Modelling Urban Traffic Flow Using Cellular Automata: A Nagel-Schreckenberg (CA) Simulation

Athetheyan Jayakumar
Student ID: 1223049
jayakuma@uoguelph.ca

Syed Mahdi Rehan
Student ID: 1210377
syedmahd@uoguelph.ca

Syed Haadi Rehan
Student ID: 1210383
syedhaad@uoguelph.ca

1. Abstract

Urban traffic congestion is a growing issue contributing to longer commute times, increased fuel consumption, and elevated pollution levels. Traditional traffic models often fail to capture the complexity of vehicle interactions and environmental conditions in real-world scenarios. This project explores an alternative approach using **Cellular Automata (CA)**, a computational model well-suited for simulating dynamic, rule-based systems such as traffic flow. Specifically, we implement the **Nagel-Schreckenberg (NaSch) model**, a unique CA model that captures essential vehicle behaviors like acceleration, deceleration, vehicle slowdowns and lane merger [5]. Our model extends the standard NaSch implementation by incorporating environmental features such as dynamic traffic signals, lane-changing logic, and merging lanes to reflect urban traffic conditions more accurately.

Simulations are conducted across varying traffic densities, signal timing strategies, and road configurations. The results are visualized through congestion heatmaps, flow-versus-density plots, and visual traffic grids. Key findings highlight how dynamic signal timing, traffic signal placement and lane management strategies influence vehicle congestion levels. Furthermore, this project demonstrates the effectiveness of Cellular Automata in modelling complex traffic systems and provides insights that may be valuable for urban planning and traffic engineering applications [4][1].

2. Introduction

Traffic congestion remains a major concern in urban environments, leading to significant economic and environmental costs. With rising urban populations and vehicle ownership, transportation systems are under increasing stress. Traditional traffic flow models, often based on equations or old data, struggle to accurately reflect the interactions between individual vehicles, road structures, and environmental variables.

Cellular Automata (CA) offers a promising alternative due to its simplicity, flexibility, and computational efficiency [2][6]. CA models operate on a grid where each cell updates its state based on localized rules, making them well-suited for simulating complex systems like traffic. One of the most widely used CA models in traffic research is the **Nagel-Schreckenberg (NaSch) model**, which simulates essential behaviors such as acceleration, slowing due to leading vehicles, random braking, and forward movement [5].

In this project, we implement and expand the NaSch model to better represent real-world urban traffic. Our enhancements include **dynamic traffic signal control, lane-changing behavior, and traffic signal-induced congestion**. These additions evaluate how environmental and behavioral factors influence traffic flow and congestion. The project is implemented in Python using a Jupyter Notebook, emphasising interactive visualization and user-adjustable simulation parameters.

3. Project Objectives

The primary objective of this project is to implement a traffic simulation based on the NaSch model using Python. The simulation is designed to replicate realistic vehicle behaviors such as acceleration, deceleration, traffic signal slowdowns, and lane changing [1][4].

Additional complexities are introduced to the model in the form of (dynamic) traffic signals, merging lanes, and slow zones. The goal

is to create a simulation that models traffic under varying conditions and provides visual outputs that allow for meaningful evaluation of traffic patterns and potential interventions. The **hypothesis** is that increasing dynamic signal control would improve flow under moderate density [8].

4. Background and Related Work

In 1992, Kai Nagel and Michael Schreckenberg developed the NaSch model [5]. This CA-based model treats the road as a one-dimensional or two-dimensional grid. Each cell in this grid can either be empty or occupied by a vehicle. Vehicles have discrete velocity values ranging from zero to a maximum speed, denoted as 'vmax'. This framework has been extensively analyzed and expanded upon in broader reviews such as those by Chowdhury et al. [1] and Maerivoet & De Moor [4], both of which survey a range of CA traffic models and highlight the significance of incorporating real-world phenomena like braking probability and lane-changing into CA-based traffic simulations.

The NaSch model applies a four-step sequence at each time step:

1. Acceleration – A vehicle increases its speed by one unit if it is below vmax.

2. Traffic Signal-Induced Slowdowns – The speed is reduced if the vehicle approaches a red traffic signal or a traffic signal zone. More traffic signals lead to more frequent vehicle braking, resulting in slower vehicles and increased slowdowns across the road network.

3. Randomization – With a defined probability, a vehicle may randomly reduce its speed.

4. Vehicle Movement – The vehicle moves forward by the number of cells equal to its speed.

In addition to transportation modelling, CA models have been used to simulate wildfire spread, disease contagion, and pedestrian flow. They are particularly valuable in traffic research due to their scalability, speed, and interpretability [6]. Downey's "Think Complexity" introduces CA and the NaSch model as educational tools for understanding emergent behavior [9]. Our project adapts these principles and enhances them with realistic traffic dynamics, expanding the simulation's applicability to urban planning scenarios.

Lastly, John Hunter's article on Matplotlib [3] supports the technical implementation of visualization methods, enabling effective representation of the simulation results through time-series plots, heatmaps, and congestion maps, which were crucial for evaluating traffic patterns under different scenarios.

5. Model Approach and Methodology

Our implementation builds on the NaSch model as a foundation to simulate and analyze urban traffic dynamics, extending it into a modular, multi-lane simulation framework written in Python [5]. The simulation operates on a 2D NumPy grid. Each row represents a traffic lane, and each column represents a cell in that lane, where each cell may either be empty or contain a vehicle moving at a certain velocity. Vehicles move according to pre-determined NaSch rules that govern acceleration, braking, and lane movement, making CA an ideal base for complex traffic pattern simulations [4][5].

The model was extended to ensure realism and flexibility with additional behavioural components, including lane-changing logic, merging behaviours, and responses to dynamic red/green traffic signals. Each of these enhancements was incorporated into the simulation code, making it easy to adjust parameters.

5.1 Simulation Architecture

The simulation is developed in Python using a modular, object-oriented design that ensures clarity, reusability, and scalability [2]. The architecture is organized around a class-based system, clearly separating responsibilities between different traffic system components. This structure facilitates parameter experimentation and scenario customization within the simulation.

5.1.1 Core Components

1. **'TrafficSimulation' Class:**
   This is the central class that co-ordinates the entire simulation. It is responsible for:
   a. Initializing the road network as a 2D grid (lanes × road length).
   b. Populating the road with vehicles based on the specified **density.**
   c. Updating vehicle states at each timestep according to NaSch rules.
   d. Handling special zones such as **slow zones** and **merging points.**
   e. Managing vehicle interactions with **traffic signals.**
   f. Recording metrics like **flow rate**, **average speed**, and **vehicle throughput.**
   g. Running full simulations over a given number of timesteps.
2. **'TrafficSignal' Class:**
   This class models **traffic light behavior**. Each signal can be configured with:
   a. A customizable **cycle time.**
   b. Green-to-red duration ratios.
   c. Signal phase tracking (i.e., determining whether the signal is currently green or red) These signal objects are strategically placed on the road and integrated into the vehicle update logic to control stop-and-go behavior.
3. **'traffic_signal_quantity' Function/Parameter:**
   a. Rather than manually placing each signal, this parameter controls the **density of traffic signals** along the road.
   b. A higher value results in more frequent signal-induced interruptions, contributing to reduced average speeds and increased vehicle braking. The logic ensures that signal distribution is consistent with real-world road

environments, simulating the effects of signal timing on congestion levels.

4. **'visualize_simulation()' Function**
   a. This function generates a **real-time heatmap-style visualization** of the simulation state. It shows the positions of vehicles across multiple lanes, the speed of each vehicle (using a color gradient), road occupancy, signal positions, and slow zones. It provides intuitive insight into how vehicle distributions and speeds evolve over time. This is complemented by lower plots showing **flow rate dynamics** across the simulation duration.

The Python implementation of the model is carried out in a Jupyter Notebook environment. The code is modular and allows for the inclusion of various enhancements and user-defined input parameters.

5.2 User Defined Input parameters

Several key parameters can be adjusted to create different test cases to make the simulation flexible and adaptable to different test cases, facilitating comparative studies of traffic behaviour under various scenarios:

1. Main Simulation Parameters (in the main() function):

road_length: The length of the road

num_lanes: Number of lanes in the road

density: Vehicle density on the road

vmax: Maximum vehicle speed (units per time step)

traffic_signal_quantity: Probability of traffic signal induced slowdowns.

2. Traffic Control Parameters (can be added/modified):

a. Traffic Signals:
   position: Where the signal is placed on the road
   cycle_time: Total time for one signal cycle
   green_ratio: Proportion of time the light is green

b. Slow Zones:
   start_pos: Starting position of the slow zone
   end_pos: Ending position of the slow zone
   speed_limit: Maximum speed allowed in the zone

c. Merge Points:
   position: Where vehicles can merge onto the road

3. Simulation Runtime Parameters:

num_steps: Number of simulation steps (executions) to run.

Here are some example test cases you could simulate by adjusting these parameters:

1. **High Traffic Density Scenario**:

```
road_length = 100
num_lanes = 3
density = 0.8  # 80% road occupancy
vmax = 5
traffic_signal_quantity = 0.1
```

2. **Low Traffic Density with Fast Vehicles**:

```
road_length = 100
num_lanes = 2
density = 0.1  # 10% road occupancy
vmax = 8  # Higher maximum speed
traffic_signal_quantity = 0.05  # Less vehicle slowdowns
```

3. **Heavy Traffic Control Scenario**:

```
road_length = 100
num_lanes = 3
density = 0.4
vmax = 5
traffic_signal_quantity = 0.2
# Add more traffic signals
sim.add_traffic_signal(position=20, cycle_time=15, green_ratio=0.5)
sim.add_traffic_signal(position=40, cycle_time=15, green_ratio=0.5)
sim.add_traffic_signal(position=60, cycle_time=15, green_ratio=0.5)
sim.add_traffic_signal(position=80, cycle_time=15, green_ratio=0.5)
```

4. **Slow Zone Scenario**:

```
road_length = 100
num_lanes = 3
density = 0.3
vmax = 5
traffic_signal_quantity = 0.1
# Add a long slow zone
sim.add_slow_zone(start_pos=30, end_pos=70, speed_limit=1)
```

5. **Highway Merge Scenario**:

```
road_length = 100
num_lanes = 3
density = 0.4
vmax = 5
traffic_signal_quantity = 0.1
# Add multiple merge points
sim.add_merge_point(position=25)
sim.add_merge_point(position=50)
sim.add_merge_point(position=75)
```

The main() function parameters would need to be modified, and the traffic control elements would need to be adjusted to run these scenarios. The visualization will show how these parameters affect traffic flow, congestion patterns, and overall system behavior. The user can also combine different elements, for example, having traffic signals and slow zones in the exact simulation or creating a high-density scenario and multiple merge points to study traffic merging behavior under heavily congested conditions.

This set of controlled simulation scenarios (low density vs. high density, static vs. dynamic signals) was implemented to observe and evaluate emergent traffic patterns such as congestion zones, flow breakdown, and stop-and-go waves. The outcomes were visualized through dynamic matplotlib animations, time-series plots, and heatmaps, demonstrating the impact of different input conditions on overall traffic efficiency [3].

5.3 Feature Enhancements

5.3.1 Traffic Signals

Each signal, implemented using the TrafficSignal class, has a position, a cycle time, and a green/red light ratio. Traffic signals are introduced as periodic red-green cycles that control the movement of vehicles at designated intersections. Vehicles approaching a red signal will decelerate and stop, simulating realistic behavior at traffic lights.

5.3.2 Lane Changing

Vehicles attempt to change lanes when blocked. The logic prioritizes movement to an adjacent lane if there is enough space and a potential to move faster than in its current lane. The `_check_lane_change()` function verifies the safety and efficiency of the maneuver.

5.3.3 Traffic Slowdowns

Traffic signal quantity slowdowns model the relationship between the number of traffic signals on a road segment and the frequency of vehicle slowdowns due to signal-induced braking. It assumes that each traffic signal introduces a potential stop point, increasing the likelihood of vehicle slowdowns increasing traffic jams likelihoods.

The model includes slow zones by reducing the maximum allowed speed in certain road sections. This mimics construction zones or high-traffic urban stretches and school zones.

5.3.3.1 Merging Lanes

Merging lanes are modeled to allow vehicles to enter the main road following rules that prevent collisions and ensure smooth merging. Priority and safety checks are incorporated to reflect real-world merging behavior.

5.4 Use of Development Tools.

AI-guided tools such as ChatGPT and Cursor AI were utilized to enhance the initial planning and design of the simulation framework. These tools assist in conceptualizing the architecture of the Cellular Automata model, including how to segment logic into well-defined functions for vehicle behavior, signal timing, and visualization modules. Additionally, they offered valuable suggestions on organizing the Jupyter Notebook and promoting readability, maintainability, and clarity of simulation stages. While the code itself was developed and iteratively refined through conventional testing and debugging practices, AI tools provided a helpful guide for laying out the foundational logic, accelerating the design process.

Key libraries such as **NumPy** were used for efficient array manipulation and state updates within the Cellular Automata grid. At the same time, **Matplotlib** was the primary tool for visualizing simulation outcomes, including congestion heatmaps, flow-density plots, and time-step animations.

Furthermore, **matplotlib.use('Agg')** is a powerful backend system for Matplotlib, which enables smooth visual rendering of the simulation. The backend efficiently handles multiple visual elements simultaneously, from moving vehicles at various speeds and slow zones to traffic signals, lane changing, merging and flow rate graphs, making it an ideal choice for dynamic simulations and applications that require reliable and responsive visual output.

In addition to using these tools and frameworks, a significant portion of the simulation logic, including vehicle update rules, lane-changing algorithms, traffic signal behaviours, and merging mechanisms, was custom-coded. The decision to implement core functionality manually allowed for greater control and adaptability in testing different and customizable traffic conditions.

5.5 Visualization and Analysis

The visualization of the simulation outcomes is a crucial part of this project. Using the matplotlib library, the notebook generates various types of visual outputs:

1. Time-Series Plots: Show the positions and velocities of vehicles at each timestep.

2. Congestion Heatmaps: Provide a lane-wise view of traffic density to identify congestion zones.

3. Flow-Density Graphs: Illustrate the relationship between vehicle density and throughput.

4. Traffic Jam Identification: Yellow highlighted zones with frequent braking and vehicle congestion.

These outputs allow users to observe the evolution of traffic conditions and identify inefficiencies.

5.6 Breakdown and Explanation of Visual Output

The simulation visualization consists of **two main sections**, each offering insight into traffic behavior and performance across the simulated environment.

1. **Upper Section - Traffic State Visualization**:

This section displays a grid-based snapshot of the road, showing the distribution and movement of vehicles across multiple lanes over time.

- **Axes**:
  - **Vertical Axis (Y-axis):** Represents Road lanes.
  - **Horizontal Axis (X-axis):** Represents positions along the road length.
- **Road Representation**:
  - **White spaces**: Indicate empty road segments, suggesting free-flowing space.
  - **Black spaces**: Indicate occupied segments, potentially due to congestion or stopped vehicles.
  - **Colored dots**: Represent vehicles, with dot color reflecting speed.
- **Speed Color Gradient**:
  - **Purple / Blue**: **Slower vehicles**
    - Clusters of purple indicate congestion or slowdowns.
  - **Green / Yellow**: **Faster vehicles**
    - Yellow or green tones reflect smooth, uninterrupted flow.
- **Additional Features**:

- **Yellow vertical highlighted strip**: Highlights a slow zone, such as a school zone, pedestrian crossing, or traffic area, where vehicles are forced to reduce speed.
- **Vertical dotted lines**: Represent traffic signal positions, if present.

2. Lower Section – Flow Rate Graph

This plot illustrates how the overall vehicle flow evolves throughout the simulation.

- **Axes**:
  - **X-axis**: Time steps of the simulation.
  - **Y-axis**: **Flow rate**, measuring the number of vehicles successfully moving through the system.
- **Line Graph**:
  - **Blue Line**: Tracks the flow rate over time.
    - Peaks reflect periods of efficient traffic movement and better vehicle traffic flow.
    - Dips in the line indicate reduced traffic flow, suggesting congestion or interruptions.

6. Simulation Results and Discussion

The simulation was tested across multiple scenarios to obtain meaningful results:

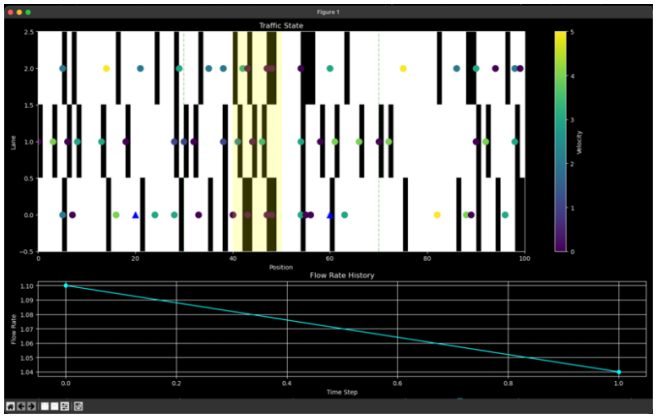

Figure 1: Low Traffic Conditions

Under low traffic conditions, the simulation demonstrates efficient vehicle flow and higher average speeds. Low vehicle density and less traffic signal quantity contribute to minimal interruptions throughout the road. As shown in the visual output, vehicles are well-distributed across lanes and maintain higher velocities, indicated by the prevalence of brighter-colored markers on the velocity scale. More lanes allow vehicles to manoeuvre freely and avoid slowdowns, further enhancing traffic flow. Moreover, the limited number of traffic signals, and increased dynamic signal probability results in fewer braking events, allowing most vehicles to accelerate to near-maximum speeds. This leads to a relatively high and stable flow rate, as illustrated in the flow rate history plot. The gradual decline is minimal, signifying small congestion buildup. Overall, the simulation validates that lower density, increased **dynamic** traffic signals, and adequate lane capacity significantly optimizes traffic flow efficiency in urban settings.

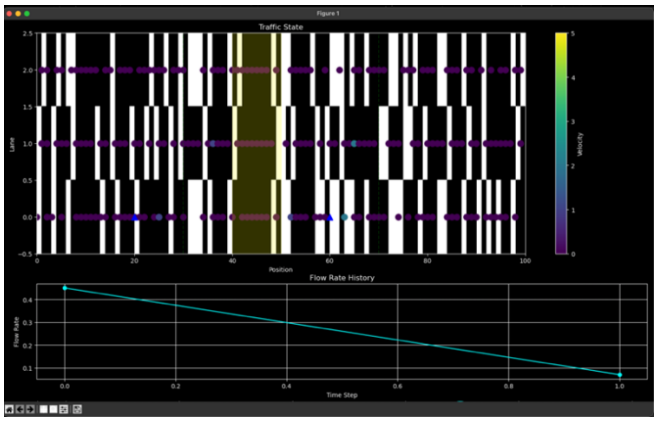

Figure 2: High Traffic Conditions

The simulation shows significant congestion and reduced traffic performance in the high-traffic scenario, with a density of 0.8 and three lanes. Most vehicles appear in dark purple, indicating slow or stopped motion due to crowding and frequent braking triggered by the high quantity of traffic signals (traffic signal quantity of 0.6), and less dynamic traffic signals. With limited free space and high vehicle interaction, flow is restricted, resulting in a sharp drop in the flow rate, as shown in the flow rate history. The higher vehicle density overwhelms lane capacity, leading to consistent slowdowns and clustering near signal zones. Hence, this confirms that increasing density and traffic signal quantity reduce overall efficiency, even when maximum speed remains high.

The results presented in this project were obtained through a structured, logical, and repeatable simulation framework based on well-established traffic modelling principles. The project adopted the Nagel-Schreckenberg (NaSch) model as its core, ensuring that the underlying logic of vehicle dynamics was realistic and validated in the research literature. The implementation involved rule-based vehicle simulation and randomization elements (e.g. slowdowns, lane mergers and traffic signal placements), all mirroring real-world traffic behavior in a simplified form.

Each experimental scenario was carefully defined by varying key traffic parameters such as vehicle density, number of lanes, maximum speed, and the presence of dynamic traffic signals. The simulation facilitated meaningful comparisons between low-traffic and high-traffic conditions, static vs. dynamic signal control, and different road configurations by isolating one or two variables at a time while holding others constant. This controlled approach allowed observing direct correlations between model inputs and emergent traffic behaviors, including flow rate trends and congestion buildup.

The results were visualized using real-time and static plots (such as congestion heatmaps and flow-density graphs) and interpreted in the context of traffic theory. Trends observed in the simulation, such as reduced vehicle densities, more lanes, and fewer traffic signals, are consistent with known behaviors in traffic systems flow, further validating the simulation's accuracy and the sensibility of its outcomes.

7. Key Findings from Results

The study revealed key patterns associated with vehicle density, dynamic traffic signals, and road configuration through systematic simulations.

1. Impact of Vehicle Density: Vehicle density has a nonlinear impact on traffic flow efficiency. At low densities, traffic remains fluid with minimal delays. However, as density increases beyond a critical threshold, congestion increases rapidly, and overall flow decreases.

2. Dynamic vs. Static Signals: Dynamic traffic signals were found to outperform static signals by adapting to current traffic conditions, hence reducing average vehicle wait times.

3. Effectiveness of Merging Logic: Intelligent merging behavior significantly improved vehicle flow at entry points, reducing vehicle congestion and enhancing lane utilization.

In **low-density scenarios**, vehicles experienced minimal interaction and were able to maintain high speeds across multiple lanes, resulting in smooth flow. Visualizations showed mostly high-speed vehicles, with flow rate graphs indicating steady and efficient movement. Fewer traffic signals further contributed to this outcome by reducing braking interruptions.

Conversely, in **high-density scenarios**, traffic flow became unstable and congested. Vehicles frequently slowed down or stopped due to proximity and repeated signal-induced braking, leading to the emergence of stop-and-go wave patterns in the flow rate graph, resulting in a decline in flow rate. The colour-coded traffic grid diagrams for these simulations illustrated clusters of slow-moving vehicles, particularly around traffic signals and merging zones.

Additionally, the comparison between static and dynamic traffic signal timing revealed that **dynamically adjusted signals** helped reduce vehicle delay and prevent unnecessary stops, especially in moderately dense traffic.

8. Validation and Model Testing

To ensure that the simulation behaves as intended and produces meaningful results, the following assessments were conducted:

Validation Checks: At low vehicle densities with no traffic signals or slow zones, vehicles were expected to move at or near the maximum speed. The simulation confirmed this behavior, with near-zero congestion and consistently high throughput, which validates correct acceleration and movement logic.

Boundary Testing: Extreme parameter values (e.g., density = 0 or density = 1) were tested. At density = 0, no vehicles were placed, and the simulation returned a blank grid with zero flow, as expected behavior. At density = 1, every cell was filled, leading to immobile traffic with zero flow, matching theoretical expectations.

Component Isolation: Each traffic feature (lane changes, signals, slow zones, merges) was independently enabled and tested to ensure that it triggered expected vehicle behaviors (e.g., stopping at red signals, slowing in speed-limited zones).

These internal checks and visually intuitive outputs (heatmaps and flow plots) confirmed that the Cellular Automata logic and enhancements were functioning as expected.

9. Limitations and Future Work

While the presented simulation captures essential vehicle behaviors and demonstrates emergent traffic patterns, some limitations restrict its scalability:

Lack of Predictive Analytics or Control Systems: While the model demonstrates emergent behavior well, it does not include predictive control mechanisms. Future development could explore adaptive signal control using AI agents, which are assigned tasks that could give predictive maintenance traffic plans to optimize traffic flow.

Limited Real-World Integration: The simulation uses artificial and simulated parameters. Future development could use parameters from real traffic datasets integrated with GPS and GIS to map out the data to test against actual congestion events.

No Consideration of Environmental Impact: The simulation does not measure emissions or energy consumption. Since many traffic interventions aim to reduce CO_2 and fuel use, adding such metrics would improve environmental impact analysis.

10. References

[1] Chowdhury, D., Santen, L., & Schadschneider, A. (2000). Statistical physics of vehicular traffic. *Physics Reports*, 329(4-6), 199–329. https://doi.org/10.1016/S0370-1573(99)00117-3

[2] Downey, A. B. (2017). Think Complexity: Explorations of Complexity Science with Python (2nd ed.). O'Reilly Media. https://greenteapress.com/wp/think-complexity-2e/

[3] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[4] Maerivoet, S., & De Moor, B. (2005). Cellular automata models of road traffic. *Physics Reports*, 419(1), 1–64. https://doi.org/10.1016/j.physrep.2005.08.005

[5] Nagel, K., & Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de Physique I*, 2(12), 2221–2229. https://doi.org/10.1051/jp1:1992277

[6] Wolfram, S. (1983). Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3), 601–644. https://doi.org/10.1103/RevModPhys.55.601

[7] Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media. https://www.wolframscience.com/nks/

[8] Helbing, D. (2001). Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4), 1067.

[9] Think Complexity (1st Edition) by Allen B. Downey. Chapters 9–12.