Combining Adaptive Junction Control with Simultaneous Green-Light-Optimal-Speed-Advisory

Jakob Erdmann
Institute of Transportation Systems
German Aerospace Center (DLR)
Berlin, Germany
jakob.erdmann@dlr.de

We introduce the AGLOSA algorithm (pat. pending) which combines an adaptive control algorithm (A) with Green-Light-Optimal-Speed-Advisory (GLOSA) to achieve near-optimal control for a single junction. The central issue which must be resolved to allow this combination is the need for stable signal plans by GLOSA versus the instability caused by adaptive control. Our algorithm exploits Vehicle-to-Infrastructure Communication (V2I) to extend the planning horizon and create sufficiently stable plans using dynamic programming. In a fully-equipped simulation scenario vehicle time-loss is reduced by between 33% and 72% in comparison to competing algorithms.

Keywords: single junction control,V2I, GLOSA, optimal control

I. Introduction

The problem of controlling traffic at an intersection is central to achieving smooth urban traffic flow. Even though the research focus appears to have shifted from single intersections to networks of intersections, we demonstrate that significant efficiency can be gained by applying V2I technologies to an isolated intersection.

While there are intersections where roundabouts are an advantageous form of control, traffic light systems (TLS) still offer better performance in many situations. The most rudimentary form of traffic signaling involves fixed signal plans which cycle through a fixed number of fixed-length phases. One way of adapting these plans to changing traffic conditions is by using pre-computed plans for different times of day or different days of the week. More sophisticated forms of adaption rely on real-time traffic measurements, i.e. using inductions loops embedded into the road.

In Germany, the most common type of adaptive TLS is the "Zeitlückensteuerung" (time gap control). It measures time headways between successive vehicles and extends the current green phase as long as the vehicles are closely spaced in time (or until the maximum phase duration of 120 seconds is reached). This algorithm allows for short (responsive) cycles in low-traffic conditions and efficient but less responsive long cycles in heavy traffic conditions [1].

A quite recent development in intersection control does not aim at matching the signal plan to the incoming vehicles but rather attempts to coordinate the vehicles with the known (and usually fixed) signal plan. This concept is called Green-Light-Optimal Speed advisory (GLOSA)[2,3]. Vehicles receive messages about the switching times of the next TLS and an on-board assistance system computes an ideal approaching speed. If the driver follows this speed advice several advantages can be realized:

- Security is increased by preventing sudden decelerations at a red light
- Energy is conserved since the vehicle may avoid stopping
- The efficiency of the intersection is increased since freely flowing traffic has higher throughput than a stopped queue which is accelerating.

Unfortunately GLOSA requires beforehand knowledge of switching times in order to work. This stands in conflict with adaptive control which may modify switching times in response to detected vehicles.

In the following we present the AGLOSA-algorithm which solves this conflict using V2I communication. Incoming vehicles announce their presence to the TLS in advance and thus allow optimization of an adapted signal plan which is sufficiently stable for the GLOSA application.

The computed plan is optimized under the constraint that there are distinct phases and clearance times[1] and under the assumption that drivers follow the GLOSA speed advice. We describe 3 variants of the AGLOSA-algorithm which offer different trade-offs between optimality and computational requirements.

We first give a general outline of the AGLOSA algorithm in section II, then explain the algorithm in detail using an intersection with two conflicting streams as an example in section III and generalize to arbitrary intersections in section IV. Simulation results are presented in section V.

[1] Higher gains in efficiency can be achieved without this constraint [4].

II. Structure of the AGLOSA Algorithm

algorithm is a cooperative algorithm which involves a TLS and the surrounding vehicles. It requires vehicles and TLS to exchange the following messages:

- vehicle to TLS: [*id, geo-position, speed*]
- TLS to individual vehicle: [*assigned_switching_time*]

Furthermore it requires vehicles to be equipped with the GLOSA assistance system which computes the optimal approach speed based on the assigned_switching_time and to advise the driver on using this speed. The driver is responsible for adapting the speed in regards to other vehicles. Our treatment of GLOSA messaging differs slightly from the definition of ETSI[2] where vehicles receive information about multiple switching times and must select the appropriate switching time themselves.

The algorithm continuously repeats the following steps:

1. vehicles send *id, position* and *speed* to TLS
2. TLS computes optimal plan
3. TLS sends *assigned_switching_time* to each vehicle
4. vehicles compute (and adapt to) speed advice

Single vehicles may occasionally receive multiple GLOSA messages but the algorithm guarantees that these messages obey acceleration and deceleration constraints of the receiving vehicle.

The steps 1, 3 and 4 employ known technologies and will not be elaborated on in this work. Our main contribution lies in the computation of the optimal signal plan in step 2. Note that this optimization problem is a so-called “online”-problem. The incoming vehicles are only known up to a certain horizon at each time step. Since online-problems have to deal with uncertainty of inputs they can only be solved optimally in regards to probability assumptions about future inputs. In contrast, the AGLOSA-algorithm solves the offline-problem for the known demand within the horizon and then uses this solution for the online-problem. The feasibility of this heuristic approach will be demonstrated with simulation results in section V. The details of our algorithm are explained in the next section.

III. Signal Plan Optimization for 2 Intersecting One-Way Streams

The problem of computing an optimal signal plan for an intersection using knowledge about the demand has been studied by Gartner [5]. He demonstrated that this problem can be solved using dynamic programming techniques. Our approach differs from Gartner’s in that we allow switching at any point in time (while keeping constraints such as clearance time and minimal phase durations) whereas Gartner only allowed switching at a limited set of points in time.

As a novel contribution to signal plan optimization we incorporate the fact that (due to GLOSA) each vehicle has a range of possible arrival times $[t_{min}, t_{max}]$ at the intersection and that a vehicle may clear the junction faster if it does not have to accelerate from a full stop. This allows us to compute a more efficient switching plan. In the following we will first define the optimization problem, then explain the general idea for solving the problem with dynamic programming. Following that we give a detailed description of the algorithm in pseudo-code.

The problem of signal plan optimization for two intersecting one-way streams *(H: horizontal, V: vertical)* can be stated as follows:

Given a list of vehicle arrival ranges from the horizontal direction $(h_1, ..., h_n)$ with $h_i = (h_{i,min}, h_{i,max})$ and from the vertical direction $(v_{0,...,}v_m)$, $v_i=(v_{i,min},v_{i,max})$ and initial state of the junction where direction $d \in (H,V)$ has green light.

Determine: What are switching times (s_0, s_1,…) and what is the *assigned_switching_time* for each vehicle that minimizes the combined time loss for all vehicles. The time loss for a vehicle is determined as the difference $(t_{actual} - t_{min})$ between the time of actually passing the junction t_{actual} and its earliest possible time for passing the junction t_{min}. Note, that the *assigned_switching_time* is a lower bound on t_{actual}. For the first vehicle on the approaching lane t_{actual}= *assigned_switching_time* will usually hold, while subsequent vehicles will pass the intersection at a later time. The underlying assumption is that the drivers always follow their preceding vehicles at a safe speed even if the GLOSA system advises on a higher speed.

The AGLOSA algorithm ensures that the phase duration will accommodate all vehicles which are given the same *assigned_switching_time* (and thus the same phase) if they maintain a reasonable time-headway to their leader[2].

As is common with dynamic programming problems we decompose our problem into repeating subproblems which are solved once but reused repeatedly in constructing the end result. Our subproblem formulation is this:

What is minimal time loss in state *(i,j)* where the vehicles $(h_{0...}\ h_i)$ and the vehicles $(v_0, ..., v_j)$ have already passed the junction? The key idea is that this can be determined from the optimal solutions for the predecessor states *(i-1, j)* and *(i, j-1)*. Thus we have a recursive problem formulation for the initial problem *(i=n, j=m)* where the solutions for different *(i,j)* reoccur frequently. Since we only need to determine predecessor solutions where *i,j >= 0* and the solution to *(0,0)* is trivial we can build the solution matrix *S* for all *i,j* up to *(n,m)* from the bottom up. To determine the optimal solution for *(i,j)* from *(i-1,j)* and *(i,j-1)* and to generate the final signal plan and *assigned_switching_time* values, we need to keep additional information in each cell *S[i,j]* of our solution matrix.

To achieve optimal results using dynamic programming the problem formulation must satisfy Bellmans principle of optimality[5]. In our case, the predecessor states must contain enough information to allow consideration for all relevant alternative switching schedules. To describe a solution for state *(i,j)* we need to know the value of the objective function (time loss) but also the state of the traffic light and the minimum

[2] This is a configurable parameter Defined as **K1** in section IV

remaining time until the next switch. (i.e. there might be two competing solutions where one has less time loss but the other one has less switch delay and thus leads to a better solution in state *(i,j+1)*). To reduce computational effort we deviate from optimality and consider simplified state descriptions:

- Variant A1 where only the predecessor states *(i-1,j)*, *(i,j-1)* with minimum time loss are considered
- Variant A2 where the predecessor states including the currently switched direction *(H,V): (i-1,j,H), (i-1,j,V), (i,j-1,H), (i,j-1,V)* with minimum time loss are considered
- Variant A3 where the predecessor states include the currently switched direction and the minimum time until the next switch

In section V we present simulation result using the two heuristic variants A1 and A2. At the time of this writing, the optimal variant A3 is yet untested.

The pseudo-code in **Listing 1** describes variant A1 of the AGLOSA algorithm.

The preliminary computation of **tmin** and **tmax** for every vehicle can be accomplished by solving a second-degree polynomial, assuming that the vehicle accelerates towards the junction and keeps driving with maximum speed (**tmin**) or decelerates to GLOSA_MIN_SPEED as quickly as possible and continues driving at this speed (**tmax**).

The heart of the algorithm in **Listing 1** is the method for computing a continuation cell defined in pseudo-code in **Listing 2**. It computes the solution cell c' for starting in cell c and moving vehicle v across the junction. A cell c = S[i,j] of the solution matrix S describes the state of the junction after i vehicles from the horizontal direction and j vehicles from the vertical direction have passed the junction with minimal time loss. To compute the continuation for vehicle v additional junction-state variables besides time loss needs to be tracked for every cell. **Listing 2** references the following state variables:

- **l:** accumulated time loss (objective function)
- **t:** time at which the next vehicle may enter the intersection
- **d:** the direction which had the green light
- **r:** earliest time for next switch (to accommodate minimum phase durations)
- **s:** time of phase change or undefined
- **p:** the previous cell or undefined

From the vehicle v the following values are obtained:

- **d':** the direction of the vehicle
- **tmin:** the earliest possible arrival
- **tmax:** the latest possible arrival

Additonal configurable parameters are used:

- **K1:** time for passing the intersection at full speed
- **K2:** time for passing the intersection when accelerating from a stop
- **K3:** clearance time between phases
- **K4:** minimum green phase duration

Of particular relevance are the parameters **K3** and **K4** since they determine the gain in junction efficiency when a vehicle

```
# step1: obtain vehicle speeds and positions via
# C2I-communication

preliminary: based on the vehicle positions and
speed, compute for every vehicle v:
tmin: the earliest possible arrival time at the
junction (using constants MAX_SPEED and
MAX_ACCELERATION),
tmax: the latest possible arrival time assuming a
configurable minimum speed GLOSA_MIN_SPEED

# step 2: create solution matrix S
# h[i] is horizontal vehicle i
# v[i] is vertical vehicle i
for i from 0 to n:
  for j from 0 to m:
     c_horiz = continuation(S[i-1,j], h[i])
     c_vert  = continuation(S[i,j-1], v[j])
     if c_horiz.time_loss < c_vert.time_loss
       S[i,j] = c_horiz
     else
       S[i,j] = c_vert

# read solution matrix backwards and inform
# vehicles using I2C-communication
platoon = [] # list of vehicles passing the light
             # in the same phase
cell = S[n,m]
until cell equals S[0,0]
  platoon.append(cell.vehicle)
  if cell.switch is defined
    # step 3
    Send time cell.switch to all vehicles
    in platoon (assigned_switching_time)
    empty the platoon list
  cell = cell.previous_cell
# note that the vehicles which pass the
# intersection in the current phase do not need
# to adapt their speed and consequently receive
# no message

## step 4
vehicles adapt their speed or continue driving at
               their maximum speed
```

Listing 2 Inner loop of the TLS control algorithm which repeats continuously (Variant A1). The subroutine continuation() is described in Listing 1

```
# computation of continuation cell
if d equals d'
  s' = undefined
  t' = max(t, tmin) + K1
  r' = max(t, tmin, r)
else:
  s' = r
  # time at which the vehicle starts driving
  # across the intersection
  z = max(s' + K3, tmin)
  r' = max(z, s' + K3 + K4)
  if tmax < s' + K3
    t' = z + K2
  else:
    t' = z + K1
l' = l + t - (tmin + K1)
p' = c
 return the solution cell with the updated values
               l',t',d',r',s',p'
```

Listing 1 Computation of a continuation cell *c'* for starting cell *c* and vehicle *v*. The continuation cell *c'* describes the state of junction starting in state *c* after the vehicle *v* has passed.

does not have to stop in front of the junction but can drive continuously instead. Thanks to the arrival time range afforded by GLOSA, many more vehicles can potentially pass the junction without stopping.

In variant A2 of the AGLOSA algorithm the solution matrix has an additional dimension with the values (H,V) and requires more continuations to be evaluated. Also, when assembling the solution from the solution matrix *S*, the initial cell must be chosen as the optimum from *S[n,m,H]* and *S[n,m,V]*. The other methods remain unchanged.

```
# step 2: create solution matrix S (Variant A2)
for i from 0 to n:
  for j from 0 to m:
     c_hhoriz = continuation(S[i-1,j,H], h[i])
     c_hvert  = continuation(S[i,j-1,H], v[j])
     if c_hhoriz.time_loss < c_hvert.time_loss
       S[i,j,H] = c_hhoriz
     else
       S[i,j,H] = c_hvert
     c_vhoriz = continuation(S[i-1,j,V], h[i])
     c_vvert  = continuation(S[i,j-1,V], v[j])
     if c_vhoriz.time_loss < c_vvert.time_loss
       S[i,j,V] = c_vhoriz
     else
       S[i,j,V] = c_vvert
```

Listing 3 Computation of the solution matrix *S* for AGLOSA variant A2 with additional matrix dimension.

To illustrate the workings of variant A1 of the AGLOSA algorithm we give an example of the solution matrix *S* for the following input: $h_1 = (6,14)$, $v_1 = (4,10)$, $v_2 = (7,15)$ consisting of 1 vehicle from the horizontal direction and 2 vehicles from the vertical direction. Table I lists cell contents from S in the order in which they are computed during step 2 (**Listing 1**). The algorithm constants where set to ***K1****=1.4,* ***K2****=2,* ***K3****=5,* ***K4****=5*.

TABLE I. CELLS OF SOLUTION MATRIX *S* FOR EXAMPLE INPUT. COLUMN NAMES ARE VARIABLES FROM **LISTING 2**

i	**j**	**l**	**t**	**d**	**r**	**s**	**p(i,j)**
0	0	0	0	H	0	-	-
0	1	1	6.4	V	10	0	0,0
0	2	1	8.4	V	10	-	0,1
1	0	0	7.4	H	6	-	0,0
1	1	7.6	13	H	16	6	0,1
1	2	10.6	17	H	20	10	0,2

The computed solution is to switch at time 0 and time 10 for a total time loss of 10.6. This is obtained by starting in the last row and working backwards according to the cell indicated in column **p** (for *p=0,2* continue with the row where *i=0* and *j=2*).

The given example input is close to a threshold where the optimal switching plan changes: If vehicle h_1 were to arrive 1 second earlier it would have been better to switch only once at time 5 for a total time loss of 10.4 (note that the junction starts with green in the horizontal direction).

IV. SIGNAL PLAN OPTIMIZATION IN THE GENERAL CASE

The time complexity of the AGLOSA algorithm applied to the scenario from the previous section is $O(n \times m)$ for variant A1 and $O(n \times m \times p)$ for variant A2 where *p* is the number of different 'green'-phases. In the example, $p = 2$ since only the directions *H* and *V* where considered. Alternatively, we can bound the complexity of variant A1 as $O(d^p)$ and A2 as $O(d^p \times p)$ respectively, where *d* is the average number of vehicles for each phase. This is, because the incoming vehicle stream for each phase adds another dimension to the solution matrix *S*.

The example scenario only considered conflicting stream of traffic. With a minor increase in book-keeping one can keep track of unconflicting streams (i.e horizontal left-to-right and right-to-left). This affects the computation of continuations since every unconflicting stream maintains its own value for ***t***, the time at which the next vehicle may enter the junction. The time complexity of the algorithm is not affected.

One possible enhancement for reducing the time complexity would be to analyze incoming vehicles for platoon-structures beforehand and perform optimization not for individual vehicles but rather for platoons. We suppose that this strategy should be effective wherever traffic lights are closely spaced and thus vehicles naturally arrive in platoon structure.

V. SIMULATION RESULTS

For the simulation scenario described in section III we generated Poisson-distributed traffic demand with flow rates between 200 and 2000 vehicles for both incoming directions for a duration of 100000 seconds. We only considered scenarios where the sum of flow-rates from both directions was below 2200 vehicles per hour. Simulations were performed using the open source traffic simulation suite SUMO [6]. The control algorithms for the traffic light were implemented in python and connected to the running simulation via SUMO's python interface (TraCI). We report average time loss and maximum time loss for the following algorithms.

- **F:** fixed control (formula according to Webster [7])
- **R:** reference adaptive control (using time headways measured at induction loops)
- **G:** fixed control as in **F** and GLOSA
- **B:** the AGLOSA algorithm variant A2 with the GLOSA-part switched off
- **A1:** the AGLOSA algorithm variant A1
- **A2:** the AGLOSA algorithm variant A2

TABLE II. AVERAGE TIME LOSS (SECONDS)

vehicles/hour H,V	**F**	**R**	**G**	**B**	**A1**	**A2**
200,200	7.99	6.53	7.35	1.65	1.35	1.24
500,200	9.30	7.77	7.93	2.92	2.38	2.13
500,500	12.39	9.32	11.52	5.40	3.98	3.73
1000,200	13.51	8.59	12.56	4.59	3.89	3.46
1000,500	24.88	12.75	24.65	9.76	7.00	6.68
1500,200	15.21	10.41	27.90	9.43	8.49	7.62
1000,1000	74.39	52.35	117.24	31.26	21.22	20.74
1500,500	63.85	48.08	117.99	25.87	17.82	17.23
2000,200	132.39	133.92	160.21	19.71	17.31	17.43

TABLE III. MAXIMUM TIME LOSS (SECONDS)

vehicles/hour H,V	**F**	**R**	**G**	**B**	**A1**	**A2**
200,200	41	34	32	36	34	25
500,200	87	44	74	76	70	76
500,500	58	51	45	46	44	43
1000,200	237	99	237	121	124	107
1000,500	121	84	134	90	103	68
1500,200	251	183	670	389	445	389
1000,1000	186	186	158	153	137	139
1500,500	460	361	463	209	151	186
2000,200	1435	1448	1783	4990	8961	9970

Table II shows the average time loss for each vehicle for different levels of demand from both directions. As can be seen both variants of the AGLOSA algorithm outperform all other algorithms in all considered demand scenarios. The improvement in time loss from A1 to A2 is 7% on average. This leads us to believe that the solutions are quite close to the, yet untested, optimal variant A3. The result for the demand scenario (2000, 200) where A1 slightly outperforms A2 may be explained by the fact that both variants are "only" heuristic and a simpler heuristic may perform better under some conditions. It also must be noted, that the performance of the AGLOSA algorithm depends on the correct choices for the parameters K1 and K2 (time for passing the intersection at full speed or when accelerating from a stop). The underwhelming performance of the plain GLOSA-algorithm (G) in high-demand scenarios is surprising and warrants further research.

Table III shows the maximum time loss for each demand scenario. One can see that the AGLOSA-algorithm variants have competitive time losses at low or symmetrical demands but excessive time losses at high asymmetrical demand. This however is simply due to the fact that the currently used objective function minimizes average time loss with no regard to maximum time loss and no constraints on maximum phase durations.

When comparing the overall performance of the different algorithms it can be seen that the biggest jump in performance lies between regular adaptive control (**R**) and optimal adaptive control (**B**). We take this as a general encouragement to exploit V2I messages for improving junction control. However, even though optimal adaptive control by itself is already quite good, it can be further improved by combining it with GLOSA technology as in the AGLOSA algorithm (variants **A1, A2**).

In conclusion, we note that the simultaneous combination of adaptive control and green-light-optimal-speed-advisory with the AGLOSA algorithm, allows for a very notable increase in the performance of controlled junctions when compared to alternative state-of-the-art algorithms

REFERENCES

[1] Forschungsgesellschaft für Straßen- und Verkehrswesen – FGSV: *Beispielsammlung zu den Richtlinien für Lichtsignalanlagen* (RiLSA). Köln 2010

[2] ETSI: Intelligent Transport Systems (ITS); Vehicular Communication, Basic Set of Applications, Part 1: Functional Requirements (TS 102 637-1 V1.1.1). European Telecommunications Standards Institute, Sep. 2010

[3] Katsaros, K., Kernchen, R., Dianati, M., Rieck, D. (2011) Performance study of a Green Light Optimized Speed Advisory (GLOSA) Application Using an Integrated Cooperative ITS Simulation Platform, International Wireless Communications and Mobile Computing Conference (IWCMC), 2011, pp.918-923

[4] Kurt Dresner and Peter Stone. Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 530–537, July 2004.Gartner.

[5] Richard Bellman: *Dynamic Programming*. Princeton University Press, 1957.

[6] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D. (2011). SUMO – Simulation of Urban MObility: An Overview. In Proceedings: SIMUL 2011, The Third International Conference on Advances in System Simulation, Barcelona, 23.-28. Oct. 2011.

[7] F. V. WEBSTER (1958) Traffic signal settings. Road Res. Tech. Paper No. 39, H.M.S.O.