Evolutionary Approach to Traffic Signal Optimization (EvATO)

Benjamin Goldstein b.goldstein@yale.edu
Department of Computer Science
Yale University
New Haven, CT 06520, USA

Luke Mozarsky luke.mozarsky@yale.edu
Department of Statistics & Data Science
Yale University
New Haven, CT 06520, USA

1. Introduction

Traffic congestion is a significant problem in many urban centers around the world. Besides the increased time commuters must wait to get to their destination, vehicle congestion wastes fuel and produces increased emissions, posing an environmental hazard; each year in the U.S., idling wastes an estimated 6 billion gallons of fuel, and personal vehicles alone generate roughly 30 million tons of CO_2 from idling (1). Slow downs and vehicle build-up at intersections are a significant source of urban traffic, and efficiency in signal programming has proven effective in reducing traffic congestion (2). Thus, optimization of traffic signals on a city-wide scale can help reduce both the monetary and temporal costs related to traffic congestion, and are currently a subject of intensive research.

In this work, we report an attempt to optimize the traffic light signals in a simulated urban road network using a simple genetic algorithm. We place a predetermined number of vehicles in a model of New Haven's road network, and we simulate their routes over a given time period. In particular, we minimize the total "time lost" during all vehicle journeys over several parameters pertaining to each traffic light in the network: the cycle time, the partition of green light time in for each direction at an intersection, and the offset in time between cycles of different lights. We provide an overview of SUMO, the network generator used to accurately recreate the New Haven road network, and algorithm implementation details. In doing so, we propose a evolutionary approach to traffic signal optimization, EvATO. Finally, we discuss our results and comment on possible next steps we could take in continuation of this work.

2. Related Work

Significant effort has been expended into algorithmic optimization of traffic signals, with seminal work conducted by the Robertson et. al. under the supervision of the British government in 1969 (3; 4). Iterative improvements on TRANSYT have become the industry

standard and variants are sold as closed source software products by both American and British transportation authorities. These models have been applied in a wide variety of scenarios to improve traffic flow, particularly in urban areas (5; 6). The basic premise of the system is to combine a simulator based traffic model with a traffic signal optimizer. Original approaches used heuristics such as the hill climb algorithm with fixed step size sequences and single parameter optimization sequences, but modern variants utilize advanced techniques such as genetic algorithms and multiperiod methods. TRANSYT uses a macroscopic traffic model that ignores individual vehicles in favor of continuous methods including platoon splitting and queue spillback. TRANSYT faces competition from similar software, but the above methodology remains the state of the art (7).

Techniques that optimize over historical data, such as reinforcement or deep learning methods, typically require many samples to realize a global optimum, which may or may not be easily acquired. Simulation-based methods can overcome the need for large quantities of data, but are often limited in how realistic the generated road network is. As we describe in the following sections, EvATO is a simulation-based approach to optimize traffic signals in a highly realistic, generated road network. Thus, we bypass the need for big-data sources while maintaining the applicability of our results to an optimization framework for a real-life urban traffic grid.

3. Optimization Problem Formulation

Consider a network of roads (edges) with n traffic lights, one at each intersection (node). Traffic light i follows a predetermined pattern defined by a finite ordered set of states S_i, with $(S_i)_k$, state k of traffic light i, being activated for a time $t_{st,i,k}$. Each state is a string of characters 'G' (unyielding green), 'g' (yielding green), 'y' (yellow), and 'r' (red). A naive approach would be to optimize over the space of potential state sets as well as durations. It becomes incredibly difficult to enforce real-world constraints without drastically restricting the state space of a given light, so we neglect this altogether. Instead, we take a signal i's state set S_i to be given and optimize over the durations of each state $t_{st,i,k}$. We only give attention to states with green lights; states with only yellow and red signals are considered to have a fixed activity time t_y (yellow light timing is typically fixed or nearly fixed across a jurisdiction). We decompose the traffic light i's timing into the following components: cycle time, offset time, active time list. The cycle time, $t_{c,i}$, is the total time taken by the light to run through its complete state list, or more precisely $t_{c,i} = \sum_{k=1}^{|S_i|} t_{st,i,k}$. The offset time, $t_{o,i}$, which is defined as the time it takes for light i's first state $(S_i)_1$ to become active in a given configuration of states. The active time list is simply $t_{st,i,k}$, though now subject to the constraints $\sum_k t_{st,i,k} = t_{c,i}$ due to defined cycle time and $t_{st,i,k} = t_y \ \forall \ k$ s.t. $'y' \in (S_i)_k$. We also constrain the active times to a range to maintain realistic behavior, that is $t_{st,min} \leq t_{st,i,k} \leq t_{st,max}$.

In a given time period, m vehicles pass through this network, following shortest-path routes between randomly selected predetermined point pairs. Upon simulation, vehicle j completes its journey in time t_j. However, suppose that the vehicle would take time t'_j to complete its journey if it traveled at the posted speed limits and was not interrupted by slow downs (red lights, traffic). We define the time lost by vehicle j on journey as $t'_j - t_j$, making the total time lost by all vehicles $\tau = \sum_{j=1}^{m}(t'_j - t_j)$.

To summarize, we formulate our optimization problem as follows: given the described network, what is the choice of cycle time, offset time, and green light partition time for each traffic light that minimizes τ, the total time lost? In this framework, total time lost τ is our objective function which we attempt to minimize. Our decision variables are:

- A vector of cycle times $[t_{c,1}, ..., t_{c,n}]$, one time for each traffic light
- A vector of offset times $[t_{o,1}, ..., t_{o,n}]$, one time for each traffic light
- A matrix of active state times, T, of dimension $n \times K$ where K is the maximum number of active states for a given traffic light, across all traffic lights. The rows of T correspond to the traffic lights, and the columns of T correspond to the active time allotted to a given state; thus, the component T_{ik} corresponds to the active time for state $(S_i)_k$—state k belonging to traffic light i—denoted by $t_{st,i,k}$. If for traffic light i we have $|S_i| < K$ (meaning traffic light i has fewer states than the maximum number of states held by any traffic light), zeros appear in row T_i following the last state of light i.

The constraints we impose on our optimization scheme are:

- $t_{c,i} = \sum_{k=1}^{|S_i|} t_{st,i,k}$ (the cycle time for a given light is the sum of time allotted to all states belonging to the light)
- $0 \leq t_{o,i} \leq t_{c,i}$ (the offset time is undefined below zero and can be at most the light's cycle time)
- $t_{st,min} \leq t_{st,i,k} \leq t_{st,max}$ where
  - $t_{st,min} = |\{S_i| \ 'y' \in (S_i)_k \text{ for } k = 1, ..., |S_i|\}| \cdot 6 + |\{S_i| \ 'G' \in (S_i)_k \cup \ 'g' \in (S_i)_k \text{ for } k = 1, ..., |S_i|\}| * 7$, where 6 seconds is the minimum time for yellow states, and 7 seconds is the minimum time for green states. In other words, the minimum state time $t_{st,min}$ is simply (number of yellow states) $\times$ (minimum yellow state time) + (number of green states) $\times$ (minimum green state time)
  - Similarly, $t_{st,min} = |\{S_i| \ 'y' \in (S_i)_k \text{ for } k = 1, ..., |S_i|\}| \cdot 6 + |\{S_i| \ 'G' \in (S_i)_k \cup \ 'g' \in (S_i)_k \text{ for } k = 1, ..., |S_i|\}| * 120$, the same expression as above except we now use the maximum green state time, defined to be 120 seconds (the minimum and maximum yellow state time are the same, at 6 seconds).

4. Methods

Time lost by vehicles in the road network is not readily formulated as any function, let alone a convex or approximately convex one, of the traffic signals' cycle time, offset time, and active time lists. Gradient methods, therefore, are not readily applicable as the objective is not differentiable with respect to the parameters. With inspiration from TRANSYT, we explore the application of a genetic algorithm to traffic signal timing optimization.

4.1 Genetic Algorithm

Genetic algorithms are inspired by principles of biological evolution and natural selection. Chromosomal crossover, genetic mutation, and fitness based selection are all implemented programmatically. Each individual is represented as a genome, usually a linear string of numbers or text similar to the biological analog of a nucleic acid sequence. Fitness is evaluated based on the objective function of optimization. The fittest individuals of a given generation are then selected for crossover. Various forms of programmatic crossover exist, but the simplest is a simple random selection of which parent to inherit each attribute from. Then a probabilistic mutation is applied to the generated genome, before it is then released to be part of the next generation. Mutation schemes are largely domain specific, and are designed to prevent convergence to a local minimum. There is also a feature of genetic algorithms analogous to a skip connection in deep architectures known as elitism, in which the most fit element of each generation is directly inserted into the subsequent one to avoid losing progress.

We apply the genetic algorithm to the traffic light optimization problem as detailed in Section 3 by considering a genome representation that includes the optimization parameters for each traffic light: the cycle time, offset time, and active time list. The offset is stored as a fraction of the cycle time ranging from 0 to 0.2. The cycle time is stored as is, and is constrained to the following range where $n_{s,i,y}$ is defined as $|\{'y' \in s' | s' \in s_i\}|$ and $n_{s,i,g}$ as $|\{'y' \in s' | s' \notin s_i\}|$:

$$t_y * n_{s,i,y} + t_{st,min} * n_{s,i,g} \leq t_{c,i} \leq t_y * n_{s,i,y} + t_{st,max} * n_{s,i,g}$$

We are simply calculating the upper and lower bounds for feasible cycle times based on the signal's state profile as well as the constraints on active times of each state. The active times $t_{a,i,k}$ are stored as a list of unnormalized factors that are later used to generate the active times for green states as follows:

$$t_{st,i,k} = \frac{t_{a,i,k}}{\sum_{k'} t_{a,i,k'}} * (t_{c,i} - n_{s,i,y} * t_y - n_{s,i,g} * t_{st,min}) + t_{st,min}, \quad \forall k \text{ s.t. } 'y' \notin s_k$$

The active times are first normalized, then multiplied by available surplus cycle time (time not taken by yellow states or the minimum green time of green states), then finally added to the minimum green time. This ensures that the active times sum to cycle time and that the genome still allows for sufficient variation in how active time is partitioned between states.

The crossover scheme used by the algorithm randomly selects which parent to select each of the cycle time, offset time, and active time list from based on a random threshold generated per pairing. Element wise selection of active times was considered as opposed to selecting the list as a whole, but it was concluded that the latter method better maintains features that made the parent configurations successful. A mutation of a timing configuration is defined as the a reset of the offset time, cycle time, and/or active time list of a random light to be reset to random values. The initial generation is randomly generated using a random cycle time within the permitted range, a random offset time between 0 and 1, and a list of random entries for active times for each light. The fitness is evaluated using a traffic simulation, which is explained in the next subsection. A link to the implementation, which uses `pyeasyga` to facilitate the genetic algorithm dynamics is included as a footnote.[1]

1. `https://github.com/bengoldstein19/traffic-light-optimization`

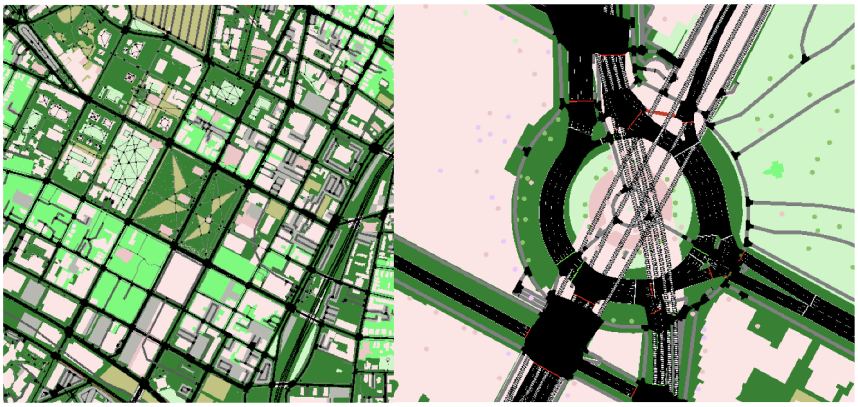

Figure 1: Caption

4.2 Network Simulation with SUMO

To generate the road network on which we simulate traffic patterns and different traffic signal configurations, we utilize the SUMO (**S**imulation of **U**rban **MO**bility), an open source software package used to generate and simulate activity on road networks[2]. While SUMO is a standalone software tool, it is fully integrated into Python with `sumolib`[3], a library of modules to interact with networks and work with simulation output, as well as `traci`[4] (Traffic Control Interface), an API that allows the user to interact with the simulation in real time.

SUMO comes equipped with a Python script, `OsmWebWizard.py`, which interacts with OpenStreetMap[5] to generate simulated networks that resemble road networks in real cities and towns. Figure 1 shows a recreation of the New Haven road network on the left, and a close-up view of Columbus Circle in New York City on the right.

We decided to use the New Haven road network as a testing site for optimization. The network is grid-like, however there are several junctions with more than two roads intersecting. Thus, it will be interesting to compare how our algorithm selects the best signal parameters for these junctions with the optimal parameters it finds for the more typical 2-way intersection.While SUMO allows for the implementation of a host of vehicles (cars, buses, bicycles, etc.) in addition to pedestrian traffic, for simplicity, we choose to only place cars in our network.

The program `OSMWizard.py` requires two parameters to simulate random vehicle routes throughout the generated network: "Through Traffic Factor," and "Count." Through Traffic Factor is how much more likely a car is to begin its route on the edge of the network as opposed to within the network itself; we set this to 10, under the assumption that we are perhaps in the earlier portion of the day and more cars are arriving in downtown New Haven (for, e.g., work) than are leaving. The Count parameter is the number of cars per kilometer per hour we want in our network; we set this to 50. Though the actual number of cars

2. https://sumo.dlr.de/docs/index.html
3. https://pypi.org/project/sumolib/
4. https://pypi.org/project/traci/
5. https://www.openstreetmap.org/

passing through New Haven per kilometer of road per hour may be larger, we found that this parameter value shortens the simulation time drastically while still generating enough traffic throughout the network to see noticeable differences in time lost between different light configurations.

5. Results

The genetic algorithm succeeded in monotonically decreasing the mean and minimum penalty across generations when run on a map that includes Yale's campus and neighboring blocks. The algorithm also demonstrated signs of convergence to a solution, shown by decreasing fitness variance across generations as well. It should be noted that the penalty is not directly taken from time loss, an additional component of 2× the vehicle's time loss is added if SUMO elects to "teleport" due to either total deadlock or another unwanted issue that would break the flow of traffic. The best configurations achieved by the algorithm have no teleports, but this is used to assess an extra penalty against earlier configurations that are unreasonably poor. The algorithm was run for 20 generations with 50 configurations per generation, a crossover probability of 0.5, and a mutation probability of 0.3. The demand generated was 30 vehicles per kilometer per lane per hour, and the simulation was car-only. Results of the algorithm run are shown below: The effects of mutations and diversity within

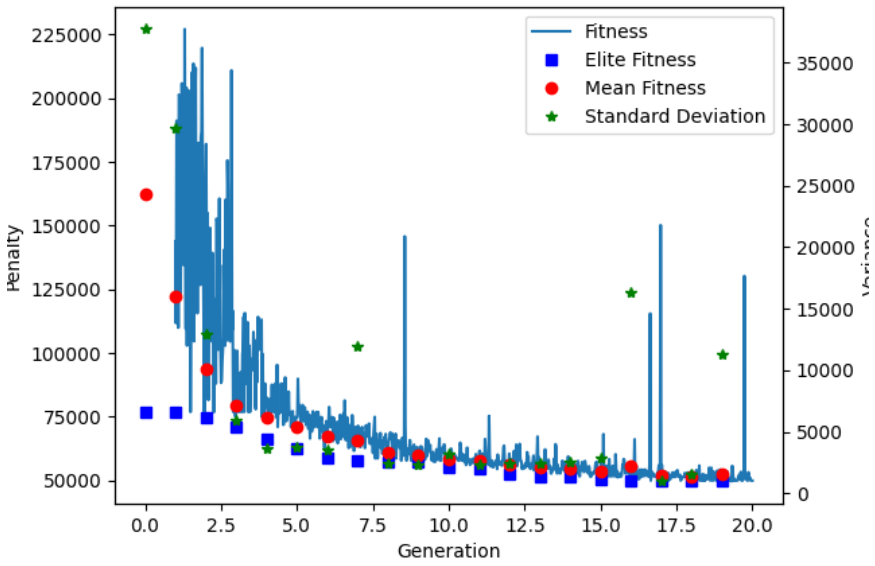

Figure 2: Evolution of best, average, variance, and individual fitness over generations

generations can be seen in the individual-wise fitness curve, which exhibits noise about the mean. It is also evident that the best, mean, and variance of fitness all decline as the algorithm evolves. Anomalies occur with negative mutations, as expected, but this overall trend dominates. The best configuration attained by the algorithm exhibited an average penalty of 290.55 per vehicle (there are 171 vehicles in the simulation). This performs well

relative to the randomly generated benchmark, which is 950.79 per vehicle taken from the mean of the first generation. The OSM generated network of this map uses real-life signal information. In busy areas, including Yale's campus, New Haven uses mostly actuated signals. These signals use sensor information from both vehicles and pedestrians to adaptively adjust durations within a range. The implementation of an actuated signal system is often an expensive undertaking that is unfeasible for many jurisdictions. The real-life actuated signal configuration

6. Conclusion

In this work, we demonstrated the capability of optimizing a static light traffic network with a genetic algorithm to a signal configuration that reduces time lost en route far more than a random signal configuration. We used only three traffic light parameters—cycle time, offset time, and the partition of active time between green states—to achieve this degree of optimization. Our model is fully general and applicable to any road network.For a simplistic model that moderately resembles a realistic road network, we are satisfied with our results.

As mentioned in section 3, one parameter we could have optimized over was the actual states of each traffic light. In our model, we take the states of each traffic light as given and optimize the distribution of active time among these states. One can imagine an optimization scheme that actually changes the green, yellow and red signals in a given state. This framework, however, introduces a significant degree of complexity. Modified states must be "allowed" in the sense that they follow the typical rules of the road; one cannot simply construct a state by a random mixture of green, yellow and red signals, as, in general, a randomly constructed state will not be "allowed" and will lead to traffic accidents. (For example, a state where all lights are green would clearly not be realistic, however this state may be given to a light if states are generated randomly.) Since each traffic light junction can have a different number of intersecting lanes, a general rule or algorithm for generating "allowed" states would need to be developed, which is a complicated task. Allowing for the modification of states also vastly increases the dimensionality of the problem, making it more difficult to implement an optimization algorithm that converges to a minimum, much less the global minimum.

In future work, we could attempt to devise such a method of altering traffic light states to improve upon existing state sets, as mentioned. We could also try to optimize a network that uses actuated lights, which would necessitate treating the minimum and maximum duration of each traffic light state as a decision variable. It is also possible to add new sources of traffic such as pedestrians and bicycles, although in doing so we would introduce new parameters into the network that would require pauses (new states with all red signals) in light cycles. Nevertheless, we produced a generalized framework that is useful optimizing simple yet realistic road networks and can be built upon for more complex simulations in subsequent research.

References

[1] U.S. Department of Energy, Office of Energy Efficiency & Renewable Energy. Idling

reduction for personal vehicles. may 2015.

[2] Yang Wang, Huizhen Zhang, Hongtao Yuan, Youqing Chen, Wenlong Yu, Cheng Wang, Jing Wang, and Yueer Gao. Traffic light optimization based on modified webster function. *Journal of Advanced Transportation*, aug 2021. doi: 10.1155/2021/3328202.

[3] D. I. Robertson. Transyt: A traffic network study tool. 1969. URL `https://api.semanticscholar.org/CorpusID:106626896`.

[4] D. I. Robertson. Esso energy award lecture, 1982: Coordinating traffic signals to reduce fuel consumption. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 387(1792):1–19, 1983. ISSN 00804630. URL `http://www.jstor.org/stable/2397455`.

[5] Rodrigo Fernandez, Eduardo Valenzuela, Federico Casanello, and Carola Jorquera. Evolution of the transyt model in a developing country. *Transportation Research Part A: Policy and Practice*, 40:386–398, 06 2006. doi: 10.1016/j.tra.2005.08.008.

[6] E. Almasri. Signal coordination for saving energy and reducing congestion using transyt-7f model and its application in gaza city. *Natural Resources*, 5:282–292, 2014. doi: 10.4236/nr.2014.57026.

[7] Chris Kennett. Understanding linsig in the real world. 2017.