vehicles, roadblocks) has distinct visual representations for clarity:
    - **Passengers:** Circles, with orange indicating that they have reached their destination and blue otherwise.

    - **Vehicles:** Rectangles, with green for active vehicles and red for broken-down ones.
    - **Roadblocks:** Black rectangles indicating blocked areas.

- **User Interface:**
  - Users can interact with the simulation through sliders to adjust parameters such as grid size, number of passengers, and vehicles.
  - A checkbox allows toggling perturbations on or off.
- **Dynamic Visuals:**
  - Agents move across the grid in real-time, with colors and shapes changing to reflect state updates (e.g., passenger boarding a vehicle, vehicle breaking down).

This visualization aids in analyzing the impact of disturbances on urban mobility and offers an intuitive understanding of the system.

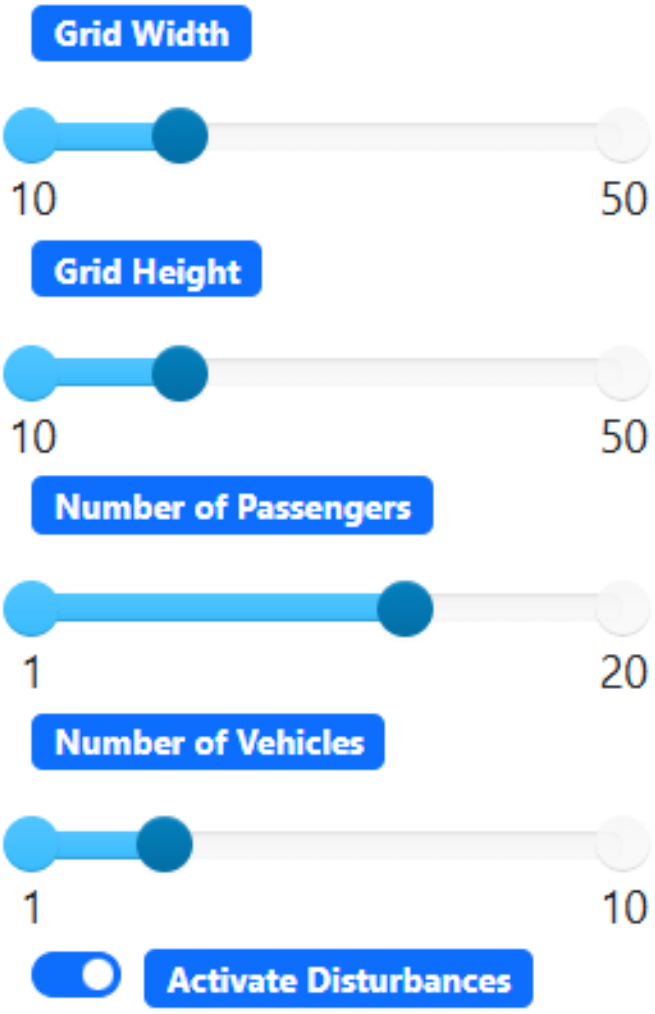

Figure 4.6: Visualization of the parameters

Chapter 5

CHALLENGES FACED

5.1 Synchronizing Agents and Grid States

Ensuring consistency between agent states in the scheduler and their positions on the grid posed significant challenges. Debugging and validating synchronization logic required careful examination of agent interactions and the simulation framework.

5.2 Handling Random Perturbations

Implementing realistic disturbances, such as vehicle breakdowns and route changes, required balancing randomness with meaningful impacts on the simulation. Managing these perturbations without disrupting the overall simulation flow was complex.

5.3 Dynamic Movement of Vehicles and Passengers

Designing efficient algorithms for passenger movement and vehicle routing was non-trivial. The movement logic had to account for obstacles, grid boundaries, and interactions with other agents while maintaining computational efficiency.

5.4 Visualization Clarity

Ensuring that the real-time visualization accurately reflected the simulation state while remaining interpretable required iterative adjustments to the portrayal of agents. Balancing visual detail and clarity was a persistent challenge.

Conclusion

This project demonstrated the potential of multi-agent systems for simulating complex urban dynamics. By integrating passengers, vehicles, and environmental disturbances, the simulation provided insights into real-world challenges and solutions in public transportation. Python and Mesa proved to be effective tools for developing and visualizing agent-based models. Future work could explore:

- Optimizing vehicle routes and schedules.
- Enhancing visualization for greater clarity and user engagement.
- Enhancing visualization for greater clarity and user engagement.

Incorporating additional agent types, such as pedestrians or traffic lights, to increase the complexity and realism of the simulation.

Overall, this project highlighted the value of multi-agent systems for understanding and addressing urban mobility challenges.