ME 308 Report

Operations Research Project

Traffic Lights Optimization

Sashreek Paul	*Akhilesh Narayan*	*Bhaskar Pegu*	*Shreejesh Barde*
20D100024	*200100016*	*20D100006*	*200100146*

Introduction

Traffic congestion is a significant issue in urban areas, leading to increased travel times, fuel consumption, and environmental pollution. This paper presents a novel optimization framework for traffic signal control to mitigate congestion and enhance mobility and safety at high-volume intersections. The framework leverages optimization techniques and algorithms to determine optimal signal timings for each lane, dynamically adjusting them based on real-time traffic data.

The framework's evaluation uses SUMO, a traffic simulation software, to assess its performance in various traffic conditions and scenarios. The results demonstrate the effectiveness of the proposed framework in reducing congestion and enhancing traffic flow efficiency.

Problem Statement

The proposed project aims to develop an optimization framework for traffic signal control using optimization techniques and algorithms. These algorithms will be used to determine the optimal signal timings for each lane at high-volume intersections, to reduce traffic congestion and improve mobility and safety. The proposed framework aims to improve traffic flow efficiency and reduce congestion by dynamically adjusting signal timings based on real-time traffic data. The framework will incorporate optimization algorithms, such as Max Pressure and Self-Organizing Traffic Lights, with constant and varying phase durations to minimize the average waiting time of vehicles traversing the network.

$$\min\left(\frac{\sum_{i=1}^{n} t_i}{n}\right)$$

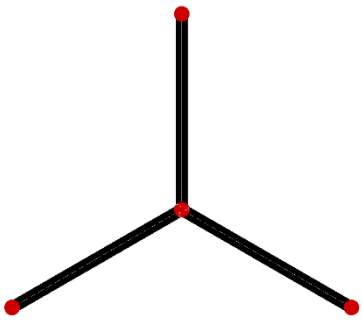
Here t_i represents the time waited by car i while traversing the network, and n is the total number of cars that have traversed the network.

The proposed framework will be evaluated using a traffic simulation software called SUMO (Simulation of Urban Mobility) to assess its performance in various traffic conditions and scenarios.

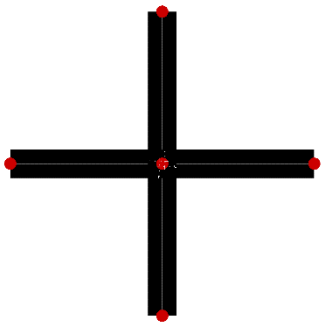
In conclusion, the proposed optimization framework uses real-time traffic data and optimization algorithms to adjust signal timings dynamically. The proposed algorithms can improve traffic flow efficiency and reduce congestion in a given network.

Road Networks

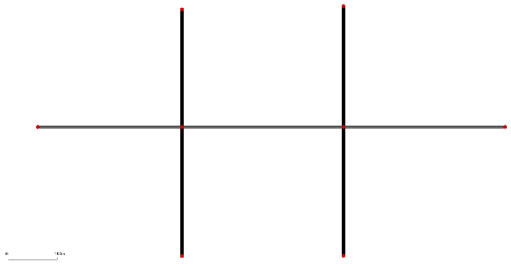
1. 3 roads junction (single lane roads):



2. 4-Roads Junction (triple lane roads):



3. 2 Junctions (single lane roads):



Algorithms

Fixed Time

In traffic engineering, controlling traffic signal timings is crucial to ensure efficient and safe vehicular and pedestrian traffic flow. One widely used approach is the fixed time algorithm, which involves changing the traffic signal lights at set intervals in a fixed order regardless of the traffic conditions.

The fixed time algorithm is a traffic signal control method in which the duration of each signal phase remains fixed and does not depend on the traffic demand at the intersection. This algorithm assumes that the traffic demand at the junction is predictable and does not change with time.

In the case of a three-way intersection, the fixed time algorithm can be implemented by dividing the intersection into three incoming lanes and assigning specified green time intervals to each lane. For example, in a scenario where each lane has a fixed green time of 20 seconds, the traffic signals will alternate between green lights for each lane in a pre-determined sequence for 20 seconds each. After the green time interval completes for an incoming lane, the traffic signals will switch to the next incoming lane, and the cycle will continue.

Although the fixed time algorithm has limitations, it is still a widely used traffic signal control strategy due to its simplicity, low cost, and ease of implementation.

Max Pressure

The Max Pressure algorithm with varying phase duration is a traffic signal control strategy that aims to maximize the throughput of a traffic intersection by dynamically adjusting the signal timings based on real-time traffic conditions.

Constant Phase

In this algorithm, the lane with the highest traffic volume is identified and given priority over other roads, and the signal phase for this lane is kept green for a constant duration of time, known as the constant phase.

In the case of a three-way intersection, the Max Pressure algorithm with constant phase can be implemented by dividing the intersection into three incoming lanes and continuously monitoring the traffic volume in each lane. The lane with the highest traffic volume is identified, and the signal phase for that lane is made green for a constant duration, like 30 seconds. After the constant phase is over, the algorithm re-evaluates the traffic volume in each lane and identifies the new lane with the highest traffic volume. The signal phase for this new lane is then made green for the next constant phase, and the cycle continues.

Compared to the fixed time algorithm, the Max Pressure algorithm with constant phase can adapt to changing traffic conditions and prioritize the lanes with the highest traffic volume, improving traffic flow efficiency. However, the algorithm requires real-time traffic data, which can be challenging to obtain in certain situations. The constant phase duration must also be carefully adjusted to balance the need for efficient traffic

flow and the risk of causing congestion in other lanes.

In conclusion, the Max Pressure algorithm with constant phase is a promising traffic signal control strategy to improve traffic flow efficiency in high-volume intersections. However, the algorithm's limitations and challenges should be carefully considered, and further research is needed to assess its performance in different traffic conditions and scenarios.

Varying Phase

Like the constant phase duration algorithm, the signal for the highest traffic volume lane is made green in this algorithm for a duration directly proportional to the number of waiting vehicles in that lane.

First, a parameter T is defined for the algorithm. After every phase duration, the number of vehicles waiting in each lane (waiting_vehicles_number) is recorded. These numbers are further normalized using the total number of cars waiting in the network (Total_vehicles) at that instant. Thus,

$$\text{traffic\_value} = \text{normalized\_waiting\_vehicles} = \frac{\text{waiting\_vehicles\_number}}{\text{Total\_vehicles}}$$

The traffic signal for the lane with the highest traffic_value (highest_traffic_value) is opened, and the next phase duration is simultaneously updated as

$$\text{phase\_duration} = T \times \frac{\text{highest\_traffic\_value}}{\text{total\_traffic\_value}}$$

In contrast to the constant phase Max Pressure algorithm, which selects the highest traffic volume lane and opens that for a fixed duration, the varying phase algorithm also updates the duration of the green signal depending on the congestion. Thus, the varying phase algorithm is more adaptable than the constant phase algorithm, but the constant phase algorithm is much simpler to implement.

Self-Organizing Traffic Lights

"Self-Organizing Traffic Lights" is a larger class of adaptive traffic control algorithms to which the Max Pressure algorithms belong. It provides much greater flexibility and many more parameters that can be varied to obtain the minimum average waiting time of a car traversing the network.

While implementing this algorithm in our project, we considered two different parameters: the number of cars waiting in each lane and the total time waited by cars in each lane.

Constant Phase

In the Constant-Phase Self-Organizing Traffic Light algorithm, the durations of the green and red phases remain constant. The algorithm dynamically adjusts the sequence of the signals based on the traffic flow.

Like the Max Pressure algorithm, a parameter T is first defined here. After every fixed phase duration, the number of vehicles waiting in each lane (waiting_vehicles_number) and the total time waited by cars in each lane (waiting_time_lane) are recorded. These numbers are normalized, respectively, using the total number of cars waiting in the network (Total_vehicles) and the total time waited by the cars in the network (Total_waiting_time) at that instant.

For each lane, we compute

$$\text{normalized\_waiting\_vehicles} = \frac{\text{waiting\_vehicles\_number}}{\text{Total\_vehicles}}$$

$$\text{normalized\_waiting\_time} = \frac{\text{waiting\_time\_lane}}{\text{Total\_waiting\_time}}$$

Next, for each lane, a parameter traffic_value is computed as

$$(\text{normalized\_waiting\_time})^{1-\frac{i}{\text{Total\_i}}} \times (\text{normalized\_waiting\_vehicles})^{\frac{i}{\text{Total\_i}}}$$

Here, i is varied from 0 to Total_i, and Total_i = 20 for 3 roads junction and 4 roads junction and Total_i = 10 for 2 junctions.

The lane with the highest traffic_value (highest_traffic_value) is opened.

It can be seen that setting i = Total_i in the above expression converts SOTL to MP.

Varying Phase

In the varying phase algorithm, traffic light signals adapt to changes in traffic patterns by adjusting the durations of the green and red phases based on real-time traffic data. The algorithm uses a decentralized approach, meaning each traffic light controller operates independently but communicates with other controllers to coordinate the traffic flow.

Like the Constant Phase algorithm, traffic_value (for all lanes) and highest_traffic_value are computed.

The lane corresponding to highest_traffic_value is opened, and the next phase duration is simultaneously updated as

$$\text{phase\_duration} = T \times \frac{\text{highest\_traffic\_value}}{\text{total\_traffic\_value}}$$

In the expression of traffic_value, to get the optimum exponents of normalized_waiting_time and normalized_waiting_vehicles, we performed multiple simulations by varying the exponents. The exponents which resulted in the minimum weighting time were the optimum exponents.

Genetic Algorithm

In this new implementation, there are always two nested simulations taking place:

1. Main GUI-based simulation (label = "simulation_1")
2. Auxiliary non-GUI-based simulation (label = "simulation_2")

A list (traffic_states of size population_size) of randomized traffic_state strings is created at the end of every STEPS steps of simulation_1. Each traffic state string is run in an auxiliary simulation for steps steps starting from the last network state in simulation_1. Another list (best_traffic_states) is created by selecting the best half from traffic_states.

In a generation of traffic states, two parent states are chosen randomly from best_traffic_states and crossed, and the resultant states are further mutated. This is repeated until the number of member states in that generation becomes at least population_size. This complete process is repeated for generations generations where generation generation serves as the parent generation of generation generation + 1.

The optimum traffic state (optimum_traffic_state) is selected as the state from the last generation of traffic states with the minimum fitness score. This optimum_traffic_state is assigned to the main simulation junction, and the simulation is run for STEPS steps from the last simulation step.

This implementation will also indirectly control the phase duration of each traffic light phase.

So, after every STEPS main simulation steps, steps auxiliary simulation steps are performed (starting from STEPS steps) to determine the optimum traffic state for the next STEPS main simulation steps.

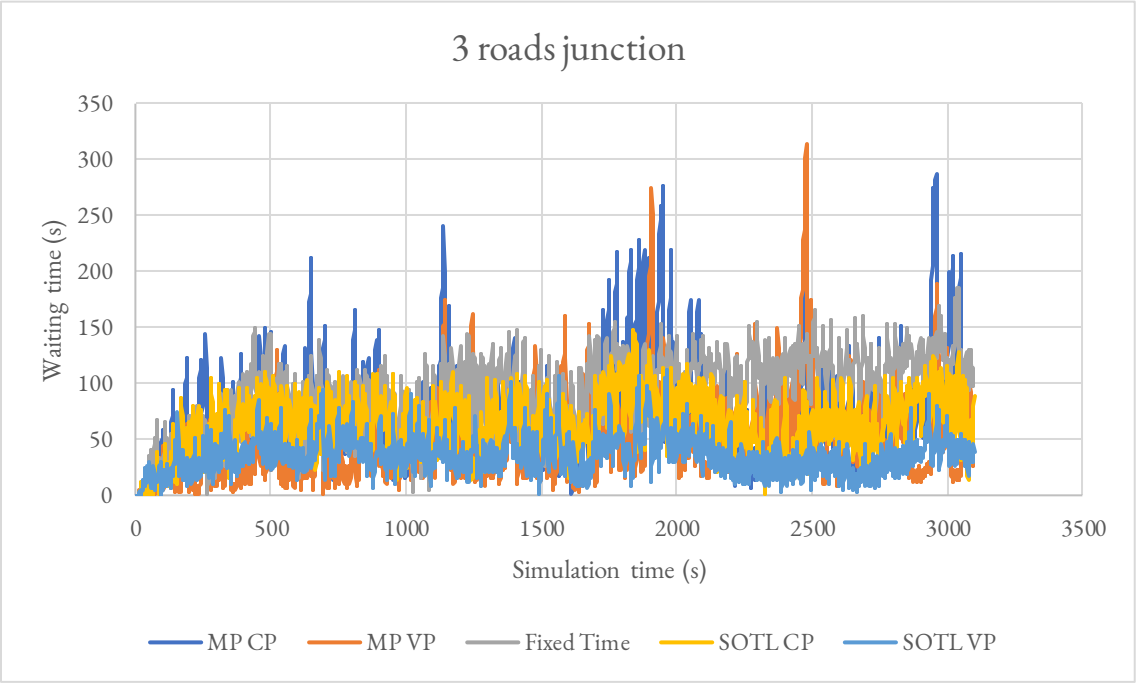
Results

We simulated the Fixed Time, MP CP*, and SOTL CP** algorithms for a constant phase duration of 10 seconds, and the value of T for the MP VP*** and SOTL VP**** algorithms was 10 seconds. The simulation duration was 3100 seconds.

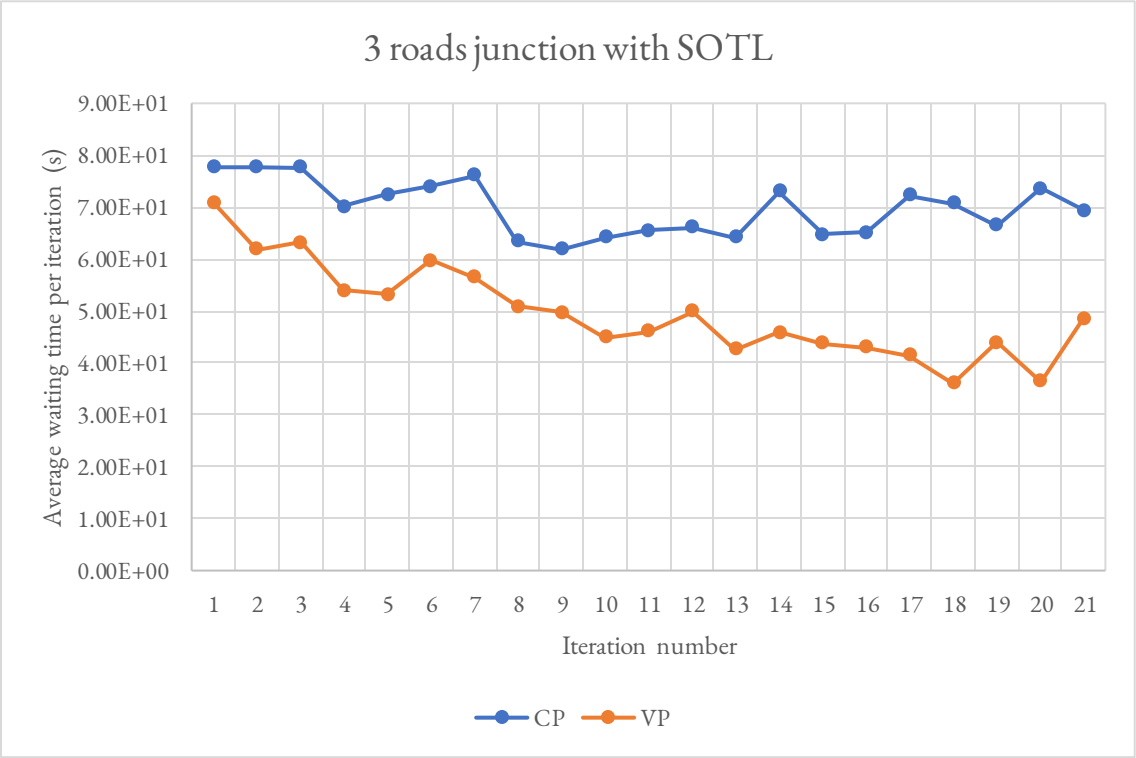
* ** *** ****

Max Pressure Constant Phase | *Self-Organizing Traffic Lights Constant Phase* | *Max Pressure Varying Phase* | *Self-Organizing Traffic Lights Varying Phase*

1. 3 roads junction with various algorithms applied:

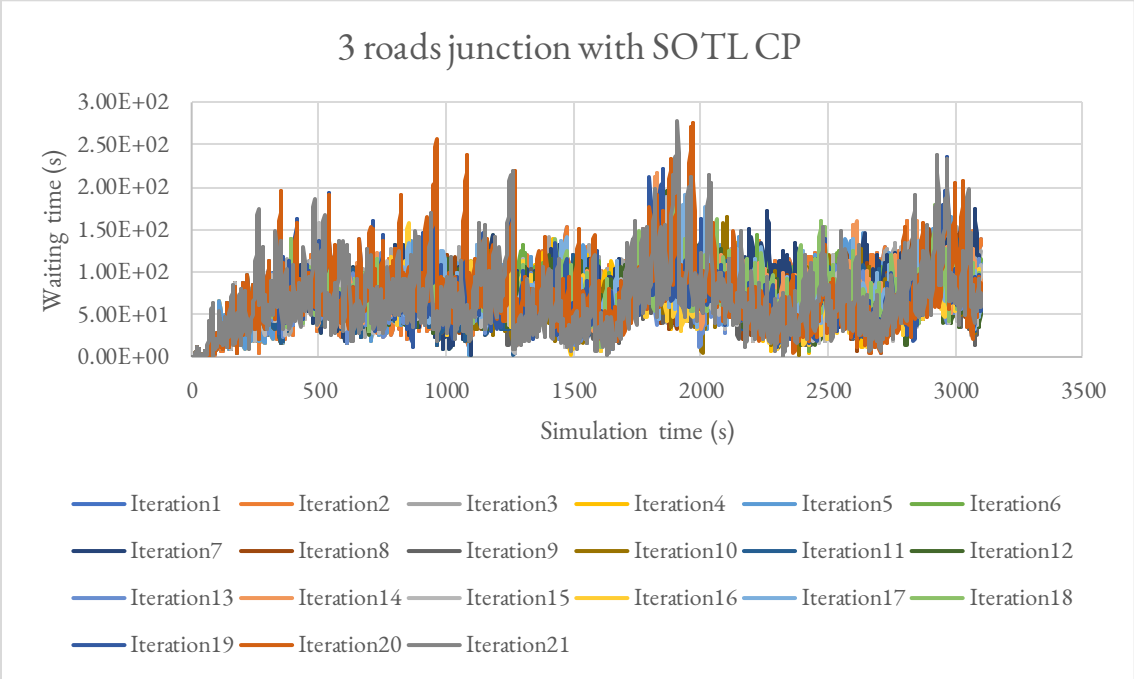


2. 3 roads junction SOTL CP and SOTL VP comparison (with varying i in $traffic\_value$):



Note: Iteration number k corresponds to $i = k - 1$ in the expression of $traffic\_value$.

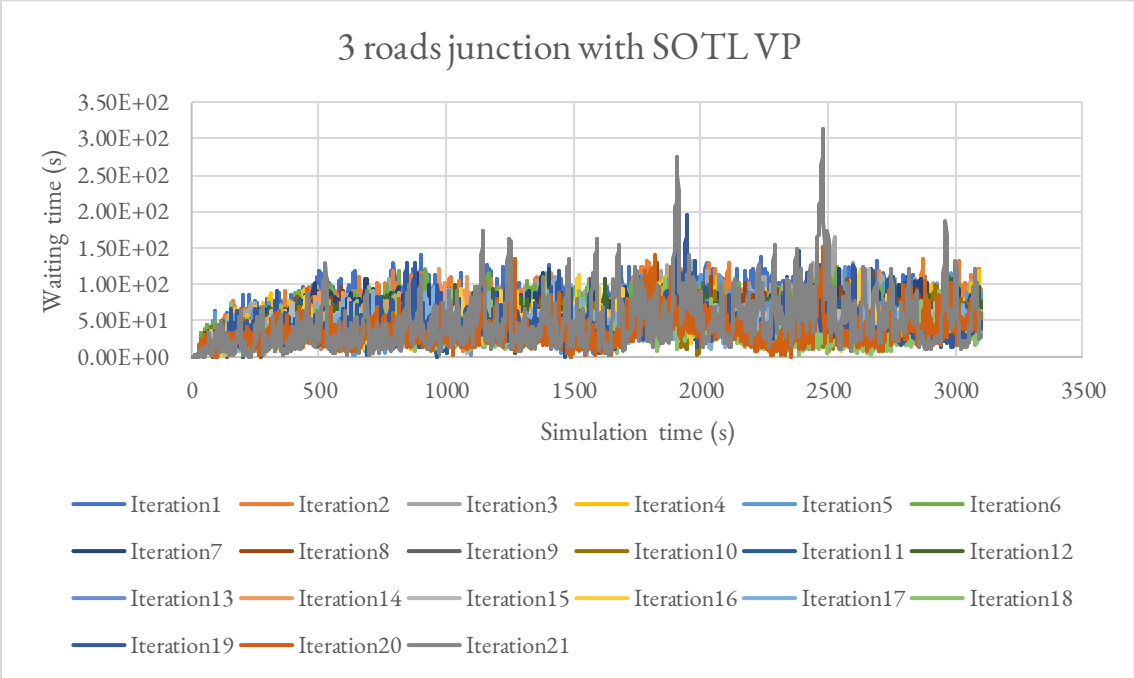
3. SOTL CP applied to 3 roads junction (with varying i in $traffic\_value$):



Optimum traffic_value expression:

$$(\text{normalized\_waiting\_time})^{0.6} \times (\text{normalized\_waiting\_vehicles})^{0.4}$$

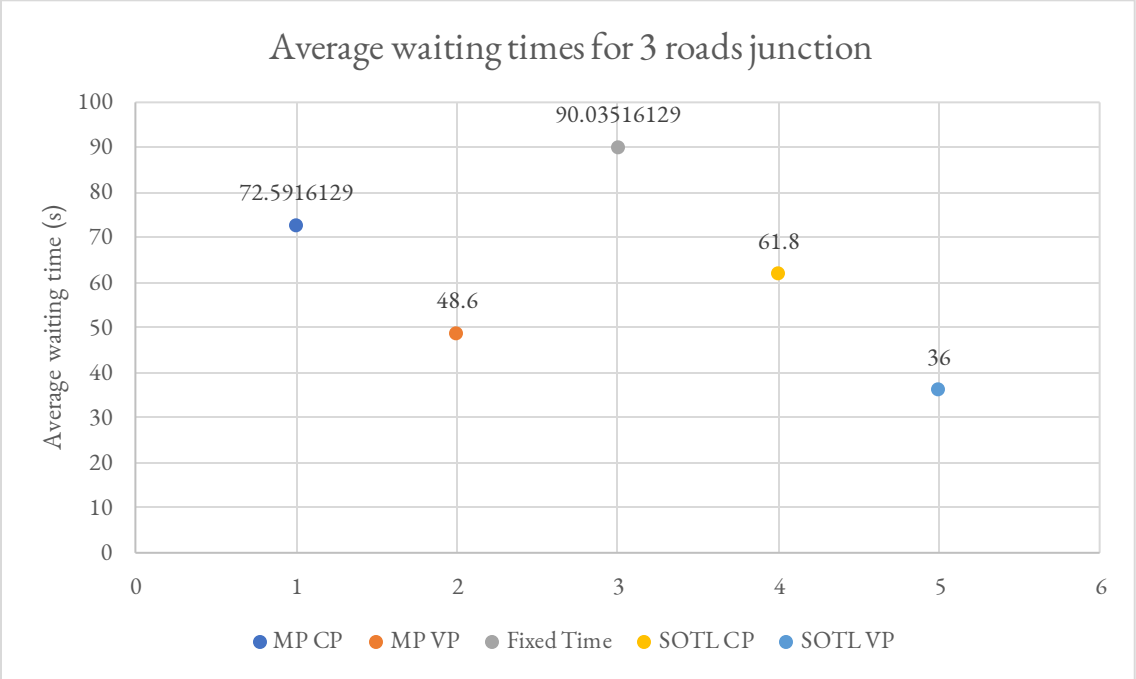
4. SOTL VP applied to 3 roads junction (with varying i in $traffic\_value$):



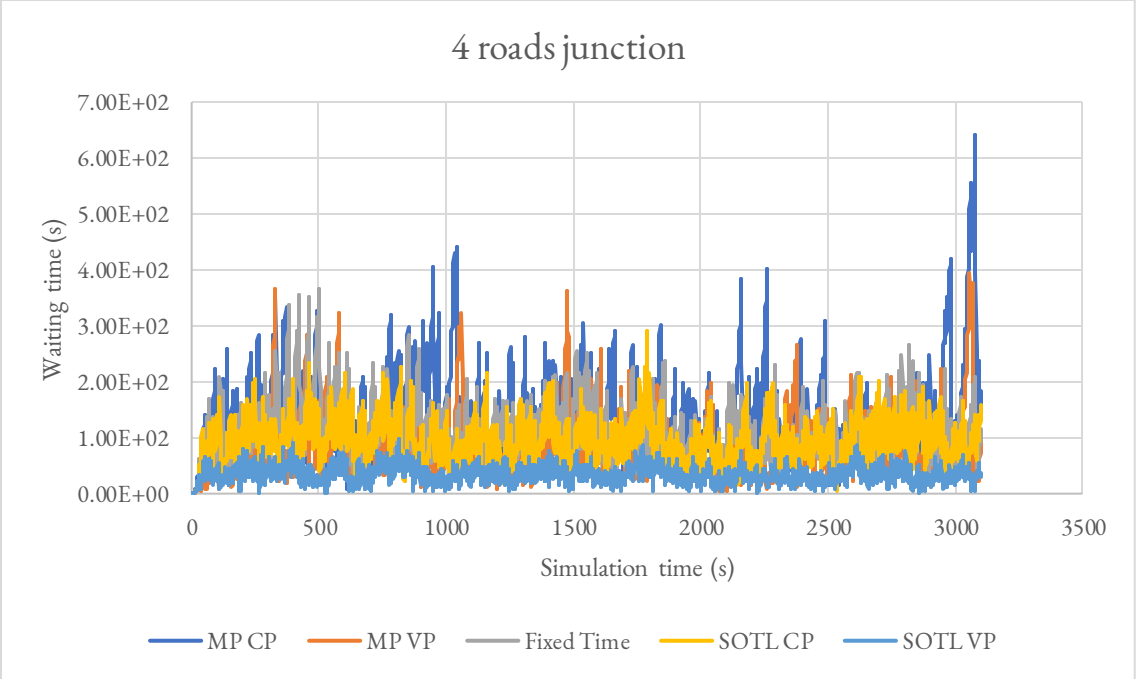
Optimum traffic_value expression:

$$(\text{normalized\_waiting\_time})^{0.15} \times (\text{normalized\_waiting\_vehicles})^{0.85}$$

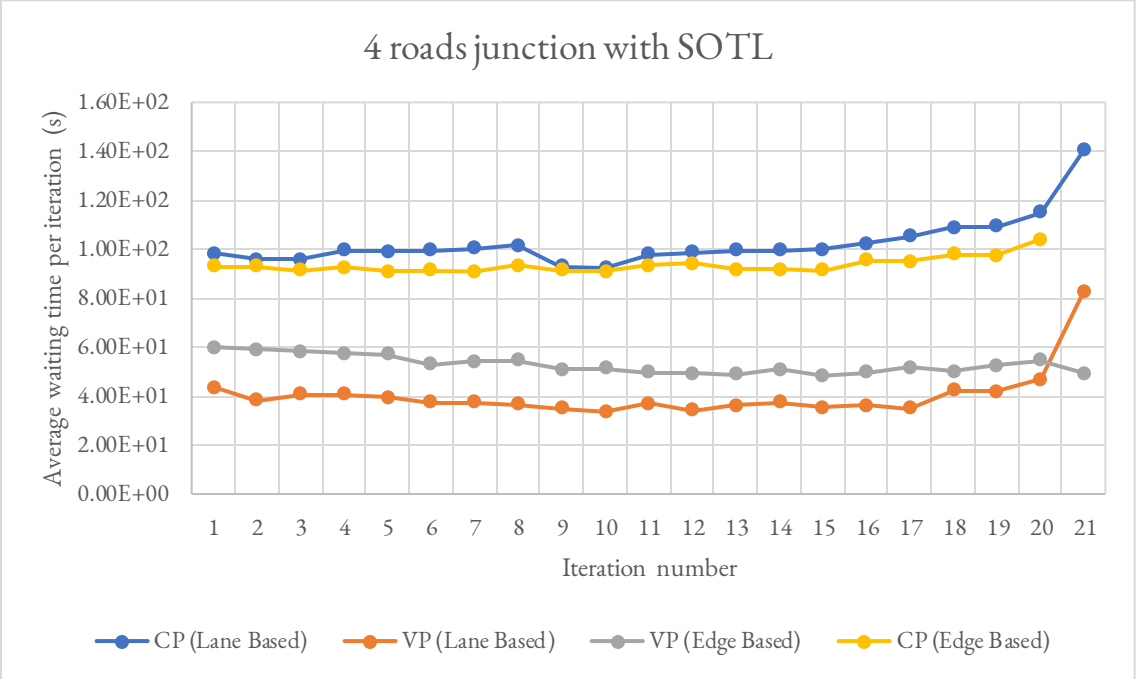
5. Performance of the various algorithms in 3 roads junction:



6. 4 roads junction with various algorithms applied:

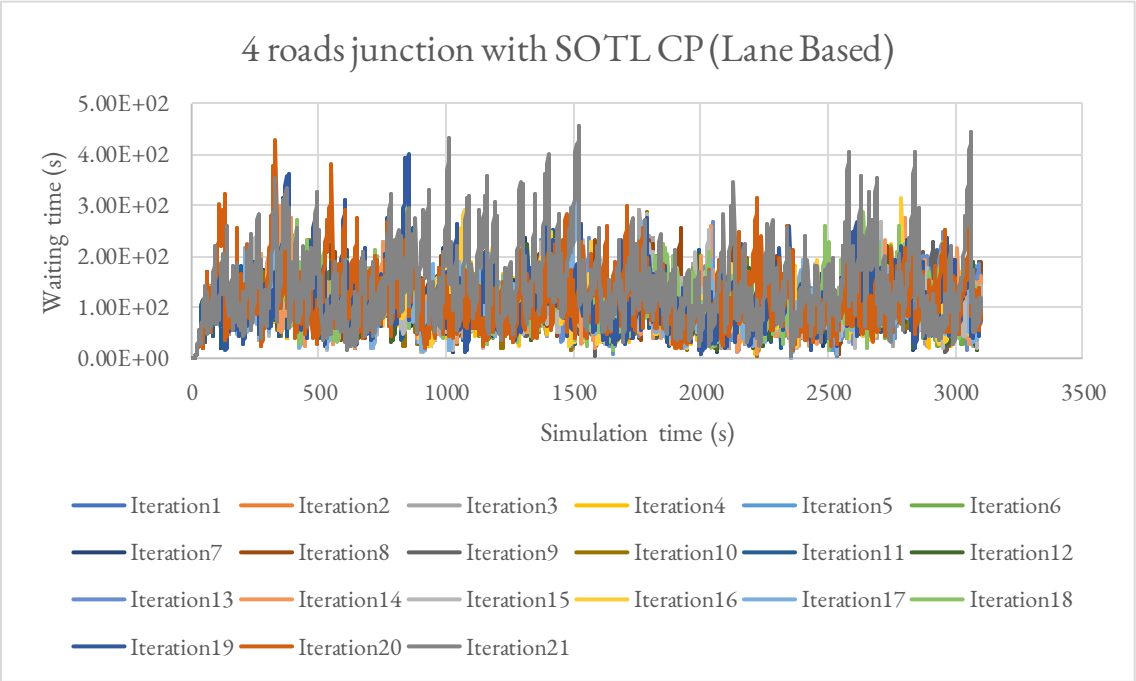


7. 4 roads junction with SOTL CP and SOTL VP applied (with varying i in $traffic\_value$):



In Lane Based, the entire edge containing the highest traffic value lane (out of 12 lanes) is opened. In this, traffic values are computed for each lane separately.
In Edge Based, the entire edge with the highest traffic value as a whole is opened. In this, traffic values are computed for entire edges without considering constituent lanes.

8. SOTL CP applied to 4 roads junction (with varying i in $traffic\_value$):



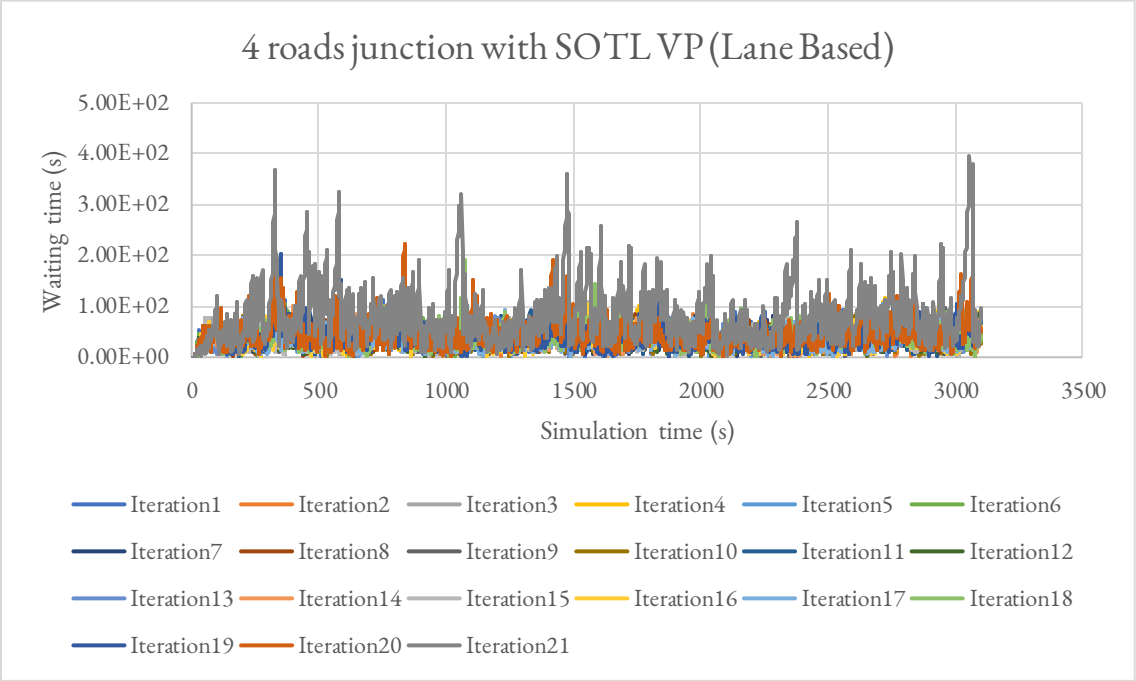
Optimum traffic_value expression (Edge Based):

$$(\text{normalized\_waiting\_time})^{0.7} \times (\text{normalized\_waiting\_vehicles})^{0.3}$$

Optimum traffic_value expression (Lane Based):

$$(\text{normalized\_waiting\_time})^{0.55} \times (\text{normalized\_waiting\_vehicles})^{0.45}$$

9. SOTL VP applied to 4 roads junction (with varying i in $traffic\_value$):



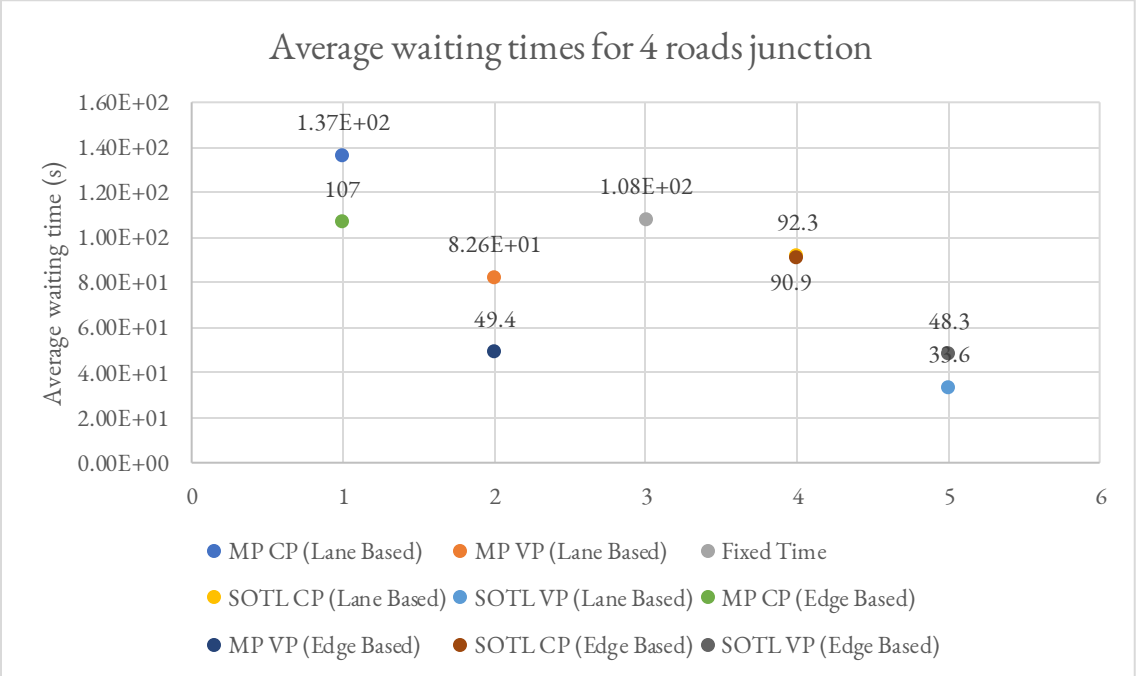
Optimum traffic_value expression (Edge Based):

$$(\text{normalized\_waiting\_time})^{0.4} \times (\text{normalized\_waiting\_vehicles})^{0.6}$$

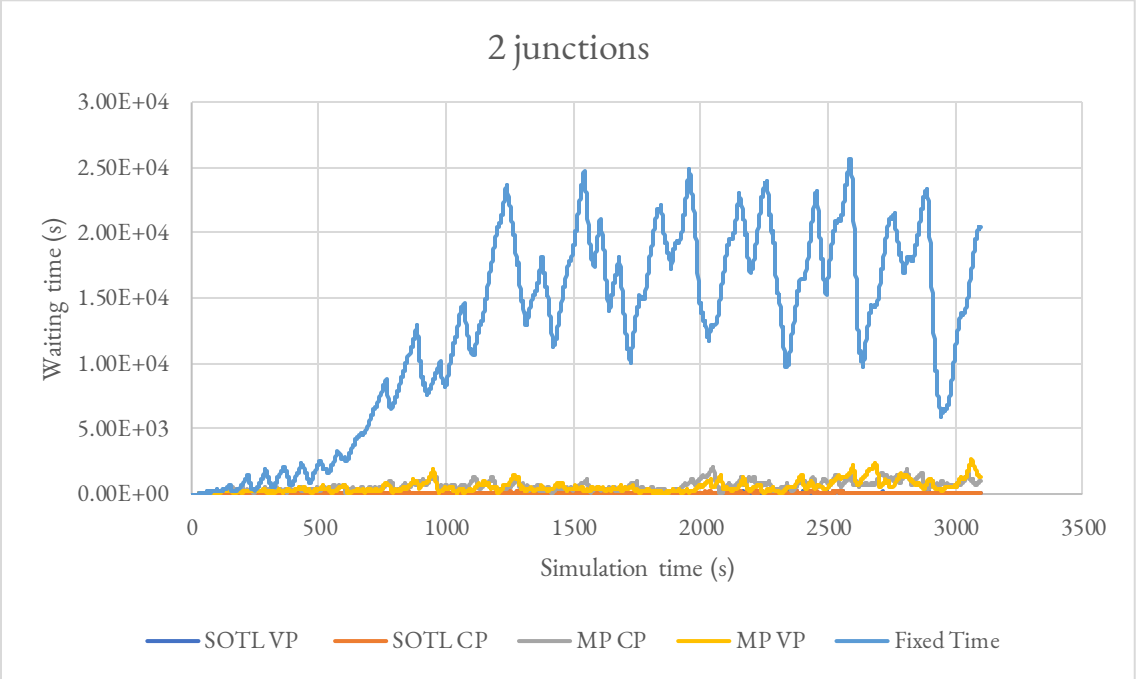
Optimum traffic_value expression (Lane Based):

$$(\text{normalized\_waiting\_time})^{0.55} \times (\text{normalized\_waiting\_vehicles})^{0.45}$$

10. Performance of the various algorithms in 4 roads junction:



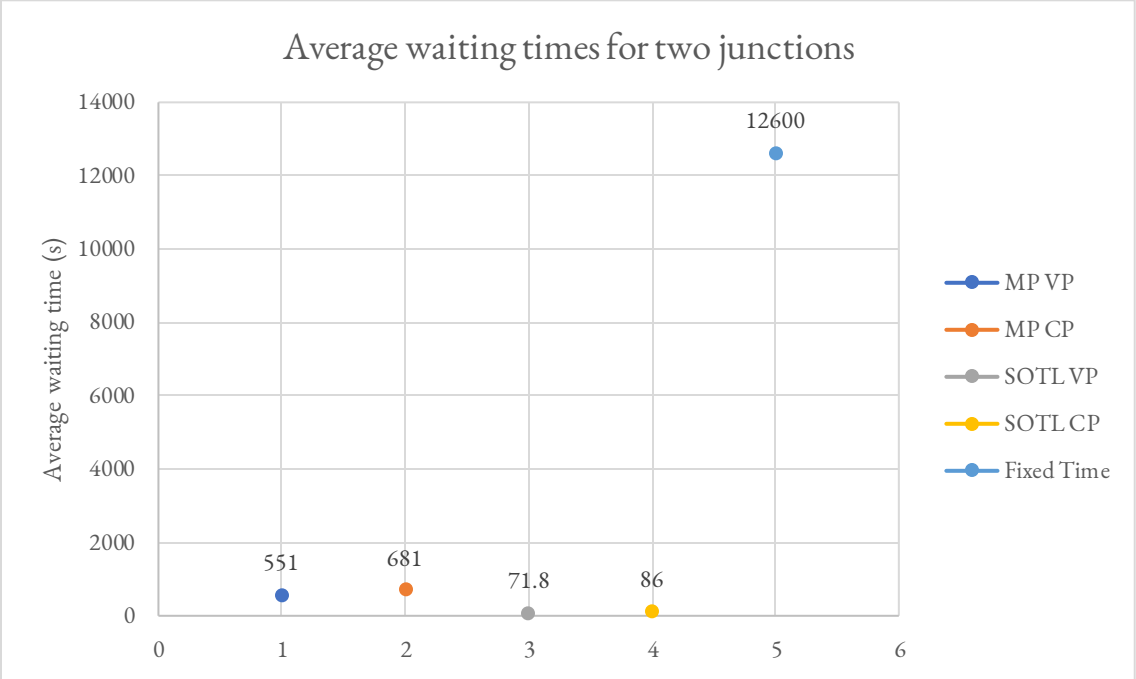
11. 2 junctions with various algorithms applied:



12. SOTL CP applied to 2 junctions (with varying i in $traffic\_value$):
    Optimum traffic_value expression for left junction:
    $$(\text{normalized\_waiting\_time})^{0.9} \times (\text{normalized\_waiting\_vehicles})^{0.1}$$
    Optimum traffic_value expression for right junction:
    $$(\text{normalized\_waiting\_time})^{0.8} \times (\text{normalized\_waiting\_vehicles})^{0.2}$$

13. SOTL VP applied to 2 junctions (with varying i in $traffic\_value$):
    Optimum traffic_value expression for left junction:
    $$(\text{normalized\_waiting\_time})^{0.5} \times (\text{normalized\_waiting\_vehicles})^{0.5}$$
    Optimum traffic_value expression for right junction:
    $$(\text{normalized\_waiting\_time})^{0.3} \times (\text{normalized\_waiting\_vehicles})^{0.7}$$

14. Performance of various algorithms in 2 junctions:



15. Table of optimum exponents in traffic_value expression in SOTL:

	(normalized_waiting_time)				(normalized_waiting_vehicles)			
	3 roads	4 roads	2 junctions		3 roads	4 roads	2 junctions	
			Left junction	Right junction			Left junction	Right junction
SOTL CP (Edge)	0.6	0.7	0.9	0.8	0.4	0.3	0.1	0.2
SOTL CP (Lane)		0.55				0.45		
SOTL VP (Edge)	0.15	0.4	0.5	0.3	0.85	0.6	0.5	0.7
SOTL VP (Lane)		0.55				0.45		

16. Table of average waiting time (s) for every algorithm and every road network:

	FT	MP CP (Lane Based)	MP CP (Edge Based)	MP VP (Lane Based)	MP VP (Edge Based)	SOTL CP (Lane Based)	SOTL CP (Edge Based)	SOTL VP (Lane Based)	SOTL VP (Edge Based)
3 roads 1 junction	90.04	72.59		48.6		61.8		36	
4 roads 1 junction	108	137	107	82.6	49.4	92.3	90.9	33.6	48.3

2 junctions	12600	681	551	86	71.8

Conclusions:

1. The simulations show that the Max Pressure algorithms are highly volatile compared to the SOTL algorithms.
2. The SOTL algorithms offer greater flexibility as they provide weight to both the number of waiting vehicles and waiting time, unlike the MP algorithms, which just give importance to the number of waiting vehicles.
3. The VP algorithms result in lower traffic congestion than the CP algorithms.
4. The SOTL algorithms result in lower traffic congestion than the MP algorithms.
5. The SOTL and MP algorithms are more profound in multi-junction networks than in single-junction road networks.
6. The SOTL and MP algorithms require real-time traffic data through sophisticated sensors or cameras installed on the roads because these algorithms are based on feedback. This may account for additional costs. On the other hand, GA does not require much-sophisticated sensors as it is an ***exploratory*** algorithm. However, applying GA to obtain the most optimum traffic state in real time requires immense computational power. Also, 6 independent parameters must be determined first for the most efficient functioning of GA.
7. In short, the long-term costs associated with SOTL and MP are higher, but the long-term costs associated with GA are much lesser.

References

1. SUMO Python Documentation
2. SUMO User Documentation

Contributions

1. Sashreek Paul:
2. Akhilesh Narayan:
3. Bhaskar Pegu:
4. Shreejesh Barde: