Traffic Light Timing Optimization Using Genetic Algorithms in SUMO

Zohaib Aslam
School of Science and Engineering
Habib University
Karachi, Pakistan
za08134@st.habib.edu.pk

Ahsan Siddiqui
School of Science and Engineering
Habib University
Karachi, Pakistan
as08155@st.habib.edu.pk

***Abstract*—Efficient traffic signal control is essential for minimizing congestion and improving travel times in urban environments. This project presents a Genetic Algorithm (GA)-based approach for optimizing traffic light timings using the Simulation of Urban MObility (SUMO) platform. The objective is to minimize the average vehicle waiting time at a signalized intersection. The optimization process is applied to two traffic scenarios—light and heavy—under both fixed and randomized vehicle flow patterns. The GA evolves signal phase durations by evaluating individual configurations through simulation-based feedback. A baseline timing configuration is used for comparison to assess the effectiveness of the GA-optimized solution. Experimental results demonstrate a significant reduction in average waiting time, validating the potential of evolutionary techniques for traffic signal optimization in both static and dynamic conditions.**

***Index Terms*—Genetic Algorithms, Traffic Signal Optimization, SUMO, Traffic Simulation, Intelligent Transportation Systems**

I. Introduction

Urban traffic congestion remains a persistent challenge in modern cities, leading to increased travel times, fuel consumption, and environmental impact. One of the most effective ways to alleviate congestion is through intelligent traffic signal control, which can significantly reduce vehicle delays at intersections. Traditional traffic light systems rely on fixed-time control strategies that are not adaptive to varying traffic conditions, often resulting in suboptimal performance during peak and off-peak hours.

To address this limitation, this project explores the use of Genetic Algorithms (GAs), a class of evolutionary optimization techniques inspired by natural selection, to automatically determine optimal traffic light phase durations. The aim is to minimize the average waiting time of vehicles at a four-way intersection controlled by traffic signals.

The traffic environment is modeled and simulated using SUMO (Simulation of Urban MObility), a widely used, open-source traffic simulator. Two traffic scenarios—light and heavy traffic flow—are considered. Each scenario is evaluated under both fixed and randomized vehicle generation to test the robustness and generalizability of the optimized signal timings.

The effectiveness of the GA-based approach is compared against a fixed baseline configuration commonly used in static signal control. Results demonstrate that the GA consistently identifies signal timings that reduce average vehicle waiting time across varying traffic patterns

II. Technical Background

This section provides an overview of the core components and technologies used in the project, including the SUMO traffic simulator and Genetic Algorithms, which form the basis for the optimization approach.

A. Simulation of Urban Mobility (SUMO)

Simulation of Urban Mobility (SUMO) is an open-source, portable, and microscopic traffic simulation tool developed by the German Aerospace Center (DLR). It allows for real-time simulation of vehicular movement on large-scale road networks. SUMO supports detailed traffic modeling, enabling researchers to evaluate traffic systems, smart mobility strategies, and optimization algorithms in a controlled environment.

B. Purpose and Capabilities

SUMO is widely used in the Intelligent Transportation Systems (ITS) community due to its flexibility and powerful feature set. It supports modeling of various vehicle types, pedestrians, emissions, and custom routing. Crucially, SUMO can be interfaced with external tools through TraCI (Traffic Control Interface), which allows step-by-step simulation control and data retrieval. This makes SUMO suitable for integration with optimization algorithms such as Genetic Algorithms (GA).

C. How We Used SUMO

In our project, SUMO was used to simulate a four-way intersection with traffic signals and varying vehicle inflow patterns. The simulation setup was defined using XML files:

- **net.xml:** Defines the road network topology, including lanes, junctions, and connections.
- **route.xml:** Specifies vehicle routes through the intersection.
- **cfg.xml:** Central configuration file that combines network, route, and simulation settings.

Initially, traffic lights operated using a fixed signal plan. We aimed to optimize the durations of green phases using a Genetic Algorithm in order to reduce average vehicle waiting time and improve traffic flow efficiency.

D. Integration with Genetic Algorithm

We interfaced SUMO with our Python-based Genetic Algorithm using the TraCI API. Each individual in the GA population represented a candidate solution encoding traffic light phase durations. For each simulation run:

1) TraCI was used to start the simulation with a specific timing configuration.
2) SUMO collected metrics such as average waiting time per vehicle and total simulation delay.
3) These metrics were used to compute a fitness score for the candidate.

Over successive generations, the GA evolved better timing configurations by selecting, crossing over, and mutating high-performing individuals. This approach allowed us to search for traffic signal plans that minimized congestion under dynamic conditions.

E. Simulation Output

The following figure shows a snapshot of the traffic simulation in SUMO. This visualization helped us observe the effect of different signal plans on vehicle flow, congestion levels, and intersection throughput.

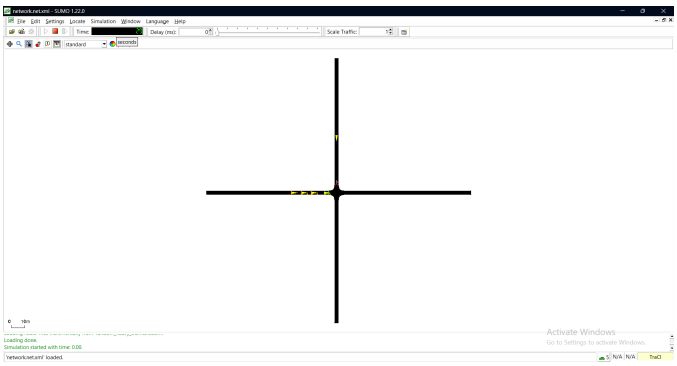

Fig. 1. SUMO Traffic Simulation of the Four-Way Intersection

The final GA-optimized traffic light configuration significantly outperformed the static controller in terms of reduced average waiting time and vehicle throughput.

Genetic Algorithms (GAs) are population-based optimization techniques inspired by the principles of natural evolution. A GA operates by evolving a population of candidate solutions (called individuals) over multiple generations. Each individual is evaluated using a fitness function, which in this project corresponds to the average vehicle waiting time at the intersection. The GA includes standard genetic operations such as:

- **Selection**: Choosing the fittest individuals to reproduce.
- **Crossover**: Combining two parent solutions to create offspring.
- **Mutation**: Randomly altering an individual to introduce diversity.

GAs are well-suited for traffic signal optimization problems due to their ability to search large, complex solution spaces without requiring gradient information.

F. Traffic Signal Control

Conventional traffic signal control typically uses static timing plans based on historical data or empirical rules. These fixed plans fail to adapt to real-time changes in traffic demand. By contrast, the approach in this project dynamically adjusts the green and yellow phase durations for a four-phase traffic signal to better accommodate varying traffic volumes in both light and heavy scenarios.

III. Implementation

A four-way intersection is modeled using a custom SUMO network file (`.net.xml`) that defines the roads, junctions, and traffic light logic. The layout consists of two incoming lanes (`north2center`, `west2center`) and two outgoing lanes (`center2south`, `center2east`), all connected to a central traffic light-controlled junction.

Traffic flow is introduced into the simulation using route files (`.rou.xml`), which define vehicle types, routes, and departure times. To simulate different traffic conditions, two scenarios are prepared:

- **Light Traffic**: Vehicles are generated with a lower frequency, simulating off-peak or moderate traffic load.
- **Heavy Traffic**: Vehicles are generated with a higher frequency, simulating peak traffic conditions.

For the random generation of traffic flow, the route files are generated using SUMO's `randomTrips.py` utility, allowing the traffic patterns to be randomized per run using different random seeds. This ensures that the GA is trained on one instance of traffic data and tested on a separate, unseen instance to assess generalizability.

A. Genetic Algorithm Design

The Genetic Algorithm is designed to optimize the timing of traffic light phases. Each individual in the GA population is represented as a 4-dimensional vector:

$$\text{Individual} = [g_1, y_1, g_2, y_2] \tag{1}$$

Where:

- g_1: Green duration for the North-South direction
- y_1: Yellow duration for the North-South direction
- g_2: Green duration for the East-West direction
- y_2: Yellow duration for the East-West direction

Each parameter is initialized within realistic constraints:

- Green durations: $10 \leq g_1, g_2 \leq 60$ seconds
- Yellow durations: $3 \leq y_1, y_2 \leq 5$ seconds

The GA proceeds through the following steps over a fixed number of generations:

1) **Initialization:** Generate a random population of traffic light configurations.
2) **Evaluation:** Each configuration is applied in SUMO. The simulation is run for a fixed number of steps, and the average waiting time of all vehicles is computed as the fitness.

3) **Parent Selection:** Tournament selection is used to select parent individuals based on their fitness values.
4) **Crossover:** Single-point crossover is applied to generate two children from each pair of parents.
5) **Mutation:** Each child has a probability of mutation, where one randomly selected parameter is reassigned within its valid range.
6) **Survivor Selection:** The best individual from each generation is retained (elitism), and the next generation is formed using a mix of elite, offspring, and top performers.

B. TraCI Integration

SUMO's Traffic Control Interface (TraCI) is used to control the simulation in real time from Python. The following functions are used:

- `traci.start()`: Launches the SUMO simulation in headless or GUI mode.
- `traci.trafficlight.setCompleteRedYellowGreenDefinition()`: Applies a new traffic light logic (set of phase durations) during simulation.
- `traci.simulationStep()`: Advances the simulation by one time step.
- `traci.vehicle.getAccumulatedWaitingTime()`: Retrieves the time a vehicle has spent stopped or moving slower than desired.

Each simulation is executed for 100 steps. At each step, the current list of vehicles is polled, and their waiting times are accumulated to compute the fitness of the current individual.

C. Baseline Comparison and Evaluation

A static baseline configuration of $[42, 3, 42, 3]$ is used as a reference, mimicking traditional fixed-time traffic control systems. After the GA completes its optimization process, the best-performing individual is tested on new randomized traffic files (test data). The average waiting time of vehicles under this optimized configuration is compared against the baseline to quantify improvement.

This evaluation is performed under unseen traffic patterns to test the generalizability of the solution. A final visualization routine displays the best solution using SUMO-GUI for interpretability and demonstration.

D. Implementation Summary

The full system is implemented in a single Python script, integrating GA logic, SUMO simulation control, data collection, and visualization. The modular design allows easy extension to more complex network topologies or additional optimization objectives such as throughput, fuel consumption, or emission reduction.

IV. Experimental Setup

This section outlines the experimental conditions under which the traffic signal optimization was carried out. The setup includes the simulation parameters used in SUMO, the configuration of the Genetic Algorithm, and the strategy for separating training and testing scenarios to ensure robust evaluation.

A. Simulation Configuration

The traffic simulation is conducted using SUMO version 1.22. A four-way intersection is constructed using custom-defined network files in SUMO's XML format. Each simulation is executed over a duration of 100 time steps, with vehicles entering the network from two directions (north and west) and exiting toward the south and east.

The simulation is run in both headless (command-line) and graphical (GUI) modes. For optimization, headless mode is used for speed, while the GUI is employed to visualize the best-performing configuration.

B. Traffic Scenarios

Two distinct traffic conditions are considered:

- **Light Traffic:** Vehicles are introduced at a lower frequency using a larger average departure period (20 seconds), resulting in sparse traffic flow.
- **Heavy Traffic:** Vehicles are introduced more frequently using a shorter average departure period (5 seconds), simulating peak traffic load.

Route files (`.rou.xml`) for each traffic type are generated using SUMO's `randomTrips.py` tool. This utility creates randomized vehicle trips with specified parameters and seeds, allowing for both controlled and reproducible randomness.

During the training phase, one random light traffic file and one heavy traffic file are generated and used consistently throughout the Genetic Algorithm run. For testing, new route files with different seeds are generated to simulate previously unseen traffic conditions.

C. Genetic Algorithm Parameters

The Genetic Algorithm is implemented in Python and follows a steady-state evolution model. The key parameters are listed below:

- **Population Size:** 10 individuals
- **Number of Generations:** 20
- **Crossover Type:** Single-point crossover
- **Mutation Rate:** 0.2 (20% probability)
- **Parent Selection Method:** Tournament selection (size = 2)

Each individual encodes four phase durations: two green lights and two yellow lights for the two signal directions (North-South and East-West). The green durations range from 10 to 60 seconds, and yellow durations from 3 to 5 seconds.

D. Training and Testing Protocol

The GA is trained using the initially generated light and heavy traffic files. Once the best traffic light configuration is identified, its performance is tested against newly generated light and heavy traffic conditions with different random seeds.

A baseline configuration of $[42, 3, 42, 3]$ is also evaluated on the same test data to allow direct comparison with the GA-optimized results. The average waiting time of all vehicles is used as the primary performance metric.

V. Results

The genetic algorithm (GA)-based optimization approach for traffic light configuration significantly reduced average vehicle waiting times at the intersection. After running the algorithm on training data, the best-performing individual was found with the following configuration:

- **Best Traffic Light Configuration:** [11, 4, 19, 4]
- **Best Training Score (Average Waiting Time):** 1.41 seconds

To evaluate the generalizability of this configuration, it was tested on previously unseen data. The performance was compared with a baseline configuration, yielding the following results:

- **Baseline Score on Test Data:** 8.49 seconds
- **GA-Optimized Score on Test Data:** 3.60 seconds
- **Improvement Over Baseline: 57.59%**

These findings indicate a substantial improvement in traffic flow efficiency through GA optimization.

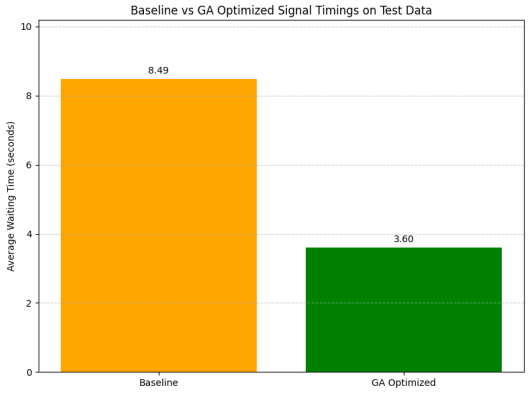

Fig. 2. Comparison of Average Waiting Time: Baseline vs. GA-Optimized Configuration

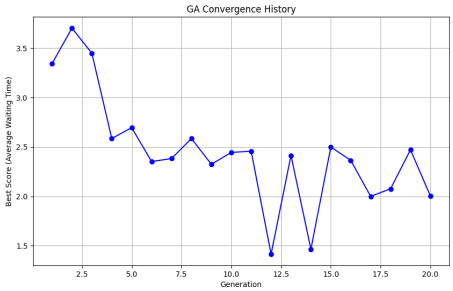

Fig. 3. Convergence of Best Score Over Generations

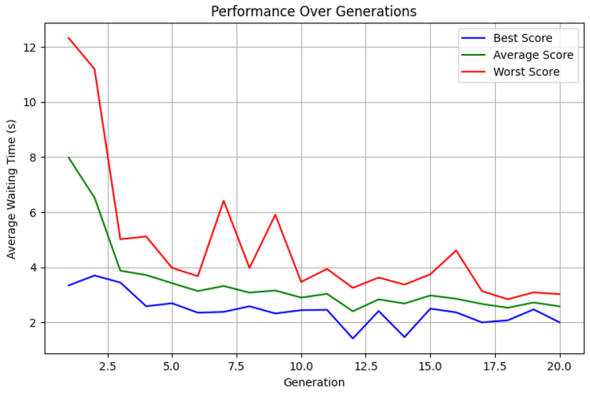

Fig. 4. Performance Over Generations

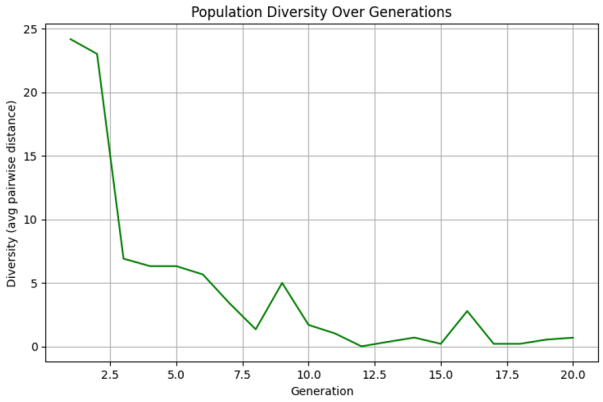

Fig. 5. Population Diversity

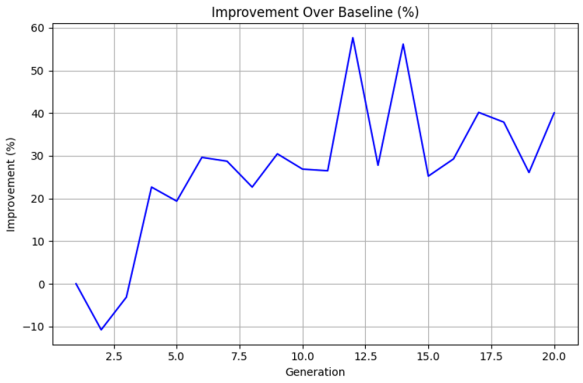

Fig. 6. Improvement Over Baseline

These visualizations further reinforce the effectiveness of the GA. The convergence plot confirms steady optimization, while the other graphs show that the algorithm reduces population diversity and continues improving performance over generations.

VI. Conclusion

This project demonstrated the effectiveness of using a Genetic Algorithm (GA) to optimize traffic light timings

at a four-way intersection. By modeling the traffic system and iteratively evolving traffic light configurations, the GA achieved a significant reduction in average vehicle waiting times compared to a baseline setup.

The best-performing configuration reduced the average waiting time from 8.49 seconds to 3.60 seconds on unseen test data, yielding an improvement of approximately 57.59%. This highlights the potential of evolutionary optimization techniques in addressing real-world traffic congestion problems.

In addition to improving performance, the GA maintained population diversity and demonstrated consistent convergence across generations, as shown by the analytical visualizations. These insights are encouraging for scaling the method to more complex intersections or real-time adaptive systems.

Future work could involve integrating reinforcement learning for dynamic adaptation, incorporating real-time traffic sensor data, and extending the model to account for pedestrian crossings, emergency vehicle prioritization, and environmental factors such as emissions or fuel consumption.

Overall, this study showcases how bio-inspired algorithms can contribute to smarter, data-driven urban traffic management solutions.

VII. References

1) R. P. Roess and E. S. Prassas, "Time Optimization for Traffic Signal Control Using Genetic Algorithm." Available: https://www.researchgate.net/publication/229029442_Time_Optimization_for_Traffic_Signal_Control_Using_Genetic_Algorithm
2) B. Park and J. D. Schneeberger, "A genetic algorithm approach for traffic signal optimization with the TRANSYT model," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3–4, pp. 281–304, 2003. Available: https://www.sciencedirect.com/science/article/abs/pii/S0191261503000158
3) X. Zhang, Y. Zhang, and J. Wang, "An Intelligent Traffic Signal Control Based on Genetic Algorithm," in *Intelligent Transportation and Smart Cities*, Springer, 2024, pp. 81–94. Available: https://link.springer.com/chapter/10.1007/978-3-031-60419-5_8