Efficient Road Traffic Control Using CPU Scheduling Algorithms

Shiv Kampani

Dhirubhai Ambani International School, Mumbai-400098, India

DOI: 10.29322/IJSRP.11.06.2021.p11483
http://dx.doi.org/10.29322/IJSRP.11.06.2021.p11483

Abstract- The aim of this study was to find an efficient traffic control algorithm that reduces the total waiting time (TWT) of all vehicles in a traffic jam at a two-lane intersection. This paper draws from time-shared computer systems and the scheduling theory to determine the optimum algorithm for scheduling traffic. In this study, four CPU Scheduling algorithms were evaluated against a set of 100 randomly-generated traffic patterns in a simulated environment to determine their relative levels of efficiency in reducing the TWTs of vehicles in a traffic jam and identify the most efficient one. The study showed significant differences in the performance of each algorithm, with the Shortest Job First Context Switch and the Shortest Job First algorithms being the most efficient of the four algorithms evaluated.

This paper also includes recommendations for an efficient traffic control system which uses cameras placed at intersections and the CPU Scheduling Algorithms described in other sections of this paper to efficiently control traffic. The system can be implemented with minimal modifications to existing infrastructure, and would reduce traffic delays at congested intersections. This would provide social, environmental and economic benefits, some of which were discussed in this paper.

Index Terms- CPU Scheduling, traffic congestion, algorithms

I. INTRODUCTION

The TomTom 2019 Traffic Study that surveyed 403 cities across 56 countries identified traffic congestion as a global problem. In cities like Moscow, Delhi, Bogota, Lima and Istanbul, the level of traffic congestion is higher than 50% [1].

The traffic congestion issue has become particularly acute in the city of Mumbai. With the city's rapidly-growing population, the number of registered vehicles doubled from 1.50 million in 2007 to 3.05 million in 2017 [2], with almost 685 new vehicles registered everyday [3]. It is little wonder that Mumbai was ranked the most congested city in the world in 2019, with a congestion level of 65% [4]. In some areas, the average speed of vehicles has decreased from 40 km/h to 20 km/h [5], thus leading to an increase in the number of traffic jams at two-way intersections. The waiting queues of vehicles not only cause huge delays for businesses and potentially harm them, but also contribute to the city's pollution problem.

The endeavor to manage Mumbai's traffic congestion problem has come at a high cost. The government is in the midst of constructing a 30-km long, 8-lane coastal road — a 5-year-long project that is costing a total of USD1.7 billion [6]. While this solution may reduce Mumbai's traffic congestion problem, it is not a feasible solution for other cities due to the high costs and time involved, not to mention its short-term effectiveness.

A far more cost-efficient fashion of dealing with traffic congestion could simply be found in improving the efficiency of traffic control systems of two-way intersections in cities. Most traffic systems implement traffic signals with timed phases to control the flow of vehicles at intersections. In *"Organic Control of Traffic Lights"* and "*Adaptive Traffic Light Timer Control (ATLTC)"* this method is described as inefficient due to fluctuations in traffic density, simply because the fixed-time traffic control systems do not react to actual traffic situations [7], [8].

Consider a two-lane intersection. If, for example, Lane 1 has low traffic density, but Lane 2 has high-traffic density, a traffic signal using timed phases will not change the duration of the green light on each lane based on the pattern of traffic density, even though, ideally, Lane 2 should have a green light for a longer duration than Lane 1.

This leads to an increase in the total waiting time (TWT) of all cars at that intersection, which culminates in traffic congestion.

The aforementioned study thus proposed a solution to this problem by switching the traffic signal to the lane with greater traffic density or a longer queue of vehicles. However, in some cases, it may not be optimal to switch to the lane with a higher traffic density.

Now consider an intersection with fast-moving vehicles on Lane 1 and a longer queue of slower vehicles on Lane 2. In this case, if the traffic signal allows the longer queue on Lane 2 to proceed first, the faster vehicles on Lane 1 would spend a lot of time waiting for the slow vehicles on Lane 2 to clear the jam.

Conversely, if the faster vehicles on Lane 1 were allowed to proceed first, they would be able to do so quickly. Thus, the vehicles on Lane 2 would not have to wait for a long time.

This study drew its inspiration from the Central Processing Unit (CPU) of a computer in order to find a more efficient algorithm to control traffic. In time-shared computer systems, the CPU of a computer has multiple tasks to process at the same time with the restriction that only one task can be processed by the CPU at a point in time. For example, the CPU has to process the requests of multiple users, i.e., allow users to run programs and download files at the same time. In order to increase its efficiency, the CPU uses a variety of scheduling algorithms to order tasks in such a manner that decreases the TWT, or queueing delay, of tasks.

Similarly, at an intersection, a traffic signal is similar to a CPU. It has multiple tasks to schedule (multiple vehicles on two lanes that need to clear a traffic jam), but it can only schedule one task at one point in time. After all, only one vehicle can clear a traffic jam at one point in time; otherwise, there would be a collision. Hence, using CPU Scheduling algorithms to maximize the efficiency of the traffic signal would help reduce the TWTs of all vehicles in a traffic jam.

This cross-disciplinary approach offers the potential of determining the optimum traffic control algorithm for reducing traffic congestion in many urban cities with minimal changes to the existing infrastructure. Vehicles are sources of pollutants such as carbon monoxide, carbon dioxide, oxides of nitrogen and particulate matter [9]. The alleviation of traffic congestion will also be accompanied by several benefits including the reduction of vehicle emissions and air pollution [10]. Moreover, less time wasted by employees in traffic jams could translate to more productive time in the workplace, which could benefit businesses and the economy. Another advantage of reducing the TWTs of vehicles in a jam would be to enable emergency vehicles (e.g., fire engines, police cars, ambulances) to reach their destinations faster.

This research study identified the optimum CPU Scheduling Algorithm for controlling traffic by testing each of the four CPU Scheduling algorithms against a set of random traffic patterns, as well as by measuring and analyzing their performances. Subsequently, recommendations for the implementation and the usage of CPU Scheduling Algorithms in traditional traffic control systems were presented.

II. DESCRIPTION OF THE RESEARCH STUDY

A. Research Aim and Approach

The aim of this research study was to compare the relative efficiencies of four CPU Scheduling Algorithms at controlling traffic and reducing the TWTs of all vehicles in a traffic jam at a two-lane intersection and identify the most efficient one. Specifically, a computerized simulation was utilized to evaluate each of the four algorithms with the same set of 100 randomly-generated trials, i.e., random patterns. The following hypotheses were tested:

- H_0 (Null Hypothesis): There are no differences between the average TWTs of the four algorithms.
- HA (Alternate Hypothesis): There are differences between the average TWTs of the four algorithms.

B. Independent Variable

The independent variable for this research study is the type of CPU Scheduling algorithm. In the section below, each algorithm and the rationale for its selection will be presented:

1. Shortest Job First (SJF)

This algorithm prioritizes the lane that has a faster vehicle at the head of its queue than one with a slower vehicle. When slower vehicles exit the intersection first (instead of last), the amount of time they take to exit the intersection can contribute to an increased number of waiting cars. Consider the following situation:

- Lane 1: Bike, Bike
- Lane 2: Truck, Truck

Based on the assumption that bikes take one second to cross the intersection and trucks take two seconds, letting the bikes go first will generate a lower TWT than letting the trucks go first instead. Conversely, if trucks were allowed to go before the bikes, this would increase the TWTs because the bikes that could have crossed quickly would have to wait for the slower trucks to cross before they could. This is the rationale behind this algorithm. Below is pseudocode that describes the algorithm:

Algorithm: Shortest Job First Scheduling Algorithm
Input: *active_lane, head_active_lane, head_other_lane Initialization:*
if *active_lane* is empty **then**
change active lane
else if *head_active_lane* is slower than *head_other_lane* **then**
change active lane **else**
do not change active lane
end if

2. Longest Queue First (LQF)

This algorithm prioritizes the lane that has a longer queue of vehicles over the lane with a shorter queue. The rationale behind choosing this algorithm is that a larger queue of vehicles has more waiting vehicles; hence, it could contribute more to the TWT than a shorter queue would. Below is a pseudocode that describes the algorithm:

Algorithm: Longest Queue First Scheduling Algorithm
Input: *active_lane, other_lane Initialization:*
if *active_lane* is longer than *other_lane* **then**
change active lane **else**
do not change active lane **end if**

3. Shortest Job First Accounting for Context Switch Time (SJFCS)

This algorithm prioritizes the lane that has a faster vehicle at the head of its queue than the one with the slower vehicle, but it also accounts for the context-switch-time of a signal (time taken for a signal to change color). The rationale behind this algorithm is that if lanes contain an alternating pattern of fast and slow vehicles, SJF will have to switch signal lights many times. This can lead to a lot of time wastage because it takes time for the signal (context-switch-time) to change colors. SJFCS is thus supposed to enhance the efficiency of SJF by switching the lights for the lanes only if the benefit derived from switching outweighs the time needed for changing colors. Below is a pseudocode that describes the algorithm:

Algorithm: Shortest Job First accounting for Context Switch Time Scheduling Algorithm
Input: *active_lane, head_active_lane, head_other_lane, context_switch_time Initialization:*
if *active_lane* is empty **then**
change active lane **else**
time_difference = head_active_lane - head_other_lane
if *time_difference > context_switch_time* **then**
change active lane **else**
do not change active lane **end if**
end if

4. Round Robin Scheduling (RR)

This algorithm operates by using a 20-second timer. Every 20 seconds, the algorithm changes its active lane (lane with the green light). This is similar to the phase-wise algorithm currently used in most traffic signals. Below is pseudocode that describes the algorithm:

Algorithm: Round Robin Scheduling Algorithm
Input: *current_time*
Initialization

```
1:      if current_time % 20 == 0 then
2:              change active lane 3:       else
4:              do not change active lane 5:        end if
```

These four algorithms have specifically been chosen as they are the among the most efficient algorithms implemented by the CPU to schedule programs. Other commonly used CPU scheduling algorithms, such as Priority Based Scheduling (PBS) or Multiple Level Queues Scheduling (MLQS), have not been included in this study as they do not apply to traffic scheduling. For example, PBS schedules tasks with highest priority first, but in the traffic situations considered in this study all vehicles have equal priority. MLQS separates the queue of tasks based on properties of each task (e.g. memory, processing time required, type of process, etc.), but in a traffic situation, it is not practically possible to separate or reorder a queue of vehicles.

C. Dependent Variable

The dependent variable for this research study is the TWT (seconds) of all vehicles waiting at the intersection. TWT is calculated using the following formula:

$$TWT = \sum_{i=1}^{n} wi + ti$$

where wi is the waiting time of an individual vehicle, including the time a vehicle spends waiting for the

traffic signal to switch colors (context switch time), which is 6 seconds, and *ti* is the time taken for a vehicle to exit the intersection.

TABLE 1
VALUES OF *ti* FOR DIFFERENT VEHICLES

Vehicle	Time taken to clear intersection (*ti*)
Bike	4 seconds
Car	8 seconds
Bus	12 seconds
Truck	16 seconds

D. Procedure for Simulation

Step 1: Generate a random traffic pattern for Trial 1.

Each trial consists of two lanes that meet at an intersection — Lane 1 and Lane 2. Both lanes will each have a queue of 4–16 vehicles of a predetermined length consisting of bikes, cars, buses and trucks placed in a predetermined order with each vehicle having a different speed and arranged in a decreasing order of speed (see Table 1).

For example, the traffic pattern can look like this (see Figure 1):

- Lane 1: Bus, Car, Truck, Bus, Truck, Bike, Bike
- Lane 2: Bike, Truck, Car, Car, Bike, Car, Bus, Truck, Bus, Truck, Bike

Step 2: Conduct the trial using one of the algorithms.

i) Algorithm receives the following inputs:
   - the lengths of Lane 1 and Lane 2,
   - the context-switch-time of the signal,
   - the speed of the vehicle at the head of the queue of Lane 1, and
   - the speed of the vehicle at the head of the queue of Lane 2.

ii) The CPU Scheduling Algorithm can choose to switch a lane's light from green to red, red to green, or not to switch the lane's light at all. However, at any given time, one of the lanes must have a red light. Both lanes cannot have a green light at the same time (there can only be one active lane).

iii) The vehicle at the head of the active lane passes through and exits the intersection.

iv) The TWTs of all vehicles at the intersection will be calculated once all the vehicles have left the intersection. The increment TWT is calculated by the amount of time the vehicle that just exited the intersection had spent waiting and taking to exit the intersection.

v) Repeat Steps i-iv until all cars have left the intersection and record the final value of TWT.

Step 3: Repeat **Step 2** with all the remaining algorithms.

Step 4: Repeat **Steps 1-3** for Trials 2-100 (see traffic patterns for the 100 trials in Appendix A).

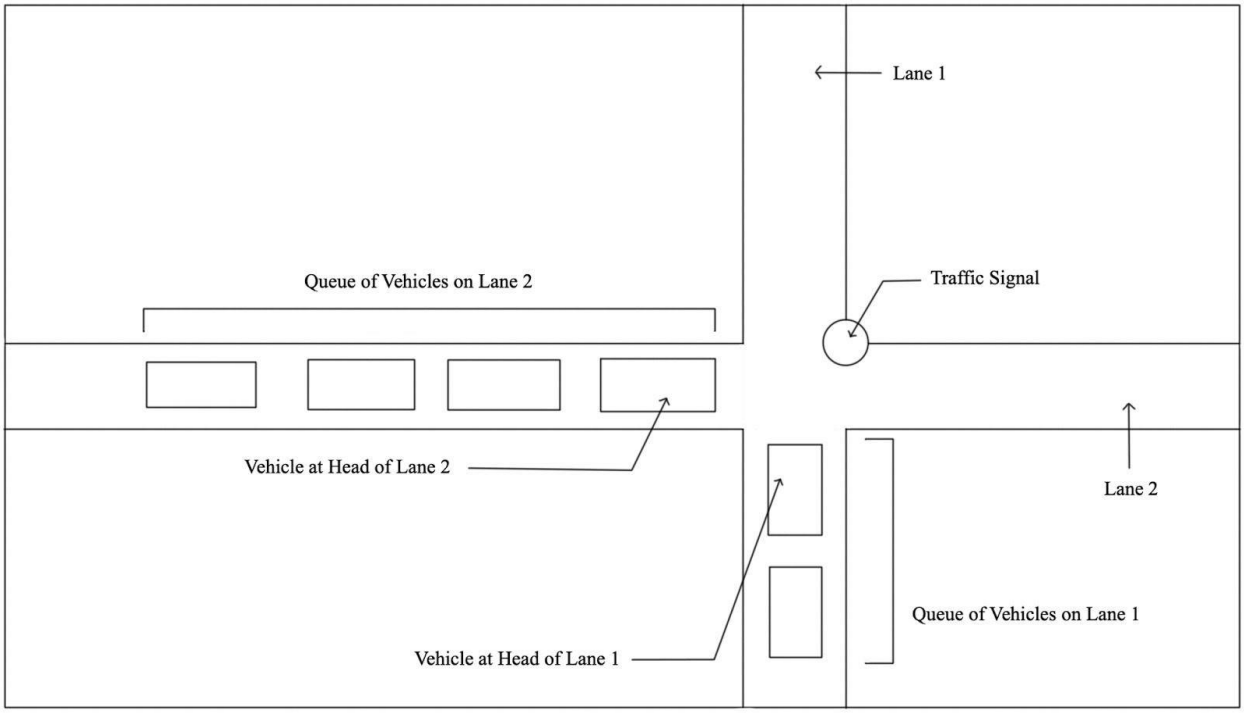


Fig. 1. Layout of simulated 2-lane intersection.

III. RESULTS AND DISCUSSION

In this section, descriptive statistics illustrating the differences between the average TWTs of the four CPU Scheduling Algorithms are presented and examined in detail (see Raw Data in Appendix B). This would help to determine which CPU Scheduling Algorithm has the lowest average TWT across Trials

1-100. An ANOVA analysis was then run to determine whether the differences in the average TWTs were statistically significant so as to validate the identification of the most efficient CPU Scheduling Algorithm for reducing TWT.

Table 2 shows the descriptive statistics calculated by using the TWT (seconds) calculated for each algorithm for each of the 100 trials. The algorithms are listed in increasing order of mean TWT. As shown in Table 2, the SJFCS has the lowest mean TWT (*M* = 2361.04 seconds, *SD* = 1168.36) amongst the four algorithms. Its mean TWT is lower than the mean TWT of the other CPU Scheduling Algorithms — Shortest Job First (*M* = 2421.32 seconds, *SD* = 1197.09); Largest Queue First (*M* = 2786.28 seconds, *SD*
= 1487.29); and Round Robin (*M* = 4595.36 seconds, *SD* = 2235.85) — by 60.28, 425.24, and 2234.32
seconds, respectively. Therefore, it is evident that the Round Robin Scheduling Algorithm was the least efficient at reducing mean TWT across most trials. In fact, its mean TWT exceeded the four algorithms by over 1800 seconds in each case.

TABLE 2
DESCRIPTIVE STATISTICS BASED ON RESULTS OF TRIALS 1–100

Groups	Mean	Variance	St. Dev.	Median
Shortest Job First Context Switch (SJFCS)	2361.04	1378851.514	1168.359105	2296
Shortest Job First (SJF)	2421.32	1447496.987	1197.088976	2298

Largest Queue First (LQF)	2786.28	2234380.365	1487.291687	2409
Round Robin Scheduling Algorithm (RR)	4595.36	5049508.112	2235.847273	4722

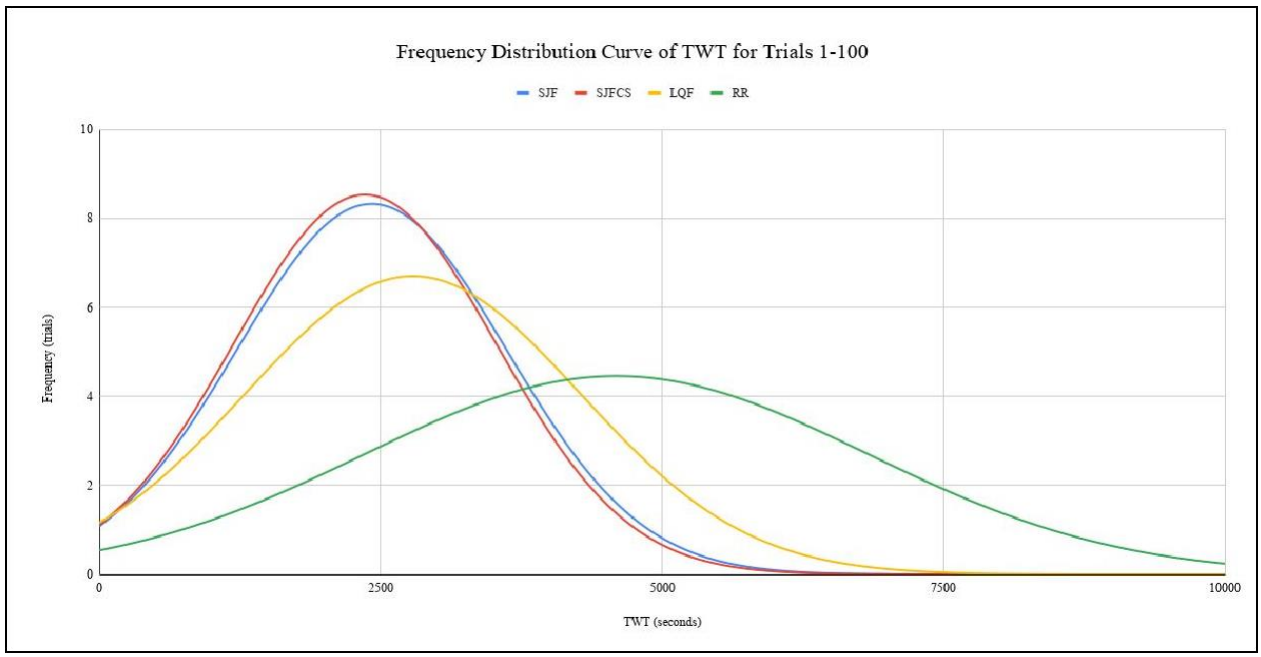


Fig. 2. Frequency Distribution Curve of TWT (seconds) for Trials 1-100.

The differences in the performances of the four CPU Scheduling Algorithm scheduling algorithms can also be seen in the frequency distribution curves of TWT (seconds) for each CPU Scheduling Algorithm for the 100 trials (see Figure 2). As can be seen readily, due to the high mean and standard deviation figures of the Round Robin Scheduling Algorithm, one can see that it has a far more inconsistent performance than all the three other algorithms, thus making it highly dependent on the traffic pattern in a given trial. Largest Queue, with its comparatively lower mean and standard deviation figure, would have a far more consistent performance. However, it is clear from this figure that the SJFCS and Shortest Job First, with their relatively low values for standard deviation, have the most consistent performance among all the CPU Scheduling Algorithms, making them less dependent on the type of traffic pattern in a given trial.

In order to determine whether the differences in the mean TWT of the four CPU Scheduling Algorithms are statistically significant, a one-way ANOVA Analysis was conducted. As shown in Table 3, the differences between the mean TWTs (seconds) across all four algorithms, for Trials 1-100, were found to be statistically significant [(F(3, 396) = 43.88 (higher than the F critical value of 2.62), p<.01]. This means that the null hypothesis can be rejected.

TABLE 3
ANOVA ONE-WAY ANALYSIS OF TRIALS 1–100 FOR ALL FOUR ALGORITHMS

Source of Variation	SS	df	MS	F	P-value	F-crit
Between Groups	332726619.2	3	110908873.1	43.87983123	0.000	2.62744077
Within Groups	1000913461.0	396	2527559.244			
Total	1333640080	399				

Further inferential testing was done due to the similarities between the mean and standard deviation figures of SJFCS and Shortest Job First. In fact, the proximity of their performance is readily apparent in Figure 1, as their Frequency Distribution Curves were almost overlapping. Therefore, a 2-sample t-test was run to determine whether the difference between mean TWTs of these two CPU Scheduling Algorithms is statistically significant.

TABLE 4
TWO-SAMPLES T-TEST OF SHORTEST JOB FIRST CONTEXT SWITCH AND SHORTEST JOB FIRST

	Shortest Job First	Shortest Job First Context Switch
Mean	2421.32	2361.04
Variance	1447496.987	1378851.514
Observations	100	100
Hypothesized Mean Difference	0	
df	198	
t Stat	0.358558802	
P(T<=t) two-tail	0.720307185	
t Critical two-tail	1.972017478	

Table 4 shows the results of the two-samples t-test conducted, using the results of Trials 1-100 for the two CPU Scheduling algorithms. There is no significant difference between the mean TWTs of SJFCS and Shortest Job First algorithms , $t(198) = 0.36$ (lower than the t critical value of 1.97), $p = 0.720$. Although there is a significant difference between the two mean TWTs when the ANOVA was run with all four algorithms, a limited statistical analysis of the mean TWTs of SJFCS and SJF reveals that there is no statistically significant difference between these two CPU Scheduling Algorithms. Since SJFCS is a modified version of SJF to account for the context-switch time of the signal, we can conclude that the modification to SJF in SJFCS would not make a statistically significant improvement to the mean TWT. This finding offers very practical useful information: improvements made on Shortest Job First in the SJFCS algorithm would not make much of a difference in the improvements in efficiency levels. Therefore, given that the SJFCS Algorithm is more complicated than the SJF Algorithm and would take more time to be processed, the SJF algorithm may be simpler and more efficient to use for controlling traffic. What is surprising about the results is that SJF and SJFCS had a significantly lower mean TWT than LQF, which indicates that the lengths of the queues of vehicles do not need to be considered while scheduling traffic. Moreover, the absence of a statistically significant difference between the mean times of SJF and SJFCS indicates that traffic algorithms do not need to account for the time taken by the traffic signal to change lights.

IV. CONCLUSION

The aim of this research study was to determine whether there were differences between the TWTs of different CPU Scheduling algorithms and which of the evaluated CPU Scheduling Algorithms was the most efficient at controlling traffic at a two-lane intersection. Based on the “Results and Discussion” section of this paper, one can comment that there are statistically significant differences between the mean TWTs across all four algorithms ($p = 0.000$). However, there are no statistically significant differences in the mean TWTs of SJF and SJFCS ($p = 0.720$). As such, the alternative hypothesis has been partially accepted. Since there are no significant benefits to using the SJFCS Algorithm over the SJF Algorithm, and given that the SJFCS Algorithm is more complicated than the SJF Algorithm and would take more time to be processed, the SJF algorithm may be simpler and more efficient to use while controlling traffic.

In conclusion, this research paper has drawn from scheduling theory and computer science to present a cross-disciplinary approach to reduce TWTs in traffic jams, through the use of an easy-to- implement efficient traffic control

system. More importantly, it has practical real-life ramifications that could resolve a pressing problem afflicting large metropolitan cities like Mumbai.

V. RECOMMENDATIONS AND IMPLEMENTATION

In this section, recommendations based on the "Results and Discussion" section are presented. The purpose of this section is to propose an efficient traffic control system and provide a plan for state or local governments to implement this system, in order to reduce mean TWTs for all vehicles in traffic jams. The following steps are recommended for the implementation of this traffic control system at negligible additional costs.

A. Traffic Cameras placed at busy intersections

In order to utilize the CPU Scheduling algorithms described in this research study, data from the real-life traffic situations, specifically the lengths of Lanes 1 and 2, the context-switch-time of the signal, the respective speeds of the vehicle at the heads of the queues of Lanes 1 and 2, need to be obtained. Data about the context-switch time is easy to obtain from the traffic signal itself. According to "*Traffic signal design-I"*, clearance intervals (context-switch time) are about 3-6 seconds long [11].

For the remaining information, traffic cameras can be placed at busy intersections to capture the data. With reference to "*An efficient vehicle queue detection system based on image processing"*, from traffic camera video feed or traffic camera images, the approximate length of a queue of vehicles can be estimated [12]. The methods elaborated in "*Measurement of Vehicle Queue Length Based on Video Processing in Intelligent Traffic Signal Control System"* can also be used to determine the length of the queue of vehicles on both Lanes 1 and 2 [13]. This data on its own is sufficient for the Largest Queue First algorithm to function, which would already improve the phase-based traffic signals used currently. Since Mumbai already has a network of over 5,000 traffic cameras that keep track of speeding and red-light-violating vehicles in order to issue *e-challans* [14], this existing infrastructure can thus be used to gather the necessary data without additional costs.

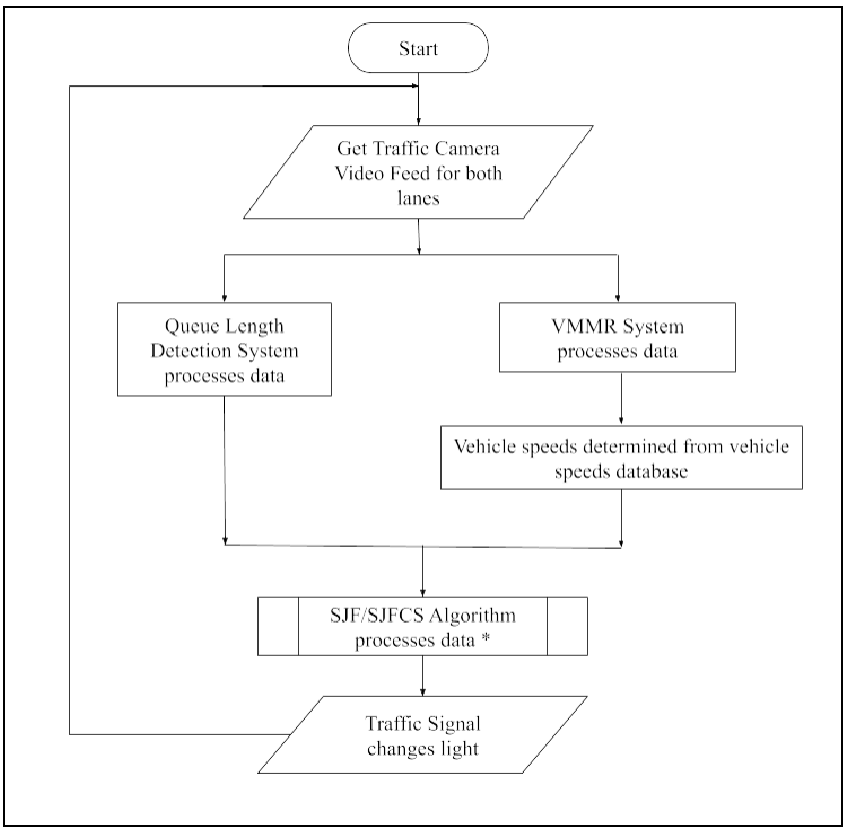
B. Vehicle Detection in real time

Although Largest Queue First is more efficient than the currently implemented traffic control algorithms, SJF or SJFCS would decrease the mean TWTs even further. In order to implement an SJF or SJFCS, estimates of the speeds of the vehicles at the heads of Lanes 1 and 2 are required. According to "*Recognition of Car Makes and Models From a Single Traffic-Camera Image"*, using video from traffic cameras placed at busy intersections, the make and model of the vehicle at the head of a traffic queue can be determined [15]. As described in "*Real-Time Vehicle Make and Model Recognition System"*, each frame of the feed is extracted, and passed through a Vehicle Make and Model Recognition (VMMR) System [16]. In addition to the usage of this VMMR system, a vehicle speed database should also be implemented. This is a database that contains the following information: the vehicle model, vehicle type (for example bike, car bus or truck) and the average speed that the vehicle travels at. Using the output from the VMMR System, the vehicle model and type can be matched with the vehicle's average speed using the vehicle speeds database. This can help us calculate the speeds of the vehicles at the heads of Lanes 1 and 2 respectively. This data are sufficient for the SJF and SJFCS algorithms to be implemented effectively.

C. Implementation of the SJF/SJFCS Algorithm at traffic intersections The CPU Scheduling algorithms evaluated in this study can only be implemented after recommendations A and B have been implemented. In their paper, "*Intelligent Traffic Control System Based On Round Robin Scheduling Algorithm",* L. Nipa and M. Islam outline a traffic control system similar to that described above, with one key difference: the fact that such systems make use of a phase based traffic control algorithm, very similar to the Round Robin Scheduling Algorithm [17]. In the "Results and Discussion" section, it has been proven that SJF and SJFCS algorithms are drastically more efficient at reducing TWTs than RR is. Hence, it would be advisable for traffic control systems to implement SJF or SJFC,S instead of RR. Together, the traffic cameras, VMMR and Queue Detection systems, and the SJF or SJFCS Scheduling Algorithm, form an efficient traffic control system whose operating procedure is outlined in the flowchart below:

Fig. 3. Operating procedure of a traffic control system implementing SJF/SJFCS Scheduling Algorithms.

* Process defined in “Description of Research Study”.



REFERENCES

[1] TomTom Traffic Index: Measuring Urban Traffic Congestion, TomTom, 2019. Accessed on: June 17, 2020. [Online]. Available: https://www.tomtom.com/blog/road-traffic/urban-traffic-congestion/

[2] India - Registered Vehicles in Mumbai 2017, Statista, 2017. Accessed on: June 17, 2020. [Online]. Available: https://www.statista.com/statistics/666663/total-number-of-vehicles-in-mumbai-india/

[3] S. Sen, Every day, 700 new vehicles hit Mumbai roads, The Times of India, December 3, 2017. Accessed on: June 17, 2020. [Online]. Available: https://timesofindia.indiatimes.com/city/mumbai/every-day-700-new-vehicles-hit-mumbai-roads/ articleshow/61908535.cms

[4] Chittaranjan Tembhekar, It’s official: Aamchi Mumbai has the worst traffic jams in the world, The Times of India, June 4, 2019. Accessed on: June 17, 2020. [Online]. Available: https://timesofindia.indiatimes.com/city/mumbai/its-official-aamchi-mumbai-has-worst-traffic-ja ms-in-the-world/articleshow/69659022.cms

[5] Debarghya Sil, Mumbai's Traffic Jams Is The Worst In The World, Says Report, The Logical Indian, June 6, 2019. Accessed on: June 17, 2020. [Online]. Available: https://thelogicalindian.com/story-feed/awareness/mumbai-traffic-worst-delhi/

[6] Mumbai’s Coastal Road Project – Civilsdaily, Civilsdaily.com, December 17, 2019. Accessed on: June 17, 2020. [Online]. Available: https://www.civilsdaily.com/news/mumbais-coastal-road-project/

[7] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck, Organic Control of Traffic Lights, Lecture Notes in Computer Science, pp. 219–233, 2020.

[8] Siddharth Srivastava, Adaptive Traffic Light Timer Control (ATLTC), NERD, March 29, 2016. Accessed on: June 17, 2020. [Online]. Available: http://www.iitk.ac.in/nerd/web/articles/adaptive-traffic-light-timer-control-atltc/#.XunNrGozZQJ

[9] K. Zhang and S. Batterman, Air pollution and health risks due to vehicle traffic, Science of The Total Environment, vol. 450–451, pp. 307–316, April 2013.

[10] J. Hermes, How Traffic Jams Affect Air Quality, Environment + Energy Leader, January 4, 2012. Accessed on: June 17, 2020. [Online]. Available: https://www.environmentalleader.com/2012/01/how-traffic-jams-affect-air-quality/

[11] Traffic signal design-I, Iitb.ac.in, 2020. Accessed on: June 17, 2020. [Online]. Available: http://www.civil.iitb.ac.in/tvm/1100_LnTse/529_lnTse/plain/

[12] M. Zanin, S. Messelodi, and C. M. Modena, An efficient vehicle queue detection system based on image processing, 12th International Conference on Image Analysis and Processing, 2003. Proceedings., 2020.

[13] Y. Cai, W. Zhang, and H. Wang, Measurement of Vehicle Queue Length Based on Video Processing in Intelligent Traffic Signal Control System, 2010 International Conference on Measuring Technology and Mechatronics Automation, March 2010.

[14] Red Light Violation Detection cameras | Mumbai Traffic PoliceMumbai Traffic Police, Maharashtra.gov.in, 2020. Accessed on: June 17, 2020. [Online]. Available: https://trafficpolicemumbai.maharashtra.gov.in/red-light-violation-detection-cameras/.

[15] Hongsheng He, Zhenzhou Shao, and Jindong Tan, Recognition of Car Makes and Models From a Single Traffic-Camera Image, ResearchGate, December, 2015. Accessed on: June 17, 2020. [Online]. Available: https://www.researchgate.net/publication/281182876_Recognition_of_Car_Makes_and_Models_ From_a_Single_Traffic-Camera_Image.

[16] Muhammad Asif Manzoor, Y. Morgan, and A. Bais, Real-Time Vehicle Make and Model Recognition System, ResearchGate, April 17, 2019. Accessed on: June 17, 2020. [Online]. Available: https://www.researchgate.net/publication/332484058_Real-Time_Vehicle_Make_and_Model_Re cognition_System.

[17] L. Nipa and M. Islam, Intelligent Traffic Control System Based On Round Robin Scheduling Algorithm, International Journal of Scientific and Research Publications, vol. 5, no. 8, 2015.

APPENDICES

APPENDIX A
TRAFFIC PATTERNS FOR TRIALS 1-100

Trials		
Trial 1	Lane 1	2, 2, 3, 0
	Lane 2	0, 3, 3, 1, 0, 3
Trial 2	Lane 1	2, 3, 2, 0, 3
	Lane 2	2, 3, 3, 3, 2, 1, 0, 0, 1, 2, 0, 3, 1, 0, 1
Trial 3	Lane 1	1, 3, 1, 0, 3, 1, 2, 1
	Lane 2	1, 1, 3, 0, 0, 0, 1, 2, 0, 1
Trial 4	Lane 1	1, 0, 0, 0, 3
	Lane 2	0, 0, 3, 2, 2, 1, 3, 3, 1
Trial 5	Lane 1	0, 2, 3, 1, 3, 3, 0
	Lane 2	3, 1, 1, 2, 0, 3, 1, 2, 0, 3
Trial 6	Lane 1	3, 3, 1, 3, 0, 0
	Lane 2	0, 0, 0, 1, 0, 2, 0
Trial 7	Lane 1	3, 3, 1, 0, 0, 1, 1, 1, 2, 2, 2, 3
	Lane 2	2, 1, 0, 0, 0, 2, 0, 1, 0, 3, 1
Trial 8	Lane 1	1, 1, 1, 1
	Lane 2	1, 2, 2, 2, 1, 0, 1, 0, 1, 3, 1, 0, 0, 3, 2, 3
Trial 9	Lane 1	2, 3, 0, 2
	Lane 2	0, 3, 3, 0, 2, 2, 1, 1, 1, 3, 2, 1
Trial 10	Lane 1	1, 3, 0, 3, 3
	Lane 2	1, 3, 1, 1, 3, 2, 2, 1, 0, 1, 2, 1
Trial 11	Lane 1	2, 3, 2, 1, 3, 0, 3, 0, 3, 1, 3, 0, 1, 3
	Lane 2	3, 2, 2, 3, 0, 0, 3, 1, 3, 3, 1, 0, 2, 3, 2, 0
Trial 12	Lane 1	1, 1, 1, 1, 1, 1, 3, 0, 2, 2

	Lane 2	2, 0, 2, 2, 0, 0, 3, 2, 2, 1, 2, 1
Trial 13	Lane 1	1, 0, 1, 2, 3, 3, 2, 3, 0, 3, 0, 0, 1, 3
	Lane 2	0, 3, 2, 0, 2, 3, 1, 2, 2
Trial 14	Lane 1	2, 0, 2, 3, 1
	Lane 2	1, 3, 2, 2, 0, 2, 0, 1, 1, 1, 3, 3, 0, 1, 1
Trial 15	Lane 1	2, 2, 0, 1, 0, 2, 0, 1, 2, 1, 3, 0, 0, 0, 1, 1
	Lane 2	1, 1, 3, 1
Trial 16	Lane 1	0, 3, 2, 2, 3, 3, 1, 0, 0, 3
	Lane 2	0, 0, 1, 3, 2, 2, 3, 3, 3, 1, 0, 2, 2, 3, 3, 3
Trial 17	Lane 1	0, 1, 0, 2, 0, 3, 2, 1, 1, 1, 0
	Lane 2	1, 3, 0, 0, 0, 2, 2, 0, 1, 2, 1
Trial 18	Lane 1	0, 0, 3, 3, 1, 2, 1
	Lane 2	1, 1, 1, 2, 0, 2, 2, 2, 3, 3, 0
Trial 19	Lane 1	1, 0, 3, 1, 1, 1, 1, 2, 3, 1, 0, 1
	Lane 2	0, 2, 0, 0, 1, 2, 0
Trial 20	Lane 1	0, 3, 1, 1, 1, 0, 2, 3
	Lane 2	2, 0, 2, 1, 2, 1, 1, 2, 2
Trial 21	Lane 1	1, 3, 1, 0, 2, 3
	Lane 2	0, 1, 3, 3, 2, 3, 3
Trial 22	Lane 1	1, 2, 3, 2, 3, 2, 1, 1, 2, 3, 1, 0, 0, 2, 3, 3
	Lane 2	2, 2, 3, 1, 3, 2, 0, 3, 0, 2
Trial 23	Lane 1	1, 0, 0, 1, 3, 0, 0, 1, 0, 2, 3, 1
	Lane 2	3, 3, 1, 1, 0, 2, 0, 2, 2, 1, 1
Trial 24	Lane 1	1, 1, 2, 3, 0, 2, 2, 2, 2, 0, 0
	Lane 2	3, 3, 1, 2, 0
Trial 25	Lane 1	0, 3, 0, 1, 3
	Lane 2	2, 3, 0, 1, 3, 3, 1, 2, 2, 3, 3, 2, 1, 1, 2, 1
Trial 26	Lane 1	2, 0, 0, 0, 2, 3, 1, 2, 0, 3, 3, 1, 0, 3, 1, 1
	Lane 2	3, 3, 3, 1, 3, 3, 3, 1, 1, 2, 0, 1

Trial 27	Lane 1	3, 0, 1, 1, 0, 1, 3, 1, 0
	Lane 2	2, 3, 0, 2, 1, 0, 1, 1, 2, 2, 2, 2, 3, 0
Trial 28	Lane 1	0, 1, 2, 3, 2, 3, 3, 0
	Lane 2	1, 2, 3, 2, 2, 1, 1, 0
Trial 29	Lane 1	0, 1, 1, 2, 2, 0, 1, 2, 0, 2, 3, 0, 0, 1, 1
	Lane 2	0, 3, 2, 0, 3, 0, 0, 1, 0, 2, 0, 1, 1
Trial 30	Lane 1	1, 1, 0, 2, 1, 3, 3, 0, 3, 1
	Lane 2	1, 0, 0, 2, 3, 3, 3, 2, 0, 3
Trial 31	Lane 1	1, 2, 1, 1, 0, 0, 3
	Lane 2	2, 3, 1, 1, 2, 2, 3, 0, 2, 0
Trial 32	Lane 1	3, 1, 3, 0
	Lane 2	3, 3, 3, 1
Trial 33	Lane 1	0, 2, 0, 1, 3, 0, 2, 1, 3, 1, 1, 1, 2, 1, 1
	Lane 2	1, 2, 2, 3, 0, 3, 0, 2
Trial 34	Lane 1	2, 2, 3, 0, 2, 0, 3, 3, 0, 0, 0, 3, 2
	Lane 2	0, 3, 1, 1, 1, 1, 2, 3, 1, 0, 0
Trial 35	Lane 1	2, 0, 3, 1, 2, 0, 2, 1, 3, 2, 3
	Lane 2	3, 2, 0, 3, 1, 1, 0, 0, 1, 1, 2, 0
Trial 36	Lane 1	2, 0, 1, 2, 0, 0, 3, 0, 3
	Lane 2	0, 1, 3, 1, 3, 3
Trial 37	Lane 1	0, 2, 2, 0, 3, 0, 1, 2, 0, 2, 2, 2, 0, 2
	Lane 2	2, 0, 3, 1, 2, 3, 2, 1, 0, 1, 3, 0, 2, 0
Trial 38	Lane 1	1, 1, 2, 3, 2, 0, 0, 3, 2, 0, 3, 3
	Lane 2	1, 0, 0, 3, 1, 2, 2
Trial 39	Lane 1	1, 1, 0, 3, 2, 2, 3
	Lane 2	1, 0, 3, 3, 1, 0, 0
Trial 40	Lane 1	1, 2, 2, 2, 3, 2, 3, 1, 3, 2, 3, 0
	Lane 2	1, 2, 3, 1, 0, 3, 3, 3, 1
Trial 41	Lane 1	1, 2, 3, 0, 1, 2, 0, 3, 0, 2, 1, 2, 3, 0

	Lane 2	1, 2, 2, 1, 3, 2, 3, 2, 3, 3, 1, 2, 2, 3
Trial 42	Lane 1	2, 2, 0, 3, 1, 3, 1
	Lane 2	2, 1, 1, 2, 3, 1, 3, 2
Trial 43	Lane 1	2, 1, 2, 0, 2, 1, 0, 1, 1, 2, 3
	Lane 2	3, 1, 3, 1, 0, 1, 2, 3, 3, 1, 1, 2, 1, 1, 1
Trial 44	Lane 1	3, 0, 3, 0, 3, 2, 1, 3, 3, 0, 0
	Lane 2	3, 0, 0, 0, 1, 0, 2, 2, 2, 1, 3, 0, 3, 1
Trial 45	Lane 1	0, 2, 2, 2, 2, 3
	Lane 2	2, 0, 1, 1
Trial 46	Lane 1	1, 2, 1, 3, 3
	Lane 2	3, 3, 2, 3, 2, 3
Trial 47	Lane 1	1, 2, 2, 3, 3, 1, 0, 1, 3, 1, 3, 1, 1, 0, 1
	Lane 2	1, 3, 2, 3, 2, 3, 2, 3, 0, 2, 0, 3, 1
Trial 48	Lane 1	0, 3, 2, 2, 2, 0, 1, 1, 2, 0, 0, 2, 0, 3, 1, 1
	Lane 2	3, 1, 3, 0, 0
Trial 49	Lane 1	2, 3, 0, 0, 3, 3, 2
	Lane 2	3, 2, 1, 0
Trial 50	Lane 1	2, 0, 3, 2, 1, 0, 3, 0
	Lane 2	2, 2, 2, 2, 1, 0, 3, 2, 3, 2, 0, 0
Trial 51	Lane 1	0, 0, 2, 2, 3, 1, 2, 0, 2, 0, 0, 3, 0, 0
	Lane 2	1, 2, 2, 2, 1, 3, 1, 3, 0, 2, 1, 1, 3, 1, 1
Trial 52	Lane 1	1, 2, 3, 3, 0, 0, 0
	Lane 2	0, 0, 0, 0, 3, 3, 2, 2, 0, 2, 2, 1, 2
Trial 53	Lane 1	2, 3, 1, 0, 2, 0, 3, 1, 2, 2
	Lane 2	2, 1, 3, 1, 0, 0, 1, 1, 1, 2, 3, 2, 1, 0, 2
Trial 54	Lane 1	1, 3, 2, 2, 2, 1, 2, 3, 1, 3, 0, 0, 0
	Lane 2	3, 2, 2, 2, 2, 1, 2, 2, 1
Trial 55	Lane 1	1, 1, 3, 0, 3, 0, 3, 1, 0, 1, 1, 1, 3, 2, 3, 1
	Lane 2	2, 3, 3, 0, 2

Trial 56	Lane 1	0, 1, 1, 1, 3, 2, 2, 1, 3, 0, 0, 1, 3, 0, 0, 1
	Lane 2	3, 1, 1, 2, 2, 1, 1, 0, 1, 0, 3, 2, 0, 0, 0
Trial 57	Lane 1	3, 2, 3, 3, 3, 0, 0, 1, 1, 2, 3
	Lane 2	1, 2, 0, 3, 0
Trial 58	Lane 1	0, 1, 1, 2, 3, 0, 0, 2, 2, 3
	Lane 2	1, 1, 0, 2, 3, 0, 2
Trial 59	Lane 1	2, 2, 1, 1, 2, 0, 0, 1, 2, 2, 3
	Lane 2	0, 1, 3, 0, 1, 1, 1, 1, 3, 2, 1, 2, 0
Trial 60	Lane 1	2, 3, 0, 1, 1, 1, 1, 2, 2
	Lane 2	2, 2, 0, 1, 1, 1, 0, 1
Trial 61	Lane 1	3, 0, 2, 3, 3, 1, 3, 3, 1, 1, 1, 0, 2
	Lane 2	3, 0, 3, 1, 1, 1, 3, 2, 0, 0, 2
Trial 62	Lane 1	2, 2, 2, 1, 1, 2, 0, 1, 1, 0, 3, 2, 1, 2, 2, 0
	Lane 2	1, 1, 2, 2, 2, 3
Trial 63	Lane 1	3, 2, 2, 3, 1, 0, 2, 3, 2, 1, 0, 1, 1
	Lane 2	2, 0, 3, 0, 3, 0, 3, 1, 0
Trial 64	Lane 1	1, 1, 1, 3, 0, 3, 0, 3, 2, 3, 2, 3, 2, 3, 1
	Lane 2	0, 2, 0, 3, 3, 0, 1, 2, 0, 1, 3
Trial 65	Lane 1	1, 1, 1, 3, 0, 3
	Lane 2	0, 2, 2, 0, 2, 0, 1, 1, 3, 2, 0, 0, 3
Trial 66	Lane 1	0, 3, 0, 2, 1, 3, 0, 3, 3, 2, 0, 1, 2
	Lane 2	3, 1, 3, 3, 1, 2, 2, 1, 1, 3, 0, 2, 2, 1, 3
Trial 67	Lane 1	0, 0, 2, 3, 1, 3, 2, 2, 0, 0, 0
	Lane 2	3, 2, 1, 3, 0, 2, 2, 0, 2, 3
Trial 68	Lane 1	1, 1, 0, 0, 2, 2, 0, 3, 0, 2, 3, 0, 3, 2, 2, 1
	Lane 2	3, 1, 0, 2, 2, 3, 0, 3, 3, 3, 0, 0, 2
Trial 69	Lane 1	2, 3, 2, 1, 2, 3, 3, 1, 2, 2, 3, 0, 2, 0, 2, 1
	Lane 2	3, 3, 3, 0, 3, 2, 2, 1, 2, 0, 2, 0, 2, 0, 2, 1
Trial 70	Lane 1	0, 0, 1, 2, 3, 0, 1, 2

	Lane 2	0, 2, 2, 0, 3, 0, 1, 3, 2
Trial 71	Lane 1	1, 3, 3, 1, 2, 1, 0
	Lane 2	3, 1, 0, 3, 3, 1, 2, 2, 3, 2, 1, 2, 0, 0, 3, 2
Trial 72	Lane 1	3, 2, 0, 0, 3, 0, 1, 3, 3, 1, 2, 2, 3, 0
	Lane 2	3, 1, 2, 2, 0, 1, 3, 0, 3, 2, 1
Trial 73	Lane 1	0, 1, 2, 3, 1, 0
	Lane 2	3, 0, 2, 0, 0, 3, 3, 0
Trial 74	Lane 1	0, 0, 1, 0, 2, 3, 2, 3, 2, 1
	Lane 2	0, 2, 2, 3, 1, 3, 2, 1, 0, 2, 0
Trial 75	Lane 1	2, 2, 2, 1, 3, 0, 1, 3, 1, 2
	Lane 2	0, 3, 2, 0, 1, 3, 2, 2, 1, 3
Trial 76	Lane 1	0, 2, 3, 3, 3, 3, 0, 2, 2, 1, 1
	Lane 2	3, 1, 1, 1, 2, 2
Trial 77	Lane 1	0, 3, 1, 3, 2
	Lane 2	3, 1, 2, 1, 3, 3, 2, 1, 1
Trial 78	Lane 1	1, 0, 1, 1, 3, 1, 2, 3, 1, 2, 0, 2, 0, 2, 0
	Lane 2	1, 1, 2, 2, 2, 0, 2, 3, 0, 3, 0, 1, 2, 1, 0
Trial 79	Lane 1	2, 0, 2, 2, 2
	Lane 2	2, 1, 0, 3, 2, 0, 3, 1, 1, 3, 0, 3, 3, 0
Trial 80	Lane 1	0, 0, 0, 1, 3, 1, 1, 2, 1, 2, 2, 1, 3, 3, 3, 3
	Lane 2	2, 1, 0, 3, 1, 1, 2, 2, 2, 1, 1, 2, 2, 0, 1, 0
Trial 81	Lane 1	0, 1, 3, 2, 3
	Lane 2	2, 2, 1, 0, 1, 3, 2, 0, 2, 1, 0, 3, 2, 2, 0
Trial 82	Lane 1	1, 1, 1, 2, 2, 3, 0, 0
	Lane 2	3, 1, 3, 1, 3, 3, 2, 2
Trial 83	Lane 1	0, 3, 0, 2, 0, 1, 3, 1, 0, 3, 3, 2, 0
	Lane 2	0, 3, 1, 0, 1, 1, 0, 2, 2, 3, 0
Trial 84	Lane 1	1, 0, 0, 0, 2, 3, 3
	Lane 2	2, 3, 0, 3, 2, 3, 3, 1

Trial 85	Lane 1	1, 2, 1, 3, 3, 3, 2, 3, 1, 0, 0, 1, 3, 1
	Lane 2	3, 1, 3, 1, 1
Trial 86	Lane 1	3, 3, 1, 1, 0, 1, 0, 1, 2, 0, 0, 3, 0, 2
	Lane 2	0, 2, 3, 3, 3, 2, 1, 2, 0, 0, 2, 3, 3
Trial 87	Lane 1	1, 0, 0, 3, 0, 1, 1, 1, 1, 3
	Lane 2	3, 2, 1, 3, 1, 1, 2, 3, 3, 3, 1, 1, 0, 3, 2
Trial 88	Lane 1	1, 3, 0, 0, 0, 2, 0, 0, 1, 0, 3, 0, 0, 2
	Lane 2	0, 3, 0, 0, 0, 3, 2, 2, 0, 3
Trial 89	Lane 1	0, 0, 3, 2, 0, 3, 1, 1
	Lane 2	3, 0, 2, 1
Trial 90	Lane 1	3, 2, 2, 0, 3, 0
	Lane 2	2, 0, 3, 0, 1, 0, 2, 3, 3, 1, 2, 2, 2, 3, 3
Trial 91	Lane 1	2, 3, 2, 0, 3, 3
	Lane 2	2, 1, 2, 2, 3, 2, 3, 2, 1, 1, 2, 0, 3, 3, 0
Trial 92	Lane 1	3, 2, 2, 3, 0, 0, 0, 3, 3, 3, 1, 2, 1
	Lane 2	3, 0, 1, 3, 1, 1, 1, 1, 2, 1, 0, 1, 1, 0, 0
Trial 93	Lane 1	3, 1, 0, 3, 1
	Lane 2	1, 1, 2, 1
Trial 94	Lane 1	0, 0, 2, 1, 2, 2, 2, 3, 0, 2, 3, 1, 0
	Lane 2	2, 3, 3, 2, 3, 1, 2, 2, 1
Trial 95	Lane 1	0, 1, 0, 0, 1, 1, 2, 3, 3, 1, 3, 3, 0
	Lane 2	3, 0, 1, 3, 0, 1, 3, 2, 2, 1, 2, 0, 0, 3, 1, 3
Trial 96	Lane 1	1, 0, 2, 3, 2, 2, 0, 3, 3, 1, 2, 2, 0, 1, 3
	Lane 2	0, 2, 1, 2
Trial 97	Lane 1	0, 1, 0, 0, 2, 3, 2, 2, 1
	Lane 2	0, 0, 1, 3, 0, 0, 1, 3, 0, 3, 0
Trial 98	Lane 1	0, 1, 3, 3
	Lane 2	2, 0, 2, 3
Trial 99	Lane 1	1, 3, 2, 1, 0, 0

	Lane 2	2, 3, 3, 3, 0, 2, 3, 2, 3, 3
Trial 100	Lane 1	2, 1, 1, 1, 0, 1, 0, 1, 0
	Lane 2	1, 2, 2, 2, 2, 0, 0, 2, 3, 0

KEY FOR APPENDIX A

Code	Vehicle
0	Bike
1	Car
2	Bus
3	Truck

APPENDIX B
RAW DATA OF TWT (s) ACROSS TRIALS 1-100 FOR ALL ALGORITHMS

Trial	Shortest Job First	Shortest Job First Context Switch	Largest Queue First	Round Robin
1	2408	2408	2774	5530
2	1636	1572	1932	2852
3	2356	2584	2292	5720
4	1542	1542	1854	3186
5	2258	2258	2982	4682
6	2170	2170	2584	4328
7	402	482	484	664
8	4916	4664	5598	9658
9	1756	1612	1754	3350
10	752	752	962	1470
11	3582	3362	4338	6462
12	1274	1166	1334	2118
13	2082	1938	2178	3570
14	2546	2546	2958	4846
15	4014	3898	4766	6994
16	5400	5400	6864	9844
17	656	656	792	1228
18	1430	1230	1618	2138
19	2192	2000	2368	3868
20	2556	2556	2510	5066
21	1390	1282	1460	2868
22	746	746	932	1176
23	3310	3082	3932	5492
24	3886	3810	4956	7544

25	3192	3192	4766	6234
26	4414	4338	5366	9230
27	2096	2032	2404	3928
28	2792	2632	3268	4944
29	2638	2478	2290	4802
30	3560	3372	4168	5776
31	3578	3342	4060	6752
32	2366	2662	2616	4056
33	1118	1086	1200	2288
34	1218	1206	1352	2140
35	2856	2828	3180	5780
36	3618	3618	4256	6560
37	704	648	738	986
38	1592	1568	1838	3174
39	2316	2292	2414	3802
40	1192	1192	1434	2186
41	1570	1458	1822	2906
42	3036	3036	3596	6112
43	2950	3058	3360	5996
44	3692	4164	4766	6754
45	3832	3580	4592	6964
46	1764	1764	2232	3328
47	964	748	1000	1540
48	3032	3000	3352	6776
49	648	508	634	970
50	724	752	892	1444
51	1626	1626	2004	2860
52	2610	2718	2842	5322
53	2674	2566	2822	6458

54	1256	1328	1514	2142
55	1752	1752	2250	3094
56	1420	1236	1506	3226
57	2914	2818	3486	5342
58	3370	3590	4274	5758
59	1934	1802	2066	3998
60	1002	1022	1226	2210
61	1528	1400	1422	2898
62	2460	2300	2616	5488
63	3136	3244	3592	5644
64	636	636	728	1012
65	1582	1394	1796	2948
66	1358	1274	1478	2790
67	1594	1594	1940	2852
68	3320	3268	4002	6282
69	2468	2364	2968	4360
70	4466	4342	4574	7374
71	4532	4488	5646	8042
72	924	928	1084	1708
73	5244	4920	6342	9754
74	5856	5584	6844	9688
75	2540	2628	2974	5378
76	1388	1344	1632	2680
77	1970	1770	1996	3768
78	3952	3952	4986	7218
79	3468	3292	3848	4920
80	3070	3070	3798	5866
81	3888	3648	4154	6994
82	1816	1704	1738	3342

83	5338	5018	6784	9304
84	932	980	1020	1408
85	2594	2494	2772	6640
86	2606	2606	2828	5984
87	1878	1886	1860	4172
88	1554	1558	1718	2942
89	2822	2738	3346	5758
90	1680	1516	1616	3056
91	1480	1352	1692	2580
92	2528	2504	2922	4762
93	2150	2354	2376	3748
94	2936	2776	3468	4928
95	1522	1578	1610	2362
96	4424	3976	5220	8224
97	2260	2260	2524	6184
98	2238	2018	2290	4998
99	2434	2434	2256	5288
100	2280	2184	2360	5700
Mean	2421.32	2361.04	2786.28	4595.36

AUTHORS

First Author – Shiv Kampani, Dhirubhai Ambani International School, Mumbai-400098, India shivkampani@gmail.com