End to End Delay Optimization for Traffic Signal Control using a Distributed Model-Based Algorithm

Taylor A. Kulp

\textsuperscript{1}Hood College, Department of Computer Science, Frederick, MD 21701 USA

Because of the increasing prevalence of urban traffic congestion, and the associated costs of wasted time and fuel, this paper presents an algorithm for minimizing the delay experienced by individual commuters as the result of traffic lights. Many current traffic control algorithms are based on isolated intersections, with no central control and hence no possibility for end to end optimizations. This is partly due to the high computing load incurred by network level modeling of traffic congestion. The algorithm proposed by this paper should defray the additional computing cost associated with network scale modeling by using a parallel computing approach. Each commuter will be separately modeled as a parallel process, going through an iterative process of computing total path delay, voting at each light for a “green” at the car’s expected time of arrival, and repeating as each traffic light computes a more optimized pattern for stop and go signaling. The process will terminate once the average change in delay between iterations falls below a predefined threshold. This approach is expected to be scalable for small to medium sized urban traffic networks because of the parallelism, but is anticipated to break down at higher traffic volumes due to the exponential nature of the graph search for determining each light’s signaling pattern as the increasing maximum delays increase the time over which the stop and go pattern must be optimized.

Index Terms — Road traffic control, distributed control, simulation, parallel computing

I. INTRODUCTION

The cost of traffic congestion in terms of time lost to commuters and wasted fuel is considerable, with current estimates in the European Union estimating losses approaching €80 billion [1]. The ability to more effectively manage the use of traffic signals is a key element in reducing the delay experienced by each commuter, and hence saving time and wasted petroleum.

The most common metric used to compute the effects of traffic congestion in recent years has been the delay on each commuter, measured either as an average, or as a marginal cost incurred by changes to the traffic conditions as described in [2]. The goal of any traffic management system using this metric is to reduce the average delay by minimizing the marginal delay for each commuter in the system. Although this delay might be reduced by a number of means ranging from increasing throughput on the roadways to providing route information to commuters to try to manipulate traffic patterns, the mechanism considered in this proposal is the control of traffic signals, commonly referred to as “stoplights”.

II. BACKGROUND

Although many algorithms have been proposed for handling traffic control systems, the strategies generally fall under one of two categories: Isolated intersection strategies or Network level strategies.

Measuring Network Delay and Congestion

1) Link-by-Link

The simplest way to model congestion is by calculating the queue saturation at each intersection. This mechanism can very quickly provide an estimate of the total number of congested links in the system, but cannot describe the spillover of congestion from one link to another as the inability of one intersection to service traffic causes other nearby intersections to fail [2].

2) START

Strategic and Regional Transport (START) is an algorithm developed by the British government to model traffic patterns. Described in detail in [1] and [2], this mechanism ignores congestion effects at the link level, and instead computes average delays based on a number of usage patterns. This can be more useful than modeling congestion in many cases because it handles average delays rather than transient phenomenon.

Isolated Intersection Strategies

Isolate intersection strategies rely on each traffic signal making decisions for its intersection independently of the system as a whole. As described in [1], this mechanism has the advantage of being easily scaled, does not require coordinated monitoring of traffic, and is, in general, the mechanism most commonly used in cities today.

1) Fixed time approaches

The simplest approach for handling traffic, and the only mechanism that works without any awareness of current traffic levels, is based on estimating the approximate load on each side of an intersection at varying times of the day, and setting the time cycle for each light to optimize servicing each traffic queue. This mechanism is widely used, but is primitive and cannot adapt to changing traffic patterns.

2) Load-based approaches

In this mechanism, each intersection gauges the amount of traffic currently queued up, and tries to change its own timing to better reflect current traffic patterns, as summarized in [1]. Unfortunately, with no communication between intersections, this method can only react to traffic congestion; it cannot reduce delays across the entire system.
Network Level Strategies

1) AI models
A number of intelligent algorithms attempt to handle the problem of traffic flow, as summarized in [3]. The advantage of the intelligent algorithms, especially reinforcement learning and neural network approaches, are their ability to both react to current traffic patterns while maintaining a hysteresis of common traffic phenomenon. These algorithms are generally applied at the traffic signal level however, and as a result they are still reactive to existing conditions, and do not predict or proactively adjust to incoming traffic.

2) Store-and-Forward
A long standing network level approach to optimizing throughput of any type of network is described in [4]. This “store-and-forward” technique helps to approximate traffic flow as a continuous, rather than discrete, system [1], and as a result allows faster modeling of traffic phenomenon over very large scale networks. Unfortunately, this tends to optimize throughput rather than delay, and queues may begin to back up as congestion occurs without careful end to end rate matching.

3) Multi-Agent simulations
The most promising parallel algorithms currently in development relate to simulating each car as its own entity, and treating them as agents in simulations, rather than flows at specific junctures. As described in [5], each car can be modeled as an entity requesting time to pass through a set of intersections on its route. Although this approach has been applied to systems of intelligent and coordinated cars, this paper expands on that methodology to apply the method to a series of unknown paths for each car (rather than planned paths), and optimize for the least average delay.

III. PROBLEM DEFINITION AND CONSTRAINTS
To provide a reasonable simulation environment to test the presented algorithm, a traffic grid based on the city of Fredrick shall be modeled using available traffic data [6]. The simulation allowed a moderate amount of traffic to be modeled using real world conditions. The objective of the simulation was to show that overall delays per user could be reduced by applying the algorithm described in this paper.
In order to operate in real world conditions, the algorithm was constrained by several factors.

1) Mutually Exclusive Traffic Flow
Not all vehicle paths can be accommodated at the same time. All possible paths that can simultaneously be supported for each of the four traffic signal states are shown below in fig. 1.

Fig. 2 Illustration of all possible traffic signal states
Note that the ability to “turn on red” is not handled in this simulation, although the structure of the algorithm does not preclude the addition of this state at some point in the future. In order to set the best possible traffic signal pattern, the signals are first set into an impossible pattern, in which, at every second, the light is set to the configuration that lets the most cars through.

2) Finite Computational Time
Because this algorithm is intended to be applied frequently in order to constantly update the scheduling of traffic signals as the number of vehicles change, the algorithm was required to run within a finite time to meet real time guarantees. This means that the algorithm’s optimization is, of necessity, a “best effort”, and only seeks to improve existing conditions, not necessarily guarantee the best possible configuration of traffic signals.

3) Discrete Traffic Signal Switching
The traffic lights cannot be switched instantaneously; there is an imposed delay of \( n \) seconds between states of the light to allow time for the light to turn yellow and alert incoming traffic. It is assumed that traffic will continue to flow at normal rate as long as the signal is set to given state, but that state can only be changed once every \( n \) seconds.

IV. ALGORITHM AND METHODOLOGY
The developed algorithm for optimizing traffic signaling consists of a simple convergence algorithm summarized in fig. 2 below.
The algorithm is essentially a convergent optimization of the traffic light pattern by repeated simulation of entering traffic. At fixed intervals of time, all of the cars on the road are assigned a random path leading them out of the traffic grid of interest. Each car will then compute the time required for it to complete this path with all lights on the graph initialized to green, and use this as a baseline for best possible time. An iterative process will then begin wherein each car computes the time that it will arrive at a given light with the current configuration, and “votes” for that light to be green at that time.

After all the cars have voted, each traffic light reschedules its green light and red light timings to maximize the number of positive votes. Then the algorithm repeats, recalculating both signal times and the delay experienced by each car. Once the average change of delay for each car falls below a given threshold between iterations, the algorithm terminates, and the resulting traffic signal pattern is returned.

1) Initialization and Path Selection
The first stage of the algorithm is to establish the paths for all of the vehicles to be accounted for the present traffic light scheduling period. It is assumed that the current placement of all of the vehicles within the traffic grid is already known. A path is then guessed for each vehicle, based on the most popular end destinations. This is accomplished by looking at the historical traffic levels at each intersection the vehicle passes through, and choosing the direction through random selection weighted by the amount of traffic known to have previously made the decision. Vehicles will not be allowed to enter the same link between intersections more than once.

Once a path has been guessed for each vehicle, the best case time for each path is computed in parallel. This computation is done assuming that the vehicles never encounters stop signal, and is simply the sum of the length of the path divided by the speed limits on each link of the path. All subsequent calculations will use that value as the baseline for the lowest possible delay. While the delay of each path is being computed, the time and direction at which the vehicle encounters each intersection shall be recorded.

Finally, the state of the traffic signals shall be initialized by a parallel process using the timing information provided by each vehicle. Every traffic light in the system shall determine its first pass state based on the times and direction of each vehicle passing through the system. This is done by looking at how many cars arrived at that intersection at this point in time, and then setting the configuration of the signal to the one that would let the most cars move. After this first guess is made, the constraint of requiring that lights not shift more than once every n seconds shall be applied, and a search algorithm will attempt to find the best “delay cost” for every node as described below.

2) Iterative Optimization and Metric Computation
Once the initial states are known, the system cycles between recomputing the times at which each car arrives at each intersection and then adapting times of the traffic signals to meet the total demands. Each step is done separately, necessarily completing and hitting a thread block before progressing to the next iteration.

The iterative computation of the timing for each vehicle is separated into a task for each vehicle. The same per link timings computed during initializations are used, but every time that an intersection for which the state at the time of arrival does not permit travel, an additional delay is added. The timings recorded are the times for arrival the intersection, but the time for arrival at the next intersection will be delayed by the time spent waiting for a favorable signal.

After the timings are collected, every intersection can compute, in parallel, the best fit for the new timing demands. This is done by running a hill climbing search algorithm against the “delay cost” for each vehicle. This metric is computed by adding the sum number of seconds a given configuration will cause the inputted requests to wait. Once this metric is minimized locally, these timings are used for the next iteration.

3) Evaluation
Between each iteration, a simple check is done summing the total delay of all the cars in the system by folding the delay results from the latest vehicle path calculation with the previous run. Once the change in delay between each iteration falls below a threshold, the algorithm will cease.

Then, the path delays are computed one final time, but this time with a different set of random paths, but using the computed optimal traffic light pattern. This should represent a real world scenario in which the paths of cars do not line up with perfectly with the guesses. The results of this second computation can then be compared with delays observed using other traffic signal configurations.

V. IMPLEMENTATION
Because of the increasing prevalence of GPU based computing clusters for low cost, large scale computing, the algorithm was implemented using NVIDIA’s Compute Unified Device Architecture (CUDA). For this paper, CUDA is assumed to always run on a GPU, although other implementations exist. This incurred several design constraints when implementing the software.

1) Kernel Design
For the two phase iterative process, which operate using a completely different set of steps, two separate kernels had to be written. When one kernel had completely finished its wave
of implementation for all given threads, the CPU would then signal the next kernel to begin, completing the iteration.

2) Memory Access
Transferring memory is a huge overhead task for the GPU, so the loading of the paths and the traffic signal arrays was done only once. Each of these arrays includes static allocation for each thread to operate on so that no conflicts between threads would arise. When the timings are to be written for arrival at each intersection, the vehicles index in that intersection array could be populated without needing to access any memory shared by another thread. By passing the same pointers to both kernels, the same memory allocation could be reused between iterations.

VI. RESULTS
It is expected that the model will converge reasonably well and find near optimal timings for traffic light signals for the paths suggested. It is expected that because the actual paths of the cars will differ from the randomly generated ones, that on a very small scale the optimizations will be inaccurate, but it is assumed that as the scale increases, a normal distribution of chosen paths will cause the predictions and reality to converge. This will result in the calculated optimizations being both useful and predictive of actual traffic patterns.

It is expected that speedup will not be linear because of the high overhead between iterations in which all of the votes placed by each agent must be tabulated. However, since both the path computation and voting section, and the individual light configuration stages can be parallelized, it is expected that significant speed up will be observed as more nodes are added.

VII. CONCLUSION
The proposed algorithm for optimizing traffic flow should be scalable for larger traffic networks, but is not recommended for very large scale optimizations because of the less than linear speed up anticipated and the complexity of the problem. It is also not recommended for very small networks in which chance events disrupt statistical patterns.

However, this proposed algorithm should provide a meaningful improvement to the quality of traffic management, and will help avoid both congestion and low throughput issues by minimizing the delay not only of the system, but also of each individual commuter. Parallel hardware will be well utilized to handle the heavy simulation load for traffic scenarios of thousands of vehicles, and will provide the ability to accurately predict the best signaling patterns for all involved vehicles.

VIII. REFERENCES