Use of car2car communications to improve efficiency of intersections

Author: Linus Meier

Supervisors: Dr. Monica Menendez
Dr. Ilgin Guler

MSc thesis
Civil Engineering

July 2013

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Index of contents

Use of car2car communications to improve efficiency of intersections

Linus Meier
Ob. Flurweg 20
3072 Ostermundigen
limeier@student.ethz.ch
July 2013

Abstract

Thanks to achievements in communication technology it might be possible for cars to transmit their current kinematical conditions such as position, speed and acceleration to an information pool. With this perfect knowledge of traffic demand intersections can be operated in a way to optimize the course of traffic. In this paper algorithms are proposed to optimize an intersection consisting of two one-way-streets in terms of total delay, number of stops and number of cars forced to reduce their speed. These algorithms are time-discrete and in every time step only one approach is allowed to discharge cars leading to the fact that only a finite number of departures curves are possible. All these departures curves are calculated and the best one is chosen.

Simulations were done for different total demands and variable flow ratios. Compared to a fix-timed traffic light all algorithms deliver significantly better results in their specific purpose. Nevertheless, only the algorithm minimizing delay is unrestrictedly feasible to implement in reality. It not only minimizes delay but also improves traffic in terms of number of stops and number of braking cars. For high demand the algorithm minimizing braking cars tends to stop all cars on one approach and no cars on the other. This is the best solution in terms of number of braking cars but generates huge delay. The algorithm minimizing number of stops improves traffic very similar to the algorithm minimizing total delay but the calculation time is almost 200 times longer.

1 Introduction

At intersections traffic needs to be controlled. This can be done with priority signalization (stop, no priority, roundabout, right-before-left) or traffic lights (fix-timed or traffic actuated). Priority signalization and fix-timed traffic lights can be inefficient. Traffic actuated traffic lights are widely used for intersections with high traffic demand. Because their algorithms use only few input (obtained by loop detectors) and the sequences of red and green are subject to restrictions such as minimal green time, minimal red time, etc. there might be space to improve efficiency at intersections in terms of total delay and number of stops.

Thanks to achievements in communication technology it might be possible that every single car shares its current position, speed, acceleration, etc. with an intersection operator, which calculates the best possible course of traffic by using its perfect knowledge of traffic demand. The intersection operator then addresses to every single car and transmit the order to stop or go. This intersection operator represents a notional traffic light. The sequences of reds and greens might change that fast that drivers have no chance to react. That's why cars need to be equipped with a fully automatic cruise control system.

Before doing a lot of work researching and developing these technologies, it is important to know how significant improvements in the course of traffic can be achieved after all. The results of this research can serve as an idea of how the algorithms to control traffic could look like and what kind of results can be expected.

The goal of this paper is to find algorithms to calculate the sequence of notional reds and greens to minimize total delay or number of stops by knowing the exact traffic demand. Mathematical models (using MATLAB) are developed for two intersecting one-way streets without turns from the one street to the other.

In terms of paper layout, initially a short literature review is presented. Subsequently, the description of the developed algorithms and the simulations and their results are discussed. Finally, there will be conclusions of this paper and suggestions of further work on this particular topic.

2 Literature review

Two papers were proposed as basic literature for this work: Zohdy and Rakha [1] developed a game theory algorithm for an intersection-based cooperative adaptive cruise control system. Lee and Park [2] developed a cooperative vehicle intersection control algorithm under the connected vehicle environment.

[1] is the first paper existing, in which the goal is to minimize total delay explicitly. The control algorithm is simulated in a micro simulation of an intersection of two streets without turning cars. In every simulation there is one car per approach arriving. With the current kinematical conditions of every single car conflicts are calculated. If conflicts occur, speeds are adjusted in such a way that total delay is minimized and conflicts are avoided. Because there is only one car per approach per simulation arriving, it is not possible to investigate the effects of discharging cars in groups.

In [2] a complete intersection with turning cars in every direction is modeled. The trajectories of the vehicles are adjusted to minimize the overlaps of stays of cars in the conflict zone. Using very detailed kinematic laws, this leads to a very complicated mathematical formulation, which is hard to solve. The used algorithms to solve this problem aren’t always able to find a solution. That’s why a control algorithm was introduced to discharge the intersection after a non-solution situation. Always when this control algorithm has to run, the traffic course is not fitted to the current demand and total delay is not optimized. If one were able to always find a solution and the control algorithm was not needed, this intersection controller should lead to optimized traffic course.

These two models use detailed kinematical laws to describe the behavior of every single car. If applied properly such models deliver realistic results but the requirements to the computers are huge. In the models proposed in this thesis the description of the traffic is much simpler (i.e. with cumulative arrival and departure curves). This leads to the fact that calculations are quickly done and solutions are always found. Nevertheless, total delay and number of stops can be calculated for every single car.

3 The models in detail

3.1 General considerations

The considered intersection consists of two one-way streets and turning is not possible. This is the most elementary intersection.

The algorithms to optimize traffic are based on cumulative curves. The arrival curves of both approaches are known due to car2car communication (i.e. every car's position and speed are known so that arrival curves at the intersection can be calculated). The departure curves are fitted to the arrivals to minimize total delay or number of stops. In Fig. 1 there is an example of departure curves, which are fitted to the arrival curves in a manner to minimize total delay.

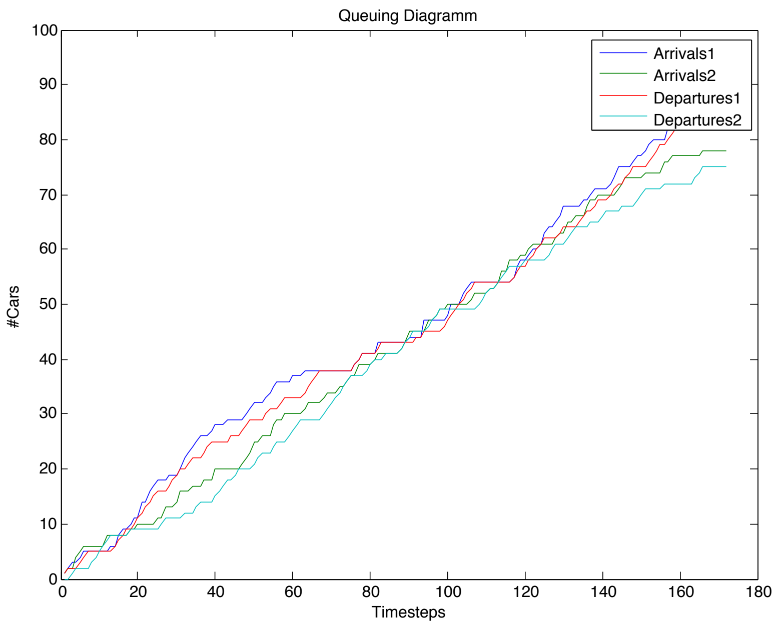

Fig. 1

In continuous time and space there is an infinite amount of possible departure curves. That's why the time is discretized into time steps. The duration of a time step is subject to some restrictions:

- At least one car has to be able to discharge during one time step.
- The product of the saturation flow times the time step duration has to be an integer (i.e. for a saturation flow of 0.5 veh./sec. and a time step duration of 3 sec. there would be 1.5 cars discharging per time step. Of course there can't be half a car, so the time for this 0.5 car would be lost).

- The shorter the time step duration is, the more possible departure curves can be computed and the higher is the possibility that the best departure curve can be found.

Every time step can feature two conditions: Either approach 1 or approach 2 is allowed to discharge. The departure curves can now be computed based on the condition of every time step:

A1 = arrivals approach 1, A2 = arrivals approach 2, D1 = departures approach 1, D2 = departures approach 2, t = time step, Δt = time step duration, μ = saturation flow

Condition 1:

$$D1(t) = \min[A1(t);\ D1(t-1) + \mu \cdot \Delta t]$$

$$D2(t) = D2(t-1)$$

Condition 2:

$$D1(t) = D1(t-1)$$

$$D2(t) = \min[A2(t);\ D2(t-1) + \mu \cdot \Delta t]$$

Using these very simple formulas, the departure curves for every possible combination of conditions of time steps can be calculated. Then, the best departure curve concerning the particular objective (i.e. minimal total delay, minimal number of stops) is chosen.

In simulation, the arrival curves have a limited duration. The best result would be obtained, if this whole duration would be treated in one single examination. In real traffic, the arrival curves are infinite and it's not possible to assume them as known in remote future. That's why there has to be decided how many future time steps are considered to calculate the best departure curves. This number n is very important for the computational speed because the number of possible departure curves m is exponentially depending of n: $m = 2^n$.

After all possible departure curves within the considered future time steps are calculated, the best one is chosen. The intersection operator then transmits the condition of the first considered time step to the cars and traffic flows according to this. That implies that of all the considered time steps only the first one is applied to the real traffic. After that the set of considered time steps moves on and the calculation of the best condition of the next time step starts again.

It might occur that several departure curves are equally good concerning the particular objective. If this is the case it is compared if more of these departure curves start with condition 1 or with condition 2. Then the more frequent condition is chosen for the next time step. If it’s still tie then the decision of the condition of the next time step is random for minimizing delay or set equal to the condition before for minimizing number of stops. As explained later in the minimizing delay algorithm a penalty for changing priority from one approach to the other is implemented. For that reason the decision is random the algorithm can’t find a better condition. If deciding equal as the time step earlier, changing priority would be penalized twice.

3.2 Minimizing total delay

The delay generated in each time step is the product of the excess accumulation (difference between arrival and departure curve at a given time) and the time step duration. The total delay is the cumulatively summed up delay per time step for both approaches. The cumulative delay can be easily calculated for every possible departure curve and then pick the best departure curve.

Two issues are neglected when calculating the delay like this:

- Only the delay generated upstream the intersection is considered. If a car passes the intersection with very low speed, it needs to accelerate downstream the intersection, what generates also a little delay there.
- All cars are treated the same. It doesn’t matter whether a car is stopped and waits directly in front of the intersection and then accelerates from zero speed or going slow in the queue and then pass the intersection with higher speed.

To handle these issues there was an additional delay implemented which every car gets as soon as it leaves the intersection. This additional delay is the time needed to pass the conflict zone. This time is derived using basic kinematical laws. The minimal penalty is given, if the car passes the intersection with free-flow speed.

t_p = penalty time, v = free-flow speed, s = length of conflict zone

$$t_p = \frac{s}{v}$$

If a car is not travelling at free-flow speed but is accelerating when passing the conflict zone, the time needed to pass additionally depends on initial speed and acceleration rate. The initial speed depends on how long the car is already accelerating when arriving the conflict zone (i.e. how long the condition has not changed and it is always the same approach discharging).

t_p = penalty time, v = free-flow speed, s = length of conflict zone, a = acceleration, t_d = discharging time

$$t_p = \frac{-a \cdot t_p + \sqrt{(a \cdot t_d)^2 + 2 \cdot a \cdot s}}{a}$$

If t_d increases (i.e. less switches between condition 1 and condition 2) the penalty gets smaller. That leads to the fact that it is more likely to discharge cars in platoons than to change condition very frequently. For a = 2 m/s^2, s = 5 m and v = 50 km/h the penalty time is plotted in Fig. 2.

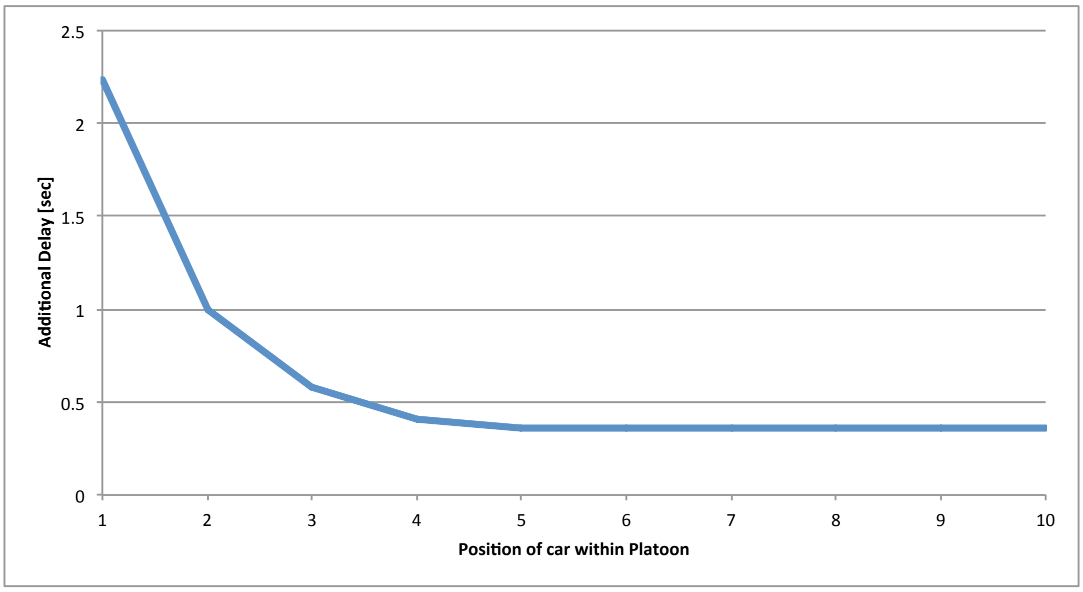

Fig. 2

With this kind of additional delay the issue that all cars are treated the same is respected. The delay, which is generated downstream the intersection is still not perfectly considered (the cars don't necessary travel at free-flow speed after passing the conflict zone) but the mistake is smaller than without this additional delay.

3.3 Minimizing number of stops

Minimizing the number of stops has one big issue: One could say that when the speed is controlled externally it is easy to let cars drive at almost zero speed until they are able to discharge. Total stops can be reduced to zero without gaining anything concerning the flow of traffic.

Because of this two algorithms are developed counting two different properties of traffic:

- Count number of stops under the assumption that cars are either driving at free-flow speed or stopping
- Count number of cars which are forced to reduce their speed neglecting the actual number of stops they perform

3.3.1 Minimizing number of stops at free-flow speed or zero speed

The basic assumption when counting stops is, that the number of stops is equal to the number of notional reds every car experiences. A cumulative counter of stops is implemented in the algorithm. Always when the notional traffic light turns from green to red plus one is added to this counter.

For every car its arrival and departure time is searched. Then its number of stops is calculated:

$N_{stop,i}$ = Number of stops of car i, $R(t_{arr,i})$ = Red counter at time of arrival, $R(t_{dep,i})$ = Red counter at time of departure

$$If\ t_{dep,i} = t_{arr,i}: N_{stop,i} = 0$$

$$Else: N_{stop,i} = R\left(t_{dep,i}\right) - R\left(t_{arr,i}\right) + 1$$

The additional +1 is explained as follows: A car arriving at a red phase and leaving in the next green phase will not see a change in the red counter and its number of stops would be calculated to zero. If a car, however, arrives in a green phase it has to stop only if there is a queue. It again will not see a change in the red counter. If now a car arrives and departs in the same green phase and there is no queue it of course does not have to stop. That’s why there has to be made a distinction.

Because the number of considered future time steps is limited it might happen (especially for high demand) that cars are not leaving in the considered time steps. If so, these cars are assumed to leave at the last considered time step. Only stops generated in the particular considered time steps are counted. If a queue is remaining at the end of the considered time steps, the number of cars in the queue is added to the actual number of stops. This is a penalty of letting grow a queue to long. It only applies for the counting of stops within the optimization algorithm. For the actual number of stops at the very end of the simulation this does not apply. If just adding the number of cars in the remaining queue all these cars are implicitly assumed to stop one additional time after the considered time steps. For high demand this might underestimate the actual number of stops. To take this fact into account, the remaining queue penalty is multiplied with a factor. This factor has to be determined before running a simulation and does not have to be identical for both approaches.

In terms of minimizing stops it might be the best solution to stop all the cars in the approach with less demand once and forever. Of course that's not at all a feasible solution. The remaining queue penalty explained above helps to reduce this problem. But still there has to be a constraint, which does not allow a queue to grow to infinity. This constraint is implemented as a maximum excess accumulation: As soon as one approach reaches the value of the constraint, the optimizing algorithm is turned of and discharging of this approach is forced until the excess accumulation is again lower than the constraint value. If both approaches exceed the value of the constraint, the optimization algorithm works as usual. The influence of the remaining queue penalty and the maximum excess accumulation is investigated later.

3.3.2 Minimizing number of braking cars

Minimizing the number of braking cars is actually the same thing as maximizing the number of cars, which don't even notice that there is an intersection. The algorithm works as follows: for every time step the excess accumulation is calculated. If the excess accumulation increases from zero to a positive number, that's the sign of the first car in the queue. The number of the first car is saved. If the excess accumulation again turns to zero, that's the sign of the last car in the queue. The number of the last car is saved, too. If a car departs after the considered time steps it is assumed to leave at the last considered time step (as above). The number of first cars is equal to the number of queues occurring. The total number of cars forced to brake can now be calculated:

N_{brake} = Number of braking cars, $N_{first,i}$ = Number of first car in queue i,
$N_{last,i}$ = Number of last car in queue i, N_{queue} = Number of queues

$$N_{brake} = \sum_{i=1}^{N_{queue}} \left(N_{last,i} - N_{first,i}\right) + N_{queue}$$

3.4 Differences in the algorithms

What the algorithms actually do and what the differences are, can be explained with a geometrical conception: The minimizing delay algorithm minimized the area between the arrival and the departure curves. The optimization works in both dimensions (i.e. time dimension and number of cars dimension). The minimizing braking cars algorithm only minimizes the total height of all areas between the arrival and departure curves it optimizes only in one dimension (i.e. number of cars dimension). The minimizing stops algorithm optimizes in the time dimension. But it’s not the time per se, which is changed but it’s the time in relation to the number of changes in the notional traffic light.

4 Simulation

4.1 General considerations

Because the arrival curves are generated using a probability distribution, the results underlay certain randomness. That’s why every simulation is done five times. The results presented later represent the mean value of these five simulations.

The algorithms are compared to a fix-timed traffic light. For this traffic light the minimum green time is set to 20 sec. According to this, the green time of the approach with less demand is 20 sec. The green time on the other approach is increased proportionally to the flow ratio (i.e. if the demand on the second approach is twice as high, the green time is twice as long, too).

The total input flow is set to 1’000, 1’500 and 2’000 veh./h and the flow ratio is varied from 1:9 to 1:1. Every simulation is done for a simulation time of 15 min

4.2 Input parameters

The input parameters for all algorithms are: both input flows, both saturation flows, the duration of a time step, the number of considered future time steps and the total number of cars arriving on approach 1 (as a proxy for the simulation time; the number on approach 2 is proportional to the flow ratio).

The minimizing delay algorithm additionally requires the acceleration rate, the free-flow speed and the length of the conflict zone. These kinematical parameters are used to calculate the additional delay for every car leaving the intersection.

The minimizing stops algorithm additionally requires the values of the maximum excess accumulation constraint and the remaining queue penalty factors. The influence of these parameters will be investigated later.

The input flows are variable and every algorithm is run for several total flows with different flow ratios. The saturation flows and the kinematical parameters are relatively easy to determine, they are set to:

- Saturation flow 1 & 2: 1'800 veh./h
- Acceleration rate: 2 m/s^2
- Free-flow speed: 50 km/h
- Length of conflict zone: 5 m

The duration of a time step and the number of considered future time steps are essential for the performance of the models. That's why a sensitivity analysis was done.

4.2.1 Sensitivity of number of considered future time steps

The number of considered future time steps n is actually the duration, which is looked in the future. The bigger n gets, the better the departure curve can be fitted to the arrival curve. n has a big influence of computational speed. Because the number of possible departure curves increases exponentially with n, computational speed gets slower quite quickly with increasing n.

For all three algorithms n was varied between 2 and 8. The simulation was done with a flow of 600 veh./h on both approaches. The results are shown in Fig. 3, Fig. 4 and Fig. 5.

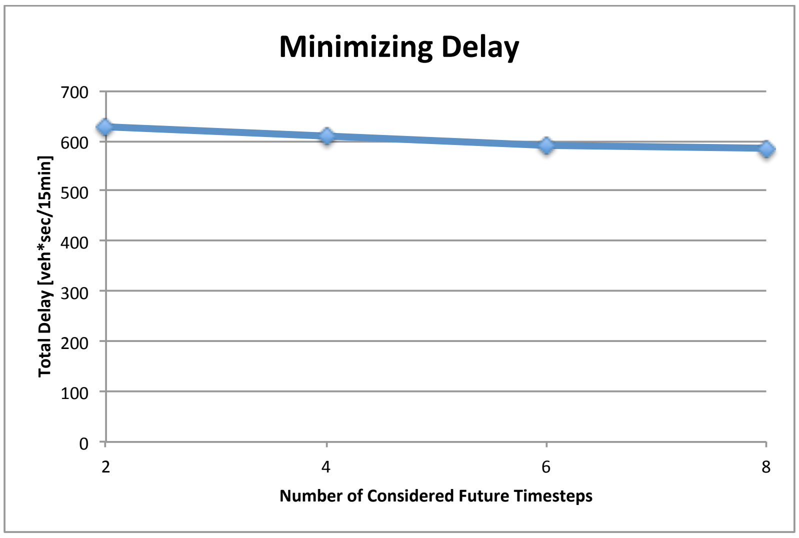

Fig. 3

Minimizing delay: On first sight there is only a slight improvement in the result when n is increased from 2 to 8. The total delay is decreased from 627 to 585 veh.·sec./15 min. This is an improvement of about 7%. Because calculation speed isn't a big issue with this algorithm, n is set to 8. An n bigger than 8 might not be feasible in reality, because then cars driving at free-flow speed would get too far away from the intersection within this time. In networks of streets this might lead to problems because it's not known whether a specific car is driving to the intersection or not.

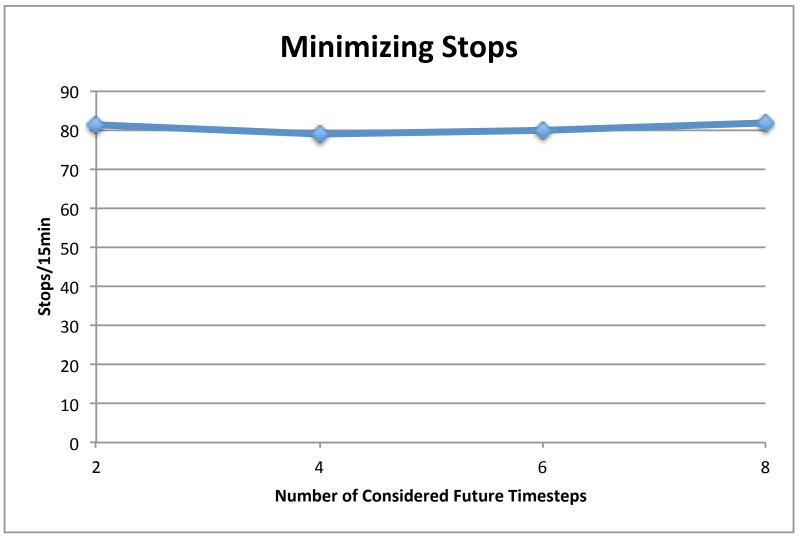

Fig. 4

Minimizing number of stops: It seems that as an increasing n produces a worse result. And actually the result for n = 8 is the worst. But the differences are very small

(79 - 82 stops/15 min). As the standard deviation of the results for n = 2, 4, 6 is 8 and for n = 8 is 7, the difference might be as well random. Because the results of all three algorithms are going to be compared, it’s better to not change the input parameters, so n is set to 8 as well.

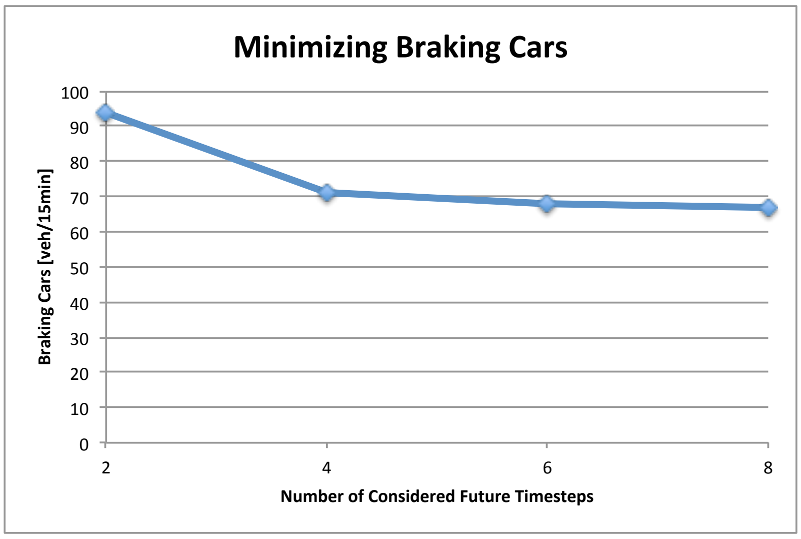

Fig. 5

Minimizing number of braking cars: The results for the minimizing braking cars algorithm show a dependency of n. Especially for low n the result is bad. From n = 4 to n = 8 the improvement of the result is not that big. But again for better compareing the results of all three algorithms, n is set to 8.

4.2.2 Sensitivity of duration of one time step

For the algorithms counting braking cars and stops it is essential that there is only one car arriving or departing per time step. As the saturation flow is set to 1’800 veh./h (= 1 veh/2 sec.), the duration of a time step is set to 2 sec. and not varied. For the minimizing delay algorithm the dependency of the duration of a time step is shown in Fig. 6. The simulations were done for a demand of 600 veh./h on both approaches and durations of a time step from 2 to 10 sec.

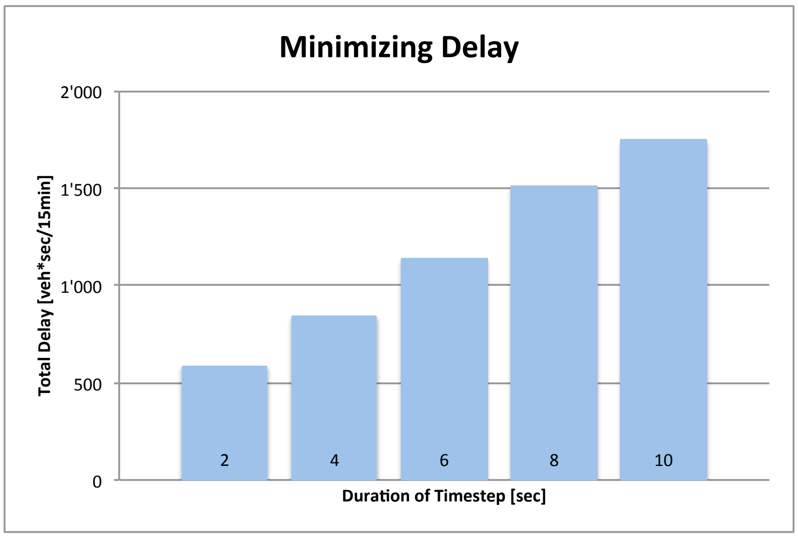

Fig. 6

A strong dependency of the result from the duration of a time step is obvious. The shorter a time step is, the better the algorithm can react to traffic. For long time steps the number of possible departure curves is limited. The duration of a time step is set to 2 sec.

In Fig. 6 the diagram shows columns and not a continuous graph. This is because the product of the duration of a time step and the saturation flow needs to be an integer (see chapter 3.1).

4.2.3 Maximum excess accumulation and remaining queue penalty

The influence of the maximum excess accumulation constraint and the remaining queue penalty is investigated for an undersaturated and an oversaturated total traffic demand and two different flow ratios, respectively. These demands are: 600/600, 400/800, 1'000/1'000 and 800/1'200 veh./h. The factor for the remaining queue penalty is varied independently for both approaches from one to five. Subsequently, particular combinations of the factors are investigated in terms of their sensitivity of the maximum excess accumulation constraint.

600/600 veh./h demand

For a maximum excess accumulation constraint equal to infinity an example of results for different combinations of remaining queue penalty factors look as follows:

Number of stops per 15 min 600/600 veh./h		Factor approach 2				
		1	2	3	4	5
Factor approach 1	1	65	65	65	65	65
	2	65	65	67	67	67
	3	65	64	65	67	67
	4	65	64	64	65	67
	5	65	64	64	64	65

Obviously, the results depend little on these factors. The best results are achieved if approach 1 is punished slightly more than approach 2.

One of the combinations of factors leading to a best result is 3 for approach 1 and 2 for approach 2. The simulation with these inputs leads to:

- Approach 1: 35 stops, 34 braking cars ⇒ 1.03 stops per braking car
- Approach 2: 29 stops, 29 braking cars ⇒ 1.0 stops per braking car

If now the factors are set according to the ratio of stops and braking cars, the remaining queue penalty is penalizing accurately in average. Nevertheless, the simulation showed no further improvement.

Due to randomness, arrivals 1 is a bit smaller than arrivals 2. Punishing the approach with less demand little (but not too much) more might lead to better results. As the differences in demand and in result are very small this statement has to be verified for unbalanced flows.

Even if it is true that punishing a bit more the approach with less demand leads to a better result, for same average demands on both approaches it initially is not known which approach actually shows a little less demand. So the factors should be set equal. In the considered example all combinations with equal factor lead to the same result. Actually, all combinations of remaining queue penalty factor leading to the best result produce exact same course of traffic. In this case the influence of the maximum excess accumulation constraint is investigated. The results of a variable constraint value are shown in the following table:

Maximum excess accumulation	1	2	3	∞
Number of stops	83	73	65	65

For constraint values equal or higher than 3, results don't change. Only for unreasonable small constraint values the result gets worse. The more restrictive the constraint, the worse is the result.

400/800 veh./h demand

For a maximum excess accumulation constraint equal to infinity an example of results for different combinations of remaining queue penalty factors look as follows:

Number of stops per 15 min 400/800 veh./h		Factor approach 2				
		1	2	3	4	5
Factor approach 1	1	53	53	53	53	53
	2	53	53	56	56	56
	3	53	53	53	54	54
	4	53	53	53	53	54
	5	53	53	53	53	53

Again, the results show little dependency of the remaining queue penalty factor. The statement from the analysis of the 600/600 veh./h demand, that more penalizing the approach with less demand leads to better results, can not be confirmed undisputedly. A higher penalty to the approach with less demand leads to the same result as equal penalties for both approaches. However, a higher penalty to the approach with higher demand can leads to worse result.

All combinations of factors leading to the best result are not only equal in terms of number of stops but also the course of traffic is exactly the same leading to:

- Approach 1: 37 stops, 36 braking cars ⇒ 1.03 stops per braking car
- Approach 2: 16 stops, 16 braking cars ⇒ 1.0 stops per braking car

If now the factors are set according to the ratio of stops and braking cars, the remaining queue penalty is penalizing accurately in average. Nevertheless, the simulation showed no further improvement.

Again, the influence of the maximum excess accumulation constraint is investigated for these factors leading to the best result. The results are shown in the following table:

Maximum excess accumulation	1	2	3	∞
Number of stops	64	57	53	53

Again, for constraint values equal or higher than 3, results don't change. Only for unreasonable small constraint values the result gets worse. The more restrictive the constraint, the worse is the result.

1'000/1'000 veh./h demand

For a maximum excess accumulation constraint equal to infinity an example of results for different combinations of remaining queue penalty factors look as follows:

Number of stops per 15 min 1'000/1'000 veh./h		Factor approach 2				
		1	2	3	4	5
Factor approach 1	1	1339	1636	906	618	496
	2	2283	847	1838	1770	1321
	3	1703	1751	634	1258	1738
	4	1616	2261	1356	581	1124
	5	1483	2156	1718	1183	533

For an oversaturated intersection the results depend strongly on the remaining queue penalty factors. First thing to mention is that the best result is achieved by penalizing approach 2 noticeably more than approach 1. This low number of stops is achieved by rarely allowing cars discharge from approach 1. This leads to huge delay and of course is not a feasible solution. Because the demand is balanced, the same phenomenon should be observed when penalizing approach 1 noticeably more than approach 2. But this is not the case. Why? Having a quick look on the queuing diagrams (see Fig. 7 and Fig. 8) helps explaining. The course of traffic does not look that different but in one case there are slightly more changes in priority. If the queue is huge as in these cases, more changes in priority quickly lead to a high number of stops.

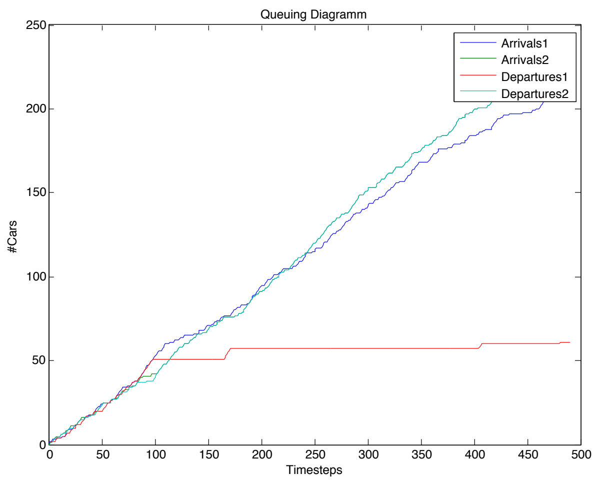

Fig. 7

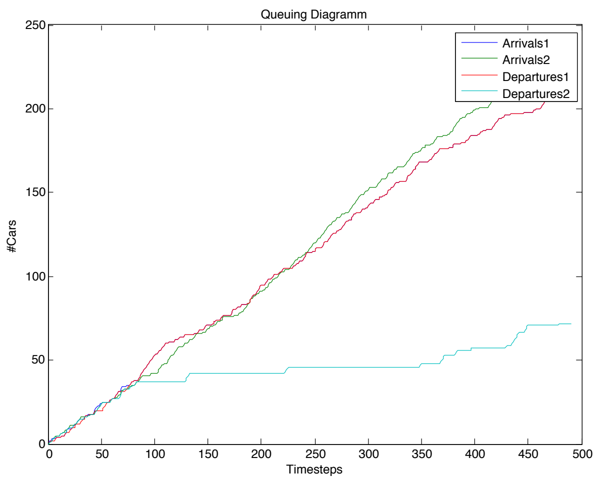

Fig. 8

A second observation is that for equal factors the result improves with increasing factors but it does not converge with factors equal to 5 or less. Further simulations showed that for factors equal to 6 or higher the result converges. The minimum number of stops achieved is 527 stops/15 min. In this case following properties if traffic are calculated:

- Approach 1: 105 stops, 71 braking cars ⇒ 1.48 stops per braking car
- Approach 2: 422 stops, 150 braking cars ⇒ 2.81 stops per braking car
- Average: 527 stops, 221 braking cars ⇒ 2.38 stops per braking car

If the remaining queue penalty factors are set to 1.48 and 2.81, respectively, in average the number of stops of the remaining queue is estimated correctly. Interestingly, doing so leads to a worse result (2'034 stops/15 min).

As the best result is achieved by setting the remaining queue penalty factors equal to 6 or higher but the actual number of stops per braking car is 2.38 in average, overestimating the number of stops for cars in the remaining queue leads to better results.

The influence of the maximum excess accumulation is investigated using both remaining queue penalty factors equal to 6. To make sure to get the best result, using these factors equal to 1 was investigated, too. The results are plotted in Fig. 9.

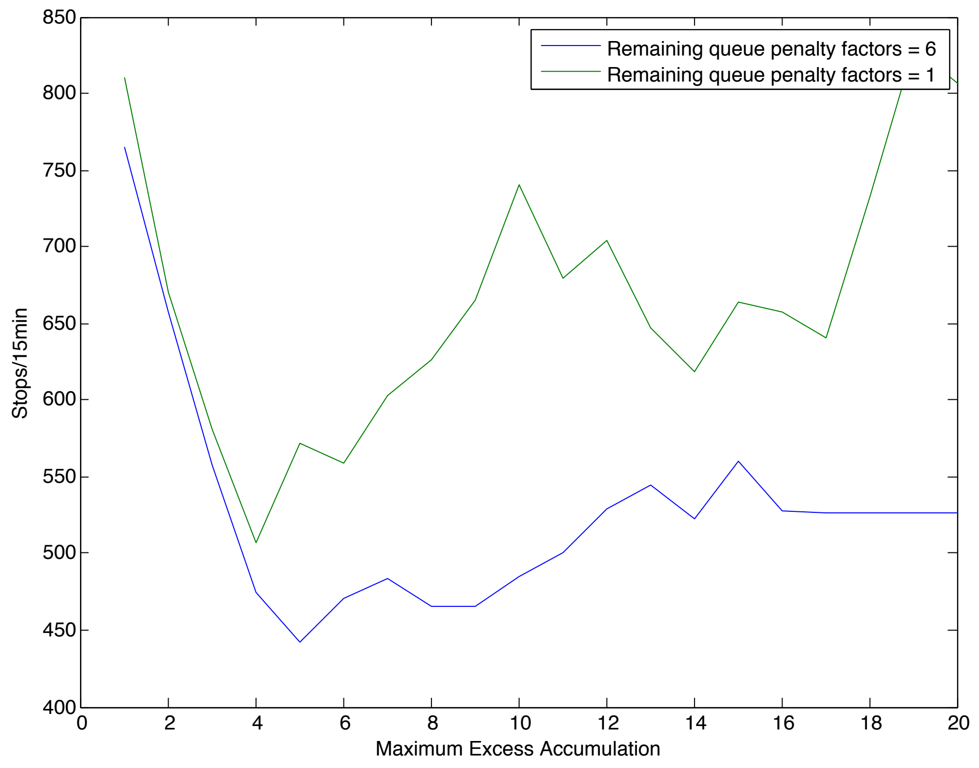

Fig. 9

First of all it is shown that for remaining queue penalty factors equal to 1 the results cannot be improved compared to factors equal to 6.

By implementing a maximum excess accumulation constraint the result is improved. Instead of 527 only 442 stops/15 min are achieved with a constraint value of 5. Improving a result by adding a constraint seems strange. Why is the result achieved with a constraint value of 5 not achieved as the value is set to infinity? This phenomenon is explained investigating the development of the number of stops over time (see Fig. 10). The number of stops without a constraint (i.e. a constraint value equal to infinity) initially is lower than the one for a constraint value of 5. In this period a queue is built up. As the queue grows larger, the remaining queue penalty causes this queue to be penalized highly, which leads to discharging from the approach with a queue. If the queue is already too large, it is not very likely to discharge the whole queue at once because this would lead to a large queue on the other approach. So the queue is moved step-by-step leading to a higher number of stops.

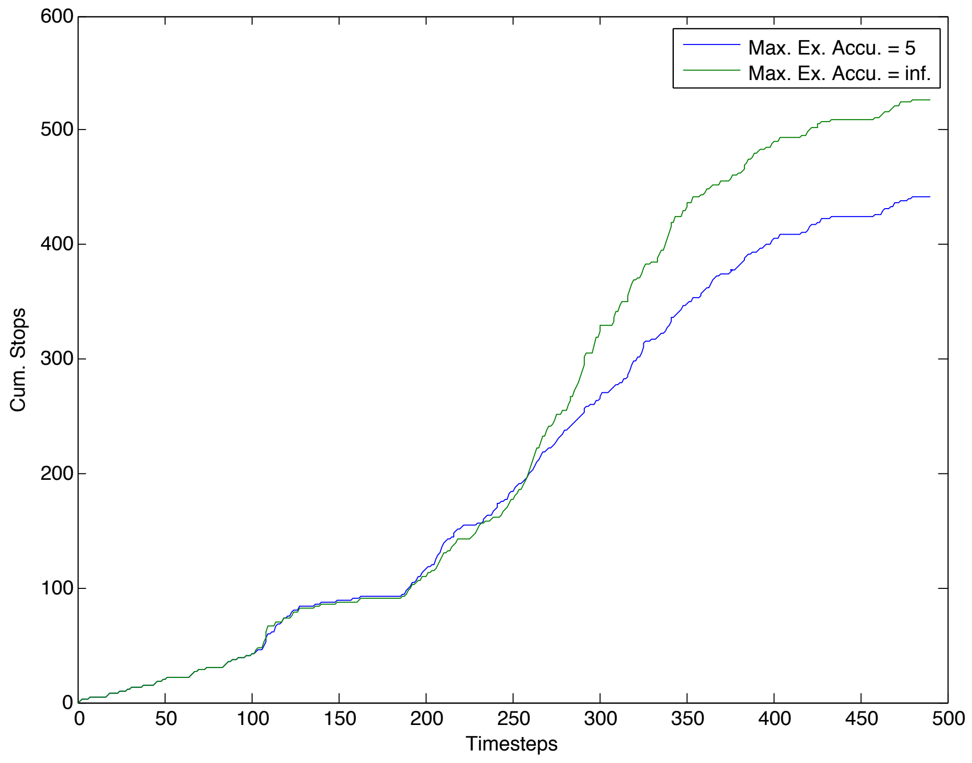

Fig. 10

Another observation in Fig. 9 is a bit of scatter as seen for a constraint value from 12 to 16. Scatter such as this does sometimes occur near the best result, too. The phenomenon of this scatter is investigated, too: Having a look on the development of the excess accumulation over time, it was seen that the total excess accumulation is exactly the same for different constraint values with a small difference. But the split of the approaches is not the same. So these local maximum occur, when a constraint value leads to the same excess accumulation as the constraint value plus one but the optimizing algorithm is turned off earlier.

The position of the global minimum is hard to predict. The constraint value has to be high enough to let the optimizing algorithm work properly but not too high to prevent a queue too long.

800/1'200 veh./h demand

For a maximum excess accumulation constraint equal to infinity an example of results for different combinations of remaining queue penalty factors look as follows:

Number of stops per 15 min 800/1'200 veh./h		Factor approach 2				
		1	2	3	4	5
Factor approach 1	1	3030	1719	980	576	365
	2	3199	1438	2388	1717	1075
	3	2533	3702	750	2302	2315
	4	2114	3572	2329	602	1474
	5	1515	2880	3702	1806	481

The dependency of the result from the remaining queue penalty factors show a quite similar picture as the one investigated for the 1'000/1'000 demand. And the explanations are the same, too. The best result (365 stops/15 min) is achieved by generating a long queue and huge delay on one approach. This is not further discussed. Again there is no converging for equal remaining queue penalty factors within a value of five. The result converges if the factors are set to 7 or higher. The minimum number of stops achieved is 478 stops/15 min. By varying the maximum excess accumulation constraint, the number of stops can even be improved to 456 stops/15 min for a constraint value of 7. All the graphs look quite similar to the ones for a 1'000/1'000 demand and explanations are exactly the same. This is not further discussed.

Conclusions

For undersaturated intersections the remaining queue penalty factors and the maximum excess accumulation constraint show little influence to the result. This is comprehensible, as there are no long queues built up. Opposite to this the influence on oversaturated intersections is very big.

The remaining queue penalty factors have to be set equally because unbalanced factors tend to create very long queues and huge delay on the approach less penalized. The values of the factors should be quite high: Overestimating the number of stops for the cars in the remaining queue never leads to a worse result but underestimating does so (i.e. for all simulations above the result for factors equal to 1'000 were the same as for the smallest value after converging). For all simulations done later the remaining queue penalty factors are set to 10, respectively.

Implementing a maximum excess accumulation constraint does lead to better results. This seems strange but a comprehensible explanation is found. The one constraint value leading to the best result is not easily found. Two different random arrivals based on the same average traffic demand, can show different constraint values leading to the best result. According to this the constraint value is implemented in the codes as variable and is varied from 1 to 8. The upper limit of eight is explained as follows:

- Not one single simulation run showed the best result for a constraint value higher than 8.
- As the number of considered future time steps is set to 8 and the saturation flow is one car per time step, for a constraint value of 8 there is always the possibility of discharging the whole queue in one calculation cycle.

4.3 Generating arrival curves

The arrival curves are generated using a probability function. The headways are assumed to follow an exponential probability distribution. The expectancy value of the headway is the inverse of the average flow. This is exact for small flows and satisfactorily good for high flows [3]. The minimum headway is according to the saturation flow set to 2 sec.

5 Results

5.1 Minimizing delay

The results of the simulations are shown in Fig. 11 - Fig. 18. Fig. 11 - Fig. 16 show the absolute result of the simulations while Fig. 17 and Fig. 18 show the difference between the minimizing algorithm and the fix-timed traffic light.

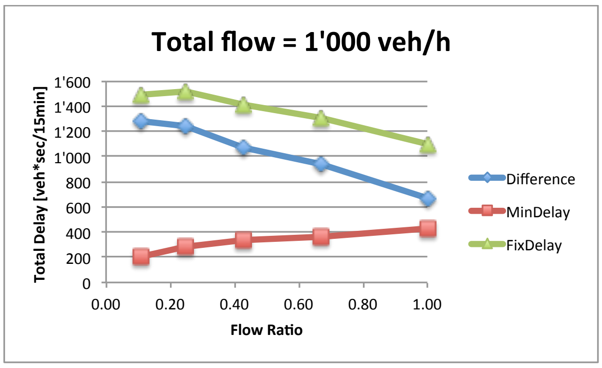

Fig. 11

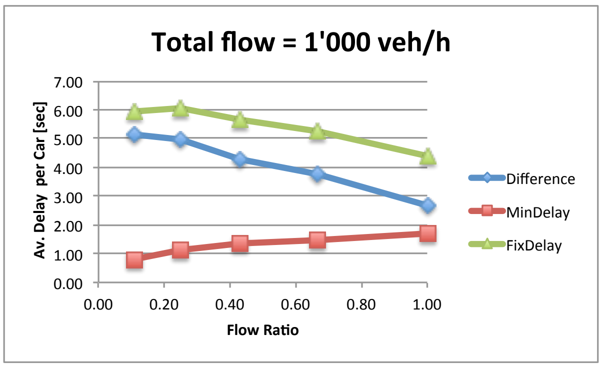

Fig. 12

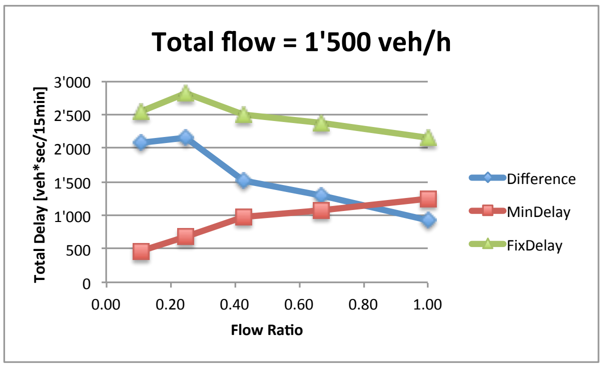

Fig. 13

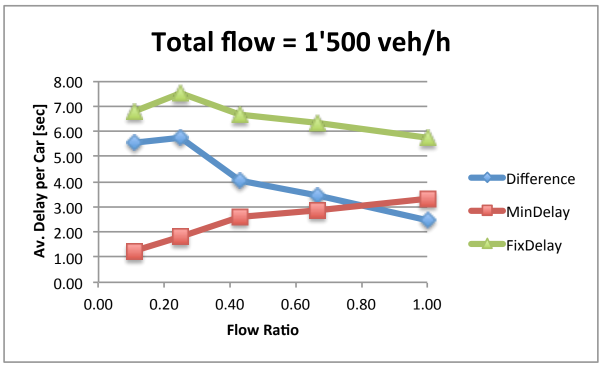

Fig. 14

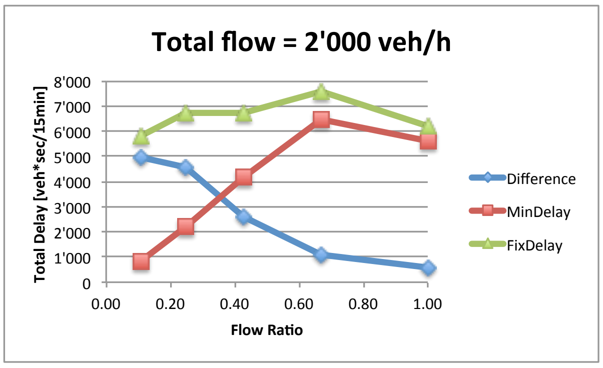

Fig. 15

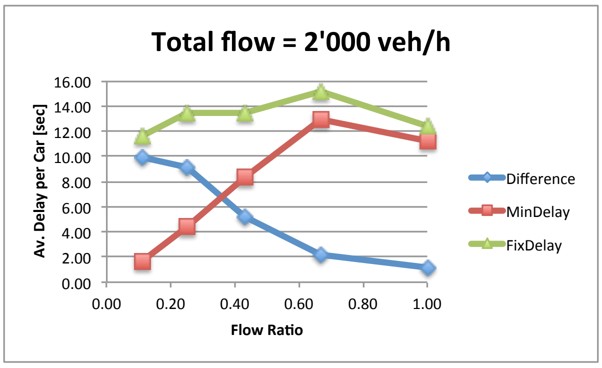

Fig. 16

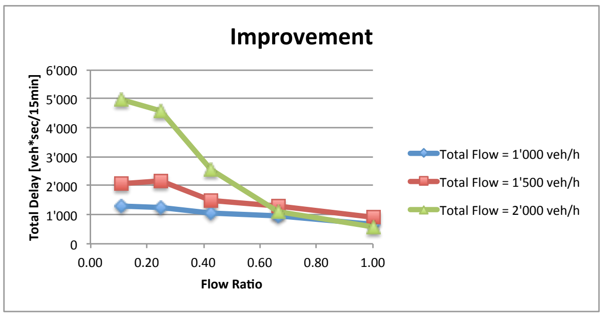

Fig. 17

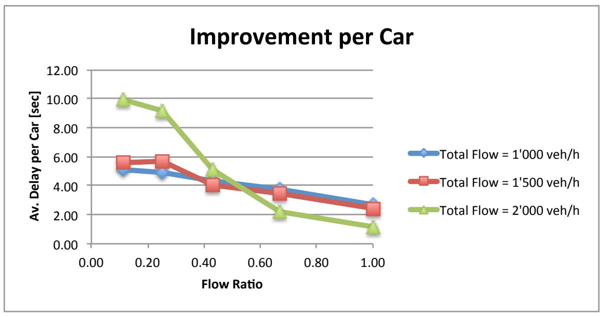

Fig. 18

Every single simulation showed an improvement in terms of total delay. The best improvement is achieved for a total flow of 2’000 veh./h and a flow ratio of 1:9. The average delay per car is reduced almost 10. sec. The least improvement is achieved for a total flow of 2’000 veh./h and a flow ratio of 1:1. The average delay per car is reduced 1.2 sec.

For total flows of 1’000 and 1’500 veh./h regularities in the curves are spotted. These regularities show that the performance of the optimization algorithm decreases with increasing flow ratio. At the same time the performance of the fix-timed traffic light increases with increasing flow ratio. This leads to the fact that the improvement is bigger for small flow ratios.

For a total flow of 2’000 veh./h, which leads to an oversaturated intersection, it’s more difficult to spot regularities in the results. Because the standard deviation of the results increases strongly with increasing total flow (see Fig. 19) more simulations should be done to make precise statements. Except for flow ratios of 2:3 and 1:1 the performance of the optimization algorithm decreases with increasing flow ratio. The improvement is again larger for small flow ratios.

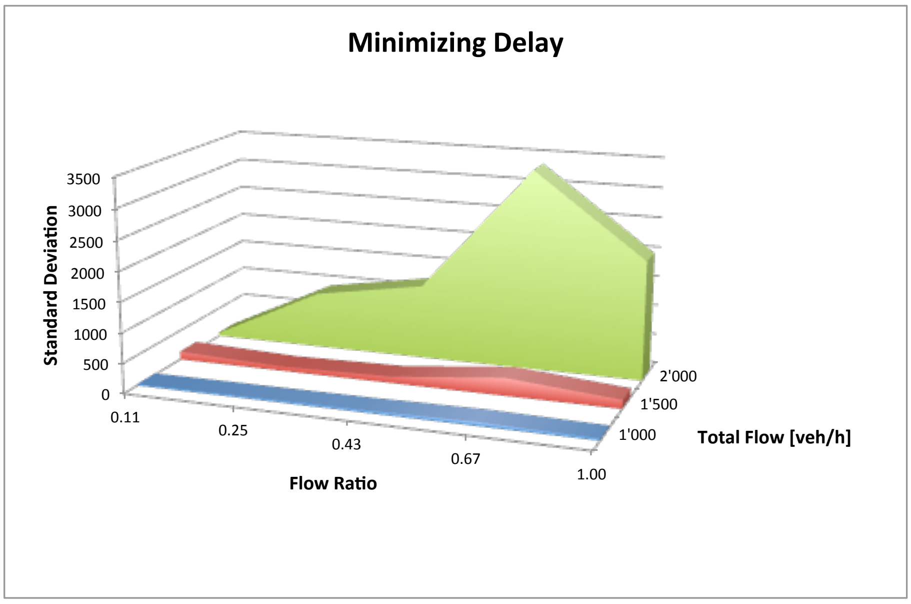

Fig. 19

5.2 Minimizing number of stops

The results of the simulations are shown in Fig. 20 - Fig. 27. Fig. 20 - Fig. 25 show the absolute result of the simulations while Fig. 26 and Fig. 27 show the difference between the minimizing algorithm and the fix-timed traffic light.

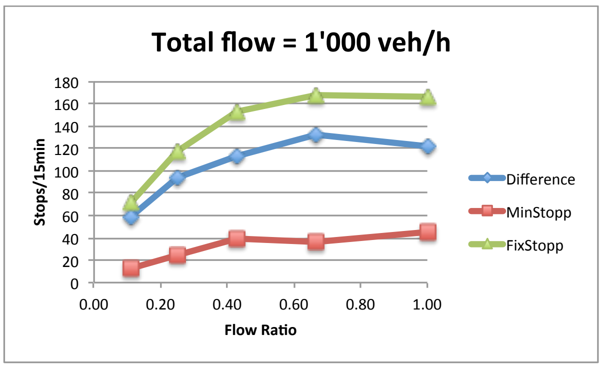

Fig. 20

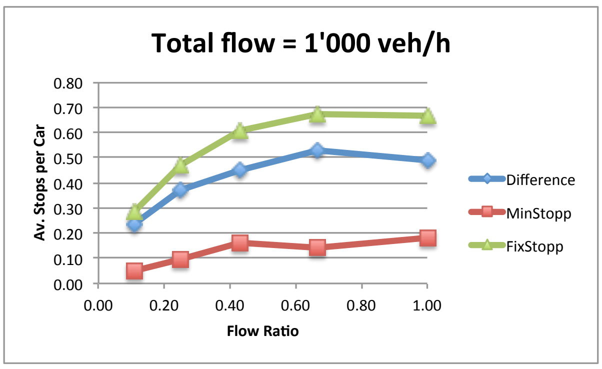

Fig. 21

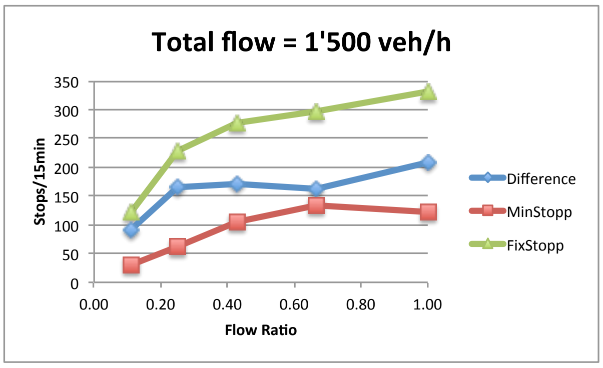

Fig. 22

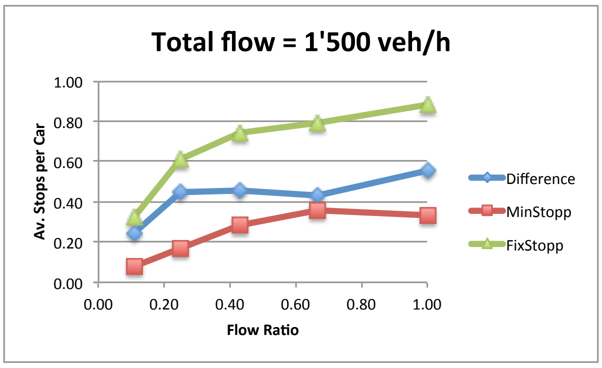

Fig. 23

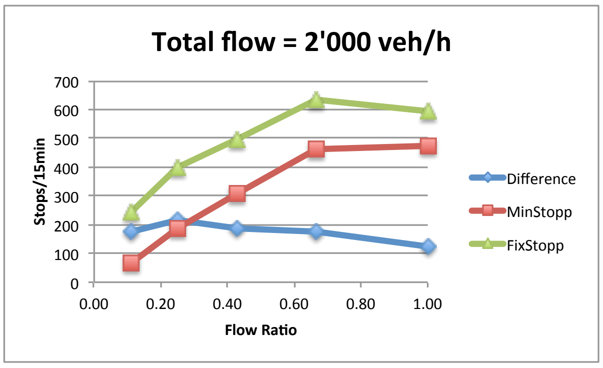

Fig. 24

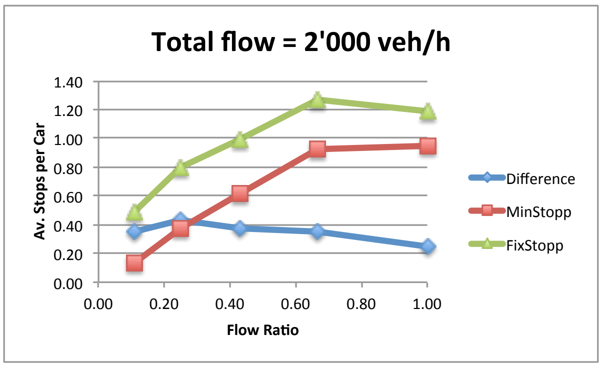

Fig. 25

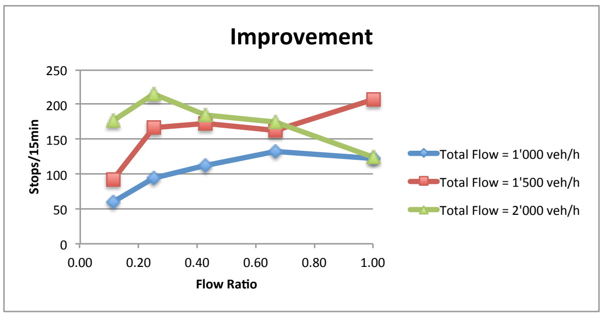

Fig. 26

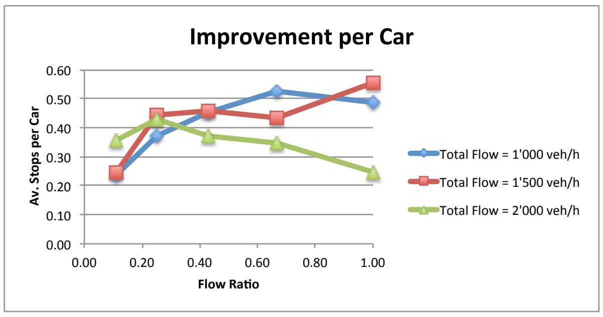

Fig. 27

Every single simulation showed an improvement in terms of number of stops. The best improvement is achieved for a total flow of 1’500 veh./h and a flow ratio of 1:1.

The average value of the improvement is -0.55 stops per car. The least improvement is achieved for a total flow of 1’000 veh./h and a flow ratio of 1:9. The average value of the improvement is -0.24 stops per car.

For all three total flows regularities in the curves are spotted. Both the optimization algorithm and the fix-timed traffic light perform worse with increasing flow ratio. For total flows of 1’000 and 1’500 veh./h the performance of the optimization algorithm decreases slower than the one of the fix-times traffic light. That leads to increasing improvement with increasing flow ratio. For a total flow of 2’000 veh./h the performance of the optimization algorithm decreases faster than the one of the fix-timed traffic light. That leads to decreasing improvement with increasing flow ratio.

The standard deviation of the results is shown in Fig. 28. It’s much larger for an oversaturated intersection.

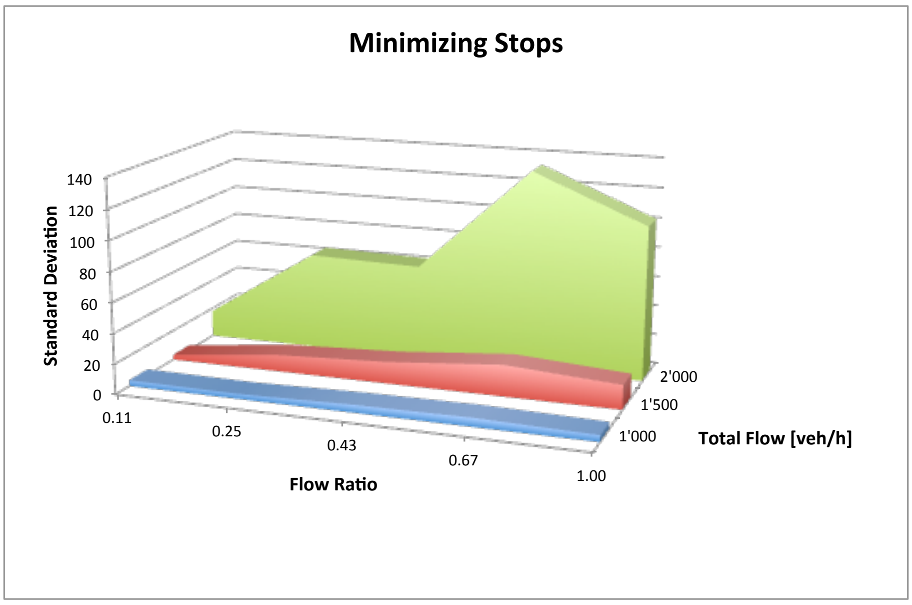

Fig. 28

5.3 Minimizing number of braking cars

The results of the simulations are shown in Fig. 29 - Fig. 36. Fig. 29 - Fig. 34 show the absolute result of the simulations while Fig. 35 and Fig. 36 show the difference between the minimizing algorithm and the fix-timed traffic light.

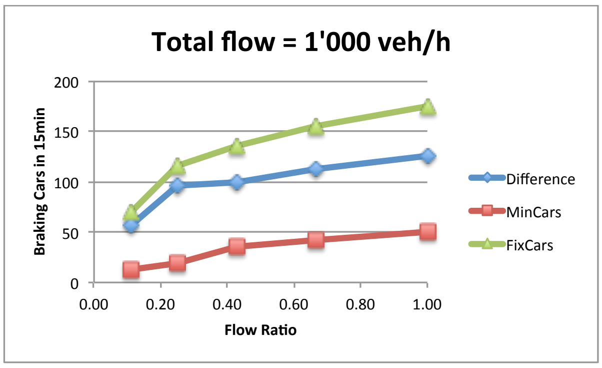

Fig. 29

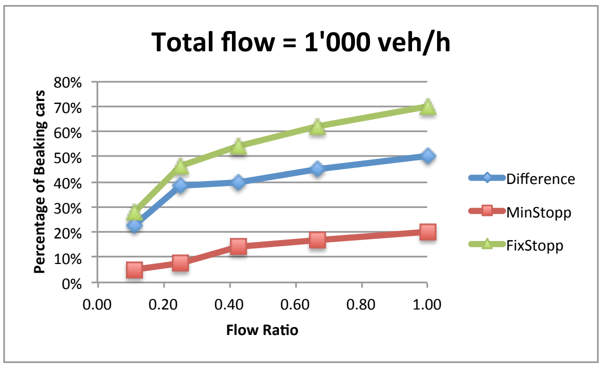

Fig. 30

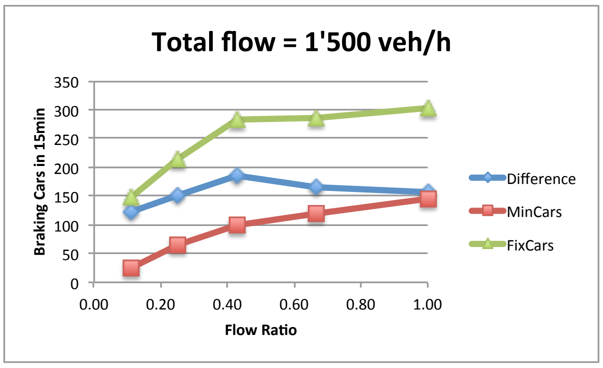

Fig. 31

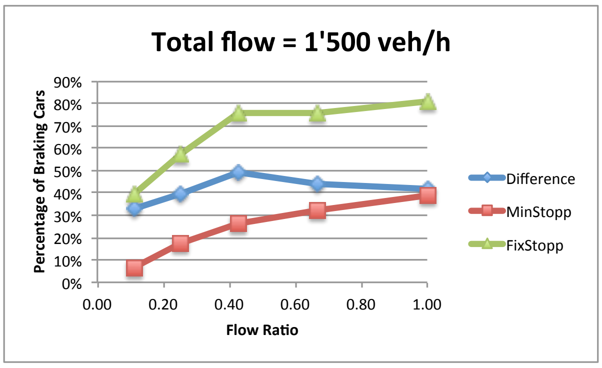

Fig. 32

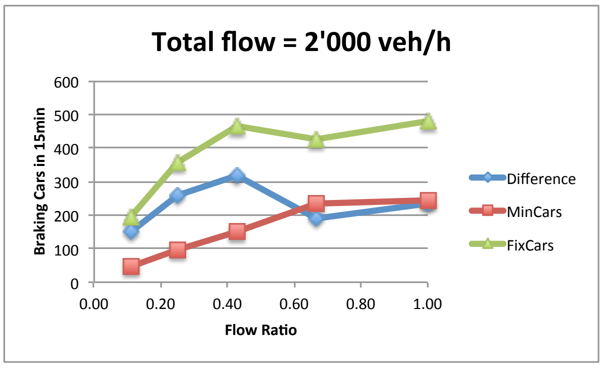

Fig. 33

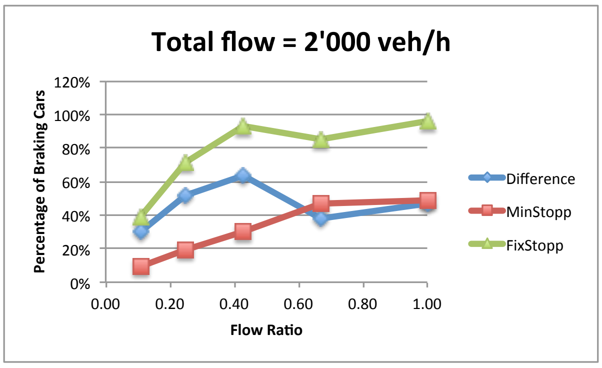

Fig. 34

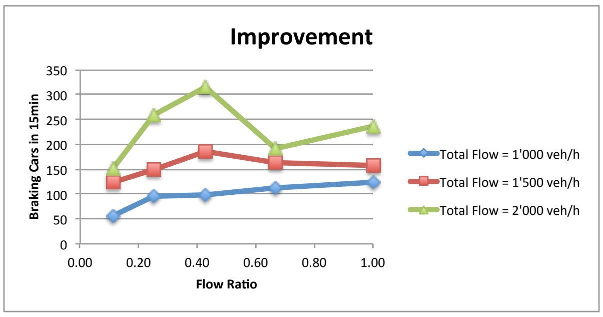

Fig. 35

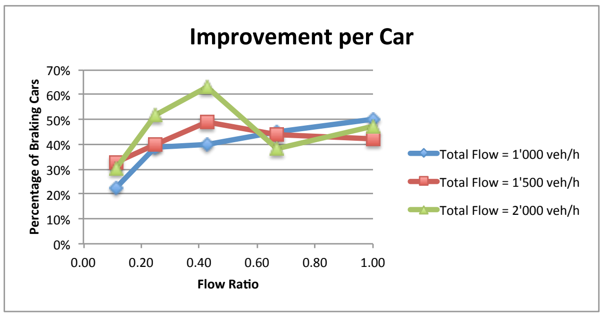

Fig. 36

Every single simulation showed an improvement in terms of number of braking cars. The best improvement is achieved for a total flow of 2’000 veh./h and a flow ratio of

3:7. The percentage of braking cars drops from 93% to 30%. The least improvement is achieved for a total flow of 1’000 veh./h and a flow ratio of 1:9. The percentage of braking cars drops from 28% to 5%.

For all three total flows regularities in the curves are spotted. Both the optimization algorithm and the fix-times traffic light perform worse with increasing flow ratio. The curves of the improvement underlay certain scatter. A quick look on the standard deviation in Fig. 37 shows even the standard deviation itself has a scattered shape (e.g. for a total flow of 2’000 veh./h: Why is it very small with a flow ratio of 3:7 but very high for a flow ratio of 2:3?). More simulations resulting in a larger sample would be needed to make more precise statements.

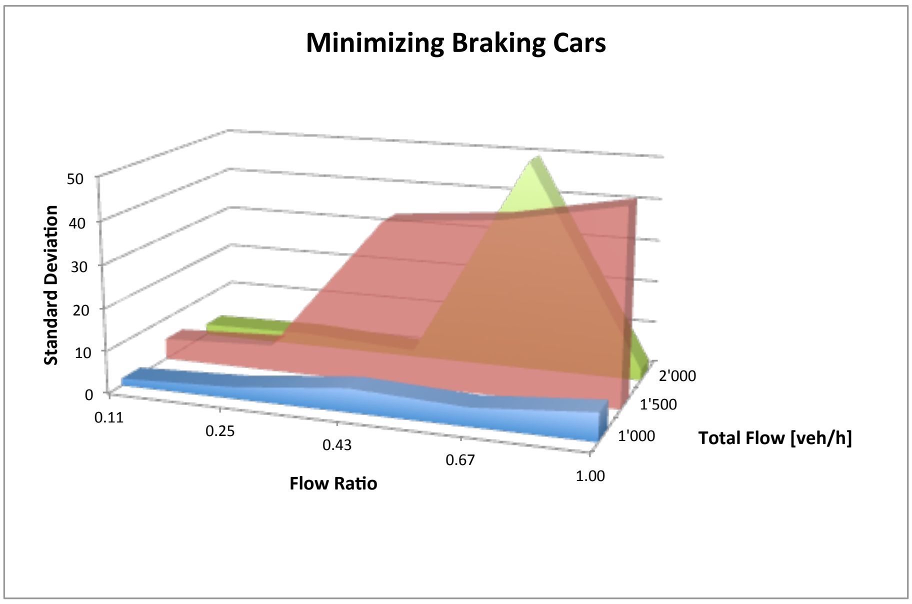

Fig. 37

5.4 Comparison of the algorithms

For every algorithm it was investigated how optimizing traffic for one specific property affects the other properties of traffic. Additionally to the properties total delay, number of stops and number of cars forced to brake a fourth property was studied: the number of changes of the notional (or real for the fix-timed traffic light) traffic light. This number of changes is a proxy for safety. The more the priority changes from one approach to the other, the more possible conflicts occur.

This investigation was done for two different total flows and two different flow ratios, respectively. One total flow represents an oversaturated intersection. The combination of flows are: 600/600, 400/800, 1’000/1’000 and 800/1’200 veh./h.

5.4.1 Changes of traffic light

Fig. 38 - Fig. 41 show the numbers of changes of the notional or real traffic light for all three optimization algorithms and for the fix-timed traffic light.

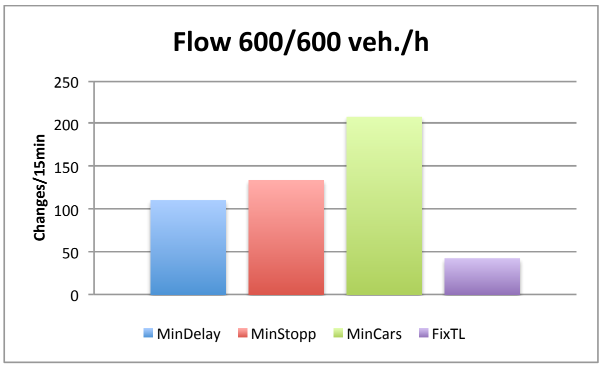

Fig. 38

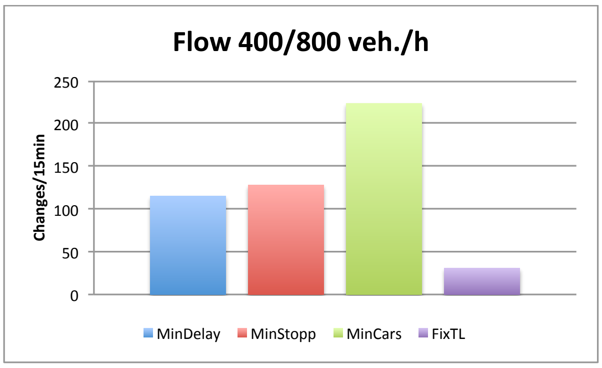

Fig. 39

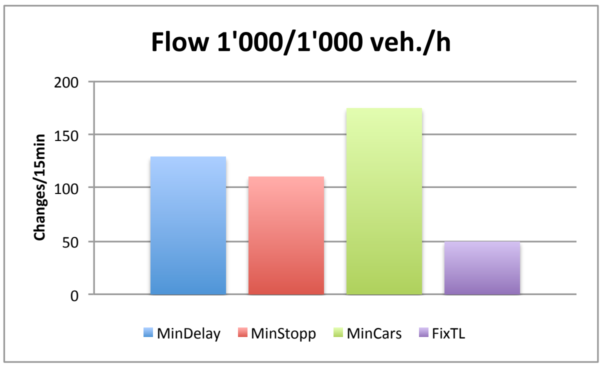

Fig. 40

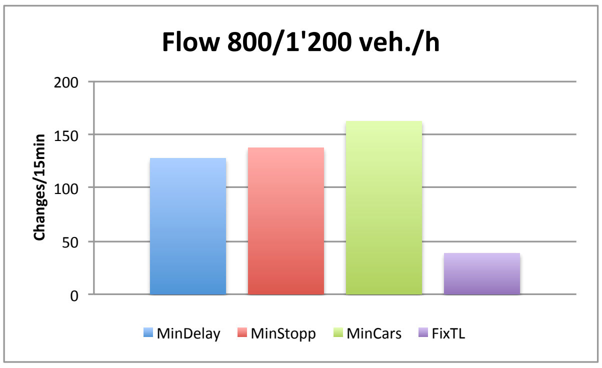

Fig. 41

All three optimization algorithms change the priority quite fast compared to the fix-timed traffic light. Except for the 1’000/1’000 veh./h demand, the minimizing delay algorithm changes the least. This is not surprising because this algorithm has the additional delay after leaving the intersection implemented. Frequent changing of priority is punished with that additional delay. The average green time for this algorithm is about 7.5 sec. meaning that in average a little less than four cars are discharges in one platoon. The additional delay for discharging cars changes only few from 4th to 5th car and remains constant after 5th car (see Fig. 2).

The average duration between the changes of priority are as follows:

Flow	600/600	400/800	1’000/1’000	800/1’200
MinDelay	8.2 sec.	7.8 sec.	7.0 sec.	7.1 sec.
MinStop	6.7 sec.	7.0 sec.	8.2 sec.	6.6 sec.
MinCars	4.3 sec.	4.0 sec.	5.2 sec.	5.5 sec.
FixTL	20 sec.	30 sec.	20 sec.	25 sec

Very short durations between the changes of priority might lead to issues concerning the implementation in reality. If cars on different approaches cross the conflict zone very closely the requirements to the accuracy of the automated cruise control are high. Especially for long and slow vehicles this might lead to problems.

Without knowing the possibilities of the technology of automated cruise control at all, it could very well be that an average duration of 4 sec. (as in the minimizing braking cars algorithm) between the changes of priority is not feasible. Taking into account that these 4 sec. are the mean value, there is a lot of tight crossing. The algorithms minimizing number of stops and total delay show an average duration between changing priority of 6.6 – 8.2 sec. This seems to be a realistic value to implement in reality.

5.4.2 Total delay

Fig. 42 - Fig. 45 show the total delay for all three optimization algorithms and for the fix-timed traffic light.

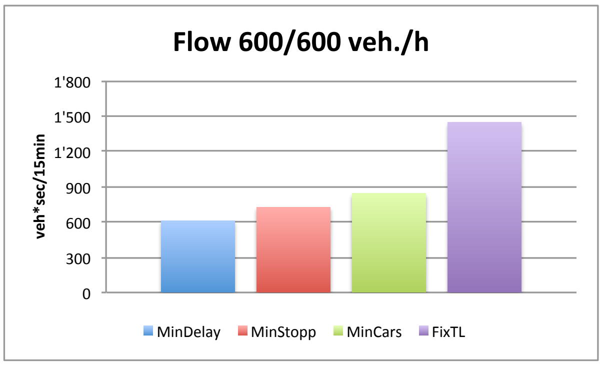

Fig. 42

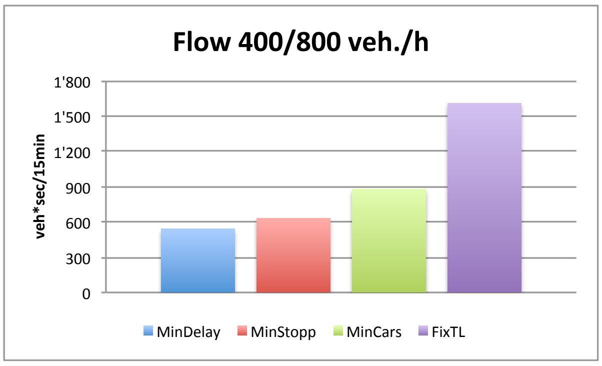

Fig. 43

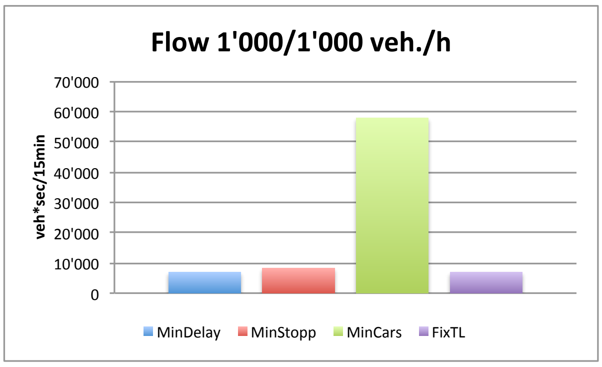

Fig. 44

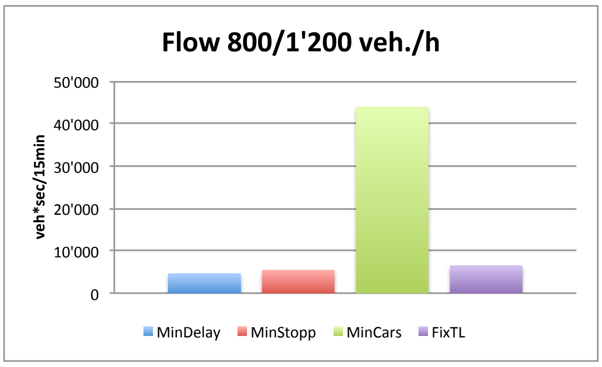

Fig. 45

For a total flow of 1’200 veh./h all three optimization algorithms deliver a course of traffic, which is clearly better then a fix-timed traffic light in terms of total delay. For a total flow of 2’000 veh./h the algorithm minimizing number of stops delivers worse results than the fix-timed traffic light for a balanced demand and slightly better results for unbalanced demand. The minimizing number of braking cars algorithm leads to huge delay. This bases on the fact that for high demand it is the best solution in terms of number of braking cars to slow down every single car on the approach with less demand and to let pass unrestrained all cars on the approach with higher demand. An example of a course of traffic like this is pictured in Fig. 46.

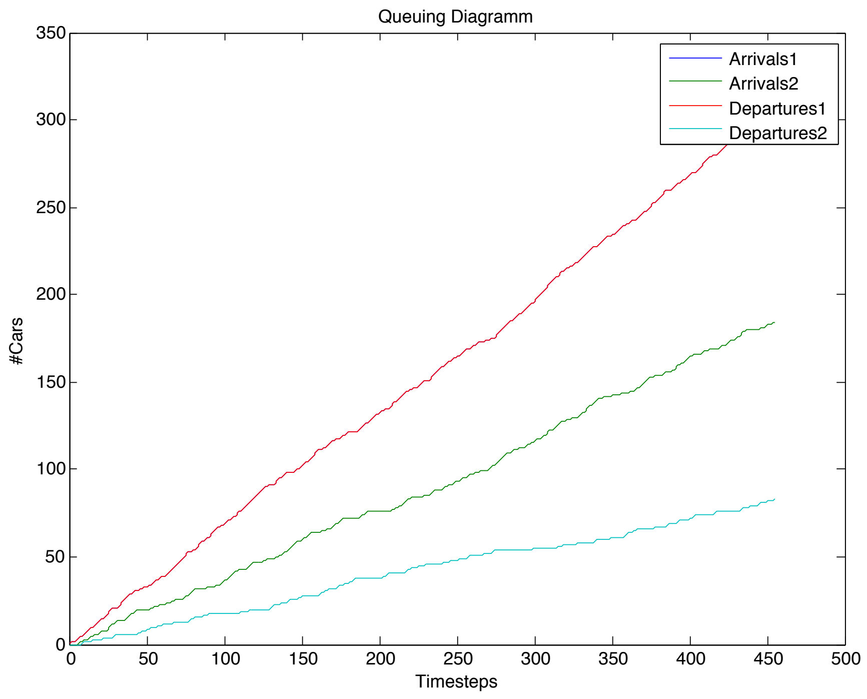

Fig. 46

The average savings per car compared to the fix-timed traffic light are as follows:

Flow	600/600	400/800	1’000/1’000	800/1’200
MinDelay	2.8 sec.	3.6 sec.	0.03 sec.	3.9 sec.
MinStop	2.4 sec.	3.3 sec.	-16.6 sec.	2.2 sec.
MinCars	2.0 sec.	2.4 sec.	-115.9 sec.	-74.9 sec.

5.4.3 Number of stops

Fig. 47 - Fig. 50 show the numbers of stops for all three optimization algorithms and for the fix-timed traffic light.

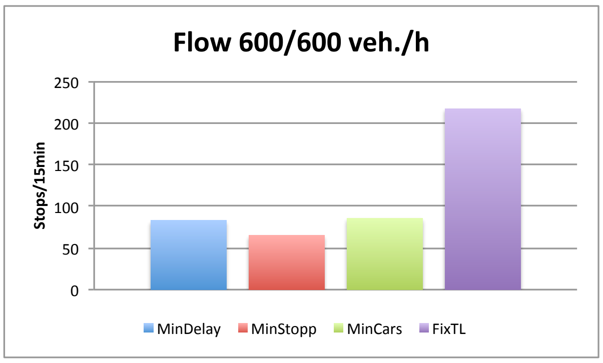

Fig. 47

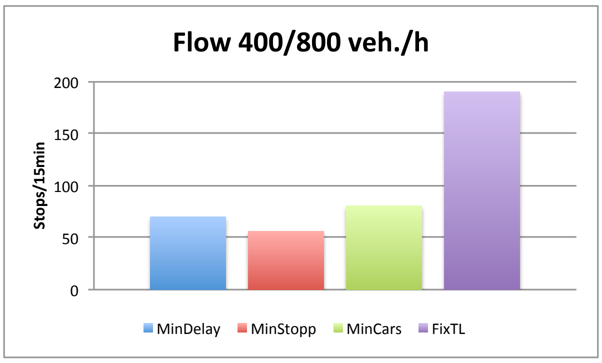

Fig. 48

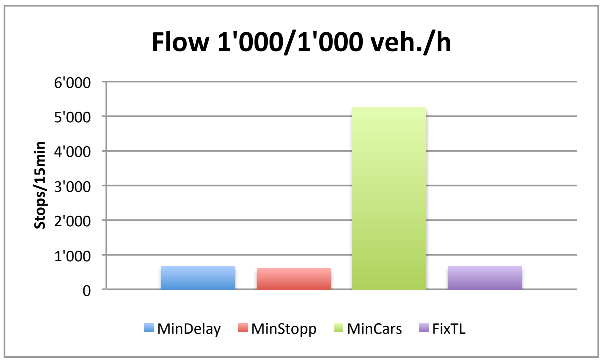

Fig. 49

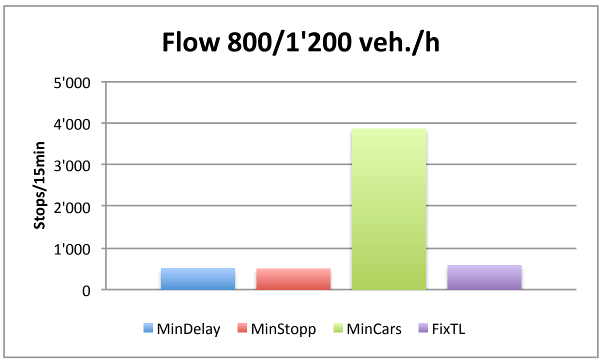

Fig. 50

The graphs for the number of stops show a pretty similar picture as the ones for the total delay. For the undersaturated intersection all optimizing algorithms deliver better results than the fix-timed traffic light. For an oversaturated intersection it’s again the minimizing number of braking cars algorithm which delivers very bad results. Having again a look in Fig. 46 one sees clearly that all cars on the approach with less demand have to stop very frequently (every step in the departure 2 curve means an additional stop for every car in the queue at this very moment).

The average savings per car compared to the fix-timed traffic light are as follows:

Flow	600/600	400/800	1’000/1’000	800/1’200
MinDelay	0.44	0.40	0.00	0.14
MinStop	0.51	0.45	0.12	0.16
MinCars	0.42	0.37	-9.2	-6.6

5.4.4 Number of braking cars

Fig. 51 - Fig. 54 show the numbers of braking cars for all three optimization algorithms and for the fix-timed traffic light.

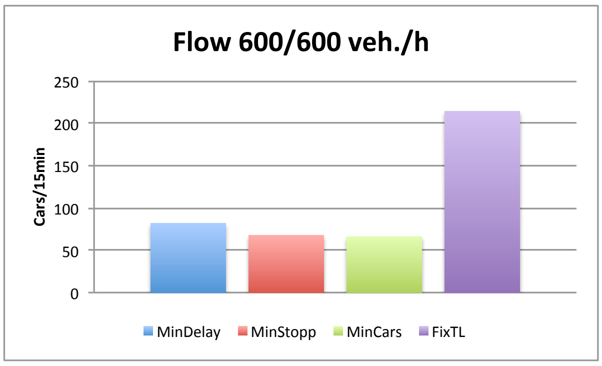

Fig. 51

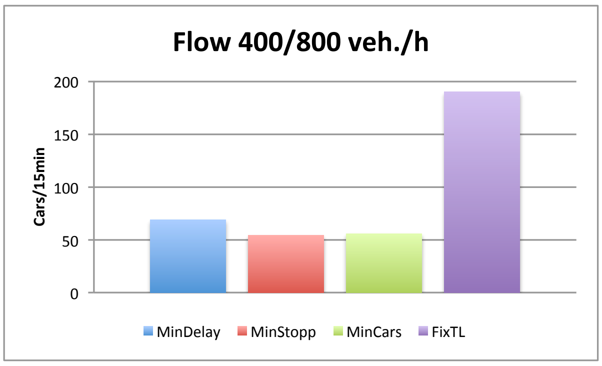

Fig. 52

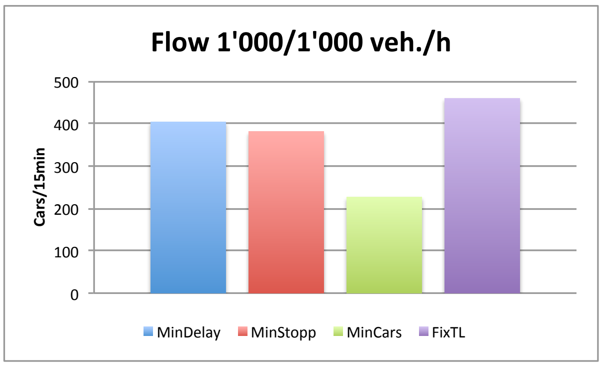

Fig. 53

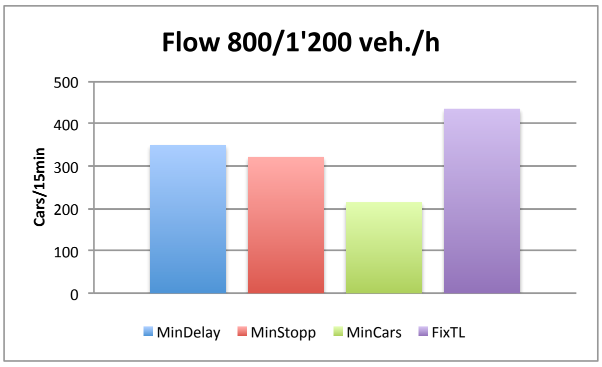

Fig. 54

In terms of number of braking cars all algorithms deliver better results than the fix-timed traffic light for all considered demands.

The average savings per car compared to the fix-timed traffic light are as follows:

Flow	600/600	400/800	1’000/1’000	800/1’200
MinDelay	0.44	0.40	0.11	0.17
MinStop	0.49	0.45	0.16	0.23
MinCars	0.49	0.45	0.47	0.44

6 Conclusions and future work

The comparison of the proposed algorithms with the fix-times traffic light always showed an improvement in the specific property of traffic regardless the traffic demand. For low and unbalanced traffic demand this comparison might be a bit unfair: The fix-timed traffic light might give green to one approach nobody is arriving at this time. For more meaningful statements it would be nice to compare the optimization algorithms with a traffic-actuated traffic light. Unfortunately, in the context of the MATLAB-codes developed for this thesis that’s not possible. Creating the same intersection with identical traffic demand in a existing simulation program such as VISSIM and implementing a traffic-actuated traffic light there, wouldn’t help, because comparing results from different simulation programs is very risky.

The algorithm to minimize the number of braking cars works well in its specific purpose. But safety issues based on very frequent changing of priority and the fact that it generates a huge total delay and many stops for an oversaturated intersection deny implementing this algorithm in reality. The algorithms to minimize the number of stops and the total delay both improve the course of traffic in terms of total delay, number of stops and number of braking cars compared to a fix-timed traffic light (Except for the minimizing number of stops algorithm with a 1’000/1’000 veh./h demand). The results of both algorithms are quite similar. But the minimizing total delay algorithm has some advantages:

- It is the only algorithm, which delivers better result for every property of traffic for every investigated traffic demand.
- It optimizes the course of traffic with the least changes of priority. This leads to safer traffic.

- The ratio between number of stops and number of braking cars is for under-saturated intersections very close to one, meaning that rarely a car has to stop twice.
- It is multiple times faster in terms of computational speed (e.g. for a simulation duration of 15 min the computational time to minimize the total delay is less than 1 sec. and to minimize number of stops it's about 115 sec.).
- It doesn't require a constraint for the length of the queue and a remaining queue penalty. These two properties of the code influence strongly the result and on top of that this influence is difficult to understand.

Concerning these facts it is suggested to concentrate on minimizing the total delay when making further research on this topic. As soon as it comes to minimize the emissions of the car, nevertheless it might be useful to return to minimizing the number of stops. Stopping and emissions sure are related. But as seen in the comparison of the algorithms the minimizing total delay algorithm works almost as well in terms of number of stops as the minimizing number of stops algorithm itself.

The modeled intersection is just the most elementary intersection possible. For future work it would be possible to extend this intersection to a full four-approach intersections with turning allowed, see Fig. 55.

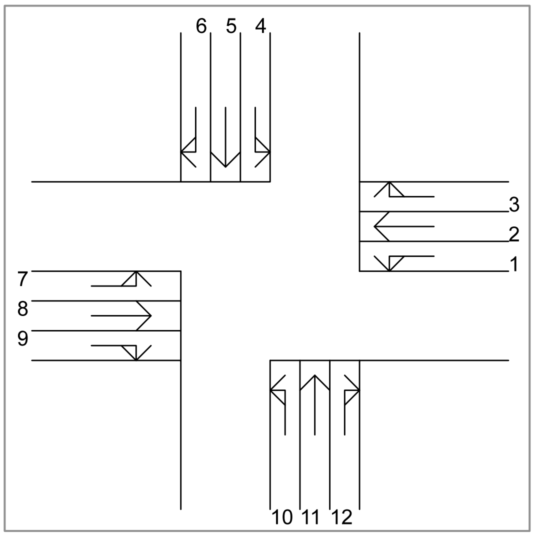

Fig. 55

Instead of just two conditions (discharging either approach 1 or approach 2) there will be quite a lot of conditions:

- Condition 1: Approaches 2, 3, 8, 9 discharging
- Condition 2: Approaches 5, 6, 11, 12 discharging
- Condition 3: Approaches 3, 4, 9, 10 discharging
- Condition 4: Approaches 1, 6, 7, 12 discharging

These are just the basic conditions. There are several more, e.g. 2 and 12, 3 and 5, 6 and 8, 9 and 11, 3 and 6 and 9 and 12, etc. The principle would stay the same: Calculate all possible departure curves and pick the best one. Of course the demands made to the computer will increase strongly. With eight considered future time steps, instead of 256 possible departure curves resulting of two conditions, there would be more than 43 Mio. possible departure curves with the nine conditions mentioned above (and there are more conditions).

To implement optimizing algorithms in real traffic the computational speed is a big issue. Of course the code has to run faster than one time step. Even if running these codes with a big computer this might be a real problem. Without knowing enough of computers and just assuming that the computational time is about proportional to the number of possible departure curves, which have to be calculated, the time to calculate a simulation of the full four-legged intersection with the conditions for every time step mentioned above would be almost 170’000 times longer than for the most elementary intersection discussed in this paper. To calculate every decision of priority within a time step duration of 2 sec. the computer would have to be more than 100 times faster than the laptop used for this thesis (For the minimizing delay algorithm, which calculates 450 time steps in 0.6 sec.). It might as well be that the computational time isn’t increased proportionally as assumed but exponentially, then optimizing in a way proposed in this paper might not be feasible at all for complete intersections in reasonable time. On top of that, for the complete intersection there are more possible conditions of a time step as these nine conditions mentioned above and the number of conditions is crucial for the number of possible departure curves and for the computational time, too.

7 References

[1] Zohdy, Ismail H. and Hesham Rakha, *Game Theory Algorithm for Intersection-based Cooperative Adaptive Cruise Control (CACC) Systems*, 15th International IEEE Conference on Intelligent Transportation Systems, 2012

[2] Lee, Joyoung and Byungkyu Park, *Development and Evaluation of a Cooperative Vehicle Intersection Control Algorithm Under the Connected Vehicle Environment,* IEEE Transaction on Intelligent Transportation Systems Vol. 13, No. 1, 2012

[3] Spacek, P., *Verkehrstechnik – Grundzüge,* Institut für Verkehrsplanung und Transportsysteme IVT, ETHZ, Zürich, 2006