NIST UNIVERSITY

Subject: DESIGN AND ANALYSIS OF ALGORITM.

Code: CSE-200

Semester: 3rd Sem   Batch: 2024-2028   BRANCH: CSE (AI & ML-A)

Student Name	Roll Number
RAJ AKASH BHUYAN	202456042
HARIPRIYA MAHARANA	202457054

Title of the Project: Traffic signal timing optimizer

Name of the Faculty :-DR. SWATIKANT MISHRA

Comment:

Mark obtained:

1. Abstract:-

Urban traffic congestion is a critical challenge that impacts travel time, fuel consumption, and environmental sustainability. Traditional fixed-time traffic signals fail to adapt to dynamic traffic conditions, leading to inefficiencies and increased delays. This project proposes a **Traffic Signal Timing Optimizer** that leverages algorithmic design principles to dynamically adjust signal cycles based on real-time traffic flow.

The system models intersections as optimization problems, where the objective is to minimize average waiting time and maximize throughput. Various algorithmic strategies—such as **Greedy methods, Dynamic Programming, and Graph-based shortest path approaches**—are analyzed to determine efficient scheduling of green phases. Additionally, heuristic and AI-driven techniques, including **Genetic Algorithms and Reinforcement Learning**, are explored to handle unpredictable traffic patterns.

The optimizer integrates sensor data (vehicle count, queue length, arrival rate) and applies algorithmic analysis to compute optimal cycle lengths. Performance is evaluated in terms of **time complexity, space complexity, and scalability** across multiple intersections. By reducing idle time and improving traffic flow, the project demonstrates how algorithmic optimization contributes to **smart city development and sustainable urban mobility**

2. Introduction

Traffic congestion is one of the most persistent problems in urban areas, leading to wasted time, higher fuel consumption, and increased environmental pollution. At the heart of this issue lies the inefficiency of traditional traffic signal systems, which often operate on fixed cycles regardless of real-time traffic conditions. Such static scheduling fails to accommodate peak hours, uneven vehicle distribution, or sudden changes in traffic flow, resulting in long queues and delays at intersections.

A Traffic Signal Timing Optimizer aims to overcome these limitations by applying algorithmic techniques to dynamically adjust signal timings. By modeling intersections as optimization problems, the system seeks to minimize average waiting time, reduce queue lengths, and maximize throughput. Real-time data such as vehicle density, arrival rates, and queue lengths can be integrated into the optimizer, enabling adaptive control of red, yellow, and green phases.The final output is an image showing how the character might have been drawn stroke by stroke.

3. Technologies Used

Sensors & IoT Devices

- Inductive loop detectors, infrared sensors, and cameras to capture vehicle count, speed, and queue length.
- IoT-enabled devices for real-time data collection and communication.

Visualization & Control Interfaces

- Dashboards for traffic authorities to monitor and adjust signals.
- Real-time visualization tools for analyzing congestion and performance metrics.

Databases & Storage

- SQL/NoSQL databases for storing traffic patterns, historical data, and optimization results.

Communication & Networking

• Wireless Sensor Networks (WSN) – to connect intersections and share traffic data.

• Vehicle-to-Infrastructure (V2I) – smart cars communicating with signals.

• Cloud Platforms – for centralized traffic management and scalability. 5. Thresholding

C PROG CODE:-

```
#include <stdio.h>


#define MAX_SIGNALS 4
#define CYCLE_TIME 120
```

```
int main() {
    int vehicles[MAX_SIGNALS];
    int greenTime[MAX_SIGNALS];
    int totalVehicles = 0;

    printf("Traffic Signal Timing Optimizer\n");

printf("=================================\n");

    for (int i = 0; i < MAX_SIGNALS; i++) {
        printf("Enter vehicle count at Signal %d: ", i + 1);
        scanf("%d", &vehicles[i]);
        totalVehicles += vehicles[i];
    }

    if (totalVehicles == 0) {
        for (int i = 0; i < MAX_SIGNALS; i++) {
            greenTime[i] = CYCLE_TIME / MAX_SIGNALS;
        }
    } else {
        for (int i = 0; i < MAX_SIGNALS; i++) {
            greenTime[i] = (vehicles[i] * CYCLE_TIME) /
totalVehicles;
            if (greenTime[i] < 10) {
                greenTime[i] = 10;
            }
        }
    }

    printf("\nOptimized Signal Timings (in seconds):\n");
    for (int i = 0; i < MAX_SIGNALS; i++) {
        printf("Signal %d: %d seconds\n", i + 1,
greenTime[i]);
    }

    return 0;
}
```

ALGORITHM:-

 Start

1. **Initialize constants:**
    - **MAX_SIGNALS = 4**
    - **CYCLE_TIME = 120**
2. **Create arrays:**
    - **vehicles[MAX_SIGNALS] to store vehicle counts**
    - **greenTime[MAX_SIGNALS] to store computed timings**
3. **Set totalVehicles = 0.**
4. **For each signal i=1 to MAX_ SIGNALS:**
    - **Prompt user to enter vehicle count.**
    - **Store input in vehicles[i].**
    - **Add to totalVehicles.**
5. **Check condition:**
    - **If totalVehicles == 0:**
        - **Assign equal time to all signals → CYCLE_TIME / MAX_SIGNALS.**
    - **Else:**
    - **For each signal i:**
    - **Compute proportional time:**

greenTime[i]=\frac{vehicles[i]\times CYCLE_ TIME}{totalVehicles}

- If [greenTime[i] < 10], set [greenTime[i] = 10] (minimum threshold).

6. Display optimized green times for all signals.
7. End

CALCULATION:-

Given:

- Number of signals = 4
- Total cycle time = 120 seconds
- Vehicle counts at each signal (example):
- Signal 1 → 40 vehicles
- Signal 2 → 20 vehicles
- Signal 3 → 30 vehicles
- Signal 4 → 10 vehicles

Step 1: Total Vehicles

{Total\ Vehicles}=40+20+30+10=100

Step 2: Proportional Green Time Allocation

Formula:

{Green\ Time}[i]= {{Vehicles}[i]*{Cycle Time}}/{ {Total Vehicles}}

- **Signal 1:**

{40* 120}/{100}=48 seconds

- **Signal 2:**

{20* 120}/{100}=24seconds

- **Signal 3:**

{30* 120}/{100}=36 seconds

- **Signal 4:**

{10* 120}/{100}=12 seconds

Step 3: Minimum Threshold Check

- **If any signal's green time < 10 seconds, set it to 10.**
- **In this example, all values ≥ 10, so no adjustment needed.**

✅ Final Optimized Timings

- **Signal 1 → 48 seconds**
- **Signal 2 → 24 seconds**
- **Signal 3 → 36 seconds**
- **Signal 4 → 12 seconds**

CONCLUSION:-

The Traffic Signal Timing Optimizer project demonstrates how algorithmic design can be applied to solve real-world urban challenges. By modeling intersections as optimization problems, the system effectively allocates green light durations based on traffic density, thereby reducing waiting time, minimizing fuel consumption, and improving overall traffic flow.