# Taxi Dispatcher Simulation

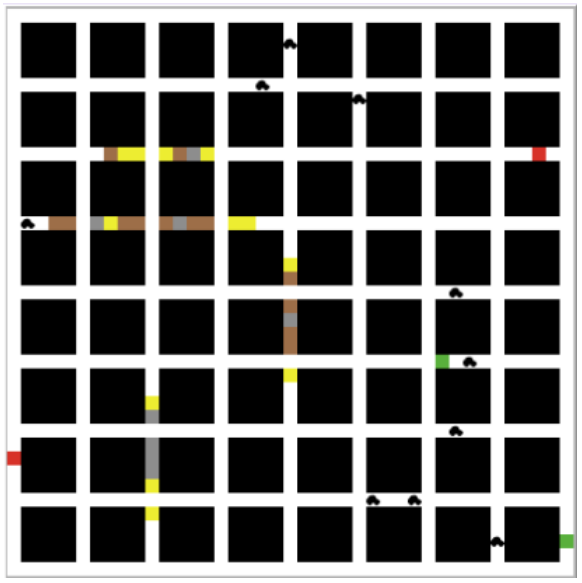
Fabio Pecora & Abayomi Shosilva

## 1. Introduction

This project presents the development of a taxi dispatch simulation designed to model urban ride-sharing dynamics within a city grid. Implemented in NetLogo, the simulation replicates the core processes involved in matching taxis with passenger ride requests while accounting for real-world complexities such as traffic congestion and dispatch strategy. The primary goal of this simulation is to evaluate the impact of different dispatching algorithms on average passenger wait times, allowing insights into how algorithmic choices affect transportation efficiency in congested urban environments.

## 2. Simulation Environment

The simulated city is constructed as a two-dimensional grid where streets appear every five patches in both horizontal and vertical directions. Only these street-designated patches are navigable by taxi agents, ensuring a realistic constraint on movement. The environment supports dynamic ride generation and includes visual indicators for ride pickups, drop-offs, and taxi states.



Traffic congestion is modeled directly onto the grid. Street patches are initialized with a default traffic level of one, representing no congestion. A small number of these streets are then randomly selected to simulate localized traffic jams, increasing their congestion levels up to a maximum of four. These levels influence how frequently a taxi can move across a patch, effectively simulating delays. Each traffic level is visually distinguished: white for no traffic, yellow for moderate congestion, brown for heavy traffic, and gray for severe congestion.

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

## 3. Agent Design and Behavior

Taxis are implemented as mobile agents that begin the simulation scattered randomly across street patches. Each taxi maintains internal state variables such as whether it has a passenger, whether it has been dispatched, its current destination, movement speed, and the time a ride was assigned.

- **Black**
  - Idle, waiting for a ride.
- **Green**
  - Moving toward a pick-up location.
- **Red**
  - Moving toward a drop-off location.

Passengers are not directly represented as agents. Instead, ride requests are abstracted as data structures containing the coordinates of pickup and drop-off locations, along with the tick count at which the request was generated. These requests are stored in a global list and are progressively assigned to taxis depending on the selected dispatching strategy.

## 4. Ride Request Generation and Assignment

Ride requests are generated probabilistically during each simulation tick, provided the total number of active requests remains below a specified threshold. Each request includes a unique pickup and drop-off location, both chosen randomly from available street patches. When generated, the pickup location is marked in green and the drop-off in red, offering a visual indication of current ride demand on the grid.

Assignment of requests to taxis occurs based on one of three dispatching strategies: random, nearest, and smart. Each strategy defines a different way to pair available taxis with waiting ride requests, influencing both how quickly passengers are picked up and how balanced the taxi workload is.

## 5. Dispatch Strategies

The random strategy performs assignment without regard to distance or traffic, simply matching taxis and requests arbitrarily. This method requires minimal computation but often results in highly inefficient service due to long travel paths and unbalanced workloads.

The nearest strategy improves upon randomness by assigning each request to the geographically closest available taxi based on Euclidean distance. While this method reduces pickup distances, it does not consider traffic levels, potentially sending taxis through congested areas.

The smart strategy introduces traffic awareness. Each taxi evaluates its three closest unassigned ride requests and selects the one with the lowest traffic level at the pickup location. This approach balances proximity and congestion, yielding better performance but at the cost of increased computational overhead.

## 6. Performance Measurement

The simulation tracks the efficiency of each strategy using average passenger wait time as the key performance metric. A wait time is defined as the number of ticks that pass between when a ride request is created and when a taxi picks up the passenger. This value is recorded for each completed ride and stored in a list from which the average can be computed at any time. This simple yet meaningful metric enables comparative evaluation of the different dispatching approaches.

## 7. Observations

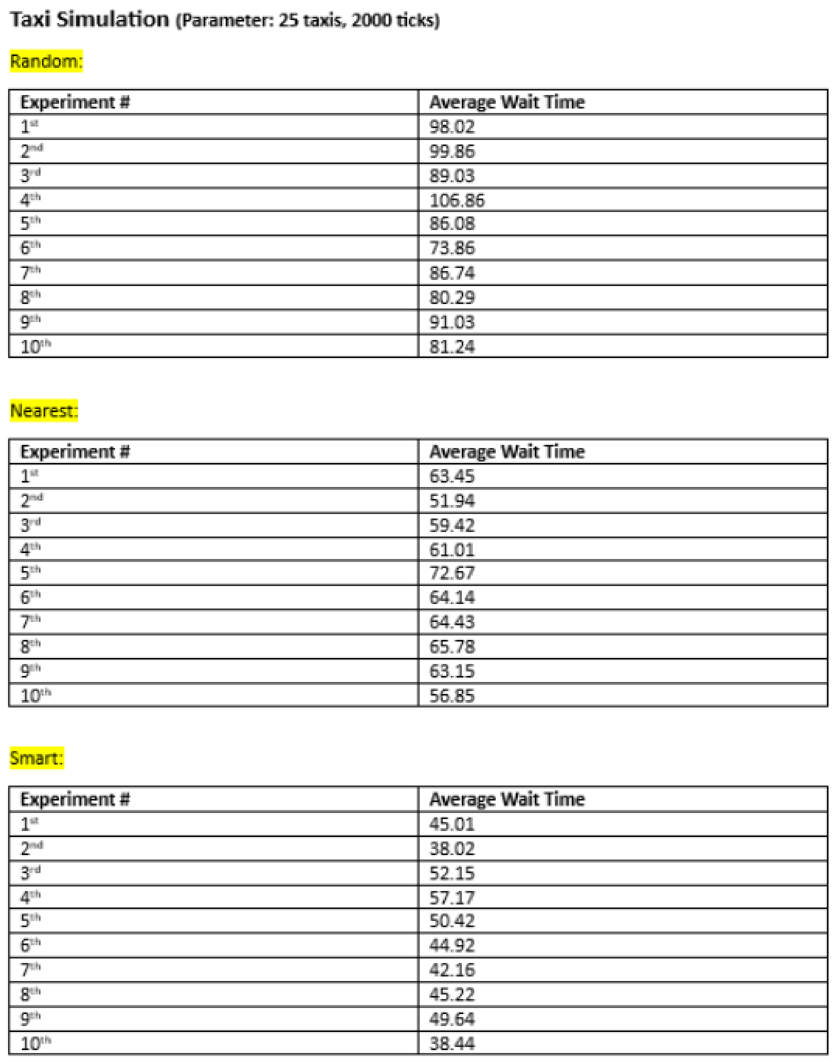
During experimental runs, the performance of each strategy was evaluated qualitatively by observing taxi behavior and quantitatively by monitoring average wait times. The random strategy consistently produced the highest wait times due to the lack of spatial logic in taxi assignment. The nearest strategy improved performance but suffered in high-traffic scenarios where Euclidean proximity did not reflect true travel cost. The smart strategy achieved better balance by avoiding congested areas at pickup time, resulting in lower average wait times in most runs.

## 8. Experiment and Evaluation

To evaluate the practical effectiveness of each dispatch strategy, we designed and conducted a controlled experiment. The simulation was executed using each of the three dispatching methods: Random, Nearest, and Smart, across multiple independent runs with identical initial conditions. Each run lasted for 2000 ticks, used 25 taxis, and key metrics such as average passenger wait time were recorded.

The Random strategy consistently produced the highest wait times with an average of 89.3 ticks, confirming its inefficiency in matching taxis with passengers effectively. The Nearest strategy reduced this average substantially, maintaining wait times around 62.3 ticks. While more efficient than Random, Nearest failed to consider traffic congestion, often routing taxis through jammed areas.

Smart dispatch showed a marked improvement, with average wait times commonly falling in the range of 40–50 ticks with an average of 46.3 ticks. By considering traffic levels at pickup points, this strategy was more effective at navigating congestion while maintaining reasonably short travel distances.



**Taxi Simulation** **(Parameter: 25 taxis, 2000 ticks)**

Random:

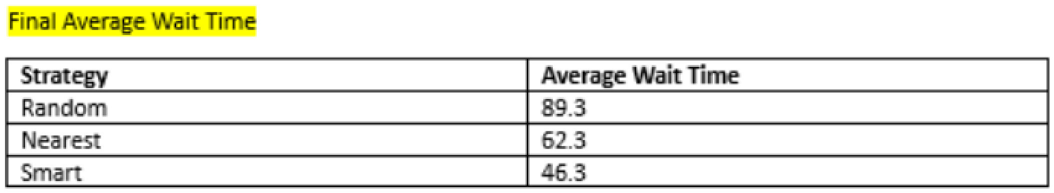
| Experiment # | Average Wait Time |
|---|---|
| 1st | 98.02 |
| 2nd | 99.86 |
| 3rd | 89.03 |
| 4th | 106.86 |
| 5th | 86.08 |
| 6th | 73.86 |
| 7th | 86.74 |
| 8th | 80.29 |
| 9th | 91.03 |
| 10th | 81.24 |

Nearest:

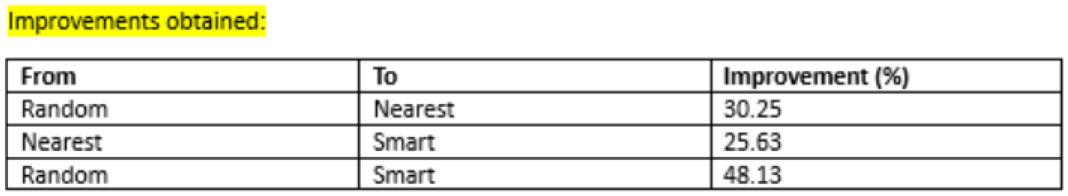
| Experiment # | Average Wait Time |
|---|---|
| 1st | 63.45 |
| 2nd | 51.94 |
| 3rd | 59.42 |
| 4th | 61.01 |
| 5th | 72.67 |
| 6th | 64.14 |
| 7th | 64.43 |
| 8th | 65.78 |
| 9th | 63.15 |
| 10th | 56.85 |

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 |


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 |

## 9. Future Work

We would love to implement in the future a full Dijkstra or A* pathfinding algorithm, because it would enable accurate cost-based dispatching and remove the guesswork from path planning. Additionally, traffic levels remain static during the simulation; incorporating real-time traffic updates would simulate urban dynamics more realistically.

Passenger behavior is also simplified, with no consideration for cancellation, impatience, or rerouting preferences. Adding such behaviors would introduce important stochastic elements and further challenge the dispatching algorithms. Finally, integrating traffic lights or intersection-based delays could better emulate real-world travel conditions and open up opportunities for more advanced traffic-aware planning.

## 9. Conclusion

This simulation highlights how different dispatching strategies impact urban mobility, especially under varying traffic conditions. Through the use of NetLogo, we created a controllable environment where strategies could be isolated and studied in a visually and computationally rich model. The results show that even modest algorithmic improvements, such as adding traffic awareness, can significantly reduce wait times and improve service quality. With further refinements such as real-time pathfinding and dynamic traffic, the simulation has the potential to model real-world dispatch systems more accurately and serve as a platform for experimenting with urban transportation policies and innovations.