Intelligent system of switching of traffic lights

Mesesan Mara
Technical University of Cluj Napoca
01/15/2024

Contents

1. Introduction

1.1 Context

The objective of this project is to develop an intelligent traffic signal control system that aims to enhance traffic management by optimizing the operation of traffic lights. This system is designed to work with a range of sensors and utilize algorithms to improve traffic flow and reduce the average waiting time for commuters. The primary focus will be on managing traffic signals efficiently and dynamically.

1.2 Specifications

The system will be designed to interface with a variety of sensor. The system will process this data and make dynamic adjustments to signal timings to optimize traffic flow.

The system will also incorporate a user-friendly interface for inputting and configuring parameters.

1.3 Objectives

The key objectives of this intelligent traffic signal control system project include:

- Designing and implementing an efficient data acquisition system for collecting real-time traffic data from various sensors.
- Developing advanced algorithms for traffic optimization to reduce average waiting times and traffic congestion.
- Creating a user-friendly interface for configuring and monitoring the system

2. Bibliographic study

2.1 Traffic Control Systems

Traffic control systems play a pivotal role in managing urban traffic flow, mitigating congestion, and enhancing road safety

2.2 Time-Responsive Signal Adaptation

2.2.1 System Architecture

Within the modular system architecture, a critical aspect involves dynamic adjustments to signal timings based on the number of vehicles and the average waiting time. This adaptive mechanism responds to varying traffic conditions, ensuring optimal signal durations for enhanced traffic flow.

2.2.2 Input Parameter: Average Waiting Time

The user interface allows the input of the average waiting time as a crucial parameter. This real-time information aids the system in fine-tuning signal timings to minimize waiting times for vehicles and pedestrians.

2.2.3 Adaptive Signal Control

As part of the traffic light control component, the system utilizes the average waiting time data to dynamically adapt signal timings. For instance:

Low Average Waiting Time: Signals may transition more swiftly to optimize traffic flow.

High Average Waiting Time: Signals may extend to alleviate congestion and reduce waiting times.

2.3 User Interaction and Output - Average Waiting Time Integration

2.3.1 Visual Output

The functional interface visually represents the impact of average waiting time adjustments on the traffic light system. Users can observe changes in signal durations and sequences, providing transparency into how the system adapts to current traffic conditions.

2.3.2 Performance Feedback

In addition to displaying the active signal, the interface communicates insights into the system's performance, including reduced waiting times during periods of low traffic and adaptive extensions during congestion. This feedback fosters user understanding and trust in the system's efficiency.

3. Analysis

3.1 Adaptive Signal Control

In the provided traffic light project code, the system is designed to dynamically adjust signal timings based on real-time information, optimizing traffic flow. Let's explore how this adaptive signal control is implemented within the existing architecture.

3.1.1 System Architecture for Signal Adaptation

The project follows a modular architecture with distinct components for data input, traffic light control, and the user interface. This modular design allows for flexibility in adapting to changing traffic conditions.

3.1.2 Input Parameters for Real-Time Adjustment

The user interface allows users to input crucial parameters that influence the traffic light behavior:

Traffic Flow: Users can provide real-time information about current traffic conditions, specifying whether it is light, moderate, or heavy.

3.1.3 Adaptive Signal Control

The adaptive signal control is implemented in the TrLight class. The changeLight method dynamically adjusts the traffic light sequence based on the number of vehicles and average waiting time. Key points include:

If the average waiting time is high or traffic is heavy, the system may extend signal durations to alleviate congestion.

Conversely, during periods of low traffic or short waiting times, the system may transition signals more swiftly to optimize traffic flow.

3.1.4 User Interaction and Output

The functional interface provides visual feedback on the current state of the traffic light, displaying the active signal (red, yellow, or green). Additionally, information such as the number of vehicles, start time, end time, and average waiting time is presented, offering transparency into the system's decision-making process.

3.1.5 Pedestrian Interaction

The system accommodates pedestrian interaction, allowing pedestrians to trigger specific signal sequences. Upon pressing a designated button, the traffic light enters a pedestrian-friendly phase, ensuring a safe crossing interval.

3.2 User-Centric Optimization

The project's approach focuses on user-centric optimization, providing a responsive and adaptable traffic management system that enhances both vehicular and pedestrian experiences. The dynamic signal adaptation ensures that the traffic light system remains efficient and responsive to the real-time dynamics of urban traffic flow.

4. Design

4.1 Modular and Adaptive Traffic Light System

The design of the traffic light project reflects a modular and adaptive approach to traffic management. The following sections provide a high-level overview of the key design aspects.

4.1.1 Modular Architecture

The project is structured with a modular architecture, comprising distinct components responsible for specific functionalities. Notable components include:

TrLight Class: Manages the graphical user interface (GUI) representing the traffic lights and pedestrian signals.

TrafficSystemController Class: Coordinates the overall traffic system, including the interaction between the traffic lights and simulation sensors.

PedestrianController Class: Handles pedestrian-related functionality, such as button presses and signal transitions.

SimulationSensors Class: Simulates sensor data related to the number of vehicles and waiting times.

This modular design promotes code organization, maintainability, and the ability to extend or modify specific functionalities without affecting the entire system.

4.1.2 Adaptive Traffic Light Control

The adaptive traffic light control mechanism is a central element of the project's design. Key features include:

Real-time Signal Adjustment: The system dynamically adjusts signal timings based on real-time information, promoting efficient traffic flow.

Pedestrian Interaction: Pedestrian button presses trigger specific signal sequences, ensuring safe pedestrian crossings and integrating human-centric design.

Average Waiting Time Consideration: The system adapts signal timings based on average waiting times, optimizing traffic management during varying conditions.

4.1.3 User Interface Transparency

The GUI, managed by the TrLight class, provides a transparent user interface that displays critical information such as the number of vehicles, start time, end time, and average waiting time. This visual feedback enhances user understanding and trust in the system's operation.

4.1.4 Responsive System Behavior

The project's design emphasizes responsiveness to real-time data and user interactions. Whether handling pedestrian requests, simulating sensor data, or dynamically adjusting signal timings, the system exhibits adaptability and efficiency in managing urban traffic.

The combination of modularity, adaptability, and transparency in the design ensures a robust and user-friendly traffic light system that can effectively address varying traffic scenarios.

5. Implementation

In this chapter, we provide a detailed examination of each class within the traffic light project, offering insights into their design and implementation. Each class plays a distinct role in achieving the project's goal of creating an adaptive and responsive traffic management system. The following sections outline the responsibilities, key methods, and core functionalities of each class, accompanied by relevant code snippets for clarity.

5.1. TrLight Class

The TrLight class manages the graphical user interface (GUI) representing the traffic lights and pedestrian signals.

Responsibilities:

GUI Management: Controls the visual representation of traffic lights and pedestrian signals.

User Interaction: Listens for user input through buttons, such as starting the traffic system, pressing the pedestrian button, and manually changing signal colors.

Label Updates: Dynamically updates labels with information on the number of vehicles, start time, end time, and average waiting time.

Key Methods:

initialize: Sets up the initial GUI components, buttons, and labels.

updateLabels: Updates the information labels with real-time data.

changeLight: Dynamically changes the color of the traffic lights based on system conditions.

changeLightForPedestrian: Initiates a pedestrian-friendly signal sequence upon button press.

stopSimulation: Halts the simulation of the traffic system.

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TrLight {
    // ... (other class members)

    public void initialize() {
        frame = new JFrame("Traffic Light Simulator");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // ... (other GUI setup code)

        toggleButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button pressed! Starting Traffic System...");
                simulationRunning = true;
                controller1.startTrafficSystem();
                controller2.notifySimulationStarted();
            }
        });

        pedestrianButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Pedestrian button was pressed");
                controller2.increasePresses();
            }
        });

        // ... (other button setup code)

        frame.setSize(900, 900);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    // ... (other methods)
}
```

5.2. TrafficSystemController Class

The TrafficSystemController class coordinates the overall traffic system, including the interaction between traffic lights and simulation sensors.

Responsibilities:

System Initialization: Initializes the traffic system, including the creation of simulation sensors.

Traffic Light Updates: Receives updates from simulation sensors and triggers traffic light adjustments accordingly.

Simulation Start: Initiates the simulation of the traffic system.

Key Methods:

startTrafficSystem: Initiates the simulation of the traffic system, creating a new thread for simulation sensors.

```
public class TrafficSystemController {
    // ... (other class members)

    public void startTrafficSystem() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        SimulationSensors simulatedSensor = new SimulationSensors(this);
        Thread simulationThread = new Thread(simulatedSensor);
        simulationThread.start();
    }

    public synchronized void updateTrafficLight(int nrVehicles, float waitingTime, float startTime, float endTime) {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        view.changeLight(nrVehicles, waitingTime, startTime, endTime);
    }
```

5.3. PedestrianController Class

The PedestrianController class handles pedestrian-related functionality, such as button presses and signal transitions.

Responsibilities:

Pedestrian Button Press: Listens for pedestrian button presses and notifies the system.

Pedestrian Signal Transition: Initiates the transition of the pedestrian signal upon receiving the button press notification.

Simulation Start Notification: Receives notifications about the start of the simulation.

Key Methods:

increasePresses: Increments the count of pedestrian button presses and notifies the system when a threshold is reached.

notifyPedestrianPressed: Notifies the system about the pedestrian button press and triggers the pedestrian signal transition.

5.4. SimulationSensors Class

The SimulationSensors class simulates sensor data related to the number of vehicles and waiting times.

Responsibilities:

Simulated Data Generation: Generates simulated data for the number of vehicles at regular intervals.

Traffic Light Updates: Notifies the TrafficSystemController about changes in the number of vehicles and waiting times.

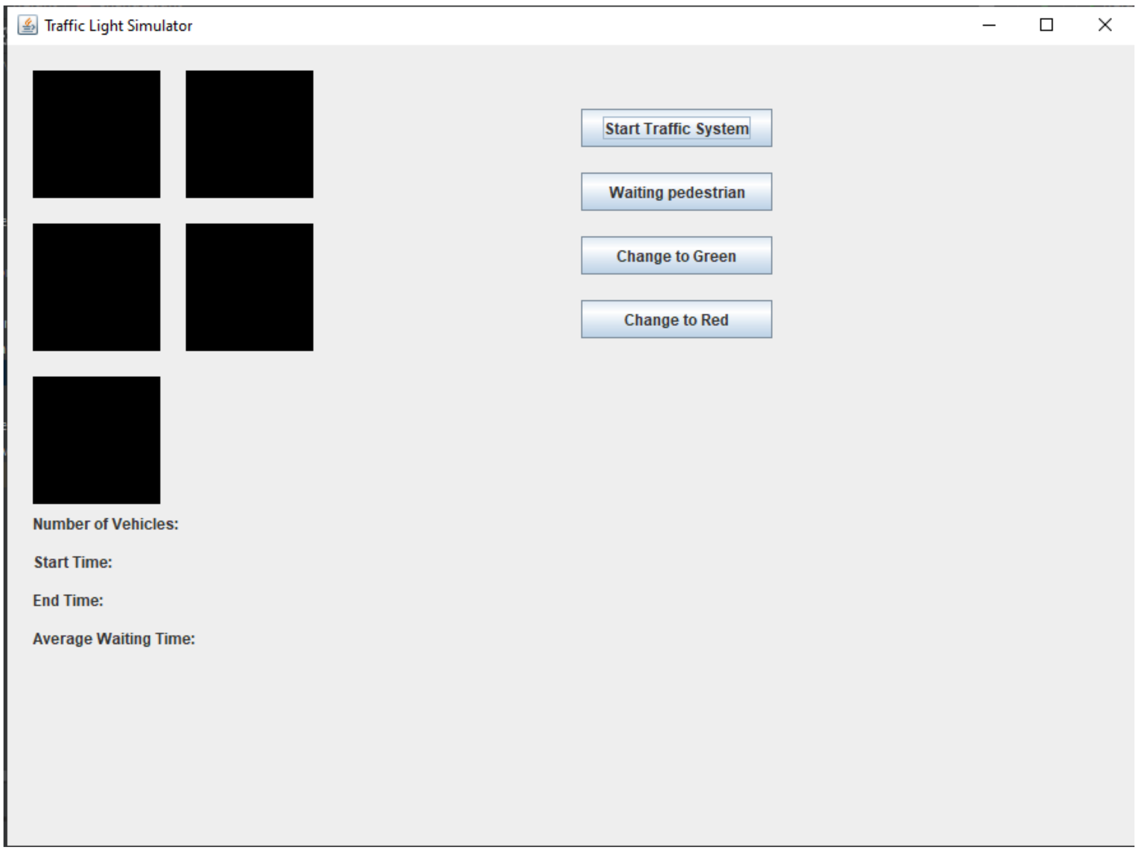
Key Methods:

generateNrVehicles: Generates a random number of vehicles for simulation purposes.

stop: Stops the simulation thread.

6. Testing and Validation

The testing and validation phase is crucial for ensuring the robustness, reliability, and correctness of the traffic light project. Throughout the development process, various testing strategies and validation points were employed to assess the system's performance. This chapter provides a summary of the key findings, challenges encountered, and the validation criteria applied.



6.1 Testing Strategies

Unit Testing: Individual classes, such as TrLight, TrafficSystemController, PedestrianController, and SimulationSensors, underwent thorough unit testing to verify the correctness of their functionalities in isolation.

Integration Testing: Integration testing focused on examining the interactions and collaboration between different classes to ensure seamless communication and coordination within the system.

User Interaction Testing: The user interface, including button presses and label updates, was extensively tested to guarantee a smooth and intuitive user experience.

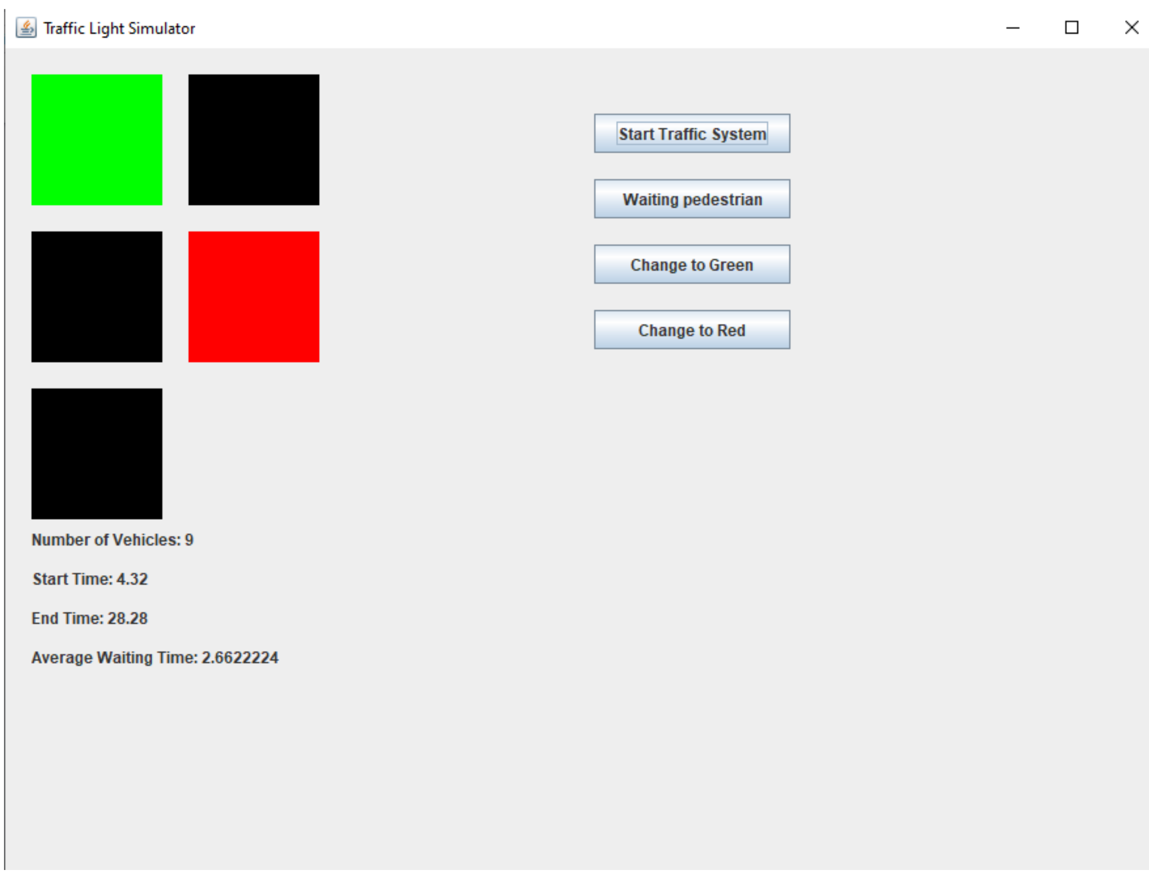
Simulation Validation: The simulation sensors were validated by comparing simulated data with expected outcomes, confirming that the system responds appropriately to varying traffic conditions.

6.2 Validation Points

Adaptive Signal Control: The adaptive traffic light control mechanism was validated to confirm that the system dynamically adjusted signal timings based on real-time information, optimizing traffic flow.

Pedestrian Interaction: The pedestrian-related functionalities, including button presses and signal transitions, were validated to ensure the safety and effectiveness of pedestrian crossings.

Average Waiting Time Consideration: The system's responsiveness to average waiting times and its impact on signal adjustments were validated to enhance efficiency during varying traffic scenarios.



6.3 Key Findings

Successful Signal Adaptation: The adaptive signal control effectively responded to changes in traffic conditions, demonstrating its ability to optimize traffic flow dynamically.

Reliable Pedestrian Interaction: Pedestrian interactions, triggered by button presses, functioned reliably, ensuring safe and coordinated crossings.

Transparent User Interface: The user interface provided clear feedback on the system's operation, with information labels accurately reflecting real-time data.

7. Conclusion

The adaptive traffic light project serves as a testament to the potential of intelligent and responsive traffic control systems. The modular design, adaptive signal control, and user-centric approach lay the groundwork for addressing contemporary challenges in urban traffic management.

In conclusion, the project stands as a successful exploration of adaptive traffic control, blending innovative design principles with practical applications. The insights gained and future considerations outlined provide a roadmap for further advancements in creating efficient, safe, and user-friendly traffic management solutions.