# Automatic Traffic Magic: Optimizing Traffic Signal Times via Genetic Algorithms

**Tyler Durkota**
tyler.durkota@gatech.edu

**Kendall Merritt**
kmerritt31@gatech.edu

**Greg Rago**
grago3@gatech.edu

**Zachary Sparks**
gth838n@gatech.edu

## Abstract

Due to lack of real world data on traffic patterns and light timings, a traffic simulation was created. This simulation models a 3x3 intersection of a metropolitan downtown road network. In order to optimize light timings in this network to maximize the average speed of all cars in the simulation, a genetic algorithm was implemented. The chromosomes of the algorithm held within them all of the light timing data for each of the intersections. After a generation of individuals were simulated and measured for fitness, a new generation was created as "offspring" of the most fit from the previous generation. Originally, the simulation was developed with some stochastic behavior. The drivers on the road could be sampled from a spectrum of aggressiveness. Additionally, the cars were given randomized routes to drive on the map. Unfortunately, these complexities proved too much for the algorithm to be effective. However, the model was simplified to a deterministic state and significant improvements were obtained.

## 1 Introduction

Traffic signal timings are something that are very close to the deepest parts of every driver's heart in a terrible way. Consider this situation: a driver is running late during rush hour and has found herself in the midst of wall-to-wall traffic in a metropolitan city block. She notices that nearest light is green, however the subsequent light is red and therefore prohibiting the traffic in her intersection to advance! She thinks to herself, "has any thought been put into the timings of these lights, or are they just randomly firing?" Most people can relate to this heart wrenching, not to mention blood boiling, situation.

Not only do poorly timed lights increase the blood pressure of drivers everywhere, inefficient light timings can increase emissions from stationary vehicles and lower gas efficiency when cars are constantly accelerating and decelerating. According to the US Energy Information Administration, an estimated 374 million gallons of gasoline are consumed each day in the United States. With such a large number number of vehicles consuming fossil fuels, even a fraction of a percentage point reduction attributed to optimized traffic lights can have a meaningful impact. Furthermore, the average worker in the US has a 25 minute commute and a city driver spends 36 hours per year stuck in gridlock. These problems, too, could be at least partially alleviated by optimal traffic signal times.

The problem's difficulty by itself makes it interesting from a machine learning perspective. Light times lie within an infinitely large hypothesis space making traffic light optimization NP-complete. It is easy to verify a good traffic light system, but coming to the best solution is very difficult. The

goal of this project is to explore the factors that are important in light timings and to learn more about the feasibility of robust light timing systems.

## 2 Related Work

The first paper of use is Traffic Optimization System: Using a Heuristic Algorithm to Optimize Traffic Flow and Reduce Net Energy Consumption [1]. It describes a traffic simulation similar to the aim of this project, and helped to establish some of the mathematical models and methodology for traffic light optimization. Saharia utilizes a congestion index to rank the intersections by amount of traffic at each light over the course of the simulation. This congestion index is calculated based on the location, density, and movement of cars at each intersection. The author found that by using this congestion index to determine light timings, the average speed of vehicles in the simulation was increased. While these methods were used explicitly in this project, it helped with initial planning of the project.

The next two resources were used extensively in this project. Congested Traffic States in Empirical Observations and Microscopic Simulations (Treiber) analyzed traffic congestion in Germany based on different road conditions. This paper defines a continuous car following model [3]. The third paper, helped define the optimization technique used in the project. Genetic Algorithms and Cellular Automata: A New Architecture for Traffic Light Cycles Optimization [2]. This resource was the basis for our genetic algorithm approach to optimizing the timings of the traffic lights in our simulation.

## 3 Approach

### 3.1 The Simulation

Machine learning methods require a sufficient amount of data to be effective. After an unfruitful search for data describing traffic flow and signal times, the only avenue available was to create the data. This goal was accomplished by a traffic simulation. The traffic simulator begins by creating a n-by-n grid of intersections where each road connecting the intersections is 275 meters long, consistent with downtown Atlanta block lengths, verified via Google Maps. At the edges of the simulation map, there exist car factories and car sinks. The factories create cars at a rate specified by the simulation. However, if the road that the factory is connected to is full, no car will be created. The car sinks at the edges of the map collect the cars as they drive out of view.

The goal of the project is to maximize the average speed of each car on the road. The average speed is calculated using equation 1:

$$\bar{v} = \frac{\sum_{i=0}^{n} d_i}{\sum_{i=0}^{n} t_i} \tag{1}$$

where $\bar{v}$ is the average velocity of all cars, $d_i$ is the distance traveled by car $i$ and $t_i$ is the time that the car was on the map. These figures are made available by the car factory "stamping" the car with a creation time, the car sink doing the same with the completion time and the car itself keeping up with the distance that it traveled. The actual behavior of the cars in the simulation is dictated by the intelligent driver model.

### 3.2 Intelligent Driver Model

The intelligent driver model (IDM) is a time continuous car following model that was developed by Treiber, Hennecke and Helbing in 2000. The model is defined by two ordinary differential equations shown below in equations 2, 3 and 4:

$$\dot{x}_\alpha = \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha \tag{2}$$

$$\dot{v}_\alpha = \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*\left(v_\alpha, \Delta v_\alpha\right)}{s_\alpha}\right)^2\right) \tag{3}$$

$$\text{with } s^*\left(v_\alpha, \Delta v_\alpha\right) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \tag{4}$$

where $x_\alpha$ is the position, $t$ is time in seconds, $v$ is the velocity, $a$ is the maximum acceleration, $v_0$ is the desired velocity, $\delta$ is the free acceleration exponent (usually set to 4), $s_\alpha$ is the space between car $\alpha$ and the car in front of it, $s_0$ is the minimum distance allowed between two vehicles, $T$ is the desired time headway, $\Delta v_\alpha$ is the difference in velocity between car $\alpha$ and the car in front of it and finally $b$ is the braking acceleration [3].

This model fully describes the behavior of the cars in the simulation. Aggressive and conservative drivers can be described easily using this model. For aggressive drivers, the desired velocity is greater than the speed limit and their acceleration and braking acceleration constants are set to higher than normal numbers. The opposite characteristics are applied to conservative drivers. These variations help to describe a robust model that imitates reality, where there is a spectrum of driving styles.

The lights in the simulation are not actually lights, but "dummy cars." These dummy cars are simply stationary cars that act as the barriers to the cars entering the intersections. Thus, when an incoming road's light is red, the first car in the queue has a stationary car in front of it, prohibiting it from advancing. When the light changes to green, the dummy car is taken away and the first car in the queue's lead car is then set to the last car in the road segment it is about to enter.

### 3.3 Car Routing

The cars in the simulation choose their route from a random mechanism. Each car has two directions that they can travel between North or South and East or West. When a factory creates a car, it's first direction is chosen. The second direction is determined by a pseudorandom number generator. For example, if a car is created traveling northwards it will then randomly choose between East and West, which determines its second direction for the duration of its "live" time. If this car chooses West then this car can now only travel North and West. At each intersection, the car determines whether it continues in the current direction or turns onto its other possible direction based on the pseudorandom number generator. In the previous example, the car would be traveling northward to the first intersection. At this intersection the car would choose between continuing northward or turning left and traveling westward. This routing method prevents the cars from traveling in cycles.

### 3.4 Optimization Algorithm

Genetic algorithms are search heuristics that mimic the mechanisms describe in natural selection. The search is initialized with a completely random group, or population, of individuals that make up the first generation. These individuals are tested against some metric and the individuals that perform best, the "survivors", are chosen to parent the next generation that is to be tested. Each individual in a generation is completely described by a "chromosome." The children of each generation are constructed by performing some kind of crossover of the parent's chromosomes. During this crossover, sections from each parent's chromosome are spliced together, forming an unique individual. Additionally, the a small number of the highest performing individuals not only contribute parts of their chromosomes to the offspring making up the next generation, but also have their chromosomes cloned to the next generation. This ensures that good solutions are not destroyed by the crossover operations.

This project's optimization technique is essentially the same as the genetic algorithm technique proposed in Sanchez et al [2]. The time intervals for the traffic lights were discretized to represent the active green light as a integer in the range [0, 3]. For example, the following chromosome, [00312], means that the north in light is green for two periods, then the west in, east in and the south in lights are green. The duration of the period describes the minimum light time, which is important as lights should not rapidly change. A drawback of this model is that only one cardinal direction

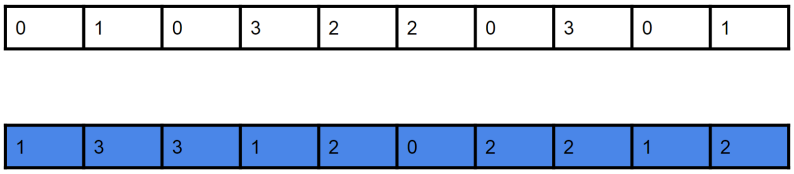


Figure 1: Two parent chromosomes to be crossed over.

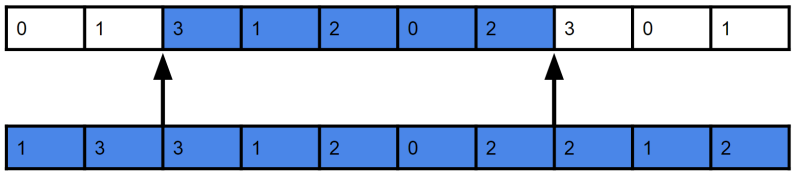


Figure 2: Two point crossover copies information from one parent and overwrites a portion of the other parent's genetic information. Therefore, a new individual is made

can be green at a time. In future work, the chromosome should be modified with a more expressive description space. The total light sequence was represented by a length-64 vector of these values. The input to an individual simulation was the concatenation of the input vectors for each light in the intersection. This codification allowed multiple simulation inputs to be "merged" together in a crossover operation (simulating cell meiosis). The general algorithm for optimizing traffic lights was as follows:

```
Generate P random chromosomes
Run simulation for each chromosome
For each generation:
        Pick the best few individuals
        Perform crossover & mutations
        Clone the best 2 chromosomes to the next generation
        Run simulation for each chromosome
```

### 3.5 Picking the best individuals

The ultimate goal of the optimization was to maximize the average speed of all vehicles over the entire simulation. Therefore, in order to pick the best individuals to survive and pass genes on to the next generation, a subroutine was created to take in an array of Results objects and return the indices of the results with the fastest average speeds. The corresponding parents are placed into a pool from which children are generated. For each child, two parents are randomly selected from the pool (never the same parent) to provide the basis for the child.

### 3.6 Crossover & mutations

With a list of chromosomes from the best-performing simulations from a generation, a way to combine characteristics was needed from these codified simulation inputs. A 2-point crossover technique was used to accomplish this, where a random section of the second parent is used to replace the corresponding values from the first parent. The resultant vector should theoretically have some characteristics of each parent. Figure 1 shows two sample parent chromosomes. Next, Figure 2 shows the two points, from which a part of the bottom chromosome's information is spliced into the top parent's chromosome.

To avoid local maxima, mutations were used in a similar fashion as found in Sanchez et al [2]. This allowed for novel behaviors that may or may not individually be more successful. The rate of mutation started at a predefined value and decreased exponentially, but at a very small rate. Our decay rates ranged from 0.97 to 0.982. Sanchez, et al. also used an elitist cloning strategy implementing within its genetic algorithm. In this system, the best individuals are cloned from one generation to

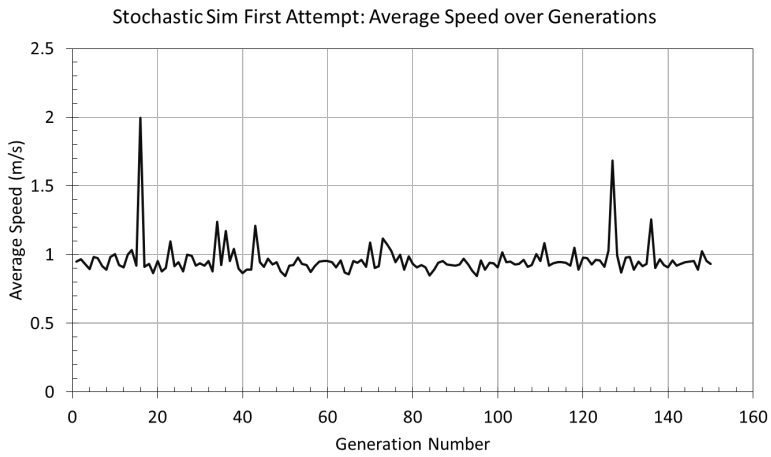


Figure 3: Results from first stochastic simulation over 150 generations. There was no cumulative improvement observed.

the next, replacing children generated by two parents. This project followed suit, and also cloned the two of individuals between generations.

### 3.7 Performance

Each simulation ran for 14,400 iterations, which took approximately 14.5 seconds to execute. Robust optimization requires hundreds or thousands of simulation executions. Therefore, this project's system executed simulations in parallel on multi-core processors. The optimization routine utilized the Executor object in the native Java API, which creates a thread pool and schedules tasks to be mapped to available threads. The executions ran in approximately 3.5 seconds on average on the target machine when scheduled in a batch fashion. The time spent in between simulation was relatively negligible.

## 4 Results

The first attempt at optimization proved unfruitful. The results from the initial run are shown in Figure 3.

Errors were found in the optimization routine that were fixed. The second attempt's results are shown in Figure 4.

After observing poor performance of the genetic algorithm in the stochastic model, the simulation was simplified so that all stochasticity was removed. There were no longer aggressive drivers, only drivers that follow the speed limit and accelerate and brake in an efficient way. Additionally, the cars were no longer allowed to turn onto different streets, limiting them to the road that the factory created them on.

With this newly deterministic model, the genetic algorithm's efficacy significantly improved. The results are shown in Figure 5. These results are very promising!

Another interesting result involved a bug in the simulation. The speed limit of the simulation was set at 16 m/s, approximately 35 mph. One run of the optimization routine ended with average speed in the 15 m/s range! After some investigation using the visualization, it was found that the genetic algorithm was exploiting a bug the simulation's code.

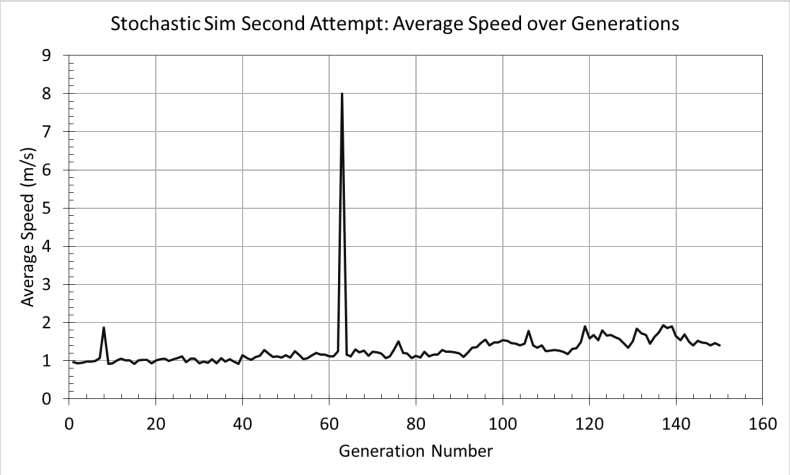


Figure 4: Results from second stochastic simulation over 150 generations. Only minor improvements were seen over all the generations.

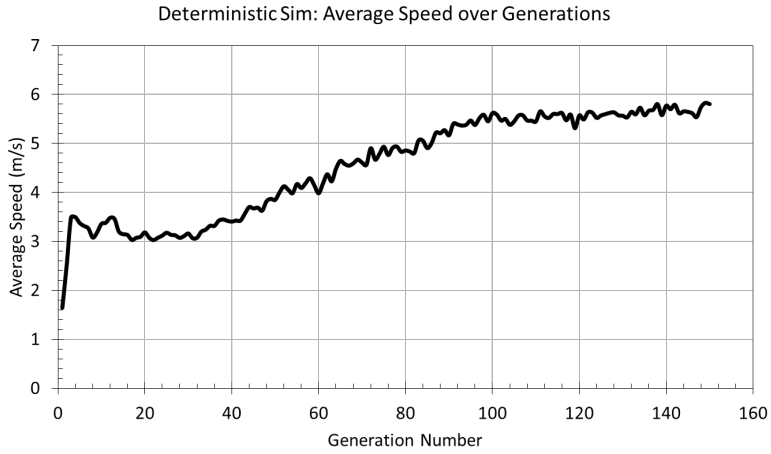


Figure 5: Results from deterministic simulation over 150 generations. The algorithm appears to be approaching the asymptotic limit.

## 5 Analysis

Unfortunately, the optimization technique was ineffective for the stochastic model in which the car could be aggressive or not and their turning behavior was random. These random features changed the model too much from generation to generation for the lights to be optimized in an effective way. When randomness is involved, a system like this would need to have on-line sensors feeding a process control algorithm real-time data so that specific decisions could be made. This just isn't possible for a genetic algorithm to tackle.

In order to accomplish some optimization, the model was simplified. The vehicles were no longer allowed to turn, so the street that they were created on by the factory was the street that they left the map on. Also, all the drivers were law-abiding and conservative. The genetic algorithm was much more capable in these circumstances! All randomness was eliminated, so every run of the simulation had, for the most part, the same cars created at the same times. Certainly, as the performance got better, the throughput of the city streets were able to see more and more cars. However, these changes were slow and only changed every generation, not every individual like the stochastic executions.

In the deterministic model that was optimized the subsequent lights acted in concert. This is one of the most easily observed inefficiencies when driving in a metropolitan inner city. The ability of the genetic algorithm to conduct the intersection to be in concert with one another is even more interesting because there is not mechanism for this to be accomplished in the chromosome data. The genetic algorithm appears to be reaching the asymptotic limit of maximum average speed. However, this result was only a local maximum.

There are certain scenarios that the simulation does not behave. For example, when a road following an intersection becomes backed up to the previous intersection, the cars waiting to get onto the clogged road just start driving over the cars in the queue. This allows for cars just to fly through the traffic unhindered. Because of this, the genetic algorithm found the situations where the lights hardly ever change so the roads stay clogged and cars are allowed to freely go from one end of the map to the other! This is where the close to speed limit average came from! In order to combat this, some debugging was done. However, the underlying problem was not found and the problem still exists. Thankfully, the genetic algorithm doesn't always find the global maximum, and portions of the search space where the bug doesn't occur were searched. A quick fix could be to simply reject the individuals that exhibited close-to-the-speed-limit results.

The results were very interesting and showed some characteristics of the genetic algorithm. The exploitation of the bug was definitely an unexpected result! The fact that it also searched a space that didn't exploit the bug showed that it doesn't always find the global maximum. The simulation performed moderately well, despite this glaring pitfall. Future work will definitely include fixing up the model so that it behaves in all scenarios.

## 6 Conclusion

The simulation's complexity hindered the team's ability to explore much of the goals that were made at the beginning of the project. A single bug in the simulation caused about a week's worth of time to be burned and much heartache. However, even through these hard times success was reached in many ways. The simulation actually runs for the most part, albeit with some edge case bugs. The genetic algorithm, while not performing well in a stochastic world, did perform well in a deterministic one. The ability of the algorithm to synchronize the lights from intersection to intersection is a definite win, and shows the power of the technique. This synchronization was achieved despite the fact that no mechanism is present for the intersections to communicate with one another.

In the future several changes should be made. One change would be to allow the simulation executions to be distributed over a network to allow several machines to aid in the computation of the traffic flow problem. This would greatly speed up the time needed to get results and would also allow for more generations to be run. More generations could potentially find even better results. However, in the successful deterministic run it appeared that the results were reaching an asymptotic limit. Additionally, other methods of genetic crossover exist, and it would be interesting to evaluate their performance as this could change the effectiveness of the genetic algorithm. Tweaking the

parameters of the simulation or the genetic algorithm could also allow the observation of what aspects most affect traffic flow. Making the simulator more deterministic or using a fluid dynamics perspective of traffic flow rather than modeling each individual car could help with the conformity of the results between simulation models.

**Acknowledgments**

We would like to thank Karl Gemayel and Kaushik Subramanian for their invaluable lectures and support during this project!

## References

[1] Heshav Saharia. Traffic optimization system: Using a heuristic algorithm to optimize traffic flow and reduce net energy consumption. *unpublished*, Aug 2011.

[2] Javier J Sanchez, Manuel Galan, and Enrique Rubio. Genetic algorithms and cellular automata: A new architecture for traffic light cycles optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1668–1674. IEEE, 2004.

[3] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Phys. Rev. E*, 62:1805–1824, Aug 2000.