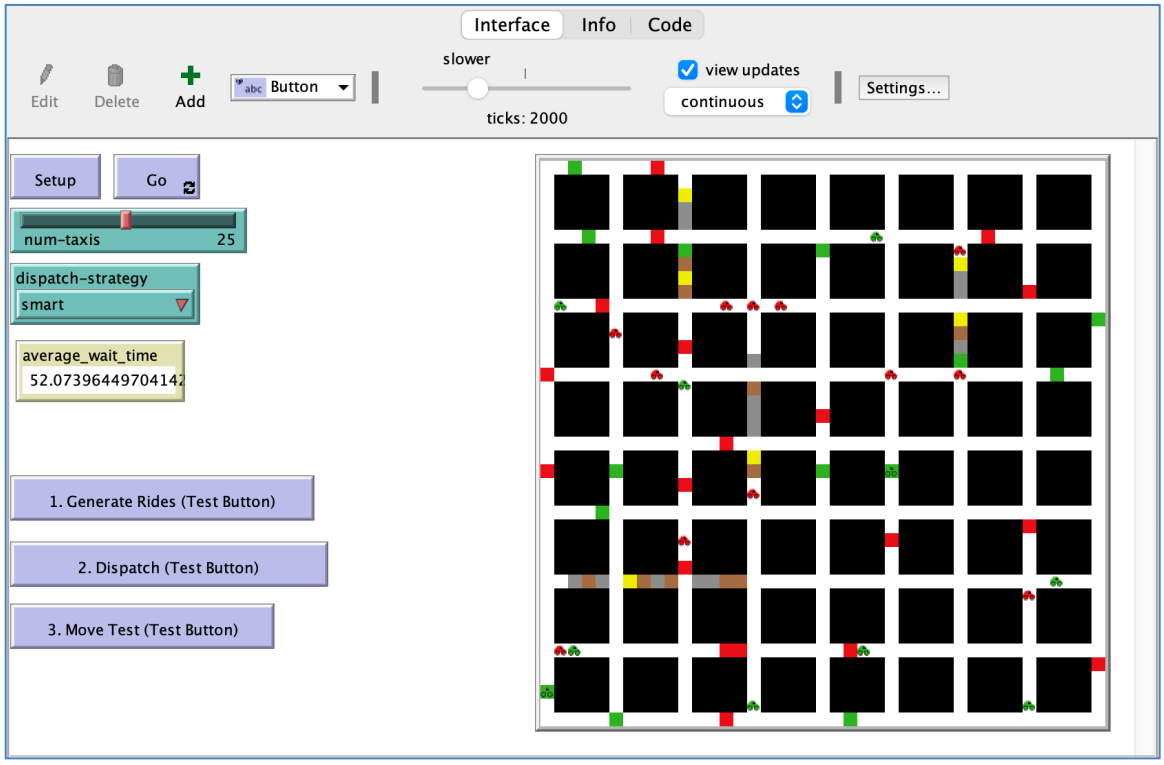
Taxi Simulation Report

Fabio Pecora & Abayomi Shosilva

Course: CSC 754 System Simulation Topics

Instructor: Dr. Feng Gu



1. Background / Problem Description

Urban mobility is an evolving challenge for modern cities. As populations increase and urban centers become more congested, efficiently managing transportation services such as taxi fleets becomes essential. One of the core tasks in these systems is dispatchin and deciding which taxi should pick up which passenger, when, and how. Poor dispatching leads to high passenger wait times, increased idle time for taxis, fuel wastage, and reduced system throughput.

Traditional dispatch systems rely on proximity, availability, or simple queueing mechanisms, which often fail in congested or dynamic environments. With the rise of app-based ride-hailing systems (e.g., Uber, Lyft), smarter algorithms that account for real-time traffic, driver distribution, and ride demand patterns have become essential.

The aim of this project is to simulate an urban taxi dispatching environment that integrates core dynamics such as ride request generation, urban congestion, and dispatching logic using the NetLogo simulation platform. This simulation allows us to test different strategies and analyze their effectiveness in minimizing average passenger wait times.

2. Modeling and Simulation (M&S) Goals

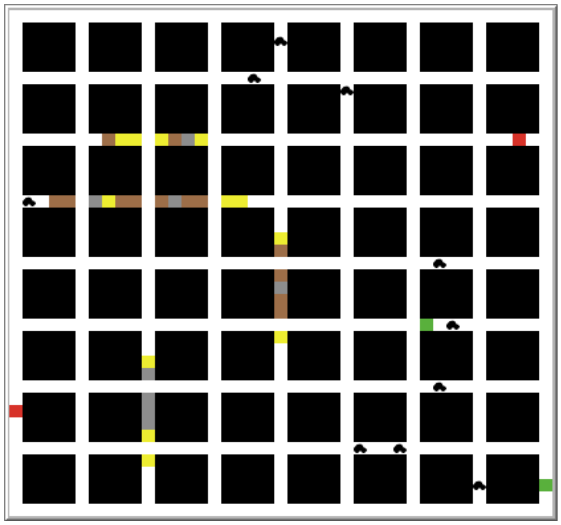
The primary goal of this project is to develop a simulation that captures the complexity and nuances of a taxi dispatching system in a city environment. By employing an agent-based modeling approach, the simulation can replicate the behaviors and interactions between taxis, passengers, and traffic conditions. Through this setup, the project aims to evaluate how different dispatching strategies influence the overall performance of the system.

More specifically, the project seeks to create a flexible and scalable simulation that can test various hypotheses related to dispatch efficiency. By analyzing metrics such as average passenger wait time, the simulation helps identify the strengths and weaknesses of different algorithms. Additionally, the visualization capabilities of NetLogo allow for a deeper understanding of the emergent behaviors in the system.

3. Developed Model

This simulation presents a top-down view of a city structured as a two-dimensional grid, where taxis serve as the primary agents.

3.1 Environment Structure

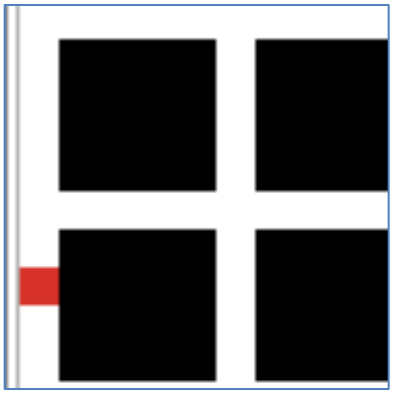


The environment in this simulation is structured as a two-dimensional grid, where each cell (or patch) represents a part of the urban landscape. Streets are laid out every five patches along both the horizontal and vertical axes, forming a network like that of a typical city grid. Taxis are constrained to move only along these street patches, ensuring a realistic representation of urban movement constraints.

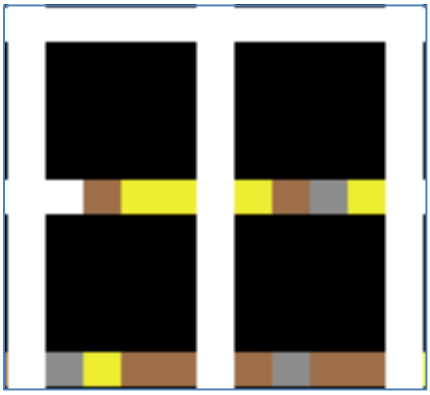
Traffic congestion is modeled directly onto the grid by assigning each street patch a traffic level ranging from 1 (no congestion) to 4 (severe congestion). These levels affect the speed at which taxis can traverse each patch. For example, patches with traffic level 4 impose significant delays, simulating the effects of real-world traffic jams. Colors are used to visually represent the traffic levels, with white indicating no traffic and gray indicating severe congestion.

3.2 Agent Types and Internal Logic

Taxis are the main agents in the simulation and are programmed with various internal states that guide their behavior. Each taxi can be idle, en-route to pick up a passenger, or transporting a passenger to their destination. These states are visually represented by color: **black** for idle, green for pick-up, and red for drop-off.



Ride requests are generated dynamically during the simulation and are stored as data structures within a global list. Each request includes a pickup and drop-off location, as well as the time it was created. While passengers themselves are not modeled as agents, the simulation tracks these requests closely and ensures they are assigned to taxis based on the current dispatching strategy.



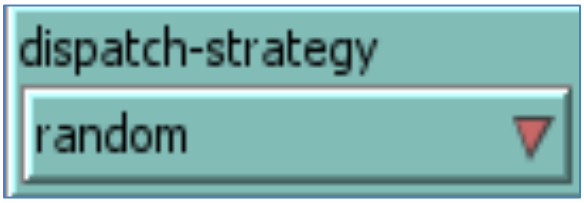
Traffic behavior is implemented through the movement delay mechanism. Depending on the traffic level of a patch, a taxi may be forced to wait several ticks before moving forward. This mechanic effectively captures the impact of congestion on travel time and route planning.

Traffic Levels:

- Level 1 - No Traffic (Color: White, Taxis move at full speed)
- Level 2 – Moderate Traffic (Color: Yellow, Minor delays in movement)
- Level 3 – Heavy Traffic (Color: Brown, Noticeable slowdowns)
- Level 4 – Severe Traffic (Color: Gray, Movement is significantly delayed)

4. Dispatch Strategies

4.1 Random Dispatch

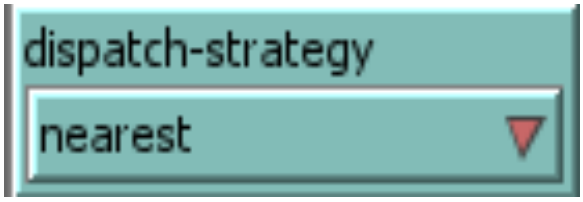


Pseudocode:
for each unassigned ride request:
    available-taxis = list of taxis that are idle and not dispatched
    if available-taxis is not empty:
        selected-taxi = randomly choose one from available-taxis
        assign request to selected-taxi
        mark selected-taxi as dispatched

In the random assignment strategy, idle taxis are paired with available ride requests without any consideration for distance or traffic conditions. This approach is extremely simple and computationally efficient, but it often leads to poor performance. Taxis may be dispatched to distant locations, resulting in long pickup times and inefficient service.

While random assignment can be used as a baseline for comparison, it does not reflect realistic dispatching logic. The absence of spatial reasoning means that the system performs poorly, especially in scenarios where traffic congestion or uneven demand distribution is present.

4.2 Nearest Dispatch



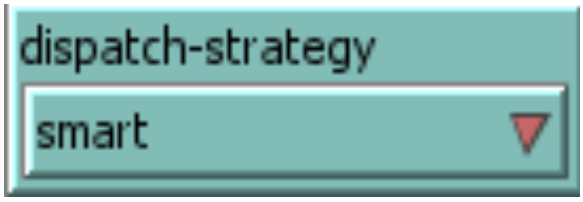
Pseudocode:
for each unassigned ride request:
    available-taxis = list of taxis that are idle and not dispatched
    if available-taxis is not empty:
        selected-taxi = taxi with the shortest Euclidean distance to request pickup location
        assign request to selected-taxi
        mark selected-taxi as dispatched

The nearest dispatch strategy improves upon random assignment by incorporating geographic distance into the decision-making process. When a ride request is generated,

the system calculates the Euclidean distance between the pickup location and each available taxi. The taxi closest to the request is then assigned the ride.

This strategy reduces travel distance and improves average wait time compared to random dispatch. However, it still lacks awareness of traffic conditions. A taxi that is geographically close but located in a high-traffic area may take longer to reach the pickup location than one further away in a low-traffic zone.

4.3 Smart Dispatch



Pseudocode:

```
for each available taxi:
    nearby-requests = find 3 closest unassigned requests
    for each request in nearby-requests:
        evaluate traffic level at pickup location
    assign request with lowest traffic to taxi
```

The smart strategy is the most sophisticated of the three and incorporates both proximity and traffic conditions into its dispatching logic. Each idle taxi evaluates a small subset of nearby ride requests (typically the three closest) and selects the one with the lowest traffic congestion at the pickup location.

This approach balances distance and travel conditions, allowing for more efficient routing and reduced passenger wait times. Although it requires more computation than the other strategies, the performance gains often justify the additional complexity. In congested environments, the smart strategy consistently outperforms the other methods.

6. Experiment Setup

Parameter: 25 taxis, 2000 ticks

Random:

Experiment #	Average Wait Time
1st	98.02
2nd	99.86
3rd	89.03
4th	106.86
5th	86.08
6th	73.86
7th	86.74
8th	80.29
9th	91.03
10th	81.24

Nearest:

Experiment #	Average Wait Time
1st	63.45
2nd	51.94
3rd	59.42
4th	61.01
5th	72.67
6th	64.14
7th	64.43
8th	65.78
9th	63.15
10th	56.85

Smart:

Experiment #	Average Wait Time
1st	45.01
2nd	38.02
3rd	52.15
4th	57.17
5th	50.42
6th	44.92
7th	42.16
8th	45.22
9th	49.64
10th	38.44

The simulation experiments were conducted under controlled conditions to ensure consistency and reliability. Each run of the simulation lasted for 2000 ticks, with a fleet of 25 taxis operating on the grid. Ride requests were generated probabilistically at each tick, provided the total number of active requests did not exceed a predefined limit.

To ensure fair comparisons, each dispatch strategy was tested using the same initial conditions across multiple runs. Key performance metrics such as average passenger wait time, number of completed rides, and taxi idle times were recorded. These metrics provided a comprehensive view of each strategy's efficiency and effectiveness.

The simulation was repeated five times for each dispatch strategy to account for variability in results. This approach allowed for the calculation of standard deviations and ensured that observed performance differences were statistically significant.

7. Experiment Results

The results of the simulation experiments highlight the differences in performance between the three dispatch strategies. The average passenger wait times for each strategy are presented in the table below:

Final Average Wait Time

Strategy	Average Wait Time
Random	89.3
Nearest	62.3
Smart	46.3

Improvements obtained:

From	To	Improvement (%)
Random	Nearest	30.25
Nearest	Smart	25.63
Random	Smart	48.13

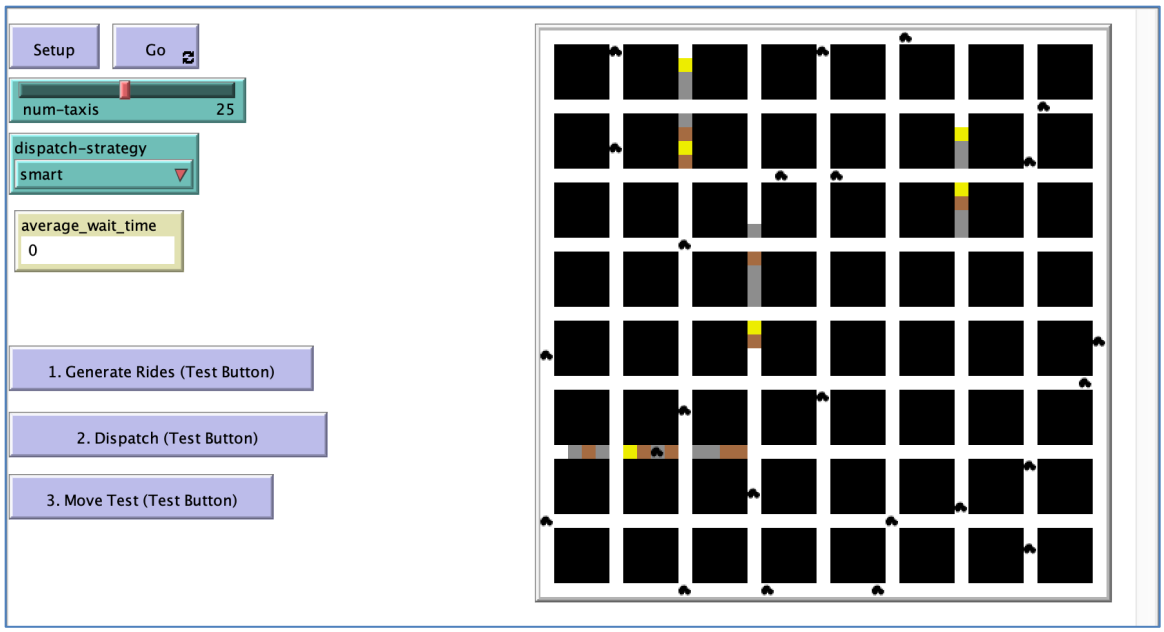
The **Random** strategy produced the highest wait times and variability, primarily due to its lack of spatial or traffic-based logic. Taxis were often sent long distances or through congested areas, leading to inefficiencies. The **Nearest** strategy improved performance by assigning the closest taxi to each request, but its disregard for traffic conditions sometimes resulted in slower pickups from high-congestion zones.

The **Smart** strategy consistently outperformed the others by considering both distance and traffic levels at the pickup location. This led to lower average wait times and a more balanced system, with higher taxi utilization and fewer unfulfilled requests. The lower standard deviation also indicates that it provided a more predictable and reliable passenger experience.

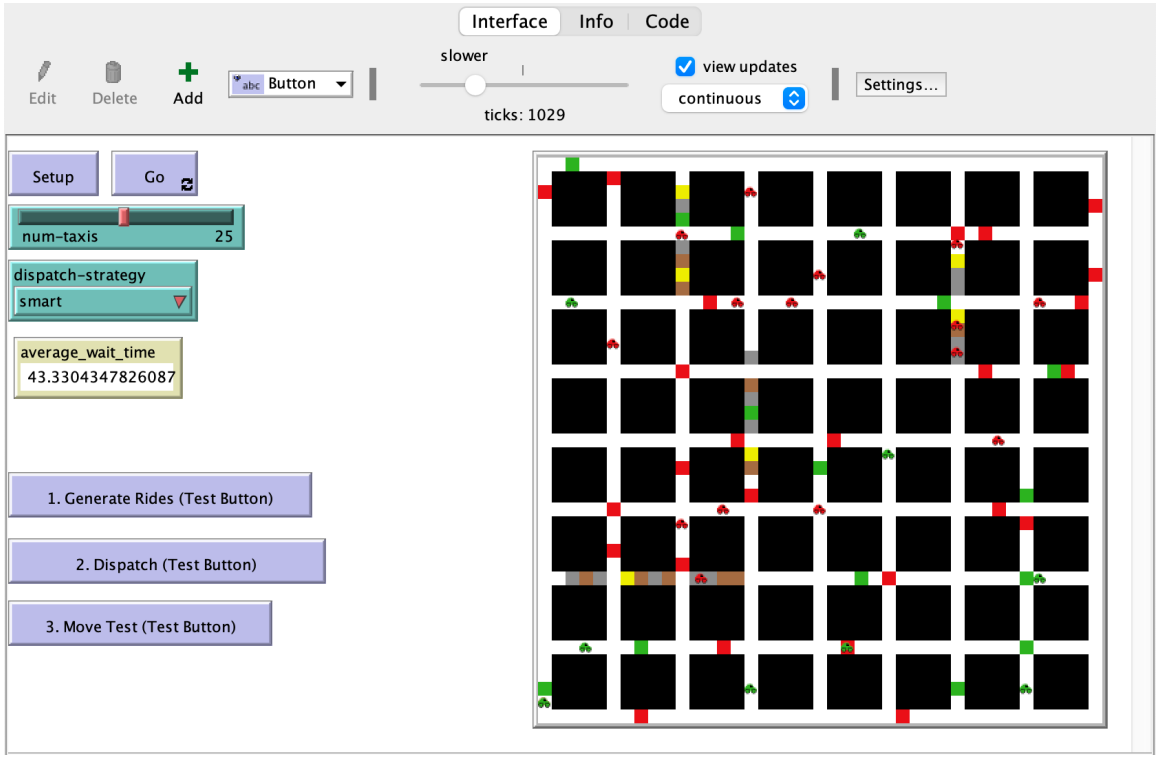
Although it introduces slightly more computational overhead, the smart strategy's performance gains suggest strong potential for use in real-world dispatch systems, particularly in urban environments where congestion is a significant factor.

8. Simulation Snapshots

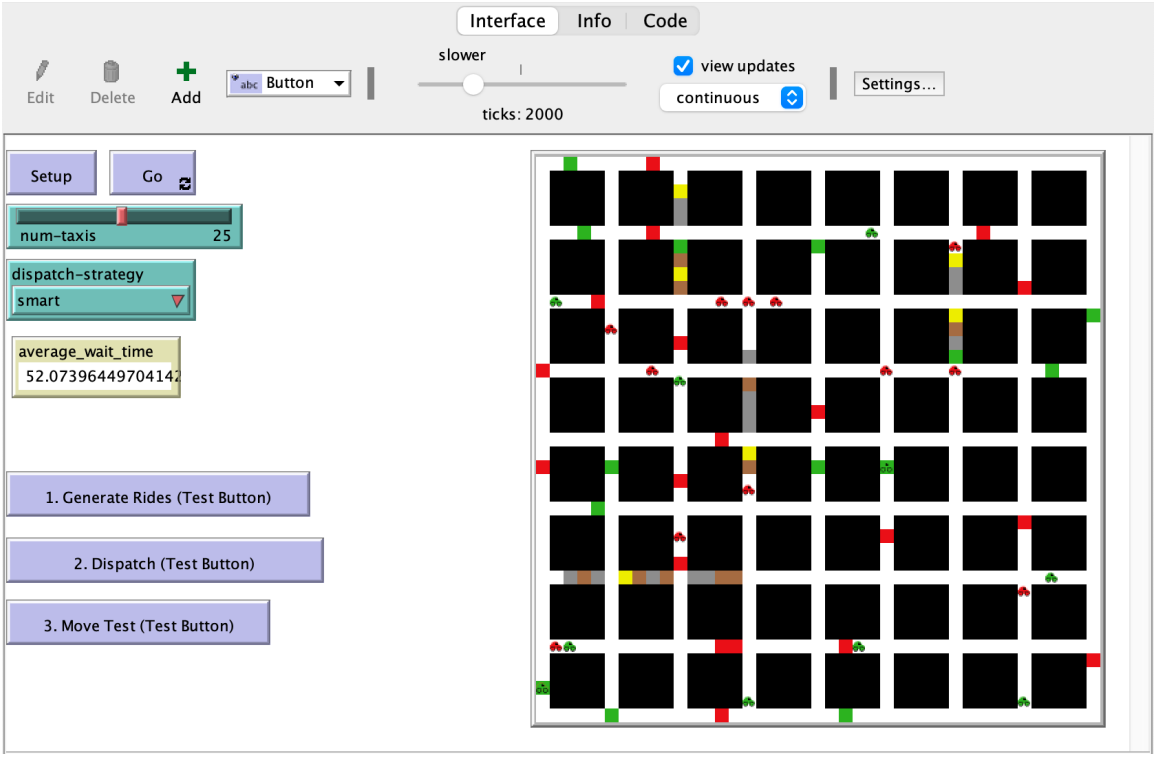
Visual snapshots from the simulation provide additional insights into the behavior of the system under each strategy. While actual images are not included here, the following descriptions illustrate key observations:



- **Initialization Stage:** Taxis (black) are randomly distributed across the grid. Ride requests begin to appear as green (pickup) and red (drop-off) markers.



- **Mid Simulation:** Under the smart strategy, taxis tend to avoid high-traffic areas and distribute themselves more evenly. In contrast, the random strategy shows taxis making inefficient, crisscross movements.



- **Late Simulation:** The smart strategy leads to a balanced system where most taxis are actively engaged in rides. The nearest strategy begins to show signs of congestion clustering, while the random strategy results in idle taxis and unfulfilled requests.

These visual patterns corroborate the quantitative results and reinforce the value of traffic-aware dispatching.

9. Conclusion

This simulation study demonstrates the impact of different dispatching strategies on the performance of a taxi transportation system. By modeling a simplified city grid with traffic congestion and dynamic ride requests, the simulation provided a controlled environment for experimentation and analysis.

The findings indicate that even modest improvements in dispatch logic—such as considering traffic conditions—can lead to substantial gains in efficiency and service quality. The smart strategy, which balances proximity and congestion, consistently outperformed simpler approaches and serves as a strong candidate for real-world implementation.

Overall, the project illustrates the power of agent-based modeling in transportation research and highlights the importance of intelligent algorithms in urban mobility.

10. Future Work

Adding several extensions to this simulation could enhance its realism and analytical depth. First, integrating full pathfinding algorithms like A* or Dijkstra's would enable taxis to plan routes based on total travel cost, including traffic delays. This would provide more accurate estimates of arrival times and improve dispatch decisions.

Second, traffic levels could be made dynamic, changing over time based on usage patterns or simulated accidents. This would reflect the fluctuating nature of urban congestion and allow for testing of adaptive dispatch strategies.

Third, passenger behavior could be modeled in greater detail. Features such as ride cancellation, impatience, and preferences for shorter routes would add complexity and realism. These elements could also serve as additional metrics for evaluating dispatch strategies.

Finally, incorporating infrastructure elements like traffic lights and stop signs could further simulate real-world travel conditions. These enhancements would allow the simulation to serve as a more comprehensive testbed for evaluating smart city transportation policies.

11. References

1. Wilensky, U. (1999). *NetLogo*. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University.