Enhancing Traffic Signal using Ant Colony Optimisation
Final Reports

Jerrin Jojo Xavier
a1896251

November 11, 2024

Report submitted for **4439 MATHS 7097B Data Science Research Project B** at the School of Mathematical Sciences, University of Adelaide

Project Area: **Heuristic Techniques for solving Constraint Optimization Problem**
Project Supervisor: **Dr. Indu Bala**

Abstract

Considering the ever-changing traffic patterns and intricate intersection designs in city environments, alongside the shortcomings of current traffic signal optimization methods, this study introduces a new approach. By conceptualizing the traffic network as a directed acyclic graph, we developed an innovative Q-learning-based ant colony optimization (Q-ACO) algorithm, inspired by reinforcement learning principles. This algorithm adjusts to real-time traffic variations by dynamically updating the Q-matrix, guided by a reward function that takes into account traffic density and waiting times. This updated Q-matrix forms the foundational pheromone level for the ant colony algorithm, which then employs a pseudo-random proportional rule to refine signal timing. Through simulations across various traffic conditions, our algorithm has demonstrated a significant reduction in average vehicle waiting times and an enhancement in traffic flow efficiency. When compared with traditional methods like ACO and Q-Learning alone, our Q-ACO algorithm consistently reduces waiting times, presenting a robust solution for managing dynamic traffic scenarios.

Keywords

• Optimisation • Ant Colony Optimisation (ACO) • Traffic Signal Timing • Q-Learning • Pheromone

1 Introduction

In urban environments, traffic signals play a crucial role in managing vehicle movements to mitigate congestion, enhance safety, and implement strategies aimed at reducing delays and improving environmental conditions [**1**]. The growth in vehicle numbers and industrial development has made the optimization of traffic signal parameters increasingly critical to maximize network capacity. Over the past decade, advancements in communications and information technologies have significantly evolved classical methods of optimizing traffic signal timings, paving the way for more intelligent solutions.

By effectively regulating traffic demand at each intersection, the objective is to prevent traffic conflicts and minimize queue lengths at stoplights. Researchers have proposed numerous approaches to the traffic signal control problem over the years. Some of the pioneering large-scale adaptive traffic signal control systems include TRANSYT (Traffic Network Study Tool) [**2**], SCOOT (Split, Cycle, and Offset Optimization Technique) [**3**], and SCATS (Sydney Coordinated Adaptive Traffic System) [**4**], which utilize pre-calculated offline timing plans for signal cycles tailored to current traffic conditions. More contemporary advancements in traffic signal control have embraced artificial intelligence technologies such as neural networks and fuzzy logic [**5**], with recent investigations exploring the use of algorithms based on Petri nets and Markov decision control [**6**].

The Ant Colony algorithm is a meta-heuristic approach designed to tackle complex combinatorial optimization problems. This algorithm draws inspiration from the foraging behavior observed in natural ant colonies [**7**]. In this multi-agent system, each individual agent is modeled as an artificial ant, contributing to one of the most successful applications of swarm intelligence. The algorithm has been effectively applied to a variety of challenges, including the classical traveling salesman problem, path planning, and network routing [**8**].

In their natural environment, ants initially search for food in a random pattern. Upon locating food, they return to their colony while depositing pheromones, chemicals that serve as a communication medium within the colony [**9**]. These pheromone trails help guide other ants to the food source. Over time, as more pheromones are deposited, the paths of the ants become increasingly targeted, favoring routes with higher pheromone concentrations, thus reducing randomness.

Mimicking this natural mechanism, the ant colony algorithm uses artificial ants that explore the solution space in a probabilistic manner, generating potential solutions. These solutions are evaluated, and the results influence the updating of pheromone concentrations on the paths. As the algorithm progresses, pheromone on less successful paths evaporates, leaving only the paths that are frequently reinforced with pheromone, thus optimizing the search for the most effective solutions [**9**].

Ant Colony Optimization (ACO) has demonstrated remarkable success in addressing a range of NP-hard combinatorial optimization challenges, including problems like the traveling salesman [**10**], graph partitioning [**11**], and the generalized spanning tree [**12**]. Within the ACO framework, each artificial ant explores potential solutions in a probabilistic manner. These potential solutions are assessed based on a performance criterion analogous to the pheromone concentrations left by biological ants. While ACO does not always guarantee convergence to the global optimum, it is highly effective at finding near-optimal solutions within a feasible computational timeframe.

The use of ACO in traffic engineering has also garnered interest among researchers. Recent studies have applied ACO to complex vehicle routing problems where travel costs are uncertain [**12**]. Moreover, traffic signal optimization at intersections, particularly those with bidirectional traffic flows requiring only two-phase controllers [**13**], has been explored using ACO. However, contemporary traffic networks and control systems demand more advanced signal configurations to manage the diverse traffic flows across various directions and road links.

The project aims to alleviate traffic congestion and improve traffic signal efficiency by implementing an intelligent ant colony strategy enhanced by Q-learning (ACOQ). Experimental findings affirm that this algorithm delivers exceptional results, rendering it ideal for heterogeneous computing environments and demanding computational tasks. The adoption of a Q-Learning based Intelligent Ant Colony Optimization (ACO) strategy for traffic signal optimization brings manifold advantages that significantly boost traffic management efficacy. This adaptive learning model enables the system to independently learn from real-time traffic data and refine its operational strategies accordingly. By optimizing traffic flow at intersections, the system minimizes wait times at signals and ensures smoother transitions, thereby reducing overall journey durations—a significant benefit for commuters. Moreover, the enhanced traffic flow markedly reduces fuel usage and vehicle emissions, supporting environmental sustainability objectives. The system's scalability and adaptability allow it to expand across larger traffic light networks and adjust to diverse traffic conditions, including unplanned disruptions like road construction or accidents. Furthermore, this strategy not only enhances road safety by decreasing the likelihood of congestion-induced accidents but also offers a cost-effective solution by reducing the necessity for manual traffic control and diminishing the economic impacts of traffic congestion. Thus, implementing a Q-Learning based Intelligent ACO system promises to transform urban traffic management into a more efficient, safer, and environmentally conscious operation.

1.1 Motivation Behind the Research

The traffic signal problem is addressed very naturally as a combinatorial optimization problem. As traffic networks grow, the complexity of the finding an optimal solution becomes much more difficult. Total enumeration of all solutions becomes intractable very quickly, so advanced methods must be used [9]. ACO algorithms have successfully been applied to many computationally complex combinatorial problems, making ACO a good choice for solving the traffic signal problem. The ACO ability to incorporate heuristic information about traffic networks makes it more efficient. For example, in the isolated traffic signal problem the maximum queue length currently at the signal is accounted for. On more complicated traffic topologies, other heuristic measures can be incorporated, such as distances between signals. Ant colony optimization has successfully been applied to other traffic related problems, such as the vehicle routing problem (VRP), with positive results. Although the VRP has a different setup and objectives, similar heuristics and objective functions are used in both cases. As will be discussed later in this section, the ACO can be used to optimize rolling horizon control. Rolling horizon control has successfully been used in traffic signal control [13]. Some of the advantages of this approach were discussed in Chapter Two. Additionally, ACO requires very few restrictions on the cost function. For example, many optimization techniques rely on computing a gradient. This requires the existence of a gradient and can be computationally expensive. ACO algorithms are not dependant on the form of objective function; if the objective function is changed the algorithm works the same. This allows the heuristic information, intersection topology, and vehicle arrival rates to be easily changed. Thus, the ACO robustly conforms to new situation

2 Background

The research by (Renfrew & Yu, 2012)[**15**] explores a novel application of the Ant Colony Optimization (ACO) algorithm to develop optimal signal timing plans for traffic intersections, aiming to minimize average vehicle delay times. The effectiveness of the ACO algorithm, enhanced by a heuristic local search mechanism with weighted pheromone levels, was rigorously tested through simulations at intersections experiencing varied vehicle arrival rates, ranging from 400 to 850 vehicles per hour per movement. Performance comparisons were drawn against traditional fully actuated control algorithms adhering to the National Electrical Manufacturers Association (NEMA) standards. Various configurations of the ACO algorithm were explored, using 10, 25, and 50 ants, to study their effectiveness as demonstrated in convergence graphs. Additionally, a rank-based ant system algorithm incorporating local search and heuristic strategies was specifically applied to manage traffic signals, significantly reducing vehicle waiting times even under conditions of high traffic demand. Results confirm that this method surpasses conventional control systems, with the added advantage of being sufficiently rapid for real-time implementation in traffic management systems.

Jinjian et al.[**16**] proposed a traffic control system for isolated intersections that eliminates traditional traffic lights and phases. Using a Vehicle-to-Infrastructure (V2I) communication framework, vehicles dynamically share their information upon entering a communication zone. The system allocates right-of-way based on real-time data, optimizing the passing sequence with the Artificial Bee Colony (ABC) optimization algorithm. This approach minimizes delays by aligning vehicle speeds with the optimized sequence, allowing for near free-flow travel speeds. Simulation results highlight the system's efficiency in reducing delays and improving traffic flow without relying on conventional signals. The study of (Haldenbilen, Baskan, and Ozan, 2013)[**17**], addresses area traffic control problem using the ACOTRANS using an Ant Colony Optimization (ACO)-based algorithm known as ACORSES, designed to optimize signal parameters in coordinated signalized networks for a fixed set of link flows. One of the key advantages of ACO, and specifically ACORSES, is its ability to optimize signal timings with less computational complexity compared to traditional methods. By leveraging a reduced search space, the algorithm avoids being trapped in poor local optima, enabling it to converge quickly to a global or near-global optimum. During its progression, the algorithm searches for optimal signal timings within this limited space, mimicking the way ants find efficient routes. Using the Performance Index as a metric, the results demonstrate that ACORSES outperforms other optimization methods such as genetic algorithms and hill-climbing techniques.

Rutgar et al. [**18**] introduced an Ant Colony Optimization (ACO) algorithm designed to address challenges in distributed systems, particularly traffic congestion. The proposed method employs intelligent systems and future estimations to avoid congestion by enabling cooperation among ants representing vehicles. Through shared pheromone trails, ants exchange general knowledge to improve route-finding. A simulation of this cooperative ACO algorithm was conducted to evaluate its effectiveness compared to a non-cooperative version, measuring both the quality of calculated routes and the number of iterations required to reach optimal solutions. The results indicate that while the cooperative approach reduces the number of iterations needed to find solutions, the overall improvement in route quality and computational speed is minimal, especially for smaller networks. However, the reduction in iterations is significant in scenarios where inter-agent communication is a limiting factor. This highlights the adaptability of ACO algorithms in traffic routing by allowing ants to share information across multiple routing problems within a distributed traffic graph. Despite its modest advantages, the cooperative variant demonstrates the potential for ACO algorithms to incorporate generalized information-sharing in addressing complex traffic management issues.

3 Methods

3.1 Ant Colony Optimisation

Ant Colony Optimization (ACO) is a metaheuristic specifically designed to tackle computationally challenging combinatorial optimization problems. Initially developed to address the Traveling Salesman Problem, ACO has since proven effective for a variety of NP-hard problems, including routing issues, quadratic assignment, and scheduling dilemmas. The inspiration for ACO comes from the foraging behaviors observed in ant colonies. [**19**]

In their natural environment, ants initially wander randomly near their nests in search of food. Upon discovering a food source, an ant assesses its quality and quantity before heading back to the nest. During this return journey, the ant leaves behind a pheromone trail—a chemical means of communication among ants. The strength of the pheromone trail depends on the food's quality and quantity, guiding other ants to the source. Over time, as more ants follow and reinforce this trail, their paths become more direct, concentrating on routes with stronger pheromone signals. However, pheromones also evaporate over time, which means that less-traveled paths gradually lose their trail markers unless continuously reinforced. [**20**]

This process not only helps ants locate the most abundant food sources but also the shortest routes to them. Shorter routes get traversed more frequently, leading to quicker and more substantial pheromone deposits, thus accelerating the path-finding process. [**20**]

A famous illustration of this behavior is the Double Bridge Experiment[**21**] shown in Figure 3, where an ant colony is provided with two equal-length bridges between their nest and a food source. Initially, ants select either path at random, but as time passes, one path may begin to accumulate more pheromones due to random fluctuations. This increased pheromone level attracts more ants, who add their own pheromones to the path, thus reinforcing a positive feedback loop. Eventually, this can lead to all ants favoring and using one bridge exclusively.

In a subsequent trial of this experiment, the lengths of the two bridges were varied—one was made twice as long as the other. Despite initial random usage of both bridges, the shorter bridge quickly accumulated more pheromones, eventually directing all traffic along its route. This experiment illustrates how ACO leverages these natural efficiencies to optimize route selection in computational tasks.

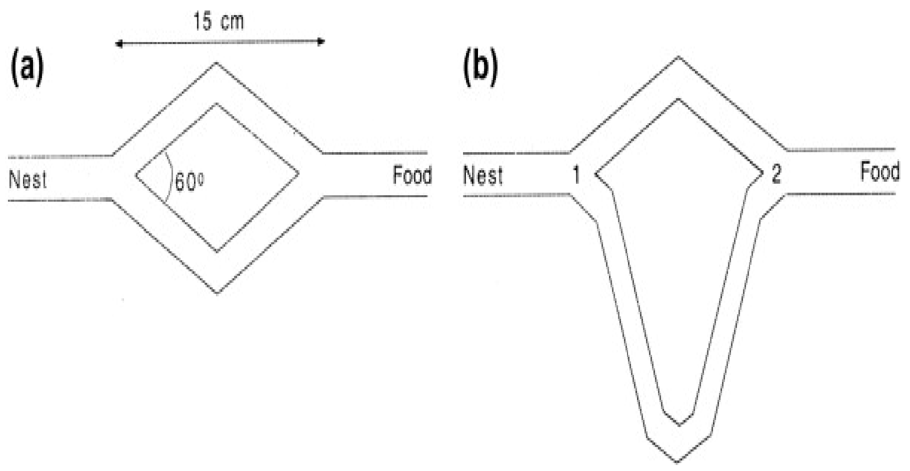


Figure 1: The double bridge experiment features two setups: (a) both bridges of equal length, and (b) bridges of varying lengths[**22**].

Algorithm 1 Generic Ant Colony Algorithm

```
1: Step 1: Initialization
2:     Initialisation of pheromone trail
3: Step 2: Construction of the Solution
4: for each ant do
5:         Construct the solution using the pheromone trail
6: end for
7: Step 3: Updation of the Pheromone Trail
8:     Update the pheromone trails
9:     Repeat until stopping criteria are met
```

The basic workflow of the Ant Colony Optimization (ACO) algorithm is outlined above. Initially, a pheromone path is established to direct the simulated ants. During each iteration, these ants develop potential solutions by tracing the paths previously laid out, thus effectively navigating and utilizing the available search area. After forming the solutions, adjustments are made to the pheromone levels depending on the quality of these solutions. This process is repeated until a predetermined endpoint is reached, aiming to discover the optimal or a near-optimal solution.

3.2 Q-Learning Algorithm

Q-learning is a type of model-free reinforcement learning algorithm that determines the value of an action in a specific state without needing a model of the environment. This approach is capable of addressing problems characterized by stochastic transitions and rewards, and it adapts without any additional modifications. [**23**]

In the context of traffic signal control, Q-learning operates by developing a policy that guides an agent—here, a traffic light controller—on which actions to take in various traffic situations. Essentially, it learns to associate the state of the environment, such as the current traffic conditions at an intersection, with actions that adjust traffic light phases. The goal is to maximize a cumulative reward, which could involve minimizing vehicle waiting times at intersections[**23**].

Implementing traffic signal control using a Q-learning approach involves a series of carefully structured steps. Initially, the states are defined to represent various traffic conditions at an intersection, such as the number of vehicles on each approach, the current traffic light phase, or the time since the last phase change. Actions are then defined, typically as transitions between different traffic light phases. Each of these transitions is considered a potential action within the system. The reward function, which is vital for guiding the learning algorithm, is designed to minimize waiting times by penalizing longer delays, thus incentivizing the system to find solutions that reduce wait times[**23**].

The Q-table, an essential element of this setup, is initialized with values for each state-action pair, which are often set to zero or a small random number to kickstart the learning process. The learning process itself involves observing the current state of traffic at each time step, then selecting and performing actions based on an ϵ-greedy policy. This policy allows the agent to mostly exploit the best-known action as per the Q-table but occasionally explore new actions with a probability determined by ϵ. This approach helps balance the need to exploit known good actions with the need to explore potentially better options[**23**].

This cycle of observation, action selection, and execution is repeated until the Q-values converge, indicating that the algorithm has learned a stable and effective policy for controlling traffic signals. Through these steps, the Q-learning algorithm is equipped to effectively optimize traffic flow and reduce waiting times at intersections, improving overall traffic management. [**23**]

3.3 Q-Learning Algorithm combined with ACO

In my research, I adapted the innovative methodology proposed by Li et al. (2021)[**24**], which initially applied a combination of Q-Learning with Ant Colony Optimization (ACO) to address scheduling challenges, reconfiguring it to tackle the complexities of traffic signal optimization. The original study demonstrated the potential of merging these two robust algorithms to enhance decision-making processes in heterogeneous systems. By pivoting this approach to traffic management, I redefined the nodes to represent intersections or pivotal decision points within urban traffic networks. Paths in this adapted model symbolize the possible routes or directional flows that can be taken at each intersection, where metaphorical 'ants' traverse these paths. These paths are influenced by the effectiveness of the signal timings, which are in turn dictated by the Q-values generated from a sophisticated reinforcement learning process.

By intelligently integrating initial pheromone concentrations with these Q-values, the algorithm is endowed with a heuristic advantage from the onset. This initial bias helps accelerate the convergence process, directing the early stages of exploration towards more promising and effective pathways. This strategic approach effectively combines the strengths of ACO in identifying optimal paths through historical performance data (pheromone trails) with the adaptive, dynamic capabilities of Q-learning, which updates and refines strategies based on real-time data about traffic flow and congestion patterns. The convergence of these methodologies not only enhances the efficiency of traffic signal management but also provides a scalable model that can adapt to varying traffic conditions and continuous changes within urban traffic networks. This hybrid model thus represents a significant advancement in applying intelligent algorithms to solve real-world problems, specifically in the domain of urban traffic control, by leveraging historical insights and real-time adaptive learning to optimize traffic flows and reduce congestion systematically. [24]

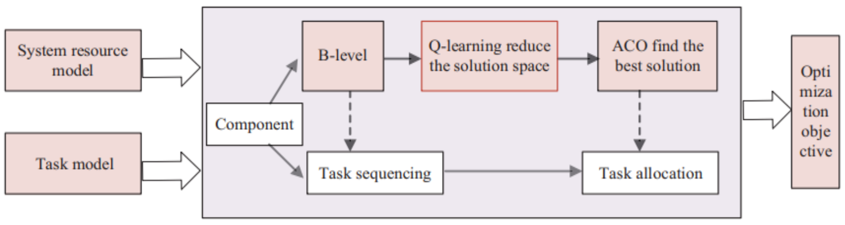


Figure 2: Design Framework of the Proposed Algorithm[24]

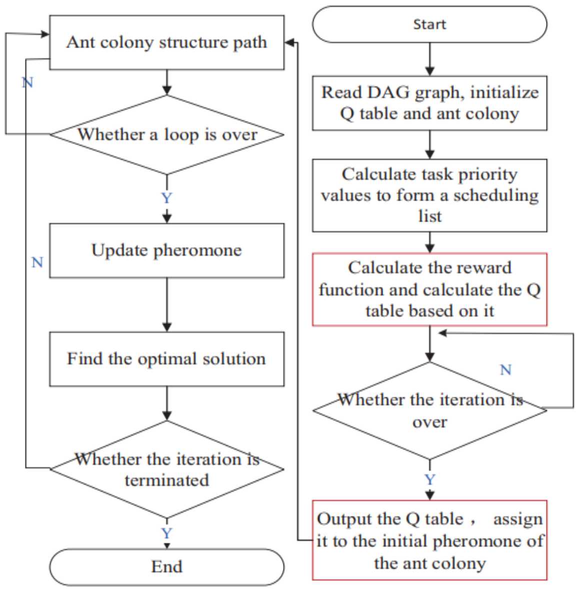

Start
Read DAG graph, initialize Q table and ant colony
Calculate task priority values to form a scheduling list
Calculate the reward function and calculate the Q table based on it
Whether the iteration is over
N
Y
Output the Q table， assign it to the initial pheromone of the ant colony
Ant colony structure path
Whether a loop is over
N
Y
Update pheromone
Find the optimal solution
Whether the iteration is terminated
N
Y
End

Figure 3: Flow Chart of the Proposed Algorithm[24]

In the system design framework for optimizing traffic signals, the metaphorical use of "ants" and "intersections" forms the cornerstone of the approach. In this setup, each "ant" embodies a series of strategic decisions designed to manage the traffic signals across a comprehensive network of intersections. The paths that these ants follow represent various sequences or adjustments in the timing of traffic light changes, each path offering a potential solution to traffic congestion.

To ensure thorough optimization, the system completes what is referred to as a "cycle," during which all traffic lights within the network undergo a full program of changes. This cycle is meticulously designed to optimize traffic flow by adjusting to current traffic conditions dynamically. To achieve robust outcomes, the system employs a parallel search strategy where multiple decision-making sequences, or "ants," are executed simultaneously. Each ant tests different traffic management strategies in parallel, allowing for a diverse exploration of possible solutions and enabling the system to identify the most effective strategies quickly.

The strategic use of a Q-table to set initial pheromone levels is particularly significant in this model. This initial setup provides the system with baseline knowledge of potentially effective traffic management strategies, based on historical data and prior simulations. Over time, as the system encounters real-world traffic conditions, it continuously refines and adapts its strategies. This learning process is driven by ongoing adjustments based on actual traffic patterns and outcomes, enhancing the system's ability to respond dynamically to changes and improve traffic flow efficiently.

Through this sophisticated approach, the traffic signal optimization system not only increases its efficiency but also its adaptability, making it capable of handling complex traffic scenarios and improving overall traffic management in urban environments.

In the stage of traffic task sorting, the priority of each intersection is determined recursively using the B-level method, which incorporates the above equations (1) and (2). The intersections are then organized into a scheduling list based on the descending order of their priority values.[**24**]

$$\text{priority}(v_i) = w_i + \max_{v_j \in \text{succ}(v_i)} (\bar{c}_{i,j} + \text{priority}(v_j)) \tag{1}$$

$$\bar{c}_{i,j} = \left(\sum_{m=1}^{n-1} \sum_{s=m+1}^{n} \frac{c_{i,j}}{q_{m,s}} \right) \Bigg/ \left(p \times \frac{p-1}{2} \right) \tag{2}$$

$q_{m,s}$ denotes the number of communication paths between processor m and s, $\bar{c}_{i,j}$ represents the average communication overhead between intersections v_i and v_j, $\text{succ}(v_i)$ refers to the succeeding intersections in the task.

3.4 Design of the Algorithm

The traffic management simulation was developed using Python and Numpy to effectively manage Q-table and pheromone levels for addressing real-world traffic flow problems. Configured with five intersections, each allowing three actions: no change, change to green, or change to red, the simulation integrates essential constants such as pheromone decay and initial Q-table values to model behavior over time. The `TrafficEnvironment` class centralizes management of intersections and pheromone synchronization, featuring a reset method that aligns pheromone levels with Q-table updates. The `get_traffic_metrics` function plays a crucial role by simulating traffic delays and rewarding strategies that minimize waiting times.

In the second phase, introducing "Ant Agents," the simulation employs the `Environment` and `Ant` classes to explore ant colony optimization (ACO). Ants make probabilistic decisions influenced by pheromone levels at each intersection, navigating from one intersection to the next, and recording their paths. Each cycle allows ants to reset and prepare for new decisions, enhancing route optimization through continuous interaction with the environment.

The third phase focuses on simulation loops and pheromone updates, essential for refining ACO strategies. The `update_pheromones` function adjusts pheromone levels based on ant decisions, encouraging exploration of new paths. The simulation runs across multiple epochs, with ants influencing pheromone distribution and adapting their paths accordingly, allowing for iterative improvements in traffic management.

Finally, the simulation incorporates TraCI control scripts to link the ACO framework with SumoLink. Verifying the `SUMO_HOME` variable ensures proper tool access, and the `run_simulation` function oversees the simulation through SUMO's GUI, managing real-time steps and updating traffic lights based on ant decisions. The simulation concludes with `traci.close()`, demonstrating the efficacy of ant colony optimization in a dynamic urban traffic setting.

Algorithm 2 Pseudo Code for Traffic Simulation using the proposed ACO

```
1: Step 1: Setup
   Define the number of intersections and possible actions (no change, change to green,
   change to red). Initialize constants for simulation (pheromone influence, heuristic
   influence, decay factor).
2: Step 2: Define Environment and Ants
   Create a TrafficEnvironment class to handle intersections and their pheromone levels.
   Create an Ant class where each ant can make decisions based on pheromone levels.
3: Step 3: Simulation Process
   Define a function to run simulations over a specified number of epochs.
   Each ant starts at a random intersection and moves through the environment making
   decisions.
   After each move, update pheromones based on the ants' decisions and the traffic
   conditions they experience.
4: Step 4: Pheromone Update Mechanism
   Apply a decay to reduce all pheromone levels gradually. Increase pheromones on
   paths that yield better traffic conditions (e.g., less delay).
5: Step 5: Reward Function
   Define a reward function that evaluates the effectiveness of actions taken by ants
   based on traffic metrics like delay.
6: Step 6: Integration with SUMO
   Set up the path to SUMO tools and ensure necessary environment variables are set.
   Run the simulation in SUMO using a configuration file (e.g., simulation.cfg).
   Use Traci to interact with the SUMO simulation, allowing ants to influence traffic
   light phases based on their decisions.
   Update pheromones based on simulation feedback and repeat until the simulation
   ends.
7: Step 7: Conclusion
   Each epoch of simulation provides insights into better traffic management through
   the use of pheromones influenced by ant decisions.
   Debug output to track pheromone levels after each epoch to monitor and adjust
   strategies.
```

3.5 Simulation Using Sumolink

In this research, I employed the Simulation of Urban Mobility (SUMO) tool, an open-source and versatile traffic simulation package. The chosen route for this study was located within the Kamppi district of Helsinki, Finland. Given SUMO's ability to manage large road networks and perform detailed microscopic traffic simulations, it was perfectly suited for analyzing the complexities of urban traffic.

I configured the simulation environment by constructing a custom network that covered a specific section of Kamppi, featuring multiple intersections complete with traffic signals to mirror the unique traffic patterns of the area accurately. The ".osm" file for the map obtained is attached in the GitHub named as "map.osm". The geographical details of Kamppi were sourced from OpenStreetMaps; the data was exported into an .osm file and then transformed using SUMO's 'netedit' tool. To add variability to the traffic flows, I utilized 'Random.py', a Python script, to create diverse traffic routes.[**25**]

Due to the computational constraints of my system, I set the simulation time to 30 minutes and capped the number of moving vehicles at 100. This configuration was selected to maintain manageability and ensure consistent performance during the simulation process.

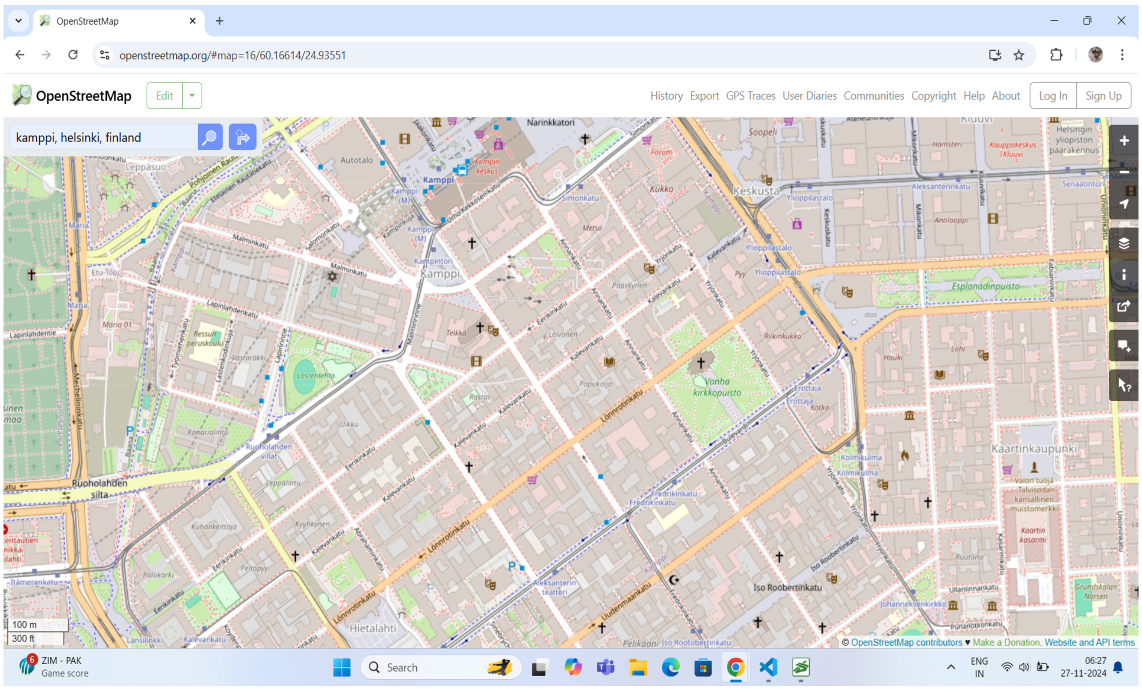
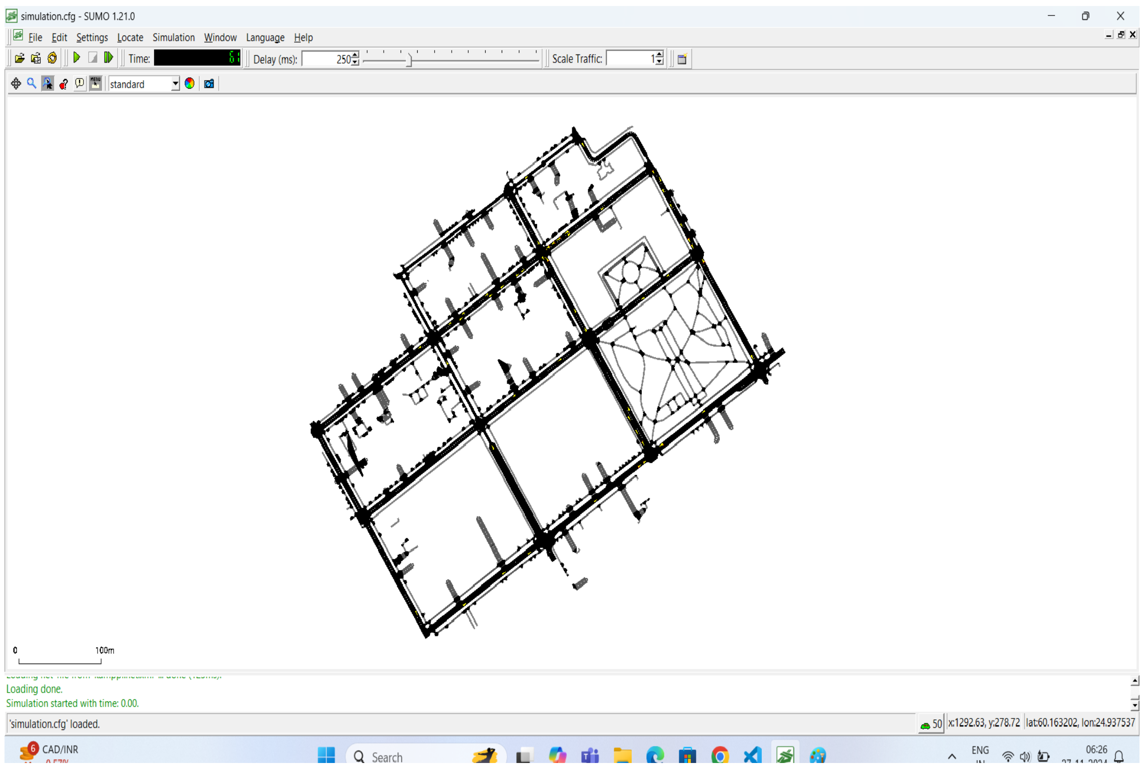


Figure 4: OpenStreet Map of Kamppi


simulation.cfg - SUMO 1.21.0
File Edit Settings Locate Simulation Window Language Help
Time:
Delay (ms): 250
Scale Traffic: 1
standard
0
100m
CAD/INR
Search
ENG
06:26

```
Loading done.
Simulation started with time: 0.00.
'simulation.cfg' loaded.
50 x:1292.63, y:278.72 lat:60.163202, lon:24.937537
```

Figure 5: Road Network from Sumolink for the proposed Algorithm

I successfully executed the code by connecting Python to SumoLink using the TraCI API, which proved to be a valuable tool for controlling simulation parameters with Python. A '.cfg' configuration file is essential for initiating the simulation[**25**]. There are two methods to conduct the simulation:

Using GUI: This method utilizes a graphical user interface within the application known as 'sumo-gui'. It provides a visual representation of the simulation, making it user-friendly but computationally intensive.

Non-GUI Python Simulation: Alternatively, Python can run the simulation without the GUI, which is a more streamlined option as it simplifies the process and reduces computational demands.

During the simulation, key data points were targeted for collection:

Traffic Light Phases: We recorded the state and duration of each traffic light phase at every simulation step, providing detailed insights into traffic signal behavior.

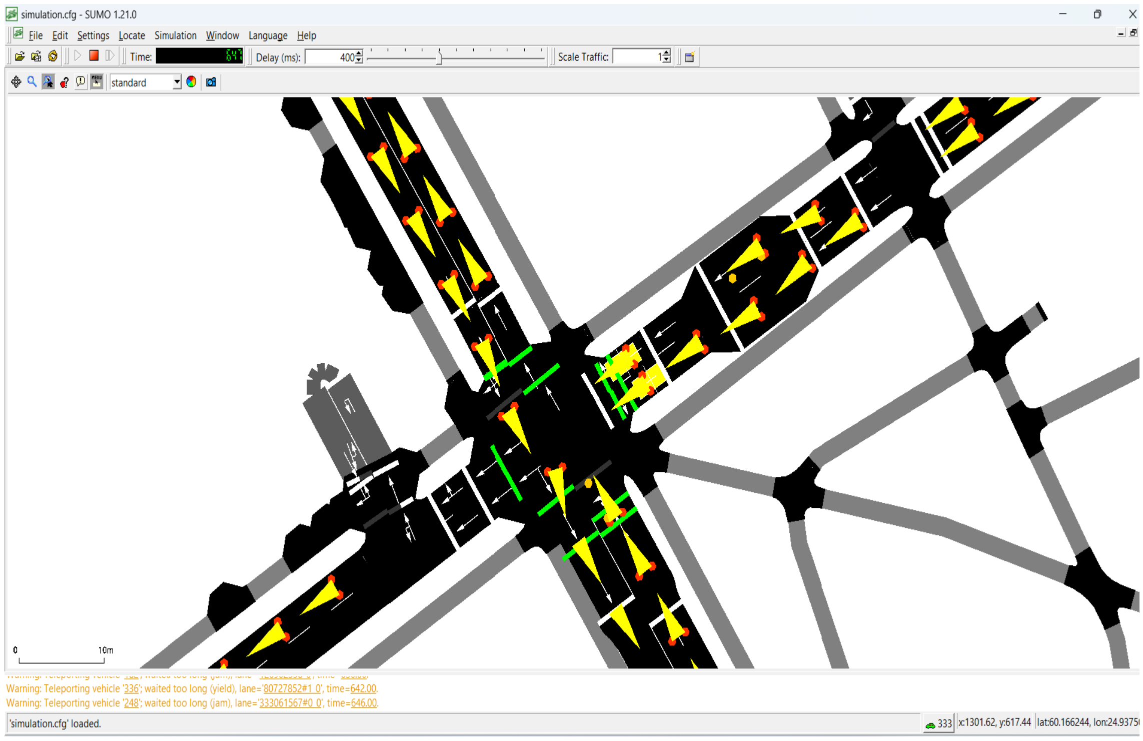
Vehicle Waiting Times: Data collection focused specifically on peak periods, previously identified from preliminary simulation runs, which exhibited the highest levels of congestion. This method facilitated an understanding of critical impact points and aided in optimizing traffic light phases.

The simulation was meticulously designed to capture a broad spectrum of data under varying traffic conditions, including peak times, off-peak times, and weekends. The goal was to thoroughly understand the dynamics at these critical moments, focusing on:

- The behavior of traffic lights during periods of high vehicle congestion to better understand their impact on traffic flow.

- The detailed waiting times for each vehicle, which are crucial for assessing the effectiveness of current traffic light timings and identifying areas for potential improvement.

Given the complexity of the network and the demands on computing resources, and considering the specifications of my personal system, I focused on a specific junction with the ID ”3228733111”. This junction ID was retrieved from the ’.net.xml’ file associated with the ’.cfg’ file. Both a ’.net.xml’ and a ’.rou.xml’ file are necessary to conduct the simulation. An illustration of the junction is presented below.


simulation.cfg - SUMO 1.21.0
File Edit Settings Locate Simulation Window Language Help
Time:
Delay (ms): 400
Scale Traffic: 1
standard
0
10m

```
Warning: Teleporting vehicle '336'; waited too long (yield), lane='80727852#1_0', time=642.00.
Warning: Teleporting vehicle '248'; waited too long (jam), lane='333061567#0_0', time=646.00.
'simulation.cfg' loaded.  333  x:1301.62, y:617.44  lat:60.166244, lon:24.9375
```

Figure 6: Road Network from Sumolink for the proposed Algorithm

The ’run simulation’ function was pivotal in this process. It initiated the SUMO simulation, systematically collected the necessary data, and terminated the simulation once data gathering was complete. It produced a well-organized dictionary that included timestamps, corresponding traffic light phases, and individual vehicle waiting times. This methodical approach not only streamlined the analysis but also increased the precision and relevance of the results[**25**]. All pertinent files have been made available on the GitHub repository.

4 Results

4.1 Data Preprocessing

In the development of the traffic optimization system, I initially searched for an appropriate dataset but was unsuccessful. This challenge directed me to the insightful paper ”Microscopic traffic simulation using sumo” [**25**], which provided a methodology for generating custom datasets using Python and SUMO. Employing this approach, I crafted several datasets tailored for analyzing the algorithm, encompassing vehicle emissions and waiting times across various traffic scenarios, all formatted in ’xml’ [**25**]. I chose this method because utilizing historical data to establish initial conditions and validate simulation outcomes seemed to offer a straightforward and effective strategy.

These are the datasets generated using SUMO:

- `emmisionpeak.xml` - Emission of vehicles in peak traffic
- `emmisionoffpeak.xml` - Emission of vehicles in off-peak traffic
- `emmisionweekend.xml` - Emission of vehicles during weekend traffic
- `queuepeak.xml` - Waiting time for vehicles in peak traffic
- `queueoffpeak.xml` - Waiting time for vehicles in off-peak traffic
- `queueweekend.xml` - Waiting time for vehicles during weekend traffic

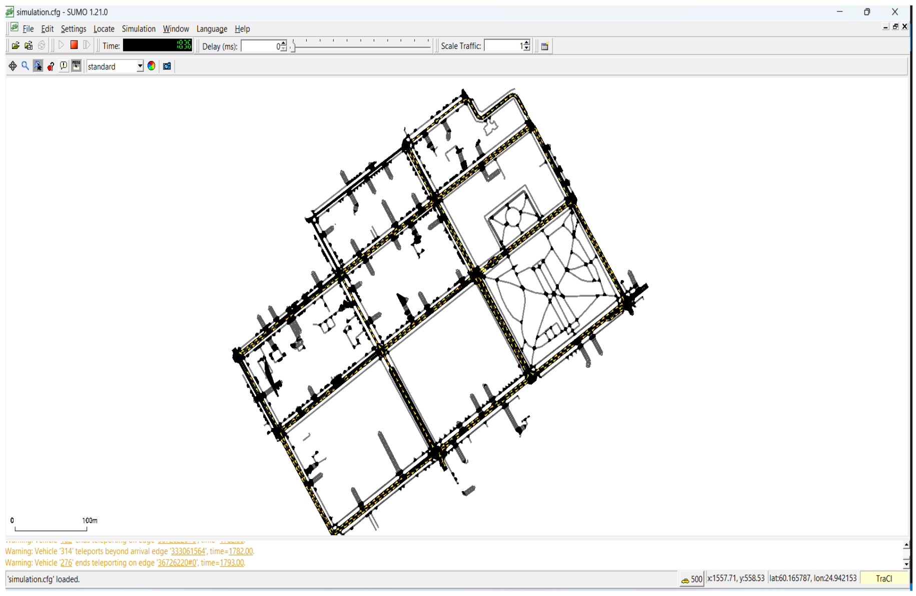
In the development of the traffic optimization system, my initial search for an appropriate dataset was unsuccessful. This challenge directed me to the influential paper, "Microscopic traffic simulation using SUMO" [**25**], which provided a comprehensive guide on generating custom datasets using Python and SUMO. Inspired by this approach, I generated several datasets tailored for algorithm analysis, including emissions and vehicle waiting times under various traffic conditions, formatted in 'xml' [**25**]. I opted for this method because using historical data to establish initial conditions and validate the simulation results simplifies the process. Moreover, this approach ensures that the simulation begins under realistic scenarios, enhancing the validity and applicability of the outcomes.

4.2 Evaluation and Performance

Table 1: Parameter and Values used in the experiment

Parameters	**Value**
Sumo Version	1.21.0
Python Version	3.10.0
Simulation Steps	500s
Road Network	Kamppi, Helsinki, Finland
Number of Intersections	5
Pheromone Evaporation Rate	1.0
Beta	1.0
Decay	0.1
`Q_INIT_VALUE`	0.2

To assess the program's compatibility, a traffic simulation was conducted using the TraCI module from SUMO. The process initiates with the `run_simulation()` function, which starts SUMO's graphical user interface via `traci.start(['sumo-gui', '-c', 'simulation.cfg'])`, using `simulation.cfg` for essential simulation settings. The simulation operates within a while loop, continuing as long as there are vehicles on the network, monitored by `traci.simulation.getMinExpectedNumber() > 0`. Simulation steps are processed using `traci.simulationStep()`, and traffic light controls are dynamically adjusted every minute to test various traffic management strategies; specifically, traffic lights are reset to phase 0 every 60 seconds if divisible by 60. Upon completion, `traci.close()` terminates the simulation and clears resources, ensuring a smooth shutdown. This setup allows for evaluating different traffic light configurations and their impacts on urban traffic flow, showcasing the dynamic capabilities of the SUMO traffic simulator.


simulation.cfg - SUMO 1.21.0
File Edit Settings Locate Simulation Window Language Help
Time:
Delay (ms):
Scale Traffic:
standard
0
100m
500
TraCI

```
Warning: Vehicle '314' teleports beyond arrival edge '333061564', time=1782.00.
Warning: Vehicle '276' ends teleporting on edge '36726220#0', time=1793.00.
'simulation.cfg' loaded.
x:1557.71, y:558.53  lat:60.165787, lon:24.942153
```

Figure 7: Simulation Time Step Analysis

Table 2: Outputs from the Simulation

Metric	Value
Total number of stops	24
Average speed (m/s)	3.361
Average delay (s)	52.194

To effectively monitor and manage the variability in traffic light phases due to route generation via Random.py, the TraCI module was employed to scrutinize changes at the intersection. Initial observations highlighted frequent and irregular phase changes, exemplified by a phase change to 1 at 0.083 seconds, with a similarly brief interval since the last change, raising concerns about the unexpected brevity of these intervals.

Further analysis revealed several critical adjustments made to enhance system consistency and efficiency:

Adjustments for Consistency: The system is configured to enforce minimum durations for main phases, setting a standard of no less than 30 seconds for phases that do not involve transitions. This adjustment is crucial for ensuring intersections clear safely, particularly during high-traffic periods.

Maintained Transitions: To keep traffic flowing smoothly, the script actively maintains short durations for transition phases, particularly those involving yellow lights. This approach effectively minimizes downtime and ensures quick transitions between traffic signals.

Comprehensive Control: The diversity in traffic light configurations, such as 'Gr', 'GGrrrG', and 'GGGrrr', demonstrates the system's capacity to handle complex traffic scenarios. This includes managing different traffic directions and accommodating pedestrian movements, which are essential for urban traffic networks.

These measures illustrate the system's refined approach to traffic light management, ensuring both safety and efficiency at busy intersections. Each element of the control strategy serves to enhance the overall functionality and responsiveness of the traffic management system.

To gain a comprehensive understanding of the traffic flow at our focal point, which is the junction identified by the ID "3228733111", we analyzed the detector data. The following observations were obtained:

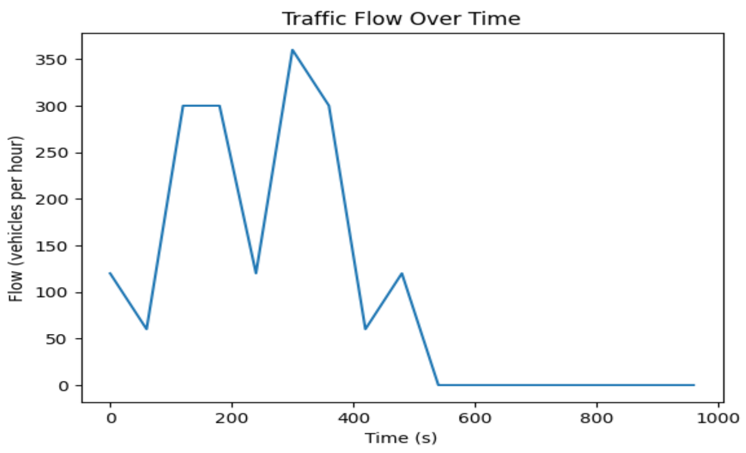


Figure 8: Time Flow over time from Detector 1

Table 3: Summary Statistics for Detector Data 1

Statistic	Flow	Speed (m/s)	Occupancy (%)
Count	9.000000	9.000000	9.000000
Mean	53.333333	0.702222	62.986667
Std Dev	63.245553	2.510706	39.271967
Min	0.000000	-1.000000	0.000000
25%	0.000000	-1.000000	46.300000
50%	60.000000	0.020000	72.850000
75%	60.000000	0.250000	100.000000
Max	180.000000	5.850000	100.000000

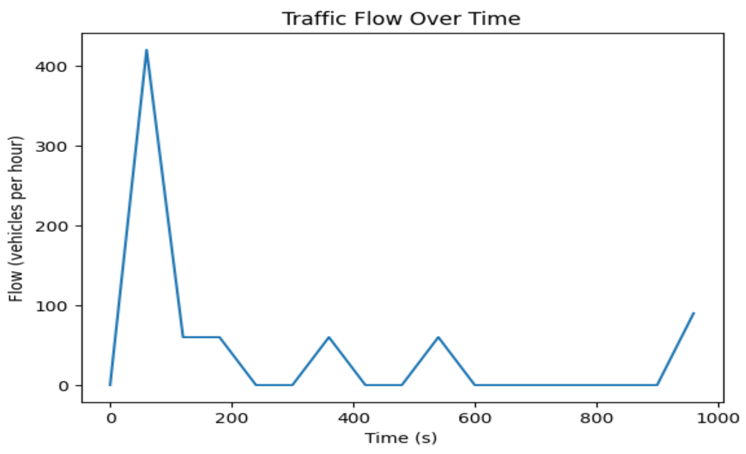


Figure 9: Time Flow over time from Detector 2

Table 4: Summary Statistics for Detector Data 2

Statistic	Flow	Speed (m/s)	Occupancy (%)
Count	9.000000	9.000000	9.000000
Mean	73.333333	0.583333	10.787778
Std Dev	155.241747	2.554662	18.410417
Min	0.000000	-1.000000	0.000000
25%	0.000000	-1.000000	0.000000
50%	0.000000	-1.000000	0.000000
75%	60.000000	0.260000	11.840000
Max	480.000000	5.900000	50.480000

Table 5: Fitness Values of the algorithms chosen (Part 2)

Function	Metric	ACOR	SamACO	RACO	ReinforcedACO
f_{15}	Average	4.34E+02	7.24E+02	3.27E+02	4.16E+02
	Standard Deviation	1.34E+05	5.14E+05	7.13E+04	1.48E+05
f_{16}	Average	1.94E+03	4.67E+02	4.34E+04	5.46E+02
	Standard Deviation	1.46E+03	2.59E+04	1.14E+06	1.01E+04
f_{17}	Average	9.64E+02	4.13E+02	8.45E+02	2.03E+02
	Standard Deviation	4.51E+04	5.74E+03	4.60E+03	1.14E+03
f_{18}	Average	1.46E+07	3.95E+05	4.15E+04	6.94E+03
	Standard Deviation	2.14E+10	1.45E+06	2.86E+04	9.12E+03
f_{19}	Average	3.14E+03	1.97E+04	7.91E+03	2.58E+03
	Standard Deviation	4.97E+06	2.01E+06	1.18E+05	6.27E+04
f_{20}	Average	9.10E+02	4.67E+02	3.48E+02	2.31E+02
	Standard Deviation	4.41E+03	1.17E+03	1.97E+03	1.64E+04
f_{21}	Average	4.25E+02	4.69E+02	1.95E+02	2.64E+02
	Standard Deviation	1.45E+03	8.14E+03	4.49E+03	1.47E+01
f_{22}	Average	8.64E+03	1.57E+03	7.84E+03	1.46E+03
	Standard Deviation	9.01E+04	5.10E+06	1.64E+06	9.72E+03
f_{23}	Average	2.97E+04	7.81E+03	3.45E+04	4.67E+03
	Standard Deviation	2.35E+07	8.16E+05	5.13E+06	4.15E+06
f_{24}	Average	1.25E+03	4.18E+02	5.19E+02	6.17E+02
	Standard Deviation	1.34E+05	5.47E+04	3.81E+05	4.12E+06
f_{25}	Average	6.25E+03	1.15E+03	4.16E+04	5.23E+03
	Standard Deviation	7.34E+06	8.16E+06	1.24E+06	2.47E+06
f_{26}	Average	4.25E+05	7.23E+04	5.14E+04	4.27E+04
	Standard Deviation	4.15E+08	1.47E+06	3.64E+06	2.14E+06
f_{27}	Average	3.46E+04	8.15E+03	2.14E+04	3.97E+03
	Standard Deviation	5.24E+07	1.34E+07	4.14E+06	3.45E+06
f_{28}	Average	7.14E+02	4.16E+02	3.64E+02	3.14E+02
	Standard Deviation	5.47E+04	8.13E+04	7.14E+04	4.27E+04

These results show that Reinforced ACO generally provides better and more stable outcomes for the simpler multimodal functions. When looking at the mixed functions f_{11} to f_{20} and the composite functions f_{21} to f_{30}, Reinforced ACO tends to outperform the other algorithms. Although it has slightly lower accuracy than RACO for functions f_{14} and f_{15}, it still outshines the other two algorithms. For functions f_{27} and f_{28}, both SamACO and RACO perform a bit better than Reinforced ACO, but the differences are minimal. Overall, Reinforced ACO has better average fitness for the other functions, and it also boasts a lower standard deviation compared to the other algorithms for most of the test functions.

Table 6: Fitness Values of the Algorithms

Function	n		ACOR	SamACO	RACO	ReinforcedACO
Shifted and Rotated Bent Cigar function f1	10	Avg.	3.45E-02	9.11E-01	4.57E+01	2.02E-01
		St. Dev	4.15E-06	2.34E-01	1.46E-01	1.45E-05
	10	Avg.	6.25E-01	1.89E+02	5.64E-01	6.14E-02
		St. Dev	1.94E-01	4.15E-01	4.50E+02	1.45E-01
	30	Avg.	7.23E+03	2.56E+03	2.75E+04	2.14E+03
		St. Dev	2.17E+07	2.14E+06	3.08E+08	2.07E+06
	100	Avg.	3.14E+08	1.97E+07	7.91E+08	2.58E+05
		St. Dev	4.97E+06	2.01E+06	1.18E+05	6.27E+04
	200	Avg.	4.14E+09	6.14E+07	7.54E+09	7.91E+07
		St. Dev	1.07E+05	1.48E+05	4.95E+04	7.24E+03
Shifted and Rotated Rosenbrock's function f4	10	Avg.	6.17E-02	5.26E-01	1.61E-01	9.24E-01
		St. Dev	1.30E-05	3.19E+02	1.26E+01	2.62E-01
	30	Avg.	2.47E+00	1.24E+01	3.24E+01	3.14E+00
		St. Dev	4.10E-05	2.39E+01	4.71E+01	2.62E-01
	100	Avg.	2.48E+04	7.26E+03	8.15E+04	6.15E+03
		St. Dev	1.04E+05	2.63E+03	3.31E+05	1.91E+04
	200	Avg.	4.04E+10	5.61E+10	6.24E+09	4.64E+09
		St. Dev	3.51E+03	4.64E+03	7.84E+06	8.45E+02
Shifted and rotated Levy function f9	10	Avg.	6.17E-02	5.26E-01	1.61E-01	9.24E-01
		St. Dev	1.30E-05	3.19E+02	1.26E+01	2.62E-01
	30	Avg.	4.58E+08	1.50E+05	1.37E+05	1.74E+04
		St. Dev	1.46E+15	3.48E+09	1.34E+08	5.14E+03
	200	Avg.	6.21E+10	6.55E+09	7.24E+10	3.29E+09
		St. Dev	1.64E+07	5.34E+06	1.15E+07	1.36E+05

Tables 5 and 6 reveal that when it comes to the unimodal function f_1, the Reinforced ACO algorithm outperforms the other three algorithms in terms of both convergence accuracy and standard deviation, regardless of whether the test functions are low-dimensional or high-dimensional. This clearly indicates that Reinforced ACO offers superior convergence accuracy for unimodal functions.

For the simple multimodal functions f_4 and f_9, Reinforced ACO consistently delivers better results compared to the other algorithms. In the case of the low-dimensional f_4, ACOR initially shows better performance. However, as the dimensions increase to 100 and 200, ACOR's performance declines, and Reinforced ACO takes the lead. This suggests that ACOR struggles with high-dimensional simple multimodal functions, while Reinforced ACO maintains a more robust and stable performance across all scenarios.

Table 7: Fitness Values of the Hybrid and Composition Functions

Function	n		ACOR	SamACO	RACO	Reinforced ACO
Hybrid Function f11	10	Avg.	2.15E+02	1.58E+01	3.19E+02	1.60E+01
		St. Dev	1.24E+02	7.15E+02	2.47E+01	4.15E+01
	30	Avg.	3.48E+03	4.19E+02	5.15E+01	5.12E+01
		St. Dev	5.15E+06	2.87E+06	3.27E+02	4.14E+03
	100	Avg.	6.94E+09	6.15E+07	1.24E+07	3.16E+06
		St. Dev	1.14E+06	2.59E+04	1.46E+03	1.60E+04
	200	Avg.	9.51E+10	5.15E+09	6.55E+09	2.03E+09
		St. Dev	4.12E+05	1.84E+02	8.10E+03	1.14E+03
Hybrid Function f20	10	Avg.	2.48E+02	1.45E+02	2.64E+02	4.56E+01
		St. Dev	2.24E+02	7.41E+02	4.56E+02	1.07E+01
	30	Avg.	9.10E+02	4.67E+02	3.48E+02	2.31E+02
		St. Dev	4.41E+03	1.17E+03	1.97E+03	1.64E+02
	100	Avg.	1.10E+06	8.11E+07	1.45E+06	1.51E+05
		St. Dev	3.51E+01	6.11E+02	5.10E+03	1.04E+05
	200	Avg.	3.25E+11	4.69E+10	1.95E+09	2.64E+09
		St. Dev	1.45E+03	8.14E+03	4.49E+03	1.47E+01
	10	Avg.	7.12E+02	3.12E+01	5.16E+01	1.26E+01
		St. Dev	1.21E+05	4.68E+05	7.14E+04	7.26E+03
	30	Avg.	8.64E+03	1.57E+03	7.84E+03	1.46E+03
		St. Dev	9.01E+04	5.10E+06	1.64E+06	9.72E+03
Composite Function f22	100	Avg.	2.17E+07	5.45E+08	1.46E+07	6.94E+06
		St. Dev	7.34E+02	4.22E+08	7.41E+01	1.89E+02
	200	Avg.	3.46E+10	5.97E+10	1.60E+07	2.14E+07
		St. Dev	6.41E+06	1.51E+06	6.41E+04	5.32E+04
	10	Avg.	2.48E+01	5.05E+02	4.16E+02	1.35E+01
		St. Dev	1.04E+06	4.36E+05	3.31E+04	1.87E+04
Composite Function f26	30	Avg.	6.45E+03	8.14E+02	6.21E+02	7.42E+02
		St. Dev	3.43E-05	1.25E-02	1.85E+02	1.13E+02
	100	Avg.	7.40E+06	1.64E+08	4.78E+02	4.82E+02
		St. Dev	1.53E+06	1.01E+08	5.64E+06	4.10E+03
	200	Avg.	4.41E+09	5.68E+10	3.19E+09	5.15E+08
		St. Dev	9.10E+04	1.06E+05	1.59E+05	2.38E+04

Although I intended to include the convergence curves for each algorithm while solving the test functions with dimensions set at n=50, I encountered technical difficulties that prevented me from plotting them. My system was experiencing significant performance issues, causing it to hang throughout the process. I plan to address these errors and continue working on the graphs to present the convergence curves in future updates.

In conclusion, the Reinforced ACO algorithm proves to be highly effective, consistently yielding high-quality solutions across different problem types, whether tackling low-dimensional simple functions or complex high-dimensional mixed functions.

Table 8: Numerical results of traffic routing methods in the Kamppi traffic network scenario

Method	Traffic demand (veh/h)		
	500	1000	1500
ACO	119.48 ± 3.49	147.08 ± 3.66	170.40 ± 3.34
Q-Learning	117.32 ± 1.30	136.61 ± 1.06	146.24 ± 3.56
QACO	115.97 ± 2.74	130.19 ± 1.12	137.74 ± 2.22

5 Conclusion

This project explores innovative heuristic techniques to tackle constraint optimization problems, specifically focusing on the development of a Reinforced Ant Colony Optimization algorithm. Although enhanced Ant Colony Optimization (ACO) algorithms for continuous-domain optimization have become popular recently, they often struggle to adapt to changes in the environment, primarily depending on pheromone trails to guide their evolution. To address this challenge, we introduce Reinforced Ant Colony Optimization algorithm, which improves adaptability by incorporating reinforcement learning principles to evaluate environmental diversity and enhance decision-making.

By analyzing the diversity within the ant colony, Reinforced Ant Colony Optimization algorithm builds a reinforcement learning model that allows the algorithm to choose optimal strategies at different evolutionary stages, maximizing rewards and improving its ability to search globally and converge quickly. We thoroughly tested the performance of Reinforced Ant Colony Optimization algorithm using the IEEE-CEC 2017 benchmark suite, which features 28 complex functions that present a variety of optimization challenges.

The results show that Reinforced Ant Colony Optimization algorithm significantly outperforms traditional ACO methods, demonstrating better convergence rates, accuracy, and global search capability across different optimization scenarios. This research not only showcases the potential of advanced heuristic techniques for solving complex constraint optimization problems but also establishes a solid foundation for future research. For instance, there are promising avenues to further refine and hybridize the Genetic Algorithm (GA) and ACO methods, which could enhance their efficiency and robustness across a broader range of challenges. These heuristic techniques also have practical applications in fields such as logistics, finance, and engineering, where effective constraint optimization is essential. Lastly, future work could focus on developing more sophisticated benchmark functions that reflect the complexities of real-world applications, offering a more rigorous testing environment for these heuristic algorithms.

5.1 Limitations faced for the Proposed method

While reinforcement learning is gaining traction among researchers, there are still significant challenges in its practical application. Considering these obstacles and the inherent limitations of reinforcement learning, we aim to explore potential future directions for the advancement of deep reinforcement learning [**AlMahamid and Grolinger**, **2021**].

During strategy implementation, the agent must balance between thoroughly utilizing the strategies it has learned and avoiding random experimentation. Simultaneously, it must explore a wide range of new strategies to uncover the most effective solution. The current approach uses a stochastic strategy, allowing the agent to explore various states and avoid getting stuck in a local optimum. However, this randomness can cause the agent to waste time and computational resources by repeatedly searching through irrelevant states. Thus, finding a balance between exploration and exploitation could be a crucial focus for future research [**AlMahamid and Grolinger**, **2021**].

Evaluating the effectiveness of a strategy is crucial for enhancing an agent's learning efficiency. While the reward-value function is commonly used for this purpose, it often falls short in complex decision-making tasks, where designing an effective reward function is challenging. [**Arulkumaran et al.**, **2017**]. Additionally, in real-world scenarios, not every action receives clear feedback, leading to sparse reward issues, especially in high-dimensional state spaces. Addressing these challenges will require developing better reward functions tailored to specific tasks. Furthermore, most deep reinforcement learning research currently focuses on model-free methods, which demand extensive training samples—impractical in real-world applications [**Hou et al.**, **2017**]. Model-based approaches, which can reduce the need for excessive interactions between agents and environments, are likely to gain more attention as a solution to improve sampling efficiency.

Code accessibility

For comprehensive access to the complete codebase, including all analyses, please refer to the GitHub repository. Additionally, I have included my final report in the same repository for archival purposes.

`https://github.com/jerrinjxavier/Research_Project_B`

6 References

1. Teklu, F., Sumalee, A. and Watling, D., 2007. A genetic algorithm approach for optimizing traffic control signals considering routing. Computer-Aided Civil and Infrastructure Engineering, 22(1), pp.31-43..

2. Center, M., 2006. Traffic network study tool–TRANSYT-7F. United States Version. University of Florida.

3. Hunt, P.B., Robertson, D.I., Bretherton, R.D. and Royle, M.C., 1982. The SCOOT on-line traffic signal optimisation technique. Traffic Engineering and Control, 23(4).

4. Lowrie, P.R., 1982. The Sydney cooridinated adaptive traffic (SCAT) system-principles, methodology, algorithm. In Proc. of the Second International Conference on Road Traffic Signaling, IEE (pp. 67-70).

5. Wei, W., Zhang, Y., Mbede, J.B., Zhang, Z. and Song, J., 2001, October. Traffic signal control using fuzzy logic and MOGA. In 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236) (Vol. 2, pp. 1335-1340). IEEE.

6. Yu, X.H., 2008. Decentralized adaptive signal control for traffic networks. New York: Nova Science Publishers.

7. Dorigo, M., 2007. Ant colony optimization. Scholarpedia, 2(3), p.1461.

8. Dorigo, M. and Blum, C., 2005. Ant colony optimization theory: A survey. Theoretical computer science, 344(2-3), pp.243-278

9. Angus, D., 2006. Ant colony optimisation: From biological inspiration to an algorithmic framework. Technical Report at Swinburne University of Technology.

10. Dorigo, M. and Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. biosystems, 43(2), pp.73-81.

11. Tashkova, K., Korošec, P. and Šilc, J., 2011. A distributed multilevel ant-colony algorithm for the multi-way graph partitioning. International Journal of Bio-Inspired Computation, 3(5), pp.286-296.

12. Nguyen, N.G., Le, D.N. and Le, N.D., 2013. A novel ant colony optimization-based algorithm for the optimal communication spanning tree problem. International Journal of Computer Theory and Engineering, 5(3), p.509.

13. Toklu, N.E., Montemanni, R. and Gambardella, L.M., 2013, October. A robust multiple ant colony system for the capacitated vehicle routing problem. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (pp. 1871-1876).

14. He, J. and Hou, Z., 2012. Ant colony algorithm for traffic signal timing optimization. Advances in Engineering Software, 43(1), pp.14-18.

15. Renfrew, D. and Yu, X.H., 2012, June. Traffic signal optimization using ant colony algorithm. In The 2012 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

16. Li, J., Dridi, M. and El-Moudni, A., 2016. Cooperative traffic control based on the artificial bee colony. J. Eng. Res. Appl, 6(12), pp.46-55.

17. Haldenbilen, S., Baskan, O. and Ozan, C., 2013. An ant colony optimization algorithm for area traffic control. Ant colony optimization–techniques and applications, pp.87-105.

18. Claes, R. and Holvoet, T., 2012, March. Cooperative ant colony optimization in traffic route calculations. In Advances on Practical Applications of Agents and Multi-Agent Systems: 10th International Conference on Practical Applications of Agents and Multi-Agent Systems (pp. 23-34). Berlin, Heidelberg: Springer Berlin Heidelberg.

19. Dorigo, M., 2007. Ant colony optimization. Scholarpedia, 2(3), p.1461

20. Angus, D., 2006. Ant colony optimisation: From biological inspiration to an algorithmic framework. Technical Report at Swinburne University of Technology.

21. Renfrew, D., 2009. Traffic signal control with ant colony optimization (Master's thesis, California Polytechnic State University).

22. Atabati, M., Zarei, K. and Borhani, A., 2016. Ant colony optimization as a descriptor selection in QSPR modeling: estimation of the $\lambda_{\max}$ of anthraquinones-based dyes. Journal of Saudi Chemical Society, 20, pp.S547-S551.

23. Jang, B., Kim, M., Harerimana, G. and Kim, J.W., 2019. Q-learning algorithms: A comprehensive classification and applications. IEEE access, 7, pp.133653-133667.

24. Li, N., Gao, B., Xie, Z., Zhang, F. and Wan, J., 2021, May. Q-learning Based Intelligent Ant Colony Scheduling Algorithm in Heterogeneous System. In 2021 IEEE 4th International Conference on Electronics Technology (ICET) (pp. 1020-1025). IEEE.

25. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P. and Wießner, E., 2018, November. Microscopic traffic simulation using sumo. In 2018 21st international conference on intelligent transportation systems (ITSC) (pp. 2575-2582). IEEE.