Traffic optimization using a novel traffic controller featuring distributed agents and forecasting

Damian M. Voorhout

Computer science department, Delft University of Technology, Netherlands

Using some sort of adaptive traffic light control system is becoming standard policy among metropolitan areas. However, controlling traffic lights efficiently on a city-wide scale is computationally intensive and theoretically complex. This paper aims to show a proof of concept of an efficient and modular traffic light controller with comparatively little computational overhead. The proposed system features distributed agents, each representing an intersection, capable of making individual decisions. These agents base their decisions on short term traffic flow forecasting and received information from neighbouring agents on incoming traffic. Testing shows the proposed controller being more efficient than less adaptive systems in terms of reduced average vehicle time loss and reduced average vehicle stop time. This paper describes the attractive properties of the system in detail, shows the shortcomings of the design choices and gives suggestions on how to improve the system in the future.

| traffic light controller | traffic flow | traffic optimization | adaptive | forecasting | distributed | ARIMA | SUMO |

I Introduction

Traffic, specifically traffic congestion, negatively impacts several societal aspects such as worker productivity, economic growth, commuter health and safety, travel reliability and the environment (1, 2). Traffic intersections with traffic lights are a way to alleviate some of the congestion as they give a great amount of control over the flow of traffic. Vast amounts of research has already taken place in optimizing traffic intersections with the use of advanced traffic controllers. More and more cities are starting to employ such systems (3–6). *Studer & Ketabdari (2015)* (7) show that existing adaptive traffic light controllers are already showing significant improvement over more static alternatives, with an up to 40% reduction of travel times. However, many of these controllers have weak points such as high computational requirements and a dependence on vehicle detectors. The aim of this paper is not to show off the most efficient traffic light controller, instead it aims to show a proof of concept of a controller that tries to fix the mentioned weak points of existing controllers while still outperforming more basic models in terms of efficiency. The main goal of this research is to find out whether this could be achieved using an agent based system and forecasting. These are the main aspects this paper focuses on.

Section II explains terms used in popular literature related to traffic light control. Section III shows some of the existing literature and controllers. Furthermore, it highlights the advantages and disadvantages of those controllers and serves as an explanation for the design choices of the proposed controller in section IV. Those design choices try to solve the shortcomings of existing controllers while iterating on their strong features.
In order to test the controller's efficiency, multiple test results were obtained which can be found in section VI, section V explains the method used to create the controller and how the test results were obtained. Finally, section VII covers the interpretation of the results section, gives a conclusion based on the interpretations and gives suggestions on what future work could be done on the proposed controller.

II Definitions

Before we take a look at some existing traffic control systems and the proposed traffic controller, some definitions are in order. Most of the upcoming explanations are from *Warberg, Larsen & Jørgensen (2008)* (8) and *Mathew (2014)* (9).

- *Lost time / time loss*: The difference in time between a trip where a vehicle can drive the speed limit at all times and a trip where the vehicle needs to brake and wait for any external factors such as other vehicles, intersections and traffic jams.
- *Stop time*: Denotes the time a vehicle stands still.
- *Controller efficiency / traffic flow optimization*: The efficiency of a traffic light controller, or the optimization of traffic flow, can have different meanings depending on the context. The most common metric to measure is the average time loss (sometimes also referred to as

the total delay) of all traffic participants in the network. Others include total average trip time, total average stop time and maximum stop time or some combination of those. The total average time loss, total average stop time and maximum stop time will be considered in this paper. Also, when no specific metric is mentioned when discussing optimization or efficiency, the minimization of those three metrics is implicitly meant.

- *Cycle*: All traffic lights on an intersection follow a cycle. A cycle is complete when a certain traffic light turns green, yellow and red and eventually green again. Note that not all traffic lights need to turn green during a cycle.

- *Cycle time*: The time it takes for a cycle to complete.

- *Phase*: A phase corresponds to a particular state of the red, yellow and green lights of the traffic lights on an intersection. For instance, on an intersection where vehicles can only go straight, there are only four phases that can take place: GRGR, YRYR, RGRG, and RYRY, where each letter stands for the color of the traffic light on a lane and the first letter corresponds to the top lane. The order of the letters follows a clockwise scheme. So a phase where the vertical lanes have green light while the horizontal lanes have red light can be denoted with GRGR. We then say the vertical lanes are experiencing a green phase while the horizontal lanes are experiencing a red phase, even though it is just one phase that is taking place.

- *Skipping phases*: Iterating on the example of the previous item in this list about phases, the simple intersection has four phases: GRGR, YRYR, RGRG, and RYRY. A traditional controller simply execute those phases in order each cycle. However, more unconventional controllers may skip a phase.
  When a phase is skipped, one of the green phases is not executed. If we assume the controller is currently in the GRGR phase and it decides switching to RGRG is inefficient, it may skip that phase entirely. It does not switch to RGRG so YRYR does not need to be executed either, and neither does RYRY as the controller does not need to switch back to GRGR as it is already in that phase.
  So when we talk about skipping a phase we always implicitly refer to a green phase. Skipping a phase ultimately results in the skipping of three phases, one green phase and two yellow phases.

- *Phase length*: Denotes the length of a particular phase.

- *Green time*: Denotes the duration of a green light one or multiple lanes get assigned during a cycle.

- *Yellow time*: Denotes the duration of a yellow light one or multiple lanes get assigned during a cycle.

- *Intergreen time*: Denotes for a given phase all time a vehicle could be moving through the intersection, but is not. This includes yellow time, driver's reaction time to the green light and vehicles getting up to speed. It can also be described as the time lost by switching phases.

- *Artery*: The dominant path most of the traffic follows in a traffic network. This can either be one way, both ways or alternating (morning commute, evening commute etc.).

- *Green wave*: A green wave occurs when traffic along the artery is given precedence over other traffic flows. Traffic along this dominant path will experience cascading green lights and can therefore continue their journey without having to stop. Doing so leads to reduced waiting times and congestion for the masses. This often disadvantages traffic participants on non-dominant paths.

- *Platoon*: Denotes a large group of vehicles.

- *Lag*: Time interval in the context of statistics.

III Background

When designing a traffic light controller many design choices need to be taken into account. This section will go over some of these choices, the rationale behind choosing one option over the other, and some existing systems which showcase some of those features. The aim is to give a better understanding of the choices that were made for the proposed controller, which can be found in section IV.

Static vs adaptive. When a traffic light controller is considered static, it means the controller does not dynamically adapt to external factors such as an increase in traffic. This does not mean the controller runs the same cycle with the same phase times all the time. It can for example change its timings during rush hour. This is however set to happen during a specific time interval.
The strength of these controllers lie in their simplicity. They require little to no computing power and are easily understood. They also don't rely on detectors and other tools outside of the junction. Static controllers are however often less efficient as their adaptive counterparts since they can't adapt to ever changing traffic situations, both *Gershenson & Rosenblueth (2012)* (10) and *Gu et al.(2012)* (11) show these

findings.

Adaptive controllers react and adapt to the incoming traffic. By measuring or predicting traffic at incoming lanes the traffic light controller can decide to change phases or phase lengths. The obvious upside is the increase in control and flexibility over static controllers which can lead to an increase in efficiency (10, 11). However, they are often more complicated and computationally intensive.

Distributed vs centralized. Coordination of traffic lights can be achieved either via a distributed system, where each agent acts on its own merit but can communicate with other agents, or via a centralized system where one server determines all actions of the network. Centralized systems have an overview of the whole system and can make informed decisions based on this information. Distributed systems often have to make decisions based on information gained from nearby nodes and usually make more short term decisions. Optimal short term decisions can due to a domino effect have negative effects in the long run (12). However, it has been shown that a generalized form of traffic light control, namely optimizing queuing network control, is considered an EXP-complete problem, assuming P != NP (13). This means the computational complexity increases exponentially as the network size grows.

Tubaishat et al. (2012) (14) proposes a distributed system with individual agents. These agents consist of control agents, which each control an intersection, and communication agents, which relay information between agents. The communication agents supervise a small number of control agents and relay information between them and other control agents. This way no single agent needs to supervise the whole network, decreasing computational complexity dramatically while still maintaining coordination between individual agents.

Green wave modeling. When a controller employs green wave modeling, it tries to enforce a green wave in some way. Many systems have been proposed that aim to optimize traffic flow by doing so. It has a number of advantages over more static and traditional controller techniques. Some advantages are described by *Gartner and Stamatiadis (2002)* (15):

- Overall reduction in congestion because of reduced delay for the majority of traffic.
- Overall less stops by vehicles, which in turn reduces vehicle emission because of cars needing to get back up to speed.
- Vehicle speeds are more uniform, there is no incentive to speed up, this only causes the driver to arrive at a red light.
- Because of the more uniform speed and less overall stops, the likelihood of accidents will decrease.

So the advantages of giving the masses preferential treatment are vast. Some examples of controllers which incorporate some form of green wave modeling are RHODES (16), DOGS (17) and Phase-by-Phase (PP) (18).

RHODES is a triple layered system where the highest layer is concerned with the macroscopic scale and keeps track of changes in the whole network. The middle layer, the mesoscopic layer, keeps track of large platoons going through the artery of the network. The lowest layer, the microscopic layer, keeps track of individual vehicles, increasing the flexibility of the controller, giving it the option to give preferential treatment to vehicles such as firetrucks and police cars. RHODES is able to predict incoming individual vehicles on a lane in the range of 20-40 seconds based on measurements of nearby intersections. It can predict platoons up to 150 seconds ahead of arrival. Because of the complex implementation of RHODES, large amounts of computing power are required.

DOGS uses a combination of static and adaptive techniques to find a middle ground between computational complexity and accuracy. When traffic load increases DOGS increases the throughput of the arterial paths. During low traffic periods, DOGS switches to a static algorithm with predetermined cycle times and paths. It uses no forecasting methods to predict traffic but instead uses measurements only. This limits the flexibility of the controller as it relies solely on detectors. When a detector fails, the system is updating or maintenance is performed on a detector, DOGS cannot switch to an adaptive system.

PP tries to solve some of the issues of other controllers such as fixed cycle lengths or fixed increments of step variation of cycle length. It also does not have fixed coordination between intersections about green wave timings. It tries to track individual vehicles with great accuracy and does so per junction. Junctions are separate agents who do not communicate with other agents and receive no information from a centralized system. PP follows a set order of phases.

Forecasting vs Measuring. Traffic flow can be measured with the use of detectors. These consist of for example cameras or magnetic strips that respond to changes in pressure exerted on the road. Measuring cars is an easy and accurate way of predicting short term incoming traffic. This can

simply be done by placing a detector some distance in front of the location in question.
Forecasting requires a model with parameters and data, but does come with a number of benefits over just using measurements to predict traffic flow. Below some of those advantages are listed, do note that at least for forecasting short term traffic flow, the data the model runs on will be collected with live measurements. So forecasting does still make use of detectors. For example, this means traffic forecasts for the next minute are based on measurements of the past 10 minutes.

- Relying solely on measurements can be impractical as we saw with the DOGS traffic controller: If a measuring device fails or maintenance is performed, the controller will in the best case have to fall back to a static model. An adaptive model that uses forecasting can still rely on earlier forecasts, or make new forecasts based on historic data.
- When relying solely on measurements, the question arises of where to place the measuring device. If the measuring device is not placed far back enough on the lane, it can happen that traffic backs up beyond the measurement device. This leads to inaccurate data being provided to the controller. If the device is placed too far back it can happen that it misses other sources of vehicles, such as intersections with no traffic lights. This also leads to inaccurate data. The only solution would be to place multiple measuring devices per lane per intersection, quickly increasing complexity and becoming impractical.
- Once a phase has been assigned it's not efficient to suddenly switch to a different phase, this could lead to very short phase times and therefor cost a lot of time in terms of overhead. By predicting traffic and scheduling phases ahead of time, no pre-emption of a phase has to take place when a bad decision has been made, assuming the predictions are accurate. This is related to the second point in this list as there is a limit to how far a measuring device can be placed from the traffic lights and therefor a limit to the information the controller has when it makes a decision. A decision that makes sense the moment it is made, can become nonsensical moments later as for example more traffic appears on the lane that was just given a red light than the lane that was given green light.

IV Design choices

In this section the proposed traffic light controller and the rationale behind it are explained. The main goal of the controller is to fix some of the weak points of existing controllers such as high computational requirements and a reliance on vehicle detectors. Other requirements for the proposed controller include simplicity, flexibility and modularity while still being efficient. The design choices were made with those ideas in mind.

Overview. Each intersection acts as an individual agent and makes its own decisions. It has full control over the phase assignment and phase time. During the initialization of a new intersection, the agent will be told by a centralized system which of its lanes it shares with which neighbouring agents and the time it takes to reach that intersection given the allowed speed of that lane. Each agent is able to send messages to neighbouring agents informing them of incoming traffic it just let through its intersection. The intersection which received information will use that to get an accurate description of the incoming traffic of that lane. For the lanes that are not directly connected to other agents, forecasting is used to predict incoming traffic.
The controller is able to switch between two different modes, reactive and predictive. The default mode is predictive, but the controller can only switch to predictive when it has gathered enough data to make forecasts. Also, when traffic volume is low predicting vehicles is difficult, being off by one can have a much larger impact when traffic volume is low versus when it is high. Therefor the controller always starts out in reactive mode and also switches to reactive mode when traffic volume falls below a certain threshold.
Algorithm 1 shows roughly how the reactive mode works in the context of a simple intersection where vehicles are only allowed to go straight. In this mode no forecasting is performed and information from neighbouring agents is ignored, instead the controller relies solely on measurements to make decisions. It simply looks at which lanes currently contain the most cars and gives that lane a certain amount of green time in order for all cars on that lane to pass the intersection. In section III it was stated that relying completely on measurements can be troublesome. However, this controller only does so during specific times and can always fall back to its predictive mode if it deems necessary. Since not much more can be said about the reactive mode, from now on when the controller is mentioned, we implicitly refer to its predictive mode.

As the literature showed, some form of green wave modeling is beneficial for overall traffic efficiency. The proposed model does not aim to enforce this explicitly but rather implicitly by favouring lanes that have a large influx of traffic. If every agent acts according to this logic a platoon will form, which will enforce the upcoming traffic lights to act

Algorithm 1: Reactive mode

```
function queueNewDecision():
    if ($totalIncoming + totalWaiting < 7$ or $size(totalMeasurements) < 10$) and $size(decisionQueue) == 0$ then
        if $switchingGreenPhase$ then
            /* switching phases, find out which phase we are in and append the relevant yellow phase to the queue */
            $yellowPhase = newYellowPhase(currentPhase)$;
            $decisionQueue.append(YellowPhase)$;
        else
            /* we stay in the same green phase, no need to add a yellow phase */
        end
        if $verticalWaiting > horizontalWaiting$ then
            /* create a green phase with length based on the amount of waiting vehicles and append it to the queue */
            $verticalGreenPhase = newGreenPhase(verticalWaiting)$;
            $decisionQueue.append(verticalGreenPhase)$;
        else
            ...    /* queue up horizontal green phase */
        end
    else
        ...    /* go into predictive mode */
    end
```

as a green wave. Most existing models model this explicitly by giving the artery lanes a minimum amount of green time and increasing this duration according to the incoming load. Often this is coupled with a set timing, which can be off due to the nature of traffic. This makes it so the green wave is enforced, but also makes the system less flexible. When for example a sudden large influx of vehicles takes place on a lane that is not considered the artery, those vehicles will not be given priority.

Originally, this system also tried to tackle a shortcoming of some existing models; the predetermined order of phases. Most existing models like the PP model are very adaptive, they have great predictive abilities, are able to change cycle length and phase lengths. They do however still follow a set order of phases each cycle and don't skip phases, even though this may lead to an increase in efficiency when a lane with little incoming traffic is not granted green time for a cycle. However, skipping phases needs to be done with care and can have the opposite desired effect when done wrong. The proposed controller skips phases when there are no vehicles currently waiting in that lane and there are no vehicles predicted to arrive in the relevant time span. It also skips phases that are allotted less time than a certain threshold. For the sake of simplicity any other type of phase skipping was left out of this prototype. Skipping cycles is discussed in more detail in section VII.

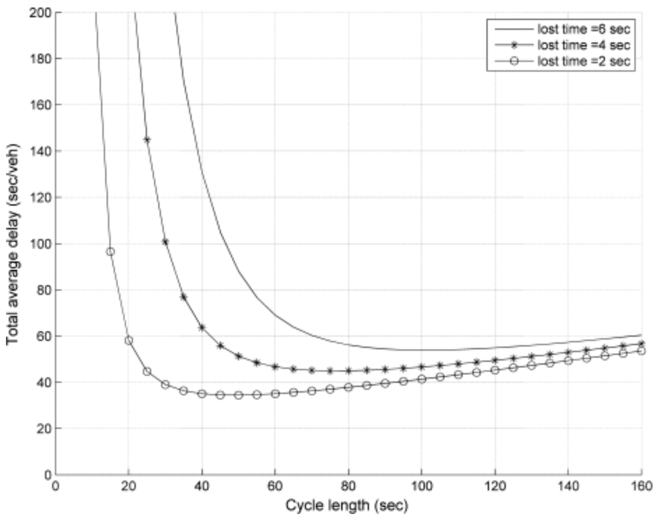

Fig. 1. Cycle length versus total average delay. Image from *Ahmed Y. Zakariya & Sherif I. Rabia (2016)* (19).

Details. During predictive mode the controller has to decide on the order and length of the phases. It does so by looking at the amount of incoming vehicles and currently waiting vehicles. It can track currently waiting vehicle by using two detectors per lane. one detector is placed some distance before the traffic light, the other at the beginning of the outgoing lane. When a vehicle passes the first detector it is added to a queue. When a vehicle passes the outgoing detector, it is removed from the queue. Each lane has its own queue, this way the controller is able to accurately track what lanes should get priority.

Weights. When precedence is given to a large influx of vehicles, the number of waiting vehicles on the non-artery lanes will start to increase until their numbers are greater than the incoming vehicles along the artery. This can make for high wait times. In order to ensure fairness, instead of just taking into account the number of waiting vehicles, the weight of each vehicle is considered. When a vehicle just showed up to a stoplight it has a weight of one, the longer it is forced to wait the more its weight grows. This is done in a linear fashion where a constant is added to the a vehicle's weight during each simulated time step, the constant is based on the cycle length:

$$w_n = w_o + \frac{1}{C_o}$$

where:

- w_n: The new weight of a particular vehicle.
- w_o: The previous weight of a particular vehicle.
- C_o: The optimum cycle time that is used by the controller.

It should be noted that this formula has no mathematical foundation as fairness is subjective and cannot be expressed in mathematical fashion. The formula and its parameters were simply chosen to find a balance between the maximum wait time of a vehicle and overall average time loss in the network.

Cycle time. In order for the controller to determine the phase lengths, it first needs to know the cycle time and divide that according to the demand, where lanes with more incoming and waiting vehicles get more green time. If the cycle time is too long a vehicle which has just missed the green light may have to wait a long time, causing delay. If the cycle time is too short, too much time gets wasted in terms of yellow time and lost time (Fig. 1) (19). The optimal cycle time can be found using Webster's equation (20):

$$C_o = \frac{1.5L + 5}{1 - \sum(V/s)}$$

Where:

- C_o: Optimum cycle length in seconds.
- L: Sum of all intergreen times.
- V: The expected amount of traffic on a particular lane in number of vehicles per second.
- s: The maximum amount of traffic on a particular lane in number of vehicles per second.

Taking V as the maximum amount of expected traffic gives us a cycle time of approximately 61.5 seconds. This is rounded down to 60 seconds for simplicity. The values that were used in this formula are found in the simulation environment that was used to create the controller, more on the simulation environment can be found in section V.

Phase order & importance factor. The order of the phases is determined by something called the importance factor:

$$I = \frac{i_{gp} + w_{gp}}{i_t + w_t}$$

Where:

- I: The importance factor.
- i_{gp}: The total amount of vehicles that are incoming on the relevant lanes of a particular green phase.
- w_{gp}: The total weights of the queues on relevant lanes of a particular green phase.
- i_t: The total amount of incoming vehicles on all lanes.
- w_t: The total weight of all queues.

So for example, let's again assume an intersection where vehicles are not allowed to turn right or left. This intersection has two green phases, namely the green phase for the vertical lanes when the horizontal lanes have a red light, and the green phase for the horizontal lanes, when the vertical lanes experience a red light. In order to determine how important it is that for example the vertical lanes get assigned a green phase, the controller looks at the incoming vehicles and the weights of the vertical lanes and compares that value to the total incoming vehicles and weights. The phase with the highest importance factor gets assigned first, then the one with the next highest importance factor etc.

Phase length. As was said earlier, the controller determines the phase lengths by dividing up the cycle length among the phases according to priority and demand. The controller assigns green light to the lane with the highest importance factor first. Then with the cycle time that remains it assigns green time to the lane with the second highest importance factor etc. The amount of green time each lane gets is based on demand and can be found with the following formula:

$$G = (lq + i_{gp}) * f + yt + bt$$

Where:

- G: Green time in seconds.

- lq: The maximum length of all queues in question. For example on an intersection where vehicles are only allowed to cross the intersection and not turn left or right, the relevant queues could be the two horizontal queues. When the green time is determined according to the longest queue, the shorter queue will by definition also be granted enough green time to clear out.
- i_{gp}: The total amount of vehicles that are incoming on the relevant lanes of a particular green phase.
- f: A constant that represents how fast vehicles are able to clear out when they have reached the speed limit, in seconds.
- yt: Time of the yellow light in seconds, only taken into account when phase switching takes place.
- bt: Time needed for the drivers to react to the green light and get up to speed, in seconds.

This formula makes sure each lane gets allotted green time according to its demand. If the allotted time for a lane falls below a threshold, that phase gets skipped. This threshold is just to make sure a phase gets enough time to make at least a single vehicle cross the intersection before the lights turn yellow again. As was said before, any more complicated phase skipping techniques are left out of this prototype.

Forecasting. To forecast incoming vehicles the controller makes use of an autoregressive integrated moving average (ARIMA) model. An ARIMA model is made up of three parts; an autoregressive model (AR), the order of integration (I) and a moving average model (MA).
The autoregressive model indicates that at time time t the values X_t of the model are based on a linear weighted combination of values at previous times:

$$X_{t+1} = \sum_{i=0}^{p} \phi_i X_{t-i} + \epsilon_t$$

where:

- X_{t+1}: The value at time $t+1$.
- p: The order of the autoregressive model, determines how many previous values are taken into account to forecast the value at time $t+1$.
- ϕ_i: The weight that is assigned to a value at time i.
- ϵ_t: A random value at time t called white noise. White noise samples are a sequence of random uncorrelated variables with zero mean and finite variance.

The moving average model is similar to the autoregressive model in the sense that it makes forecasts based on a linear combination of previous values. However, instead of relying on the previous values themselves, the moving average model relies on the error of the previous forecasts compared to the actual value:

$$X_{t+1} = \sum_{i=1}^{q} \phi_i \epsilon_{t-i} + \epsilon_t + \mu$$

where:

- X_{t+1}: The value at time $t+1$.
- q: The order of the moving average model, determines how many previous values are taken into account to forecast the value at time $t+1$.
- ϕ_i: The weight that is assigned to a value at time i.
- ϵ_t: The white noise error term. It describes the error of the forecast at time t compared to the actual value at time t. These terms are represented as white noise.
- μ: The mean of the data series.

Finally, the order of integration value reports the minimum number of differences required for the data series to become stationary. Differencing a series gives instead of two neighbouring values n_t and n_{t+1} the difference between those values. A series is considered stationary when its statistical properties are no longer dependent on time. This often means the series doesn't contain a clear seasonal trend.

The ARIMA model can be described in terms of its non-negative parameters p, d and q which represent the orders of the autoregressive, integrated and moving average indicators respectively. For example, an ARIMA(1,2,0) model is similar to a twice differenced first order autoregressive model.
The quality of a model can be described, among other methods, by the Akaike information criterion (AIC):

$$AIC = 2k - 2ln(\hat{L})$$

where:

- AIC: The Akaike information criterion.
- k: The total value of the parameters in the model. For example, ARIMA(1,2,0) has a k of three.
- $\hat{L}$: The maximum value of likelihood function of the constructed model. Simply put, it estimates the probability that the given data is represented by the constructed model and its parameters.

The AIC tries to find a balance between the simplicity of the constructed model and the accuracy of the model. A lower AIC is better.
Hamed et al. (1995) (21) showed that ARIMA(0,1,1) is able to accurately predict traffic flow for a one minute horizon. Furthermore, *Williams et al. (2003)* (22) showed that even though traffic flow can be considered a non-linear time series, and ARIMA is a linear model, it does well in predicting short term traffic data.

The controller makes forecasts one lag ahead, so for the upcoming cycle, which in this case is 60 seconds. It makes use of auto ARIMA to construct a new model for every forecasted lag and does so for every lane that requires forecasting. Auto ARIMA is a way to brute force the three parameters of an ARIMA model which minimize its AIC. While this does require some computing power, sticking to a single model for each lane is not an option. In a short time frame such as 60 seconds, models change continually, the controller needs to be able to adapt to those changes.

V Method

This section describes the steps and tools that were required to create the controller, run simulations and gather meaningful test results.

SUMO. Simulation of Urban MObility (SUMO) "is an open source, highly portable, microscopic and continuous road traffic simulation package" (23). SUMO was used to create and test controller, the controller itself is written in Python. SUMO offers the ability to retrieve and sometimes alter values such as the speed of vehicles, the gap between vehicles, total average delay , total average trip time, total average stop time and maximum stop time. The TRaCI library that comes with SUMO was used to control the traffic lights during simulations. TRaCI offers the ability to switch between phases, set phase lengths and implement completely new cycles during simulation.

Scenario. To test the proposed controller a simple custom scenario was built with the help of the SUMO net editor. The scenario consists of two intersections located next to each other in a horizontal fashion. On the intersection vehicles are only allowed to go straight, figure 2 shows one of the intersections. Each lane includes ingoing and outgoing detectors, even the lanes that are connected to another intersection so the controller can keep track of waiting vehicles on that lane.

Data. The data that was used to generate the simulated vehicles was obtained from the public database of Highways England (24), a government-owned company that handles

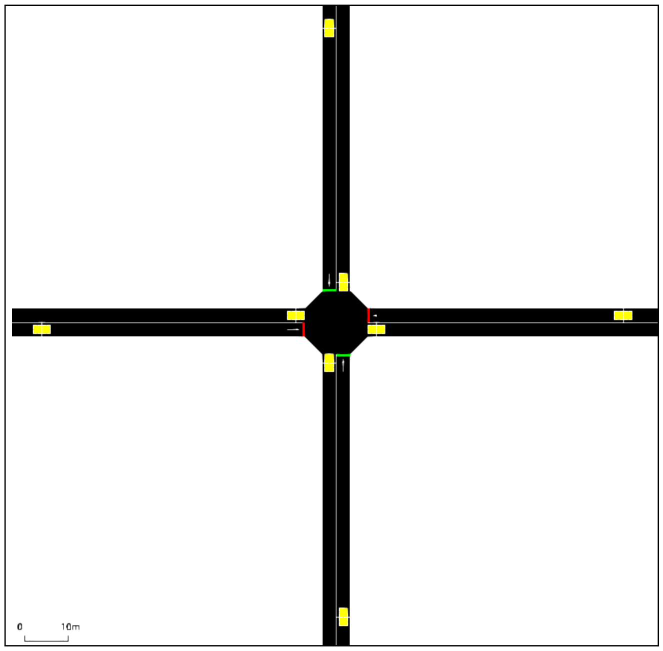

Fig. 2. Intersection controlled by agent. Shows one of the intersections of the custom scenario, including its ingoing and outgoing detectors. The controller is depicting a GRGR phase.

the operation and maintenance of many England roads.
The England Highway data aggregates its data over 15 minute intervals. Simulating a week exactly according to this data would take $15 * 60 * 4 * 24 * 7 = 604800$ time steps if each second is simulated. Because of the way the controller works this would take a significant amount of time to simulate. In order to reduce the simulation time the average amount of recorded vehicles each second is calculated from the data and used to generate vehicles for one minute intervals. This cuts down simulation time by a factor of 15 while staying true to the original data. One of the goals of this controller was to implicitly create some sort of green wave model. This can only be done when there is an artery among the network. To simulate this two different data sources were used with similar total amount of traffic volume on average, one for the vertical lanes and one for the horizontal lanes. The total amount of vehicles on the vertical lanes is then reduced by a factor. This simulates an artery along the horizontal path as there will be more vehicles on the horizontal path compared to the vertical path. The vehicles on the vertical and horizontal lanes cannot both be simulated by the same data source, otherwise incoming traffic would be too predictable, as the vertical vehicle stream is just a certain fraction of the horizontal vehicle stream.

Testing. The proposed controller is pitted against two other models, a static fair model and a static optimized

model. These were chosen as they are relatively simple to implement and are also representative of many existing traditional controllers that are currently in use.
The fair model simply rotates phases and distributes green time evenly. As was stated in section IV, the optimal cycle length is approximately 60 seconds, both competing models will make use of this. For the static fair model this means that each green phase plus its previous yellow phase lasts 30 seconds.
The static optimized model also rotates phases deterministically with a cycle time of 60 second, but distributes green time according to demand. In this custom scenario where the horizontal path is an artery, that path will experience significantly more green time than the vertical paths. The ratio between the horizontal green time and the vertical green time is set at the beginning of the simulation and is based on the average of all the traffic data that is used to generate the vehicles. In a real life scenario this ratio would be determined from historical data from the intersection in question, or a similar intersection.

Each test involves 36000 time steps which roughly equates to seven simulated days. The metrics of relevance are the total average time loss, total average stop time and maximum wait time. The three controllers will be tested for those three metrics in three different settings; low, average and high congestion, which feature approximately 5000, 7800 and 10300 vehicles during the simulation respectively. This is to assess how models function in different traffic situations and gives a more complete understanding of the model's behaviour. The ratio between the volume of vehicles on non-artery and artery lanes is approximately 1:4.

VI Results

In this section the acquired results from testing the proposed controller against other models are shown. As the proposed controller depends heavily on forecasts and information supplied by neighbours to make decisions, this section also includes subsections on the accuracy of those information streams.

Accuracy of forecasts. Figure 3 shows the controller's measurements and forecasts of a horizontal lane that is not connected to another traffic light intersection. At every lag the controller forecasted one lag ahead, where each lag is 60 seconds in the simulation, note that 60 seconds within the simulation does not equate to 60 seconds in real life, as was discussed in section V. The figure also shows how the controller didn't make any forecasts for the first 10 lags as it didn't have enough historical data to base predictions on. The controller is able to make fairly accurate predictions about the

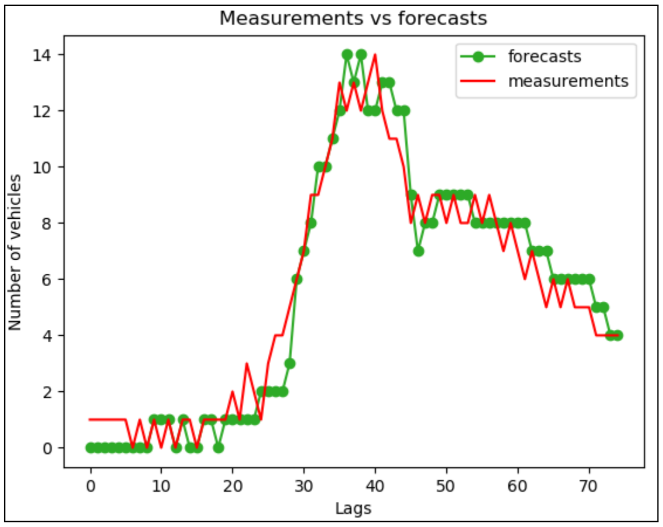

Fig. 3. Measurements versus forecasts. Each lag denotes 60 seconds.

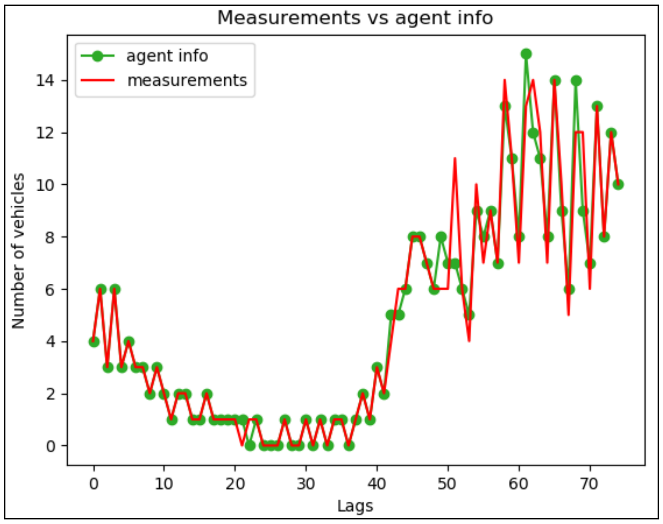

Fig. 4. Measurements versus agent supplied information. Each lag denotes 60 seconds.

amount of cars that are incoming, except for when there is a sudden large drop or spike in traffic volume.

Accuracy of agent to agent communication. Figure 4 shows the information the agent received from the neighbouring agent versus the actual measurements that were done by the agent itself. This information is more accurate than the forecasts the controller makes. However, there are still slight inaccuracies.

Simulation results. Figure 5 shows how much time the controller spent in either predictive or reactive mode. As can be expected the higher the congestion, the less time the controller spends in reactive mode. During high congestion the roads are so congested the controller rarely switches to the reactive mode at all.

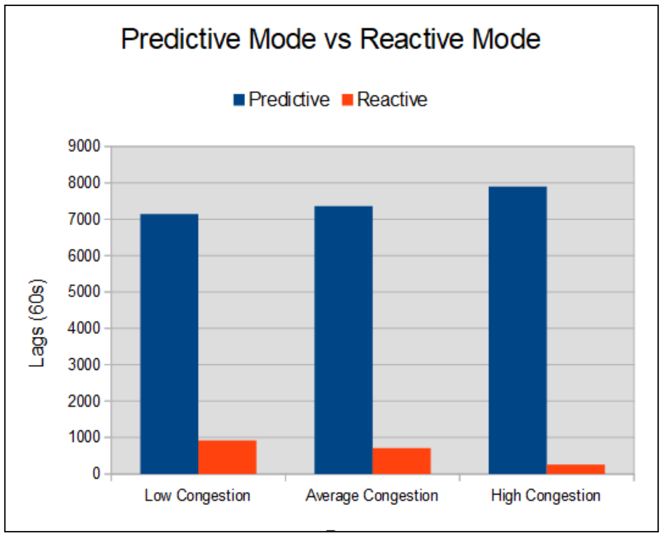

Fig. 5. Time spent in Reactive mode versus Predictive mode. Shows how many lags, which are 60 seconds each, was spent by the controller in either reactive mode or predictive mode.

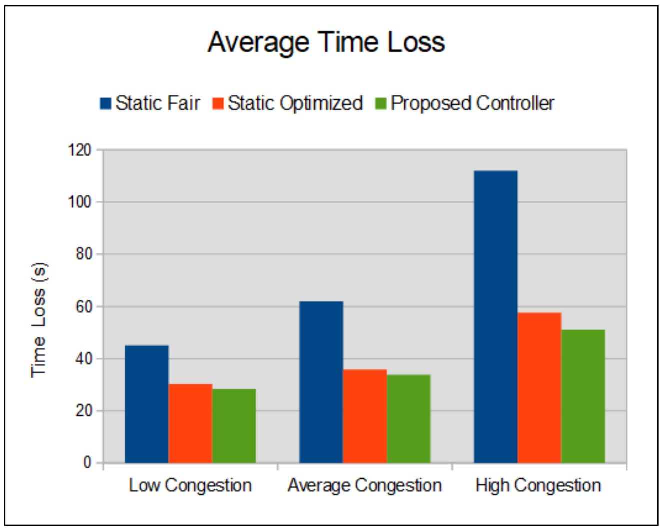

Fig. 6. Average time loss. Shows the average time loss of a vehicle from each model during low, average and high congestion. Lower values are better.

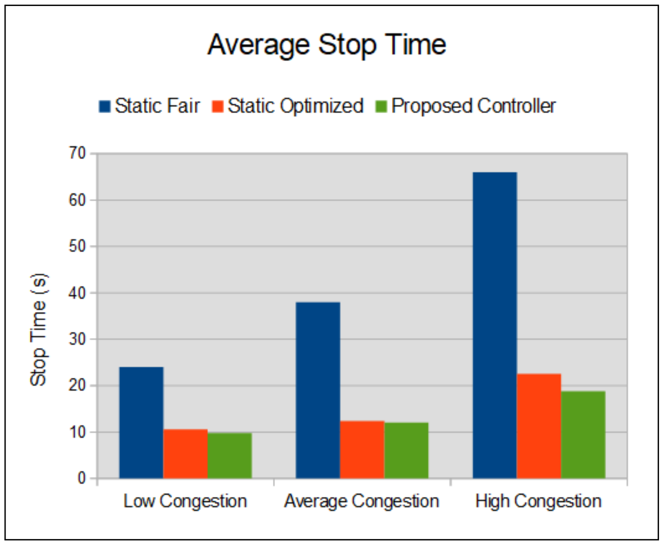

Fig. 7. Measurements versus agent supplied information. Shows the average time vehicles stand still in each model during low, average and high congestion. Lower values are better.

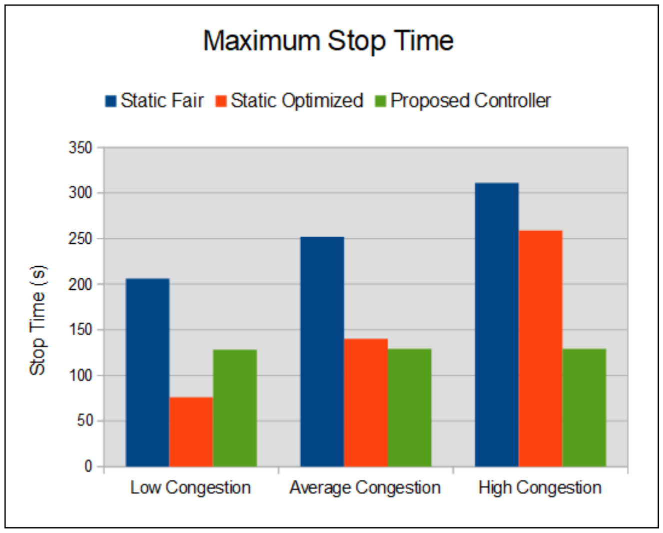

Fig. 8. Measurements versus agent supplied information. Shows the maximum time vehicles stand in each model during low, average and high congestion. Lower values are better.

Figure 6 shows the time the average vehicle loses on a journey through the custom scenario and figure 7 shows the time the average vehicle stood still during its journey. As expected the static fair controller performs worst out of the three and performs progressively worse as the amount of vehicles in the simulation increases. While the other controllers also perform worse as traffic increases, this only seems to become significant during high congestion.

The proposed controller performs best overall in average time loss and average stop time categories, but only marginally. During low and average congestion the proposed controller outperforms the static optimized controller on average by approximately 6% when it comes to average time loss and 4% when it comes to average stop time. Interestingly, the largest performance gap between the proposed controller and the static optimized controller occurs during high congestion, which is 11% for the average time loss values and 14% for the average stop time values.

The maximum stop time in figure 8 tells a less clear cut story. The maximum stop time of the static fair controller rises steadily as traffic increases, and so does that of the static optimized controller. Somewhat surprisingly the proposed controller's maximum stop time stays constant during all types of traffic volumes, which makes it so the proposed controller's maximum stop time is less than halve the maximum stop time of the static optimized controller during high congestion. Nearly exactly the opposite is true during low congestion.

VII Discussion

This section covers the interpretation of the gathered results, a general conclusion and suggestions on what research can be done in the future.

Interpretation of results. The ARIMA forecasts are quite accurate, which is line with the existing literature. The forecasts could be improved by tuning the amount of previous lags the ARIMA model takes into account to make forecasts. Reducing this amount could lead to faster adaptation to sudden changes in traffic volumes, which the model is currently somewhat slow to pick up on.

The inaccuracies of the information supplied by a neighbouring agent highlights one of the largest shortcomings of the proposed controller; the difficulty of synchronization. The cycle time the controller operates in is different from the time it takes for a vehicle to get from one junction to the next. The controller aggregates the data received from the neighbouring agent into the next forecasted lag, even though the travel time in between the two intersections may be more or less than the length of a lag. This can be remedied as long as the travel time is longer than the lag by aggregating the received information not into the next lag but into the second or third next lag depending on the travel time. When the travel time in between intersections is less than the duration of a lag, the information received from the neighbouring agent becomes useless as vehicles often arrive in the same lag as they left in, which means the information can't be used for decisions in the next lag. In other words, vehicles have already showed up at the intersection before the controller is able to use the information of those incoming vehicles to queue up a new decision.

Looking at the average time loss and average stop time, it seems at least in a simple scenario as this one there is not much to be gained from a more complex model compared to a simple model when traffic volumes are low. This could suggest that when congestion is low there is more room for error, or the controller simply does not function as well in those situations. However, the former option is more likely as the controller switches to reactive mode relatively often when traffic volume is low, and simply operates according to demand. In other words, the proposed controller seems to better live up to its potential during heavy congestion and at the same time there may be no need for complex controllers at intersections which experience little traffic.

The maximum stop time results are due to the weight based system where vehicles that have been waiting for a long time get priority. When using the proposed controller a vehicle's weight increases similarly in each setting, regardless of whether congestion is high or low which explains why the maximum stop times for the proposed controller stay constant over multiple levels of congestion. Since this is disadvantageous at low congestion levels, it again suggests that switching to a static model during low congestion is beneficial.

Conclusion. The goal of this research was to find out whether it was possible to create a traffic light controller that fixes some of the limitations of existing controllers while also meeting other criteria. The controller had to be flexible, modular, require little in terms of computing power, be efficient and not fully rely on detectors. The proposed controller meets all criteria, to varying degrees.
Testing showed the proposed controller performing well during multiple stages of traffic congestion, which can be attributed to its ability to react to different traffic scenarios.
Each controller handles its own computations instead of one system having to keep track of the whole network, leading to relatively little required computing power.
In terms of modularity, the software side only requires a newly initialized agent to know about its neighbouring agent. However, any agent does need two detectors per lane to function properly, making the initialization of a new traffic intersection at least non-trivial.
The proposed controller spends most of its time in predictive mode, which doesn't fully rely on detectors and it can also function without detectors at all.
The proposed controller outperforms the static fair and static optimized models in terms of total average time loss and total average stop time, but only marginally. Also, the results suggest there is merit in switching to a completely static cycle during low congestion.
Finally, communication between agents has its limitations, suggesting the use of forecasting on all lanes to be more advantageous.

We can conclude that although the proposed controller does meet the criteria, it is hard to recommend the controller in its current state as the benefits arguably do not weigh up to the increased complexity compared to static models. The next section highlights some changes which could make the proposed controller more efficient.

Future work. One of the conjectured reasons the proposed controller only performed slightly better than the static optimized controller is the simplicity of the scenario.

The scenario featured just two intersection with two phases on which vehicles can only go straight. It is possible the controller can better live up to its potential when there are more phases to assign, so on intersections where cars are able to also turn left and right. Also, one of its goals was to create an implicit green wave by making cars move in platoons, this could be better achieved by having more intersections. It would be interesting to see if making those changes has a positive effect on the proposed controller's performance compared to the static models.

This paper showcased the latest version of the proposed controller, which didn't feature elaborate phase skipping techniques. However, previous versions of the controller did, which performed noticeably worse. Intuition would suggest that making a small amount of vehicles wait for a large amount of vehicles makes for lower average time loss, and this is the case, but only up to a certain point. If the controller decides to make the non-artery lanes wait for a couple of cycles before it grants them green light and later grants them enough green light to clear out those lanes, it could actually cause more average time loss than was initially gained by giving the artery lanes priority. This is because the vehicles on the artery lanes quickly pile up as the non-artery lanes clear out. The time it takes then for the artery lanes to clear out takes a significant amount time and a lot of time is lost in terms of start-up. This process repeats itself as the non-artery lanes pile up again. This suggests that phase skipping needs to be done with care and is an interesting topic to pursue in the future.

Lastly, it would be worthwhile to test this proposed controller, but maybe more interestingly an improved version of it, against more elaborate controllers. One of which could be a static controller which enforces green waves in some way. Others could also include existing models such as the ones mentioned in the Background section.

Bibliography

1. Lomendra Vencataya, Sharmila Pudaruth, Ganess Dirpal, and Vandisha Narain. Assessing the causes and impacts of traffic congestion on the society, economy and individual: A case of mauritius as an emerging economy. *Studies in Business and Economics*, 13(3):230–242, 2018. doi: 10.2478/sbe-2018-0045.
2. Kai Zhang and Stuart Batterman. Air pollution and health risks due to vehicle traffic. *Science of The Total Environment*, 450-451:307–316, 2013. doi: 10.1016/j.scitotenv.2013.01.074.
3. Megan Stacey. London city hall: Committee briefs, https://lfpress.com/2017/10/24/london-city-hall-committee-briefs/wcm/bdd9d6ad-2b9b-92e3-0621-58878145fd6d, Oct 2017.
4. Douglas. Fighting miami-dade's traffic war, one green light at a time, https://www.miamiherald.com/news/local/community/miami-dade/article156709334.html, Jun 2017.
5. T4america blog, http://t4america.org/2010/10/13/smarter-transportation-case-study-5-traffic-signal-optimization-portland-oregon/, transportation for america.
6. Hemanta Pradhan and Tnn. Traffic signals: Adaptive traffic signals installed at 50 junctions in bhubaneswar | bhubaneswar news - times of india, https://timesofindia.indiatimes.com/city/bhubaneswar/adaptive-traffic-signals-installed-at-50-junctions-in-bhubaneswar/articleshow/63400658.cms, Mar 2018.
7. Luca Studer and Misagh Ketabdari. Analysis of adaptive traffic control systems design of a decision support system for better choices. *Journal of Civil & Environmental Engineering*, 05(06), 2015. doi: 10.4172/2165-784x.1000195.
8. Andreas Warberg, Jesper Larsen, and Rene Munk Jørgensen. Green wave traffic optimization-a survey. 2008.
9. Tom V. Mathew. Design principles of traffic signal. *Transportation Systems Engineering*, pages 1–13, 2014.
10. Carlos Gershenson and David A. Rosenblueth. Adaptive self-organization vs static optimization: A qualitative comparison in traffic light coordination. *Kybernetes*, 41(3/4):386–403, 2012.
11. Yitian Gu, Vipul Singh, Liyan Sun, and Aditi Bhaumick. Adaptive traffic light control of traffic network. *Consumer Communications and Networking Conference*, 4:187–191, 2012.
12. Jimmy Krozel, Mark Peters, Karl D. Bilimoria, Changkil Lee, and Joseph SB Mitchell. System performance characteristics of centralized and decentralized air traffic separation strategies. *Air Traffic Control Quarterly*, 9(4):311–332, 2001.
13. Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2):293–305, 1999.
14. Malik Tubaishat, Yi Shang, and Hongchi Shi. Adaptive traffic light control with wireless sensor networks. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 187–191. IEEE, 2007.
15. Nathan H. Gartner and Chronis Stamatiadis. Arterial-based control of traffic flow in urban grid networks. *Mathematical and computer modelling*, 35(5-6):657–671, 2002.
16. Pitu Mirchandani and Fei-Yue Wang. Rhodes to intelligent transportation systems. *IEEE Intelligent Systems*, 20(1):10–15, 2005.
17. Steen Merlach Lauritzen. Evaluation of dogs. *Road Directorate*, 2004.
18. Michael Shenoda. *Development of a phase-by-phase, arrival-based, delay-optimized adaptive traffic signal control methodology with metaheuristic search.* PhD thesis, 2006.
19. Ahmed Y. Zakariya and Sherif I. Rabia. Estimating the minimum delay optimal cycle length based on a time-dependent delay formula. *Alexandria Engineering Journal*, 55(3):2509–2514, September 2016. doi: 10.1016/j.aej.2016.07.029.
20. Signal timing design - cycle length determination, https://www.webpages.uidaho.edu/niatt_labmanual/chapters/signaltimingdesign/theoryandconcepts/cyclelengthdetermination.htm.
21. Mohammad M. Hamed, Hashem R. Al-Masaeid, and Zahi M. Bani Said. Short-term prediction of traffic volume in urban arterials. *Journal of Transportation Engineering*, 121(3):249–254, 1995.
22. Billy M. Williams and Lester A. Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.
23. Simulation of urban mobility, https://sumo.dlr.de/index.html.
24. Highways england, http://tris.highwaysengland.co.uk/detail/trafficflowdata.