A Project Report

On

A Discrete-Event Simulation Model for Traffic Flow at an Intersection

BY
BINITA POKHAREL
221710

UNDER THE SUPERVISION OF
RESHA DEO

TO
DEPARTMENT OF SOFTWARE ENGINEERING
NEPAL COLLEGE OF INFORMATION TECHNOLOGY,
LALITPUR, NEPAL
POKHARA UNIVERSITY

COURSE TITLE: SIMULATION AND MODELING
17, February, 2025

Contents

Abstract

Traffic congestion at intersections is a common challenge in urban transportation, affecting travel time, fuel consumption, and overall road efficiency. This project presents a discrete-event simulation (DES) of a single intersection controlled by a traffic light to analyze traffic flow dynamics. The simulation is implemented in Python using the SimPy library, modeling vehicle arrivals as a stochastic process and incorporating a traffic light with fixed green and red durations. Key performance metrics, including vehicle waiting times and queue lengths, are recorded and visualized using Matplotlib. The results provide insights into traffic behavior under fixed signal timing, demonstrating the usefulness of simulation in evaluating and optimizing traffic control strategies. This study highlights the effectiveness of DES in analyzing traffic flow and serves as a foundation for future enhancements, such as adaptive signal control and multi-lane modeling.

1 Introduction

Traffic management plays a vital role in the efficiency of urban transportation systems. As cities continue to grow, managing traffic flow effectively becomes increasingly challenging. Simulating traffic flow allows researchers and engineers to explore various traffic control strategies, assess their performance, and make data-driven decisions before implementing them in real-world scenarios. Discrete-event simulation (DES) is an effective method for modeling these types of systems, where events occur at specific time intervals and the state of the system changes as a result. In this project, DES is employed to model a single intersection controlled by a traffic light. The simulation focuses on critical performance indicators, such as vehicle waiting times and queue lengths, offering valuable insights into the impact of traffic light cycles on traffic flow. By studying these key metrics, the project aims to demonstrate how simulation can inform the design of better traffic management strategies and optimize intersection performance.

2 Objectives

The primary objectives of this simulation are:

- To model vehicle arrivals using an exponential interarrival time distribution.
- To simulate the operation of a traffic light that alternates between green and red phases.
- To analyze the impact of traffic light operation on vehicle waiting times and queue lengths.
- To visualize the simulation results for further insights.

3 Literature Review

3.1 Discrete-Event Simulation in Traffic Modeling

Discrete-event simulation (DES) is widely used in transportation research to model traffic systems. Law and Kelton (2000) describe DES as a method where the system's state changes at discrete points in time, triggered by events such as vehicle arrivals, departures, or signal changes.

Previous studies have used DES to simulate traffic at intersections. Banks et al. (2010) highlight that queueing theory combined with DES is an effective tool for modeling congestion and optimizing traffic light timing. The SimPy library, as described in its official documentation (SimPy Docs, 2023), provides a robust framework for implementing such models.

3.2 Traffic Light Control and Queueing Theory

Traffic light systems control vehicle flow and reduce congestion at intersections. Traditional fixed-time traffic signals, as discussed by Webster (1958), use predetermined green and red durations based on empirical data. However, more recent studies explore adaptive traffic control, which dynamically adjusts signal timings based on real-time traffic conditions (Gershenson, 2005).

This study uses a fixed-time traffic light model to analyze its effect on waiting times and queue lengths. While adaptive methods offer greater efficiency in real-world applications, fixed-time models are still widely used in many urban settings due to their simplicity and ease of implementation.

3.3 Monte Carlo Methods for Traffic Flow Analysis

Monte Carlo simulations, often used in traffic modeling, rely on random sampling to predict system performance. The exponential distribution used for vehicle arrivals in this study aligns with queuing theory principles (Newell, 1982), where interarrival times follow a stochastic process. By simulating multiple vehicles over time, the Monte Carlo approach allows for statistical analysis of traffic flow behavior under controlled conditions.

The use of Monte Carlo-based stochastic modeling in traffic simulation provides valuable insights into variability in wait times and queue lengths, helping urban planners make data-driven decisions.

4 Methodology

4.1 Simulation Approach

The simulation is built using SimPy, a Python library that provides a process-based discrete-event simulation framework. In a DES model, the system is represented as a sequence of events that change the state of the system at specific times. In this project, the events include vehicle arrivals, changes in the traffic light state, and vehicle departures.

4.2 Model Components

Traffic Light: The traffic light is modeled as an independent process that cyclically changes its state between green and red. Each state lasts for a fixed duration (10 seconds for both green and red). When the light is green, waiting vehicles are allowed to proceed; when red, vehicles must wait.

Vehicle Generation: Vehicle arrivals are generated based on an exponential distribution, a common assumption in traffic flow theory. The average interarrival time is set to 5 seconds. Each vehicle is represented as a process that:

- Records its arrival time.
- Waits until the traffic light is green.
- Measures its waiting time before passing through the intersection.

Queue Monitoring: A separate process monitors the queue length at the intersection. The queue length is recorded every second, providing a time series that reflects the dynamics of the traffic flow over the simulation period.

5 Data Collection and Visualization

Two primary metrics are collected:

Car Waiting Times: The duration each vehicle waits from arrival until the light turns green.

Queue Lengths Over Time: The number of vehicles waiting at the intersection recorded at regular time intervals.

These metrics are visualized using histograms (for waiting times) and line charts (for queue lengths), providing insights into the performance of the simulated traffic system.

6 Implementation

The simulation is implemented in Python with the following key components:

TrafficLight Class: Defines the traffic light behavior, cycling between green and red.

generatetraffic Function: Simulates random vehicle arrivals using an exponential distribution.

carFunction: Represents each vehicle's behavior, including waiting for the green light and recording its wait time.

monitorqueue Function: Periodically records the queue length.

Visualization Functions: Use Matplotlib to plot the results.

6.1 Code Implementation

6.2 Importing Required Libraries

```python
!pip install simpy matplotlib -q
import simpy
import random
import matplotlib.pyplot as plt
```

Listing 1: Importing Required Libraries

6.3 Defining Simulation Parameters

```python
SIMULATION_TIME = 100
ARRIVAL_RATE = 5
GREEN_LIGHT_DURATION = 10
RED_LIGHT_DURATION = 10
```

Listing 2: Defining Simulation Parameters

6.4 Traffic Light Model

```python
class TrafficLight:
    def __init__(self, env):
        self.env = env
        self.green = True
        self.process = env.process(self.operate())

    def operate(self):
        """Traffic light cycles between green and red."""
        while True:
            self.green = True
            yield self.env.timeout(GREEN_LIGHT_DURATION)

            self.green = False
            yield self.env.timeout(RED_LIGHT_DURATION)
```

Listing 3: Traffic Light Model

6.5 Vehicle Generation

```python
def generate_traffic(env, traffic_light, queue):
    """Generates cars at random intervals."""
    car_id = 0
    while True:
        yield env.timeout(random.expovariate(1 / ARRIVAL_RATE))
        car_id += 1
        env.process(car(env, f"Car-{car_id}", traffic_light, queue))
```

Listing 4: Vehicle Generation

6.6 Vehicle Behavior

```python
def car(env, name, traffic_light, queue):
    arrival_time = env.now
    queue.append(arrival_time)

    while not traffic_light.green:
        yield env.timeout(1)

    wait_time = env.now - arrival_time
    car_wait_times.append(wait_time)
    queue.pop(0)
```

Listing 5: Vehicle Behavior

6.7 Queue Monitoring

```python
def monitor_queue(env, traffic_light, queue):
    while True:
        queue_lengths.append((env.now, len(queue)))
        yield env.timeout(1)
```

Listing 6: Queue Monitoring

6.8 Running the Simulation

```python
env = simpy.Environment()
traffic_light = TrafficLight(env)

queue = []
car_wait_times = []
queue_lengths = []

env.process(generate_traffic(env, traffic_light, queue))
env.process(monitor_queue(env, traffic_light, queue))
env.run(until=SIMULATION_TIME)
```

Listing 7: Running the Simulation

6.9 Visualizing Results

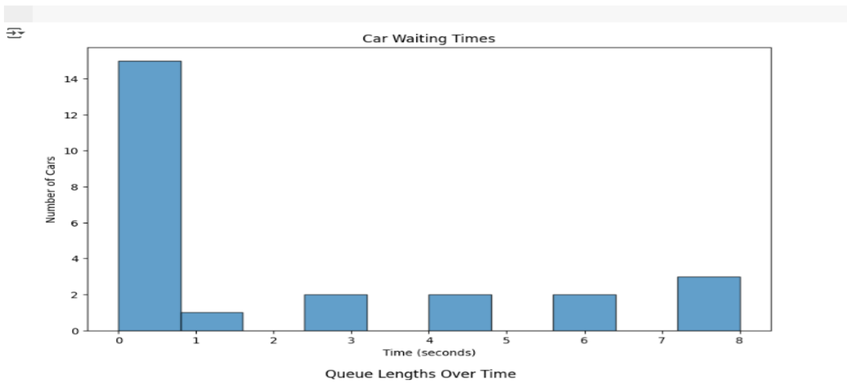
```python
def visualize_results():
    plt.figure(figsize=(10, 6))
    plt.hist(car_wait_times, bins=10, edgecolor="black", alpha=0.7)
    plt.title("Car Waiting Times")
    plt.xlabel("Time (seconds)")
    plt.ylabel("Number of Cars")
    plt.show()

    times, lengths = zip(*queue_lengths)
    plt.figure(figsize=(10, 6))
    plt.plot(times, lengths, marker="o", linestyle="-")
    plt.title("Queue Lengths Over Time")
    plt.xlabel("Time (seconds)")
    plt.ylabel("Queue Length")
    plt.show()

visualize_results()
```

Listing 8: Visualizing Results

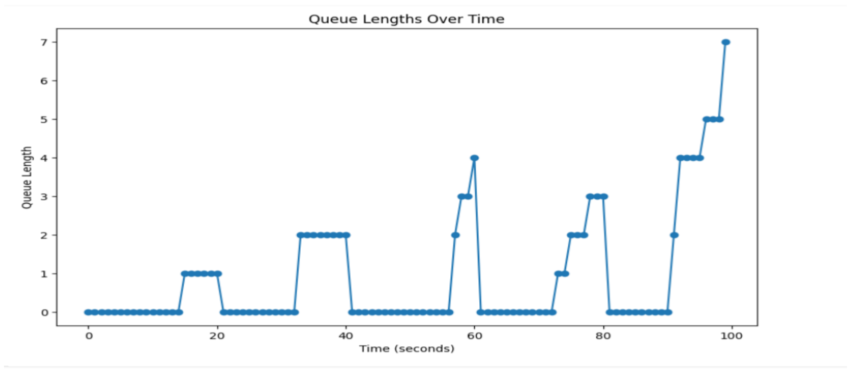
7 Results and Discussion

7.1 Results

Car Waiting Times: The simulation generates a histogram of vehicle waiting times. The distribution of waiting times offers insight into how the fixed cycle of the traffic light (green for 10 seconds, red for 10 seconds) affects vehicles arriving at different times. For instance, vehicles arriving just after the light turns red may experience longer wait times compared to those arriving during the green phase.



Queue Length Dynamics: A time series plot of the queue length provides a dynamic view of how the queue builds up and dissipates. The results demonstrate peaks in the queue length during the red phases and a rapid decline once the light turns green. This observation is consistent with expectations from traffic flow theory.



7.2 Discussion

The simulation successfully illustrates key traffic dynamics at an intersection:

Variability in wait times: Stemming from the random arrival process and fixed traffic light cycle.

Queue fluctuations: Showing accumulation during red phases and dissipation during green phases. While the model is simplified (e.g., a single lane, fixed signal timing), it serves as a foundational framework. More sophisticated models could incorporate multiple lanes, adaptive signal timing, or interactions with pedestrian flows.

8 Conclusion

This project demonstrates the use of discrete-event simulation to model traffic flow at a signalized intersection. Using SimPy, the simulation models vehicle arrivals and traffic light operations, capturing essential performance metrics such as waiting times and queue lengths. The visualizations provide clear insights into the traffic behavior under the assumed conditions.

Future work could extend this model to more complex traffic scenarios, such as intersections with variable timing, multi-lane roads, and networked intersections, thereby enhancing the model's realism and applicability to urban traffic management.

9 References

References

[1] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, 5th ed. Pearson, 2010.

[2] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. New York, NY, USA: McGraw-Hill, 2000.

[3] SimPy Documentation. Available: `https://simpy.readthedocs.io`