Combining Model Predictive Intersection Control with Green Light Optimal Speed Advisory in a Connected Vehicle Environment

Simon Stebbins1, Jiwon Kim1, Mark Hickman1, Hai L. Vu2

1School of Civil Engineering, University of Queensland

2Intelligent Transport Systems Lab, Swinburne University of Technology

Email for correspondence: uqsstebb@uq.edu.au

Abstract

In recent years there has been much interest in Connected Vehicle (CV) technology. Vehicle-to-infrastructure (V2I) communication has the potential to reduce delays, stoppage time, fuel usage and emissions, because it allows fine-grained traffic movement data to be shared with greater frequency. Previously, traffic control algorithms have been based on macroscopic, fluid mechanical traffic models, but since V2I communication allows for fine-grained traffic data, a more accurate, microscopic, car-following traffic model will be used instead in this paper.

At an intersection there are essentially two ways to improve traffic conditions – by improving the intersection control schedule, and by modifying vehicle approach trajectories. In order to best utilise traffic data that varies second-by-second, it is proposed that the optimal control schedule that minimises delay can be found via model predictive control (MPC) with suitable state space reduction techniques. In addition, since the control algorithm utilises an underlying microscopic model, entering vehicles’ trajectories can be modified with Green Light Optimal Speed Advisory (GLOSA). This allows drivers to adjust their speed profiles in order to have an efficient approach trajectory. CV technology allows MPC to be integrated with GLOSA, making the best use of this future technology to improve traffic conditions for all motorists.

1. Introduction

Traditional signal control was performed offline. Traffic engineers utilised traffic data collected throughout the hours of each day and used this data to devise a signal timing strategy. This resulted in the operating parameters *cycle time*, *splits* and *offsets*. These parameters were generally derived from standard procedures, such as those described in the Highway Capacity Manual (Transportation research board, 2000) or signal-timing software such as TRANSYT (Robertson, 1969) and SYNCHRO (Husch and Albeck, 2006).

More modern traffic control systems are able to react to current traffic control conditions by sensing the presence of vehicles using inductive loops in the road or with alternative technologies such as traffic cameras. However, they still often assume that for the next several minutes or hours, traffic conditions can be accurately characterized by summarising averages. Control strategies that operate in this fashion include SCOOT (Robertson and Bretherton, 1991) and SCATS (Sims and Dobinson, 1980). In reality, traffic patterns constantly vary are better characterised as a discrete sequence of platoons of vehicles rather than approximated as a continuous flow.

Some researchers have developed an alternative approach to signal control that predicts, ahead of time, what the traffic pattern will be as it approaches the intersection and determines the control schedule based on this information. This is a form of model prediction control (MPC). In contrast to earlier traffic control systems that calculate average flows over time, this MPC approach incorporates a feedforward mechanism instead of a feedback mechanism, because it predicts traffic movements before they arrive. Such control systems include OPAC (Gartner et al., 2001), PRODYN (Henry et al., 1984), UTOPIA (Mauro and Di Taranto, 1990) and RHODES (Mirchandani and Head, 2001). More recently, (Xie et al., 2012) devised an MPC traffic control algorithm that was formulated as a job shop scheduling problem. They clustered vehicles into platoons and used elimination criteria to reduce the search state space.

Some authors have formulated traffic control as a mixed integer linear programming (MILP) problem. (Lo, 1999) used a hydrodynamic (macroscopic) model, which captures kinematic waves. The timing plans produced were consistent with models that work for unsaturated conditions. In gridlock conditions, the produced timing plans were better than conventional queue management practices. Several other authors have formulated their own MILP traffic control schemes including (Lin and Wang, 2004) and (Beard and Ziliaskopoulos, 2006). (Han et al., 2012) formulated a MILP problem using a link-based model instead of using the Cell Transmission Model (Daganzo, 1994). A variational type argument was applied so that the system dynamics can be determined without knowledge of the traffic state in the interior of each link. This resulted in a reduced number of binary variables and computational effort. (Guilliard et al., 2016) formulated their model using non-homogenous time intervals, resulting in lower delay solutions.

Since an MPC approach to traffic control can be computationally expensive, other authors have employed a store-and-forward traffic model and applied linear-quadratic regulator theory (Diakaki et al., 2002) or quadratic programming (Aboudolas et al., 2010), (Le et al., 2013) to devise a tractable control strategy.

Since inductive loops are fixed in location they are unable to provide frequent updates about individual approaching vehicles. However, upcoming V2I technology will allow vehicles to communicate more accurate movement data more frequently. It is also capable of providing information back to vehicles approaching the intersection. This means that V2I communication is able to influence driving behaviour and trajectories. Several authors have proposed mechanisms for determining Green Light Optimal Speed Advisory (GLOSA) for vehicles approaching an intersection including (Seredynski et al., 2013), (Li et al., 2014) and (Kamalanathsharma and Rakha, 2013).

(Tachet et al., 2016) developed a method of traffic control whereby vehicles are slotted through an intersection. This approach may be considered as a combination of speed advisory and traffic control. However, they do not use MPC to find optimal solutions.

In this paper, a combination of GLOSA and MPC is used to find the optimal control schedule for a given approaching traffic pattern. Since there is a high computational cost associated with searching through every potential schedule, a few techniques are used to reduce the computational complexity. These include using elimination criteria to prune the search space, and clustering vehicles into platoons to reduce the length of each control sequence.

2. Scheduling algorithm

A traffic network composed of a single intersection can be modelled as a set of routes,

Each route in the set has an inflow and an outflow, that is,

where , the set of inflows, and , the set of outflows. For the purposes of this paper each flow is considered to be a single lane. Each inflow and outflow contains an ordered list of vehicles, and each vehicle can be described by the continuous movement variables location, speed and acceleration. Note that each vehicle starts within an inflow. Once it has exited the intersection it no longer belongs to the inflow and joins the outflow for its route. This means that the lists of vehicles for the inflows and outflows are always changing. In particular, vehicles on multiple inflows may join the same outflow once they have exited the intersection.

Traditionally, signal control has been determined by the parameters cycle time, phase splits and offsets. However, an alternative formulation considers traffic control to be a scheduling problem. The scheduling algorithm takes as input a set of vehicles, and the intended routes for each vehicle. In addition, each vehicle has a location, speed and acceleration at the current time. All of this information could be sent to the intersection controller using V2I communication. The control algorithm works by enumerating possible future schedules and calculating their associated total cumulative delays, attempting to find the optimal schedule with minimum delay. Since there are a finite number of arriving vehicles there are a finite number of ways to sequence vehicles through the intersection. Theoretically, every possible sequence can be enumerated and the optimal solution could be determined. However, it is impractical to enumerate them all in real-time because the number of possible sequences is (at least) exponential compared to the number of vehicles on all routes (Papageorgiou et al., 2003). To overcome this computational complexity problem, (Xie et al., 2012) developed a polynomial time algorithm by framing single intersection traffic control as a job shop scheduling problem, and using elimination criteria to reduce the size of the search state space. These elimination criteria are explained and their optimality is examined later in the paper. Unlike their work, in this paper, an algorithm is developed that works with a more realistic, car-following traffic model, and the effect of combining GLOSA with MPC signal control is examined.

Figure 1 illustrates how all future schedules can be enumerated. There are two waiting queues on the northbound and eastbound approaches. After each vehicle is scheduled through the intersection, there are up to two options for choosing the next vehicle to have right of way. Alternatively, the state space may be constructed differently. Each node in the tree may be a set of vehicles that can be given simultaneous right of way. For example, vehicles travelling north and south can be given right of way simultaneously during a single phase. As discussed later, it is more efficient to cluster nearby vehicles together into platoons. One such platoon is then effectively one node in the state space.

|  |  |
| --- | --- |
| N4N3N2N1E3E2E1 |  |

Figure : The left part of the figure shows an example of the current state of traffic at the intersection. Each vehicle is identified by its direction (E or N) and its index in the entry queue. The right part of the figure illustrates a partial construction of the state space. Each node contains the next vehicle to have right of way. The state space is, therefore, a tree and each path from the root to a node in the tree is a description of a (partial) schedule.

Listing 1 describes a generic scheduling algorithm that utilises the structure of the state space. It could be used with any microscopic traffic model to enumerate all feasible, future schedules. Once they have been enumerated, the optimal schedule can be chosen which will result in minimum total cumulative delay. On the line marked with \* a non-deterministic choice is made to select the next set of vehicles to be given right of way. However, real computers are deterministic. One way to run the algorithm on a deterministic machine is to try each possible choice in series. However, as pointed out by Xie et al. (Xie et al., 2012), if each vehicle is sequentially given right of way, the number of possible complete schedules is

where is the number of vehicles on entry . This means the state space grows rapidly and therefore the algorithm is intractable for even moderate queue sizes, so some techniques are required to reduce the computational complexity of the algorithm.

One such technique is the use of a prediction horizon. The prediction horizon is generally a time boundary a fixed period of time into the future from the present time. The algorithm examines schedules up until the prediction horizon, and vehicles that are not expected to enter the intersection within the prediction horizon are eliminated from consideration. However, it was found to be more convenient to use a prediction horizon that was, instead, a set distance upstream of the intersection. Vehicles beyond that location at the present time were not considered by the scheduling algorithm. The scheduling algorithm runs frequently in real-time and is executed wheever a vehicle crosses the prediction horizon. The algorithm is guaranteed to be optimal only when the prediction horizon is infinite, which is not practically possible. However, decent results are obtained by setting the prediction horizon to a suitable distance (for example 500 metres).

Programmatically, is a data structure that records the phase timing for each route by appending green and intergreen (amber and red) time for each route. The algorithm requires intersection entry and exit times to be determined for each vehicle. These can be obtained through simulation into the future, or by estimation. Note that it is possible there are a few vehicles at the beginning of the scheduling period that must be given right of way initially because, despite having not entered the intersection, they are unable to stop in time for the intersection. Generally, it is not advisable to split platoons of vehicles, and since the same signalling system is used for all vehicles, sometimes it is not possible to split platoons in any case. These generated, but impossible, schedules may be safely discarded from consideration.

Even though this generic scheduling algorithm is intractable it provides insight into developing a more practical algorithm. In particular, it shows that the main hurdle to overcome is the large number of potential schedules, rather than the computation of the underlying traffic model. For example, a macroscopic model may describe all traffic on one queue as a single flow represented by flow rate and density. A microscopic, car-following model, on the other hand, would represent traffic as individual vehicles. At each time step of the simulator, a macroscopic model would require (at best) time to update, and a microscopic model would require time to update, where n is the number of vehicles in all queues. A macroscopic model, therefore, requires less computation time. However, at best a linear speed up would result from switching from a microscopic to a macroscopic model, which is insignificant compared to the number of potential schedules. In other words, the “big O” complexity of the algorithm is not reduced by switching traffic models. Hence, in this paper, a microscopic car-following model is used, but in order to reduce computation time vehicles are clustered together into platoons.

Listing : Annotated pseudocode for a generic scheduling algorithm

enumerateSchedules():

// H is the set of all vehicles next to enter the intersection on // each entry queue, that is, the heads of all queues.

// P is the power set of H, excluding the empty set.

// p is chosen from P with a non-deterministic choice. All vehicles // in p are on compatible routes.

\* choose

// Determine the exit time for the last vehicle in p.

// Append green time to the schedule for (the routes of) all vehicles // in p.

.appendGreenTime

// Append intergreen (amber & red) time for all routes until *maxTime*.

.appendIntergreen

// Run the traffic simulator forwards in time until *maxTime*.

.run

// Remove from the entry data structures the vehicles that have now // exited the intersection in the simulator. Note that this also // indirectly updates .

// Recursively call enumerateSchedules with the updated schedule and // entries data structure, if any vehicles are yet to be scheduled.

 enumerateSchedules()

where

 is the schedule object that is progressively built with each call to enumerateSchedules(). It tracks the phase schedule for each route through the intersection. is the updated schedule.

 is the set of entry queues. All vehicles that have not exited belong to an entry. is the updated set of entries. denotes the th vehicle on entry .

 is the set of all vehicles within the prediction horizon.

and is a binary relation for all pairs of routes that have compatible movements.

3. Car-following traffic model

The following update rules were used at each time step for each vehicle:

|  |  |
| --- | --- |
|   | () |
|   | () |
|   | () |
|   | () |
|   | () |

where

 is the vehicle’s current location

 is the preceding vehicle’s current location

 is the vehicle’s current speed

 is the preceding vehicle’s current speed

 is the free flow speed (speed limit)

 is the vehicle’s updated location for the following time step

 is the vehicle’s updated speed for the following time step

 is the vehicle’s updated acceleration for the following simulated time step

 is the standstill distance between vehicles

 is the length of the preceding vehicle

 is the vehicle’s maximum acceleration rate

 is the vehicle’s maximum braking (deceleration) rate

 is the preceding vehicle’s maximum braking (deceleration) rate

 is the remaining intergreen time

 is the remaining distance to the intersection stop line

 is the length of one simulated time step

 is the desired time headway between vehicles

Rule (1) ensures that, if a vehicle is within the standstill distance of the preceding vehicle, it will brake at the maximum rate. Rule (2) guarantees that collisions will not occur in the simulator by comparing the current locations and stopping distances of the vehicle and preceding vehicle. If it is determined that a collision could occur if the preceding vehicle were to brake at the maximum rate, braking for the current vehicle occurs at the maximum rate for the current time step. Rule (3) is employed only if the antecedents of rules (1) and (2) are false. In this case, for the following time step is the minimum of two computed values – the first being the acceleration advice based on the remaining intergreen time and distance to the intersection, and the second being the acceleration required to maintain desired headway between the current and preceding vehicles. The acceleration amount ensures that , that is, at the next time step, the current vehicle will be positioned the desired headway distance behind the preceding vehicle. Note that this calculated value may be outside the range . If this is the case, rules (4) and (5) adjust to be within the range.

Once has been determined, the updated values and can be calculated according to these rules:

|  |  |
| --- | --- |
|   | () |
|   | () |
|   | () |
|   | () |

Rules (8) and (9) are used only if calculated by rule (6) violates the speed constraints.

Two acceleration advice functions were tested. The plainAccelerationAdvice function mimics drivers who brake as late as possible, that is, it mimics drivers who are not given GLOSA advice. This function returns , unless the vehicle must brake because , where is the location of the stop line at the intersection. In this case, is returned.

The minSpeedRangeAccelerationAdvice function maximises the minimum required speed over the entire trajectory of the vehicle, given the remaining intergreen time and distance to the intersection. This conserves the vehicle’s kinetic energy as much as possible. The authors’ of this paper have described how the GLOSA function behaves in a separate paper. (Stebbins et al., 2016) described this function in detail, and showed that it results in reductions for delay, stoppage time and fuel usage when the advice is followed by vehicles operating under fixed time signal control.

4. State space reduction

Since the algorithm described by listing 1 grows at least exponentially, based on the number of vehicles on waiting approaches, a few techniques are used to reduce the state space.

4.1. Platoon clustering

Generally, it is not beneficial to split platoons of vehicles apart due to intersection signalisation. Therefore, platoons are kept together in order to reduce the computational complexity of the algorithm. If the running time of the algorithm without platoon clustering is – where is the number of vehicles in entry queues – its running time with platoon clustering is , where and is inversely proportional to the size of each cluster. Thus, the running time is reduced, but the characteristics of remain unaltered. In particular, if is exponential, clustering vehicles into platoons does not bound the running time of the scheduling algorithm by a polynomial function. Even so, at high flow rates, many vehicles will join a platoon, significantly reducing the number of objects to consider on each queue.

The second line of rule (3) provides the mechanism for testing whether a vehicle has joined a platoon. If calculated from this formula alone, the vehicle is considered to have joined a platoon with the preceding vehicle. Within the scheduling algorithm, nodes of the state space no longer correspond to vehicles. Instead, they correspond to platoons.

4.2. Elimination criteria

In order to reduce the computational complexity of the scheduling algorithm, elimination criteria were introduced to prune the state space. These criteria are similar to those used by (Xie et al., 2012).

A data structure keeps track of green times and delays while the control algorithm examines potential future schedules by expanding the state space. A record in this data structure is created whenever a state transition occurs as the algorithm is executed. Each record is a 4-tuple

where

 is the set of vehicles chosen to have right of way next

 is the set of vehicles that have just exited in simulation

 is the starting green time for

 is the estimated total cumulative delay for all exited vehicles

As the scheduling algorithm is run, only one record for each unique pair is kept. Suppose the green time for the currently examined partial schedule is and the estimated total cumulative delay is . If there is a pre-existing record for the pair , where and apply to the current partial schedule, the values and are retrieved from the corresponding record and updated in the record as follows.

|  |  |
| --- | --- |
|  |  |
|  |  |

In addition, if then the current state is eliminated and no sub-states are examined. (Xie et al., 2012) showed that, if these elimination criteria are used, the scheduling algorithm’s running time is bounded by a polynomial function.

The justification for the using elimination criteria can be explained as follows. Consider two partial schedules and . If they can both be described by the pair then the elimination criteria can be used to compare them. Once a vehicle has exited the intersection its total delay can be reliably calculated. This means the total delay of all exited vehicles for both and can be compared. The second part of the elimination criteria compares green times. The assumption being that if the green time for is and the green time for is and then the total resultant delay for unexited vehicles of will be less than the total resultant delay for unexited vehicles of .

4.2.1. A caveat on optimality using elimination criteria

As the results demonstrate, the scheduling algorithm running with elimination criteria is a significant improvement over fixed time signal control when compared according to the evaluation criteria. However, there exist exceptional circumstances where the optimal solution may be excluded by the elimination criteria.

If the underlying traffic model is a car-following model, there are cases where the elimination criteria may fail. Consider the case in figure 2, which compares two schedules and . For both schedules, has right of way next and is preceded by , which has already exited. Due to scheduling differences, exits the intersection earlier in compared to . This results in being closer to for . Depending on relative speeds, this could result in being delayed by after exits the intersection when the schedule is applied; but will not be delayed by when is applied. Thus, under the schedule , even if and , the delay of caused by is not taken into account by the elimination criteria. Although this additional delay may not be significant for , vehicles following in the same platoon will also experience delay caused initially by . This may result in the total delay of being higher than the total delay of , even though may be excluded by the elimination criteria.

Figure 2: This case demonstrates that, in some cases, a schedule may be eliminated even though it could be optimal. The left image shows the resultant traffic pattern when is applied and the right image is the resultant traffic pattern of .

|  |  |
| --- | --- |
|  |  |

5. Results

Results were obtained from a custom-built traffic simulator that implements a microscopic, car-following model and is capable of examining different possible future schedules. The simulation is visualised using an HTML5 canvas and can be viewed in a Web browser. A network was defined with a single intersection and tested for two conflicting approaches, one eastbound and the other northbound. Four scenarios were tested: MPC with GLOSA, MPC without GLOSA, fixed time signal control with GLOSA, and fixed time signal control without GLOSA. For the fixed time signal control cases, equal green time was allocated to both approaches, since their average flow rates were equal. All tests were run over an hour of simulated time and the average measurements per vehicle were obtained. For each test, the same average flow rate was used for both approaches, and each vehicle arrived into the network according to an exponential distribution, but its initial location was adjusted if necessary so it was at the desired following distance behind its preceding vehicle. Each test scenario was repeated 10 times, and a Student’s t-test was performed to obtain a 90% confidence interval for each measure, as indicated by the error bars in figures 3-6. If the confidence interval was very small, indicating little variation between tests, it was excluded from the figure for that data point. The prediction horizon was set at 500 metres before the intersection. The simulation was run until all vehicles that had crossed the prediction horizon within one hour had exited the network. A running version of the simulator with configurable parameters (average flow rates, control method, trajectory strategy) can be found at <http://sstebbins-traffic-control.appspot.com/>.

Figure 3: Average delay over a range of average flow rates

Figure 3 illustrates how average delay per vehicle changes for different average flow rates. As expected, using MPC to find the optimal schedule significantly reduces delay per vehicle. Reductions in delay are also achieved by giving vehicles GLOSA advice. This is clearly the case with fixed time signal control, but the benefit is not as pronounced when the control algorithm uses MPC. In the MPC case, using GLOSA does not appear to be very beneficial except for high flow rates near the saturation flow rate (700-800 vehicles per hour). Based on queue length data at the end of one hour, it appears the saturation flow rate for fixed time signal control is lower (600-700 vehicles per hour).

Figure : Fuel usage over a range of average flow rates

Figure 4 plots fuel usage per vehicle for increasing average flow rates. The VT-Micro emissions model (Ahn et al., 2002) was used to track fuel consumption. Sample model coefficients for estimating fuel consumption rates for a composite vehicle were taken from the study by (Alsabaan et al., 2012). At lower flow rates, MPC’s performance is not quite as good, probably because vehicle’s average speeds are higher, requiring higher fuel usage. MPC outperforms fixed time signal control for higher flow rates. This occurs because MPC tends to reduce stop-and-start behaviour for vehicles travelling through the intersection. Fuel usage is reduced when using GLOSA with fixed time signal control, because GLOSA tends to smooth vehicle trajectories, avoiding oscillating speed variations.

Figure 5 shows how stoppage time changes as average flow rates increase. Average stoppage time is low for low flow rates. In both the MPC and fixed time cases, the effect of GLOSA is to reduce average stoppage time. This occurs because the GLOSA algorithm seeks to maximise the minimum required speed for each vehicle. Interestingly, MPC tends to increase average stopping time at high flow rates, even though average delays are reduced compared to fixed time signal control. MPC tends to reduce speed variations in the vehicle queues, thus if the average speed of all vehicles is lower, there will be more vehicles stopped. For fixed time signal control, there is more speed variation, hence there are fewer vehicles stopped. This explanation also accords with the fuel usage results. Even at high flow rates, very few vehicles are required to stop when fixed time signal control is used with GLOSA.

Figure : Stoppage time over a range of average flow rates

Figure 6 indicates the average and maximum CPU time for the two MPC cases. These are the times taken for a single run of the scheduling algorithm, executed whenever a vehicle crosses the prediction horizon. The average running times, in both cases, are always < 0.4 s for all average flow rates. This means that if the time step interval is 0.5 s or greater, the scheduling algorithm will run within one time step on average. The maximum CPU time results indicate the worst case running times. Interestingly, the MPC + GLOSA case outperforms the MPC only case. MPC + GLOSA finished computing the optimal schedule within 2.5 s for all flow rates across all test cases. Despite having a good average CPU time, the MPC only algorithm was less predictable. Sometimes it required up to 8 s to complete. It should be noted that the algorithm was run in JavaScript on an i5-4590 processor, not in an optimised programming language and hardware.

Overall, the performance indicators reveal that model predictive signal control significantly outperforms fixed time signal control. Combining MPC with GLOSA improves performance further.

6. Conclusion

This paper presented a traffic control strategy which utilises model predictive control to find the optimal schedule for all vehicles within the prediction horizon. Unlike many other traffic control strategies, the underlying traffic model is microscopic and different schedules are tested through simulation. A prediction horizon, platoon clustering and elimination criteria are employed to reduce computational load, while maintaining results that are near optimal. In conjunction, since the underlying traffic model is microscopic, car-following can be replaced with Green Light Optimal Speed Advisory (GLOSA) for each approaching vehicle to direct it to follow an optimal trajectory. The results demonstrate that there are marked improvements with regard to delay, fuel usage and stoppage time when using these control and trajectory strategies. In addition, computation time remains tractable. The control algorithms work well when traffic data is accurate and complete, however V2I communication may be necessary to provide traffic data with sufficient granularity for the algorithms to work well. Future work may determine how fine-grained traffic data needs to be in order to derive benefit from an MPC approach.

Figure : Average and maximum CPU times for a single run of the scheduling algorithm over a range of average flow rates

References

Aboudolas, K., Papageorgiou, M., Kouvelas, A., Kosmatopoulos, E., 2010. A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. Transportation Research Part C: Emerging Technologies 18, 680–694. doi:10.1016/j.trc.2009.06.003

Ahn, K., Rakha, H., Trani, A., Van Aerde, M., 2002. Estimating Vehicle Fuel Consumption and Emissions based on Instantaneous Speed and Acceleration Levels. Journal of Transportation Engineering 128, 182–190. doi:10.1061/(ASCE)0733-947X(2002)128:2(182)

Alsabaan, M., Nayak, A., Naik, K., Khalifa, T., 2012. Applying Vehicular Networks for Reduced Vehicle Fuel Consumption and CO2 Emissions. INTECH Open Access Publisher.

Beard, C., Ziliaskopoulos, A., 2006. System Optimal Signal Optimization Formulation. Transportation Research Record: Journal of the Transportation Research Board 1978, 102–112. doi:10.3141/1978-15

Daganzo, C.F., 1994. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. Transportation Research Part B: Methodological 28, 269–287. doi:10.1016/0191-2615(94)90002-7

Diakaki, C., Papageorgiou, M., Aboudolas, K., 2002. A multivariable regulator approach to traffic-responsive network-wide signal control. Control Engineering Practice 10, 183–195. doi:10.1016/S0967-0661(01)00121-6

Gartner, N.H., Pooran, F.J., Andrews, C.M., 2001. Implementation of the OPAC adaptive control strategy in a traffic signal network, in: 2001 IEEE Intelligent Transportation Systems, 2001. Proceedings. Presented at the 2001 IEEE Intelligent Transportation Systems, 2001. Proceedings, pp. 195–200. doi:10.1109/ITSC.2001.948655

Guilliard, I., Sanner, S., Trevizan, F.W., Williams, B.C., 2016. A Non-homogeneous Time Mixed Integer LP Formulation for Traffic Signal Control, in: Transportation Research Board 95th Annual Meeting.

Han, K., Friesz, T.L., Yao, T., 2012. A Link-based Mixed Integer LP Approach for Adaptive Traffic Signal Control. arXiv:1211.4625 [math].

Henry, J.-J., Farges, J.L., Tuffal, J., 1984. The PRODYN real time traffic algorithm, in: IFACIFIPIFORS Conference on Control in.

Husch, D., Albeck, J., 2006. Synchro studio 7 user guide. Trafficware Ltd., Sugar Land, TX.

Kamalanathsharma, R.K., Rakha, H.A., 2013. Multi-stage dynamic programming algorithm for eco-speed control at traffic signalized intersections, in: 2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC). Presented at the 2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC), pp. 2094–2099. doi:10.1109/ITSC.2013.6728538

Le, T., Vu, H.L., Nazarathy, Y., Vo, B., Hoogendoorn, S., 2013. Linear-Quadratic Model Predictive Control for Urban Traffic Networks. Procedia - Social and Behavioral Sciences, 20th International Symposium on Transportation and Traffic Theory (ISTTT 2013) 80, 512–530. doi:10.1016/j.sbspro.2013.05.028

Li, J., Dridi, M., El-Moudni, A., 2014. Multi-vehicles green light optimal speed advisory based on the augmented lagrangian genetic algorithm, in: 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC). Presented at the 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), pp. 2434–2439. doi:10.1109/ITSC.2014.6958080

Lin, W.-H., Wang, C., 2004. An enhanced 0-1 mixed-integer LP formulation for traffic signal control. IEEE Transactions on Intelligent Transportation Systems 5, 238–245. doi:10.1109/TITS.2004.838217

Lo, H.K., 1999. A novel traffic signal control formulation. Transportation Research Part A: Policy and Practice 33, 433–448. doi:10.1016/S0965-8564(98)00049-4

Mauro, V., Di Taranto, C., 1990. Utopia. Control, computers, communications in transportation.

Mirchandani, P., Head, L., 2001. A real-time traffic signal control system: architecture, algorithms, and analysis. Transportation Research Part C: Emerging Technologies 9, 415–432. doi:10.1016/S0968-090X(00)00047-4

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., Wang, Y., 2003. Review of road traffic control strategies. Proceedings of the IEEE 91, 2043–2067. doi:10.1109/JPROC.2003.819610

Robertson, D.I., 1969. “ TRANSYT” method for area traffic control. Traffic Engineering & Control 11.

Robertson, D.I., Bretherton, R.D., 1991. Optimizing networks of traffic signals in real time-the SCOOT method. IEEE Transactions on Vehicular Technology 40, 11–15. doi:10.1109/25.69966

Seredynski, M., Dorronsoro, B., Khadraoui, D., 2013. Comparison of green light optimal speed advisory approaches, in: Proc. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 2187–2192.

Sims, A.G., Dobinson, K.W., 1980. The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. IEEE Transactions on Vehicular Technology 29, 130–137. doi:10.1109/T-VT.1980.23833

Stebbins, S., Kim, J., Hickman, M., Vu, H.L., 2016. Characterising Green Light Optimal Speed Advisory Trajectories for Platoon-Based Optimisation, in: Transportation Research Board 95th Annual Meeting. Washington, DC, USA.

Tachet, R., Santi, P., Sobolevsky, S., Reyes-Castro, L.I., Frazzoli, E., Helbing, D., Ratti, C., 2016. Revisiting Street Intersections Using Slot-Based Systems. PLOS ONE 11, e0149607. doi:10.1371/journal.pone.0149607

Transportation research board, 2000. Highway Capacity Manual. National Research Council, Washington, DC, USA.

Xie, X.-F., Smith, S.F., Lu, L., Barlow, G.J., 2012. Schedule-driven intersection control. Transportation Research Part C: Emerging Technologies 24, 168–189. doi:10.1016/j.trc.2012.03.004