This paper presents a new perspective on traffic flow control by describing the mathematical models and methodology required to optimize the timing of traffic lights.

The model is designed to decrease the number of times a vehicle must waste energy at an intersection by stopping, idling, then reaccelerating. The system takes into account the motion of both idling and approaching cars at an intersection, using it to generate a “congestion index”. The congestion index determines the amount of congestion a traffic light is causing relative to other traffic lights at the intersection, and uses the ratio to determine the time a light should be green.

The algorithm to calculate the congestion index is determined through a heuristic optimization process. Constants in the equation cannot be defined due to the random and uneven nature of traffic flow, so a guess-and-check method was implemented to allow the system to autonomously ascertain the optimal values.

This proposed “green technology” will revolutionize the outdated transportation infrastructure. Problems caused by the rising number of cars on the road, along with rising economic dependence on gasoline, can be dampened through the implementation of a smarter traffic control system.

Introduction

The United States uses roughly 378 million gallons of gasoline a day. Because of this enormous figure, even the smallest decreases in gasoline consumption can reap considerable energy savings. The Traffic Optimization System (TOS) discussed in this paper is designed to accomplish this: small reductions in net vehicular gasoline consumption that accumulate into large long-term gasoline conservation.

Vehicles use significantly more energy to accelerate than to maintain a certain speed. This means that control peripherals like traffic lights cause energy to be wasted, by forcing vehicles to come to a halt and then accelerate back to their original speed. If the number of decelerations and accelerations can be minimized, net energy wastage at traffic lights can be significantly reduced.

Reduction in total acceleration can be accomplished by applying concepts of flow theory to the movement of traffic. Using heuristics, a methodology can be developed for a Traffic Optimization System (TOS) to “learn” how to enable and disable traffic flows at the right time. The algorithm is called a heuristic optimization process because it autonomously alters certain constants in the algorithm until it reaches a maximum efficiency level, in which energy wasted at a traffic light is minimized.

The algorithm functions by assigning a congestion index to each flow (each traffic light or pair of traffic lights that control unidirectional movement). The congestion index is a function determined by the position and speed of cars in a flow. Traffic lights controlling flows with large congestion indexes will be “enabled” (green light) for longer than flows with smaller congestion indexes. This new approach to traffic flow theory is designed to emphasize movement of heavier flows without disregarding lighter ones.

The implementation of such a system would be relatively inexpensive. This paper details an implementation that uses image analysis to determine the speed and position of vehicles while they are in the proximity of an intersection. This proposed “green technology” can revolutionize the outdated transportation infrastructure in the United States, as well as around the world. Problems caused by the rising number of cars on the road, along with rising economic dependence on gasoline, can be
dampened through the implementation of a smarter traffic control system.

**Project Goals**
The goal of this project is the design and construction of an algorithm that can produce the optimal timings for traffic lights at an intersection for the minimization of energy wastage.

This is done through

(1) **Improving the net efficiency of vehicles** by reducing the number of decelerations and accelerations caused by traffic flow controllers.

(2) **Reducing traffic congestion** by minimizing the number of cars that must stop.

(3) **Building an algorithm on an easily implemented platform** to allow traffic control model to be easily implemented into existing traffic controllers.

**Conventional Techniques**
To address the issue of traffic congestion, several methods have already been implemented into existing traffic infrastructure.

**Green Waves**
“Green waves” allow smoother flow of traffic over long distances by coordinating the timings of several traffic lights. A driver on the green wave will see successive green lights, timed so that they do not have to stop. This means less energy is wasted through braking and re-accelerating, while minimizing congestion. However, this method is based on pre-set timings, meaning that a single slow driver can bring down the entire wave.

**Sensor-operated lanes**
Many lanes have a sensor that only turns the light to green if there is a car in the lane. However, these sensors, usually inductive loops, are prone to human error, where a driver who does not pull forward fast enough will cause the sensor to read an empty lane, turning off the signal and ending the flow prematurely. This causes frustration in drivers, wastage of time, and overall inefficiency.

**Traffic Simulation**
Virtual simulation can be used with empirical data gathered from traffic intersections to determine the optimal timing of traffic lights. Traffic data can be collected through a vehicle detection system and analyzed through a computer simulation, which uses statistical analysis to determine the distribution of cars at certain times of the day, and times the traffic lights to deal most effectively with it. However, this method is expensive to implement due to hardware requirements, and cannot effectively deal with random variance in traffic flow.

**Heuristics**
The algorithm will define a “congestion index” for each traffic light, which is a function of the location, density, and movement of traffic over time. The congestion index is an easily-comparable factor of the amount of congestion the traffic light is causing by halting movement of traffic. The congestion index is used to calculate the amount of time that a traffic light must be “enabled” (green) or “disabled” (red). The function to determine congestion index is determined through a heuristic optimization algorithm, which alters certain variables in an equation using a guess-and-check method until a most-desirable outcome is reached. The algorithm will take into account the motion of approaching cars, as well as the presence of idling cars at the intersection, and use readings from sensor inputs to determine an ideal timing for the traffic light.

**Congestion Index**
To assign a congestion index (or “ranking”, denoted by $r$) to a stoplight, several factors must be considered:

$r_i(t,c)$ A congestion index based on the number of cars idling and how long they have been idling at the intersection.

$r_d(c,v,x)$ A congestion index based on the number of cars that are approaching, their velocity, and
their distance from the intersection.

This function for \( r(c,t) \) gives the sum of the time that the cars in a queue have been idling, multiplied by a factor \( k \). As the cars idle for longer, the congestion index increases, and for every additional car that joins the waiting queue, it increases more. The constant \( k \) is determined through heuristic optimization, which is essentially a guess-and-check method.

The function below is the equation for the congestion index caused by approaching cars, or \( r(c,v,x) \). This function takes into account the velocity, position, and mass of each approaching car. The constant \( k_a \) is tested by the same guess-and-check method to find the ideal value. The variables \( m_n, v_n, \) and \( x_n \) give the mass, speed, and position of each car approaching the intersection. The constant \( k_m \) is equal to the average weight of a vehicle on the road (about 1600 kg). This means that larger vehicles like buses and trucks (that require more energy to accelerate) will increase the congestion index faster than smaller cars because \( m_n/k_m \) will be greater than 1. The function being summed approaches the value \( j \) as the velocity and position approach 0. When velocity reaches zero, the car is added to the waiting queue and its congestion index is given by the idling congestion index function shown above.

\[
r = k_i \sum_{n=0}^{c} t_{n,i} + k_a \sum_{n=0}^{c} \frac{v_n \cdot m_n}{k_m x_n + \frac{v_n}{j}}
\]

This equation gives the total congestion index for a flow at an intersection. The equation provides the framework for a program that can automatically calculate the congestion index by compounding data from various sensor inputs. The values of the experimental constants (\( k_i \) and \( k_j \)) are tested heuristically, where a guess-and-check method will ascertain their optimal values.

This heuristic optimization process is shown in 4.2, where testing is performed in a virtual environment.

### 3.3 Timing

The congestion index is used to describe the “flows” at an intersection. A flow is any path or device used to move objects from one place to another. Each flow controller (traffic light) is equipped with sensors to collect traffic data and use it to compute a congestion index. Each flow’s congestion index will determine the time for which the flow is enabled by taking the ratio between the flow’s congestion index and the total congestion index of the intersection. This ratio, a number ranging from 0 to 1, will determine the fraction of the total time allotted to each flow. Flows with a larger congestion index will be allotted more time, while flows with a smaller congestion index will receive less time, but will not be neglected altogether. This system is more efficient than the conventional traffic light timing system because it allots more time to heavier flows without neglecting lighter ones. The total time for one cycle of flows to complete is a logistic function of the total congestion index. As \( r \) increases, \( t \) increases until it reaches \( t_{\text{max}} \), a constant. The total time allotted for each cycle is rationally distributed between flows. This logistic model
allows lights to be quickly cycling during periods of low congestion, while remaining enabled for longer during periods of high congestion.

**Hardware Implementation**
The implementation of this methodology requires hardware that can sense the movement and position of cars. This means that the conventional system of inductance loops to sense vehicle presence will be unable to provide the information that such a system would need.

A cheap and inexpensive hardware implementation would be through the use of image analysis, where a camera placed on top of a stoplight can sense moving objects and use successive images to ascertain the approximate speed of the vehicle. The diagram above shows a sample calculation of velocity, using only two measured quantities (distance and time) and a known constant (elevation, $y$). Cameras can be placed in high elevation areas such as the top of street lights or traffic signals. These cameras would provide the real-time sensor input to the algorithm for determining the congestion index.

**Virtual Testing**
Since the algorithm could not be tested in a real traffic environment, a virtual simulator was created that attempts to replicate the distribution, movement, and randomness of vehicle flow as closely as possible. The equation constants, average vehicle speed, and other quantitative data were measured during the heuristic optimization process.

**Environment**
The virtual model was of a square of four Intersection objects connected by roads that are modeled by Flow objects. The model allows the distribution of cars at an intersection as well as the distribution between intersections to be measured. To generate this model, a Java program was created where each intersection is an array of queues, which represent queues for oncoming traffic. The roads are also modeled by queues that pass cars between intersections over periods of time. Certain “rules” were built into the code:

- All cars are of equal and constant weight.
- Cars uniformly accelerate and decelerate until they are a certain distance behind another car or traveling at a certain speed, whichever comes first.
- Cars begin accelerating immediately after a light turns green, and begin decelerating at a set distance from the intersection.
- An intersection controller can only gather data from cars that are within a certain distance of the intersection. The intersection can measure velocity and position.
- There is a 2% chance that a car will randomly decrease its speed, and a 1% chance that a car will “disappear” (exit road).
- Cars are randomly being added to and removed from the flows leading into the simulator. The density of cars thus changes randomly, simulating a real traffic environment.
- Cars randomly choose the direction they want to go at an intersection.

**Java Objects**
The Java simulation has several classes that are used in conjunction with each other to model traffic flow through an intersection.

**Flow.java**
A Flow object takes objects from an input class, then allows it to be accessed by an output class after a certain amount of time has passed. It is essentially a queue that is adding and removing a value for every interval of time. An Object that has a `getFlowTime()` method can set the time it spends in the Flow.

**FlowController.java**
The FlowController class extends the Flow object, but releases objects from its queue after a time designated by a function, $f(t)$. This allows local Object functions to control the time they spend in the queue.
**CarQueue.java** A CarQueue is a FlowController that releases every object in its queue after a `releaseAll()` method is called. The objects are released according to a preset timing pattern.

**Intersection.java** An Intersection takes objects from Flow(s) and adds them to CarQueue(s) in the Intersection object. The Intersection object has a local function that calculates the congestion index for each CarQueue and uses these indexes to calculate when to enable and disable the `releaseAll()` method. When a CarQueue is enabled (the virtual light turns green), the Objects in the CarQueue are passed to different Flows so they can be sent to other Intersections.

Intersections calculate congestion index once every time pulse. The simulator carries out the actions of each object, all of which have a function that is performed during every time pulse. The simulation's time runs faster than normal time, to allow extensive testing without large time requirements. Cars are randomly added to the outer eight flows, and cars leaving the outer flows are replaced by a new car randomly placed in the system.

The algorithm used to calculate congestion index has two variables which are determined through a heuristic optimization process. The index is stored in every Intersection object. The heuristic optimization process is shown below.

The diagram above shows how the virtual simulator is structured. Twelve Flow objects send virtual cars between the four intersections.

The process is essentially guess-and-check: If the experimental variable increases the average vehicle speed (vehicles have to stop less), then it becomes the new base value from which the next experimental values are chosen. Then, another set of trials is conducted, and the process is repeated until a most-desirable outcome is reached. During this outcome, neither increasing nor decreasing the experimental variables will cause efficiency to increase, thus locking the loop at a constant variable value. This loop will also allow intersections to adjust for environmental constants, thus making each optimization algorithm unique to the intersection it is optimizing.
Basic Simulator Framework
The simulator works in a recurrent three step process:
1. Cars put in and taken out of the simulator.
2. Position and speed of cars analyzed by Intersection objects, and appropriate action is taken.
3. The time variable is incremented, and the process is repeated.
The simulator essentially performs all the movements and actions of the simulated traffic environment, increments time, moves the cars in accordance with the time increment, and so on.

Traffic Movement
The simulator performs car movements by analyzing the car’s position, velocity, and proximity to an intersection.

```java
public static void moveCar(Car c) {
    Intersection i;
    Flow tempflow;
    boolean direction;
    if (FLOW_MAX - c.position() <= DECEL) {
        direction = c.direction();
        c.decel(DECEL_RATE);
        tempflow = c.flow();
        i = tempflow.intersection(direction);
        Intersection.add(c, tempflow);
        tempflow.remove(c);
    } else {
        c.move(MOVE_RATE);
    }
}
```

The sample segment of code above shows the moveCar() function, which processes all the Car objects in the simulation and moves them to their respective new location. Functions like this analyze vehicle locations and make the appropriate movements and modifications.

Every time pulse, the car is moved by a small distance, until it reaches the proximity of an intersection. When it reaches an intersection, it is added to the queue of the Intersection object. After being queued, the Intersection uses the position and velocity information of the Car object to analyze congestion indexes and time the release of the queues accordingly.

Results
The results of the simulation were analyzed, and are shown below.

Graph 1. The average velocity of vehicles increased as the algorithm finds optimal traffic conditions.

Graph 2. The experimental constants of the algorithm reached optimal values over the course of many trials. The idling constant (shown in blue) reaches an optimal value through the heuristic process, while the approach constant (shown in red) appears to have not yet found an optimal value.

As the algorithm found optimal values, the experimental constants used by the Intersection objects to move cars facilitated a greater average vehicle velocity.
Data Analysis
The data was plotted, and an approximate regression was found ($r^2 = 0.583$).

$$v_{av} = 63.93k_i - 1.98k_a + 7.67$$

This linear regression plane shows the correlation between the experimental constants and the average vehicle velocity. Due to the relatively large $r^2$ value and the trend of the data points, there is a correlation between the experimental constants and the resulting traffic flow.

Implications
The optimization of traffic flow will address several issues.

- **Wasted opportunity cost**- the time that people spend idling at stoplights is time that could be put towards productive purposes.
- **Wasted energy**- less gasoline consumption translates to less air pollution, reduced carbon dioxide emissions, and a cleaner environment.
- **Wear and tear on vehicles**- frequent accelerations and decelerations cause vehicles to gradually wear down over time.
- **Stress and road rage**- motorists are more likely to feel stress and road rage in congested traffic conditions.
- **Emergency response**- an easily-manipulated traffic control infrastructure can facilitate faster movement of emergency vehicles through dense traffic.

Conclusion
Through extensive testing and redesigning, an algorithm was developed for the automated optimization of traffic light infrastructure. The algorithm has the ability to reduce net energy waste: however, I envision more advanced algorithms being developed in the future that can increase efficiency even further. Also, a more thorough virtual simulator that takes into account the numerous other factors that affect traffic flow may give a more comprehensive method of analyzing a heuristic optimization process. However, the results of the algorithm show that the Traffic Optimization System will save energy, decrease traffic congestion, and open a new field of research in the optimization of traffic flow.


