# Learning Traffic Light Simulation

Lynn Jepsen
Computer Systems Lab, 2006-2007

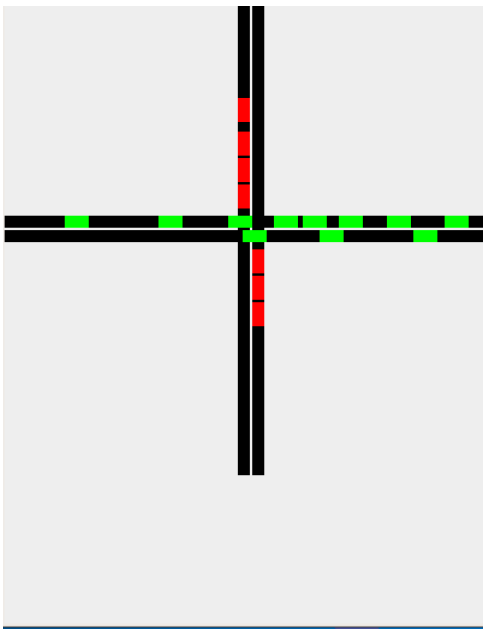
March 30, 2007

**Abstract**

This project is meant to simulate a busy traffic light. The program recognize patterns in the intersection and then uses that information to make the light as efficient as possible. These patterns could be related to the time of day and/or the day of the week. At first I would purposely input recognizable patterns and see if the program would catch it, but eventually the plan would be to possibly use this program at a real intersection. There are many varibles that the program takes into account including traffic desnity, number of lanes, etc.
**Keywords:** traffic simulation, efficiency, cars, red light: stop, green light: go, pedal to the metal, voom voom, honk

## 1 Introduction

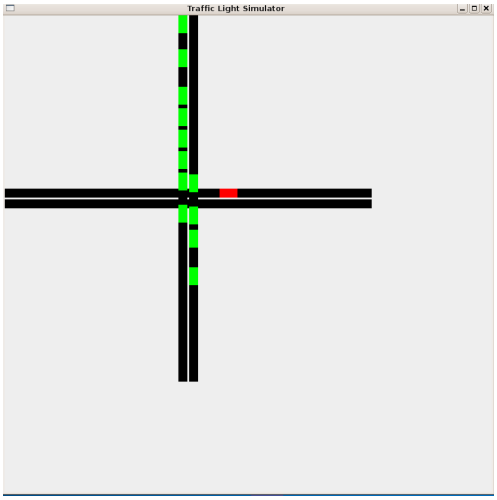
I want to develop an algorithm that will decrease queue length and wait time while also increasing green light usage by changing the cycle length and the ratio of green light time in each direction. In order to see if this algorithm is working, and gather lots of data very quickly, I built a simulation of a traffic intersection to test the algorithm on. The simulation follows basic rules of the road. Each car takes up so much space on the lane, travels so fast, and has a max acceleration. It is a realistic simulation because it obeys physical laws as well as human laws. People only travel so fast and do not peel out of intersections. The data gathered from the intersection are five separate numbers. First, the traffic density, the number of cars per minute that pass through one direction. Then, the queue length, maximum wait



time, and green light usage of the past cycle in the intersection. These are the efficiency variables. We want to optimize them in order to create an efficient intersection. To do this the light algorithm chooses an appropriate cycle length and ratio for green light time in each direction. These factors are chosen by looking at what happened to the efficiency variables when previous cycle length and ratios where used on given traffic densities. Once the cycle is complete we store all five data points so that the light algorithm can use them again.

# 2 Background

Traffic lights have always caused problems with traffic flow. If the light has no information about the intersection it is controlling, then you can sit at a red light for what seems hours. Even worse is when the light is much to short and you have to wait through lots of cycles. We have all experienced this. There has been a lot of research done to try and automate cars, using GPS, so that they all pass through the intersection harmlessly. However, this technology can be expensive, and it would take a long time to install it into all cars in order for the project to work. I think a much cheaper




and temporary solution would be to fix the traffic lights our society already has with brand new algorithms that would make the light efficient. In order to see if this algorithm is working, and gather lots of data very quickly, I designed a simulation of a traffic intersection to test the algorithm on. This way I can test pictorially, the aerial view of the intersection, but also graph the efficiency variables to see general trends.

# 3 Developments

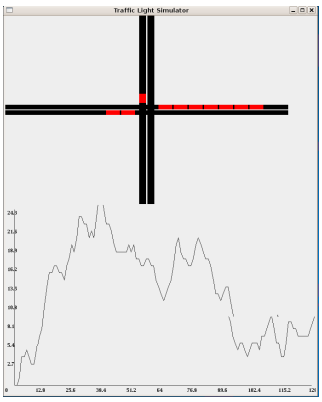
## 3.1 Simulation

The first section that I developed was a bsaic simulation of a 4 lane intersection. It has no light algorithm. Instead it changes eevry 15 secnods. It took some time to program in cars that move with realistic speeds and accerleration. Once I had the entire program working togehter to store the information that kept track of each car, I made a visual component to show the intersection. This made it easy to see if cars were indeed acting realistically. I could simply look at the simulation and decide if that is what a traffic intersection would look like.

## 3.2 Light Algorithm

The next thing I worked on was the most important part, the light algorithm. The light algorithm is what decides when to change the lights from red to green in all four directions. In order to do this there are two variable that must be found. These are cycle length and ratio. Cycle length is how long it takes the intersection to go through an entire cycle with both sides having their chance to be green. Ratio is the ratio of green light time in the north/suoth direction compared to the east/west direction.

I wanted to find the correct variables for the light cycle at each instant on an intersection that would maximize efficiency. I defined efficiency with three other variables. They are queue length, wait time, and green light usage.[2] Each instant on the intersection has one main variable that I cannot effect with my light algorithm. This is the traffic density. So if we consider traffic density, cycle length, and ratio to be our independent variables and our efficiency variables are our dependent variables. The experiment is to change th cycle length and ratio (because you can't change traffic density) in order to optimize efficiency (all three variables). The trick comes is getting all three efficiency variables optimized at once. At first I made the two functions to find cycle length and ratio. Cycle length simply took the max wait time in each direction and added them together. I figured this would give enough time for lots of cars to get through the intersection. Ratio was decided by taking the queue length in each direction and dividing. I later changed this to refelct the optimization better. In the beginning, when there is little data for the intersection to work off of, it uses the before mentioned functions. But once it has some information about previous traffic densities and the cycle length and ratio I used on the intersection in that situation, I will try and find which combination of cycle length and ratio best optimized all three variables. Sometime the algorithm compromises to make sure that even though wait time might have been lowest at this cycle, queue length was much too high. The intersection continues to compromise and eventually begins to use similar cycle lengths and ratios. It approaches an optimization after enough time. Considering that an intersection recieves tons of data every day, I feel like an algorithm that needs lots of informatino is appropriate for the situation.

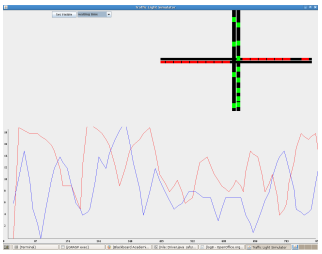


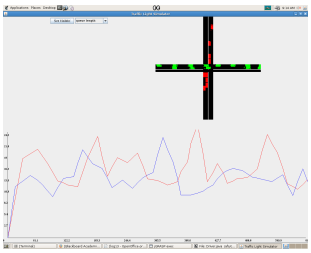

## 3.3 Graphs

I added another visual component to my program once I had finished my first draft of the light algorithm. It allows you to graph all three efficiency variables. This way you can see if the algorithm is really optimizing the intersection. I made is so that it grapheda north/south line and an east/west line. The closer the two lines are to each other, the better the optimization. The intersection can only be so efficient. If there are just too many cars for the number of lanes, then the algorithm will not work as well as it would in a lighter traffic density. This is not an exact science. There are too many variables in effect here. You cant possibly minimize wait time to the point where no one waits. Traffic flow is just too erratic to predict well.

## 3.4 Mulitiple Lanes

I updated my simulation to be able to include multiple lanes in any direction. While this does tend to lighten the traffic density, it has little other effect on the light algorithm. I did have to increase the yellow light time when there are too many lanes, because other wise too many cars run red lights. I have yet to make it so that cars can change lanes, but that is my next task.





# 4 Results and Discussion

...I will do this once i finish el project...