Carnegie Mellon University

Intersection Traffic Light Optimization Problem

Vanessa Jiang, Maggie Liu,

Cara Yi, Jessica Zhao

21-393 Operation Research Ⅱ

Professor Zhang

Introduction

Traffic signals are a common type of intersection control. The goal in signal movement design is to improve overall safety, to decrease average waiting time, and to equalize the level of services for all (or most) traffic flows. Several traffic intersections in Pittsburgh stop traffic from all directions and allow pedestrians to simultaneously cross in all directions, including diagonally, and this type of traffic signal movement is called a pedestrian scramble. First introduced in the 1940s, the pedestrian scramble became less and less popular, because it prioritizes the flow of pedestrians over the flow of cars, leading to moments where no cars nor people can cross an intersection and potentially increasing certain parties' wait times, which may also result in jaywalkers.

Currently, traffic signals operate according to several methods. There are three types of traffic signals: pre-timed, semi-actuated, and fully actuated. Pre-timed intersections operate on a fixed cycle length; semi-actuated intersections have sensors on some or all movements except the main flow(s). Non-detected flows are controlled on a pre-timed policy, and fully actuated intersections have sensors in all directions. Each movement has an initial green light cycle length for queuing vehicles to get through; this cycle length is extended if the sensor detects a car moving through the intersection.

Problem Statement

Because traffic is so notorious in Pittsburgh, we decided to see how we can optimize traffic flow at four-way intersections. Using what we know about the ways traffic signals currently operate, in addition to traffic rules and other intersection properties, how can we effectively analyze and improve the overall traffic in Pittsburgh? This problem is nontrivial because we must consider a variety of factors including car flow, pedestrian flow, car speeds, pedestrian walk speeds, car and pedestrian intended directions, etc. We aim to design a method and choose parameters that will "satisfy" both pedestrians and cars. For example, we would not want to only cater to pedestrians and hence letting the cars wait for longer times. Therefore we consider both the average pedestrian and car wait times when evaluating the performance of our methods.

Some questions we answer through this research are: How often should the lights change for cars and for pedestrians? How does this affect the pedestrian and car wait times? Would having sensors to situationally change light behavior improve traffic? Specifically, which of the three approaches, non-adaptive, time-adaptive, or order-adaptive perform the best? How do other parameters in our model such as the rates of car and pedestrian arrivals affect car and pedestrian wait times?

To answer these questions, we took the following general approach: We first generated sample pedestrian and car flow distributions and designed and implemented three logical algorithms to model the intersection traffic according to a set of basic traffic rules and conditions. We then ran our simulation on a variety of parameters and analyzed how changing each parameter affected car and pedestrian wait times. Lastly, we demonstrate some of our conclusions via graphs and plots.

Mathematical modeling

To simulate the traffic flow, we assume a Poisson arrival process for both pedestrians and cars and test out different intensity parameters for each. The rates we examined range from 10 arrivals /sec to 100 arrivals /sec with an increment of 10.

We have 3 different types of traffic signal movements at the intersection: 1) Horizontal lights are green, which means cars can go East to West and people who want to cross horizontally can go; 2) Vertical lights are green, which is for cars going North to South and pedestrians in that direction; 3) All lights for cars are red, so pedestrians can cross in whatever direction they want, including diagonally.

We also have 3 different algorithms we use to control the traffic lights: 1) The non-adaptive version has the same length for all 3 lights, and goes in order of horizontal green, vertical green, and all red; 2) The time-adaptive version has the same order of lights, but we assume there are sensors that can detect how many pedestrians and cars are at the intersection waiting and can use that to shorten the current light if there is significant flow in another direction; 3) The order-adaptive version has the same fixed length for all lights, but when it is time to switch, we assume there are sensors again and use that to determine which light should come up next, that would be the most efficient choice for the current traffic.

We needed several assumptions for the model to be close to reality. The minimum time it takes for a pedestrian to cross is assumed to be 10 seconds, and at the intersection, only one car is allowed to cross every two seconds during a green light. In addition, there is a 2 second reaction time for cars when there is a light switch, which penalizes switching lights unnecessarily often in the adaptive case, which we will explain in more detail in the solution section.

Each simulation runs for 30 minutes, so there is enough time to mitigate some variation in data and so that the adaptive versions can make enough changes to reflect their advantages and disadvantages. To have a more direct comparison, for each set of parameters, we generate 400 iterations of traffic flow(with 4 different light intervals, and 10 different pedestrian's rate of arrival and car's rate of arrival). We use the same traffic flow for the 3 algorithms, and see which one does better by comparing the average wait time for cars and the average wait time for pedestrians. We also applied linear regression for the above-defined variables and average waiting times from 3 algorithms to see if there is any relationship that could help us compare our algorithms and understand the model further.

Solution

A main part of the project is building a simulator that can closely reflect what would happen in the real world. To start tackling this problem, we first defined two classes: Car and Pedestrian. The attributes stored by a Car object include the arrival time of the car, and which direction the car came from, which could be North, South, West, or East, since we are looking at an intersection. The attributes stored by a Pedestrian object include arrival time of the person, as well as the direction the person wants to travel in, which could be Horizontal, Vertical, or Diagonal. The second attribute of both Car and Pedestrian is randomly chosen from the list of possible values upon initialization of the object, since the arrival time generator mentioned above only generates the arrival time of cars and pedestrians, and not their directions etc.

For the actual simulator itself, we relied heavily on queues to keep track of how many cars and pedestrians are waiting at the intersection at any given time. Our simulator uses seconds as its unit, so at every second we do several things: First, we check if there are people or cars that arrive at this second, and add them to the corresponding queue based on their direction or location. We mentioned in the previous paragraph that the direction and location are attributes to the class objects, so it is simple to retrieve these values, which makes our choice of structure advantageous for clean implementation. Second, we check which lights are currently on, and let pedestrians and cars pass accordingly. One thing to note is pedestrians cannot start walking if the time remaining is less than the minimum walk time. Finally, we check if it is time to switch lights, and if it is, we update relevant parameters to reflect this change in state. The last step is different for adaptive and non-adaptive approaches, which will be discussed next, but for the basic case, it uses a variable that keeps track of how long the current light has lasted, and compares it to how long the light should last.

For the adaptive case, we focused on developing two different approaches, and will discuss their advantages and disadvantages in the result section in detail. We called the first approach “time-adaptive”, as the basic idea is that light durations may be shortened depending on the current traffic flow. For example, if the lights are green in the horizontal direction, but there is significantly more traffic in some other direction, we can shorten the current light so that lights in another direction would turn green sooner.

One problem that we faced initially while working on the time-adaptive approach was: if we just add a condition that shortens the current light and immediately changes the light, it could lead to pedestrians being stranded in the middle of the road as they cross. This is because a pedestrian would start crossing the road if they think the light will last longer than the minimum walk time, but if we alter the light lengths like this, there will be no way to know how much time is left until the next light switch. The solution we came up with was adding a variable called “flashing”, which is analogous to the count down you would see at a real pedestrian crossing light. So when the condition of shortening a light is met, we change “flashing” to true and keep a tracker of how long the light has been flashing for. No pedestrian should cross when the lights are flashing, if they have not started crossing already. We spent some time thinking about what the condition should be, to best reflect the traffic flow and the best decision. This is not trivial because pedestrians can cross together, but cars have a certain speed they can go at and they also have to wait in line to go through an intersection, so a buildup of cars should be worse than a buildup of pedestrians, since each car in the traffic jam will wait for even longer. We decided to use a factor that is the number of cars that can pass per second, which in the model is set to be some reasonable constant, and to scale cars to be comparable to pedestrians, we divide the number of cars by this factor. As we will later show in the results, this scheme works pretty well, but it could probably be improved and made more sophisticated, which is what can be worked on in the future.

The second of the two approaches is called the “order-adaptive” approach, where we fix the length of each light interval but make the order of the lights flexible, instead of going in a cycle between the 3 lights. To achieve this, we first calculate the total traffic flow(cars and pedestrians), where cars are scaled in the same way as the time-adaptive version, for all 3 light states. So for the horizontal green lights, the total traffic flow would be cars in the East and West direction and people who want to cross horizontally; for vertical green lights it would be North and South, and for 4-way red lights it would be all the pedestrians. Then, we find which state has the heaviest traffic flow, and we make that light state the next one. Initially, we did not check for all 3 states, and simply compared the two lights that are not currently on. However, the results were not good, so we proceeded to alter the algorithm and check for all 3 states, which led to significant improvements in the outcome.

Results & Discussion

By running a correlation test, we discover that by doing our first approach, the non-adaptive method, the pedestrian waiting time is positively correlated with the light interval, and the car waiting time is positively correlated with the car's rate of arrival. These two correlations are reasonable since we restrict the number of cars passing by the intersection while we have no restrictions on the number of people walking during green light. This property can be also seen in the correlation between car waiting time and car's rate of arrival in the other two algorithms, whereas pedestrian rate of arrival has little influence on any of the pedestrian waiting times. In the order-adaptive and non-adaptive approaches, pedestrian waiting times are correlated with the light interval. However, in the time-adaptive approach pedestrian waiting time has very little correlation with all parameters while maintaining a low waiting time. Therefore, the shorter the interval, the better the adaptive and order-adaptive algorithms perform on pedestrian waiting time, and the car waiting time in all three is mostly affected by the car's rate of arrival.

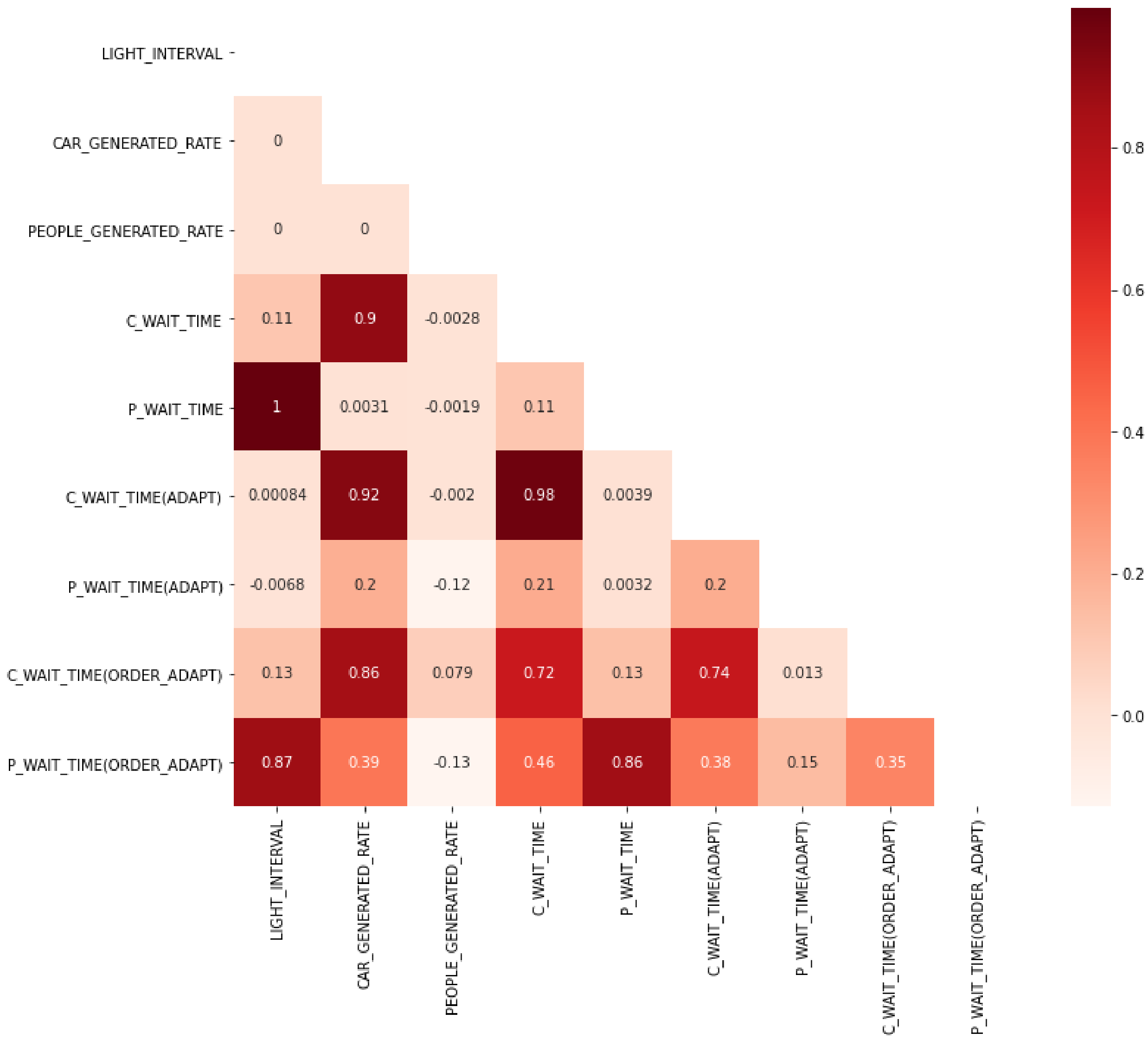
Among the three approaches, each model has its own advantages and disadvantages depending on the traffic conditions. In general, the 30 second interval order-adaptive approach performs better when cars have heavier traffic, and time-adaptive approach performs extremely well for all circumstances for pedestrians (Appendix B). We speculate that since the pedestrian wait time depends on the light interval, the time-adaptive approach having the option to shorten the interval plays heavily into their favor and therefore greatly shortens the waiting time of pedestrians. The order-adaptive approach performs well on cars. Specifically, when there is medium traffic(40 to 80 cars per minute), it does significantly better than the non-adaptive and time-adaptive cases. This is probably due to the fact that it prioritizes longer lines, and switches to the corresponding light, which in turn achieves an overall reduction of average wait time for cars. It is also important to note that when the car rate is around 90 to 100 per minute, the wait time of cars is over 200 seconds for all the algorithms, which indicates that when there is too much traffic, the traffic jam and therefore longer wait time is unavoidable. However, we should remember that these are the results we got under our assumptions about the environment, which includes the penalty we add for switching lights as well as the car speed we use to track car traffic. If a different set of assumptions is used, such as having a heavier penalty for switching lights often, we could get different results that may indicate a longer time interval is advantageous.

For future work, we can develop a new adaptive algorithm that incorporates both of the ones we have developed. As we have analyzed above, the two adaptive algorithms work well in different scenarios and have different advantages. We could combine the two and have an algorithm that can adapt in both ways, which could potentially reduce the wait time for both pedestrians and cars in a meaningful way. Also as mentioned before, we could do more rigorous testing and find a good conversion factor to take into account the speed of cars so that the car count can be comparable to number of pedestrians when we work on adaptive algorithms.

In addition, we can build a more complicated model by adding additional features. We can change some existing fixed features into hyperparameters, such as the penalty for switching lights too quickly, the constant used to balance the pedestrian and car buildup in time adaptive approach, and the travel speed of cars. Some of the ones we have considered include adding left and right turns for cars, which would not only interact with the pedestrians crossing, but also potentially hold up traffic behind the turning cars. Also, if we look at the Murray & Forbes intersection, which has a bus stop, we can take that into account and have buses that have to stop there regardless of the traffic light, which could hold up traffic and people who come off of the bus could add to those that are waiting at the intersection to cross. To incorporate these features, we can continue to use the framework we have now, but add an attribute to class Car to indicate which way it is going, and have additional conditions and variables that simulate the more complicated traffic interaction.

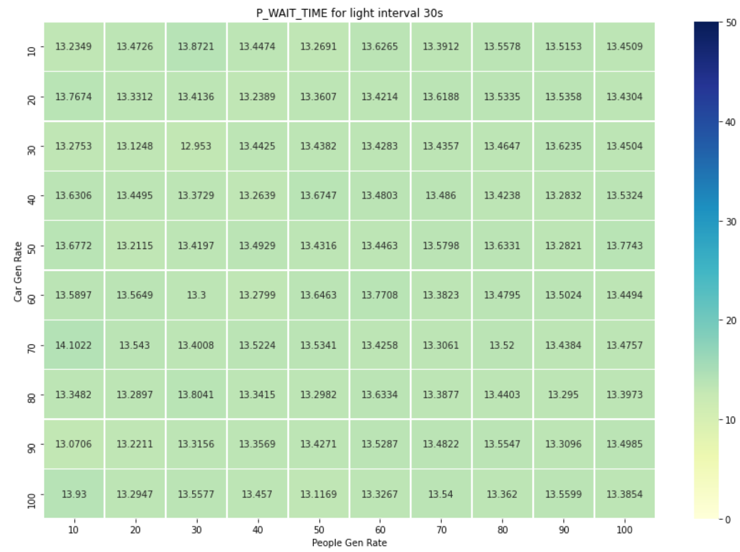
Appendix A

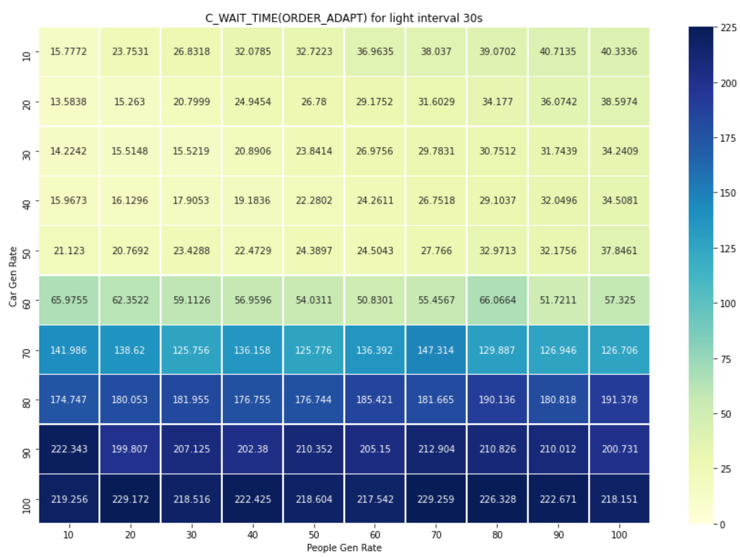
Correlation Graph

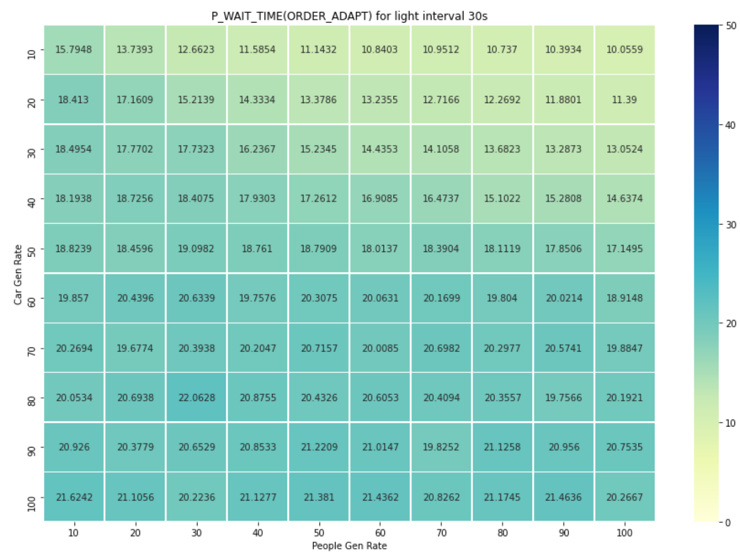


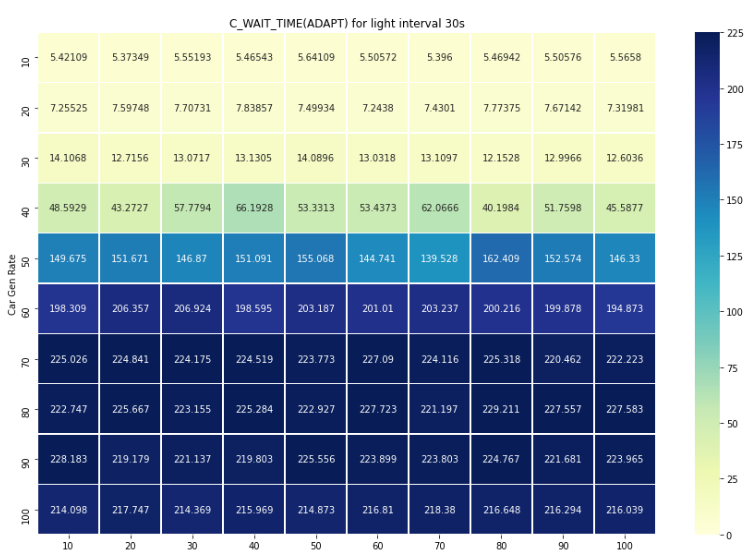
Appendix B

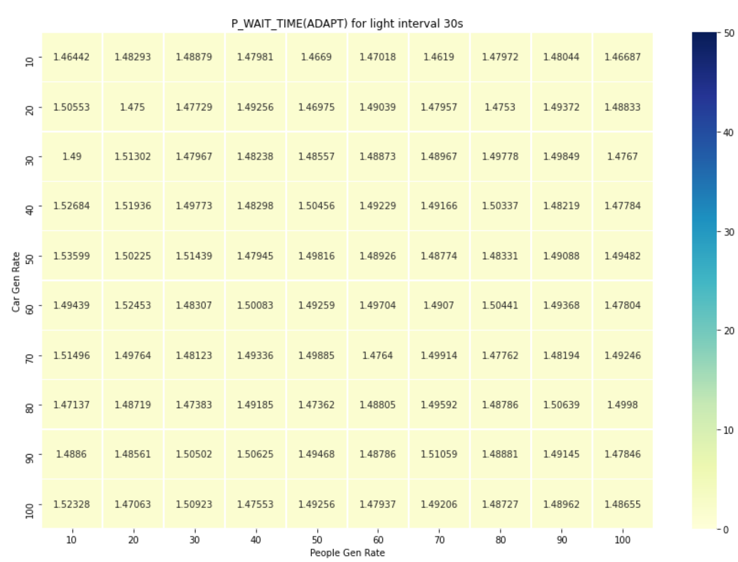
Light Interval = 30 (for normal and order adaptive)



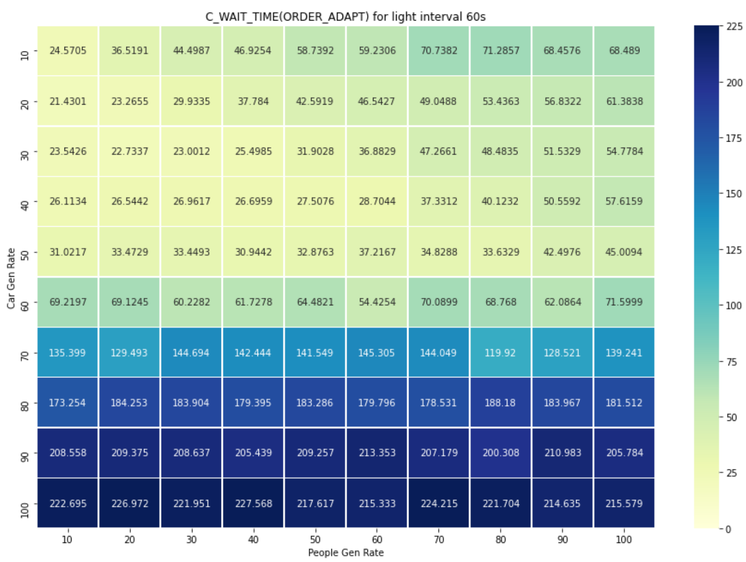


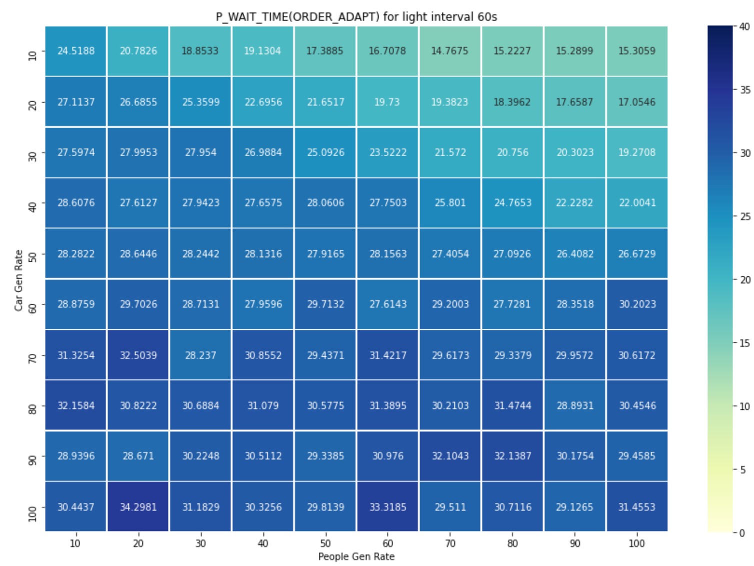


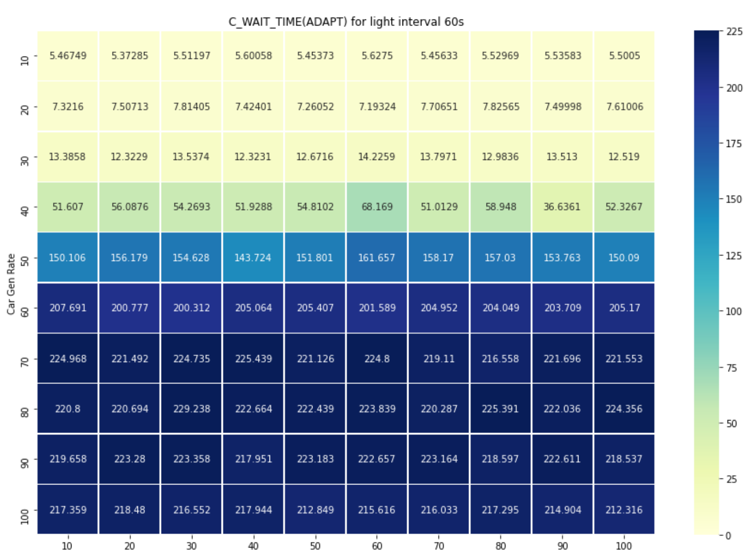


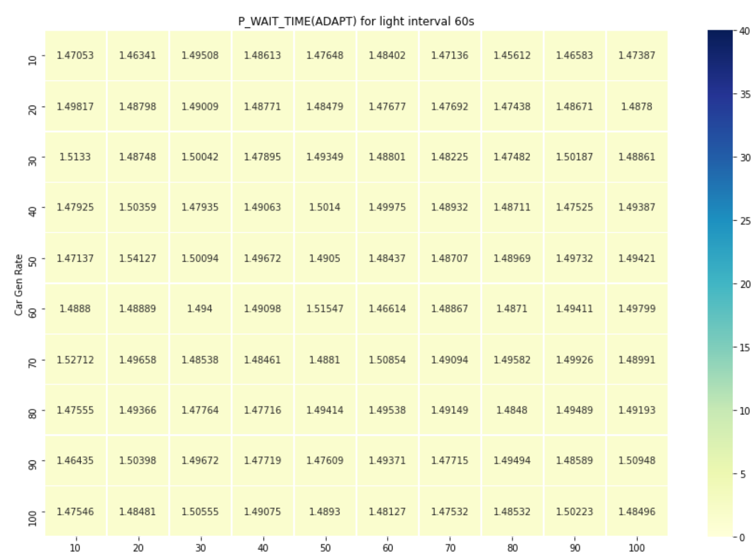


Light Interval = 60(for normal and order adaptive)

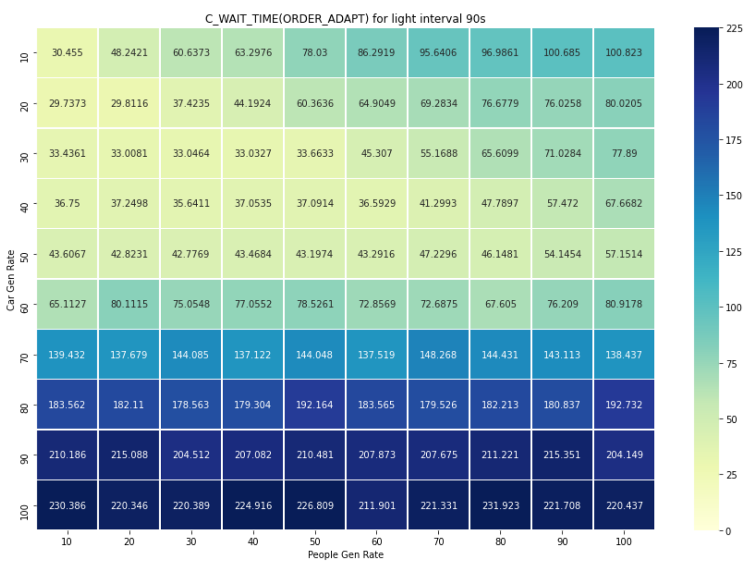


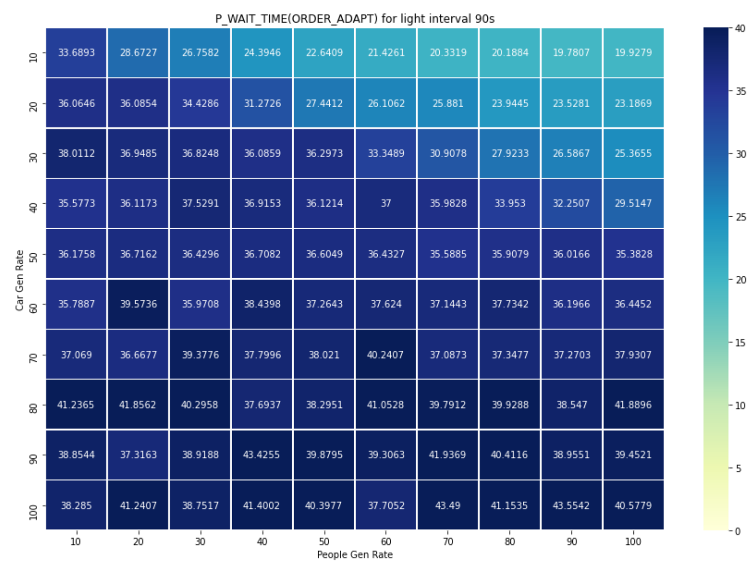


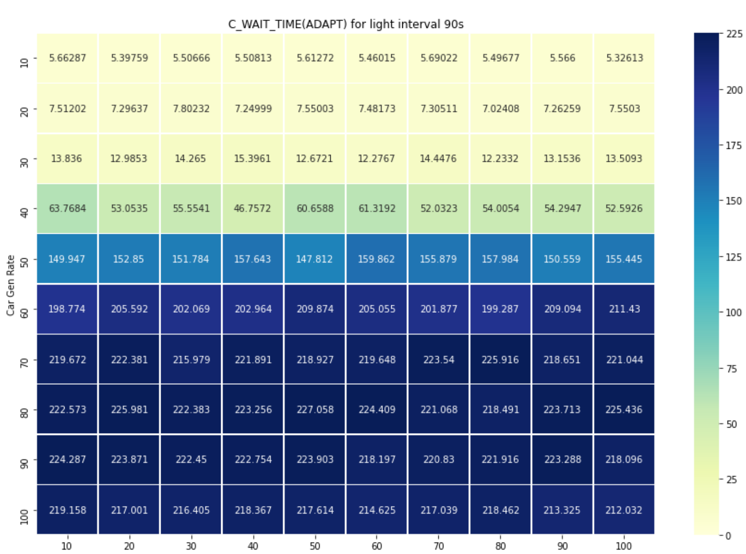


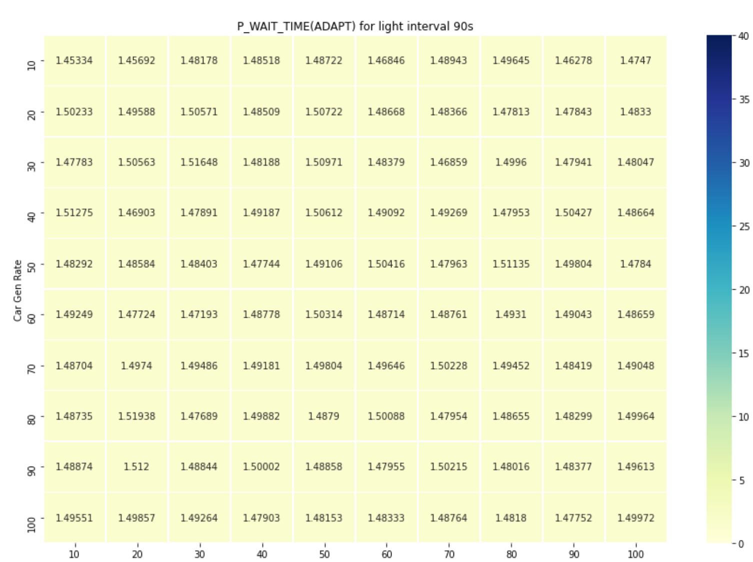


Light Interval = 90(for normal and order adaptive)









Interval = 120 (for normal and order adaptive)