Simple algorithms for preemptive traffic control, and an appraisal of their quality

Denis Roegel
LORIA, Nancy[*]

November 10, 2005

Abstract

It is a fact that traffic lights are seldom managed optimally when preemptive vehicles are allowed. More and more attention has been given to these problems, often with a practical perspective. It is the examination of several real cases that led us to take a closer look at some of the problems that occur, which are often a prejudice to the driver. Here, we show that given certain assumptions, simple solutions do exist, and that they can significatively improve the quality of traffic. We use simple criteria to measure the quality of traffic and support our claim.

Contents

[*]LORIA, BP 239, 54506 Vandœuvre-lès-Nancy cedex, `roegel@loria.fr`

1 Introduction

Every driver knows that there are two kinds of intersections. On the first hand, there are intersections with no interference. They are made of a number of intersecting lanes, each being completely separate from the others in time, and each being "open" only when the others are "closed." For instance, in a standard two-lane crossing, one lane has green lights when the other has red lights. There is no interference between the two lanes. However, this kind of intersection is rather rare, and the more common variety, on the other hand, involves lanes which do interfere. For reasons of efficiency, most crossings are designed with priorities and probabilities in mind. An example of a common interference is a crossing where cars from one lane want to turn left, but have to yield to the cars coming from the opposite direction and which are going straight. The assumption here is that once all cars that go straight have gone through, there will be enough time for the cars that want to go left to go through.

Although many crossings have their traffic lights modulated during the day in order to adjust to changing traffic, the previous case has an inherent flaw in that the left gate is usually open only as long as the straight gate is open. And the more the straight gate is opened, the more the cars going left have to wait. We might call this problem a "self-interference" problem.

Other common problems involve the need to yield to pedestrians who are usually only incompletely protected by priority and not by a full closing of the gates. Actually, managing a crossing is a compromise between the need to ensure safety, and the need to disrupt traffic as little as possible. In spite of the examples given above, such a compromise can be attained to a reasonable extent in regular traffic, even if this regular traffic changes during the day.

But crossings evolve. New lanes may be added or flows from parts of the city may be diverted and create endemic problems at certain intersections. Our focus here is on such a change, namely the insertions of certain kinds of preemptive traffic within an existing intersection. A typical example which led to this study is the case of streetcars having a higher priority than cars and disrupting normal traffic.

2 Two real examples

Two such examples are given in figures 1 and 2. In the first, which is a real crossing near the Nancy (France) railway station, there are three car lanes (1, 2, 3) which are (normally [1]) non interfering. A few years ago, a streetcar was built and this gave two lanes 4 intersecting the car lane 1. Since then, going through this intersection has often been a lot slower than before, especially when a streetcar happens to pass by.

In the second example, near Strasbourg (France), there are four separate car lanes (1, 2, 3, 4) and a central streetcar (5) interfering with three of the four car lanes. Here too, traffic can be significantly affected when a streetcar happens to go through the intersection.

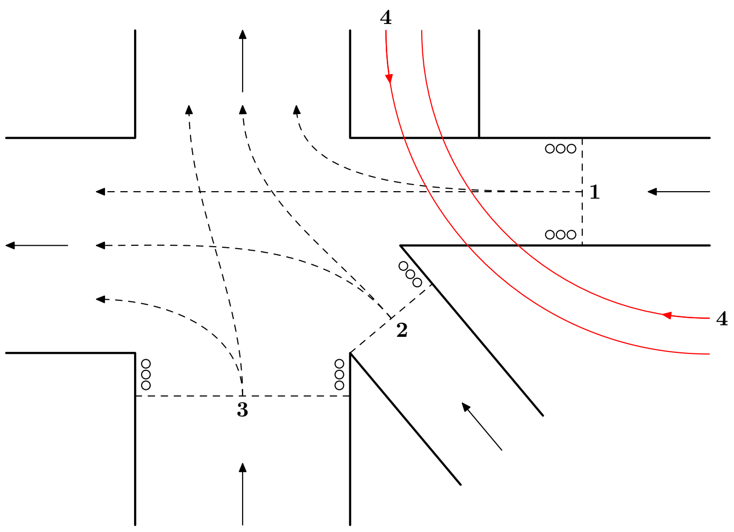

Figure 1: Crossing 'N'. The streetcar lanes (4) interfere with one other lane (1).

In the years since these crossings have been introduced, we have observed two curious phenomena. First, the traffic problems may have become differ-

[1]It should be remember that one tendency of drivers is to go through the crossing, even when the lights are about to become red, and even when the traffic is very heavy and cars move slowly. This then typically leads to traffic jams. In our study, we do assume that this is not the case and that traffic is light before and beyond the intersection.

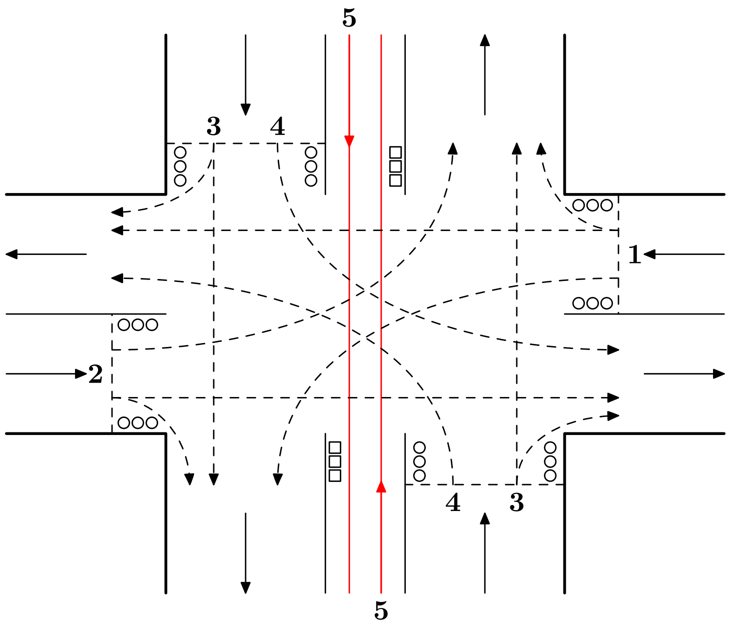

Figure 2: Crossing 'S'. The streetcar lanes (5) interfere with three other lanes (1, 2 and 4).

ent, because of slight alterations of the traffic light scheduling, or because streetcars sometimes stop (in case of severe jam, one assumes), but they haven't significantly been reduced. In both examples, it is not uncommon to wait several rounds of traffic lights, and this becomes worse when several streetcars do come in a row, producing some sort of resonance between car lanes and streetcar lanes.

A second phenomena that was observed is the resignation of the users: they may complain, and they know that traffic is difficult, but they think it can't be any different if streetcars are to have the highest priority.

In this study, we will show that this assumption is wrong, and that traffic can be significantly improved in both examples, without altering the priority of the streetcars.

3 A simulation model

A first model of the traffic lights is by the infinite sequence (F_1, t_1), (F_2, t_2), (F_3, t_3), ..., (F_n, t_n), (F_1, t_{n+1}),... where there are n different streams F_1, F_2, ..., F_n.

So, stream 1 starts at t_1 and lasts until stream 2 starting at t_2, and so on. At t_{n+1}, we return to stream 1. An example with four lanes is given in figure 3. We assume for simplification that each lane is open the same amount of time, and has similar traffic.

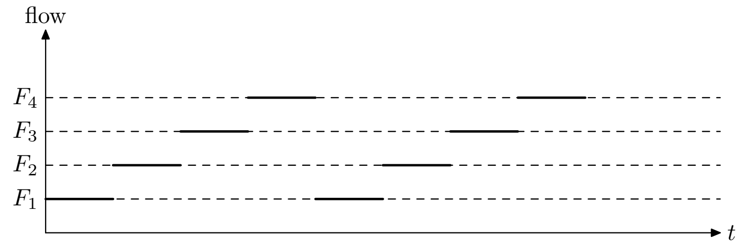

Figure 3: The standard flow of cars on four streams. The streams are F_1, F_2, F_3 and F_4 and at a given time only one stream is open. We go through the streams in a cyclic way.

Instead of this simple cycle, the traffic lights could be made to work in a more random way, but it would complicate the matter, and make driving more dangerous, as many drivers tend to anticipate how the lights will behave.

We will also assume that a stream is one-way and that it is seen as a queue. The 'N' type crossing is clearly of such a kind, but the 'S' type can also be considered such, as the concurrent lanes (such as the two lanes 4, for example) do not interfere. We assume there is no case of self-interference (as defined above) or interactions with pedestrians.

In order to model the traffic faithfully, we have to model the cars. When a vehicle v_i arrives at t on stream s, it will be added at the end of queue i. When a car goes through the intersection, it is removed from the queue. We assume that a car that goes through an intersection does no longer need to be considered for future traffic. We assume that c_i is the capacity that stream i can handle. This means that if there are less than c_i cars waiting in stream i, then the green light on stream i will allow all cars to go through.

Cars are regularly added, at random, but monotonous, times on all the lanes.

Finally, everytime a car goes through, we keep track of the *badness* of the waiting. We define this badness (on some time interval) as being the longest time a car waits, between its arrival and clearance, on that interval. This will be the *worst badness*. We will also keep track of the *average badness*. Both informations will provide a good measure of the quality of preemptive traffic control.

4 The standard, or free, algorithm

Figure 4 shows the first algorithm of our simulation in a C style. At any given time, only one lane is open, and it is called the "active lane." The time is divided in slices of ΔT and in every such period, we both remove cars from the active lane, and we add a random number of cars to all the lanes (including the active one). Adding a car is deciding when the car arrives in a lane. Only when cars have been added do we let cars pass the green light.

```
t=0; // time
active_lane=0;
lane_open=20; // Delta T
while (t<100000) { // max time
  next_change=t+lane_open;
  add_cars_to_all_lanes(t,next_change);
  // have cars pass in active lane until the next lane change:
  let_cars_pass(active_lane,t,next_change);
  active_lane=(active_lane+1) % nlanes; // change lanes
  t=next_change;
}
```

Figure 4: Standard, or free, algorithm.

The algorithm can be parameterized in a number of ways, by changing ΔT globally, by varying it during traffic, by changing the arrival or departure density. ΔT could also be made dependent on the lane.

This algorithm deals with normal intersections and no preemptive traffic. A typical simulation until time 100000 (seconds) and three lanes gives:

lane	longest wait	average wait	number of cars
0	223.510	198.391	23089
1	222.993	198.335	23094
2	223.613	198.391	23083

That is, in the worst case, a car had to wait 224 seconds, and this was on about 70000 cars. The worst average wait was 198 seconds. Runs vary and other ones are:

lane	longest wait	average wait	number of cars
0	222.992	198.290	23100
1	224.445	198.156	23117
2	223.703	198.164	23110

lane	longest wait	average wait	number of cars
0	223.148	198.561	23069
1	223.827	198.314	23102
2	224.354	198.587	23063

No lane is singled out by the algorithm, and the scheduling is *fair*.

5 Preemptive algorithms

We now assume that there is a streetcar lane and that streetcars come at more or less regular intervals and interfere with one of the streams. The regularity of streetcars is an essential assumption for our fairness conclusions. We assume that there is only an interference with one stream (such as the crossing in figure 1), although it would be easy to generalize the algorithm to multiple interferences.

We consider several algorithms, from the naïve to the complex ones.

5.1 The naïve algorithm

In this algorithm, the streetcar traffic isn't changed, but traffic on a lane is inhibited whenever there is an intersection with a streetcar. Figure 5 shows an example where lane F_2 is inhibited during part of its normal operation. As a consequence, lane F_2 will be open a shorter time than the other lanes.

The corresponding algorithm is given in figure 6. It introduces new features. First, an interference criteria is needed. This in defined in the macro "interference." There is an interference when the active lane is the one that can be inhibited and when either the next streetcar would arrive before the next scheduled lane change, or if a streetcar is not yet through when the active lane becomes active. A special variable decides when the inhibition of the lane starts.

A typical simulation until time 100000, with three lanes and an interference on lane 0, gives:

lane	longest wait	average wait	number of cars
0	687.051	304.124	15491
1	223.051	198.128	23122
2	223.742	197.987	23126

The obvious consequence of this algorithm is that cars in lane 0 have to wait significantly longer, not only in the worst case, but even more in the average case. The average wait is of course shorter for the two other lanes.

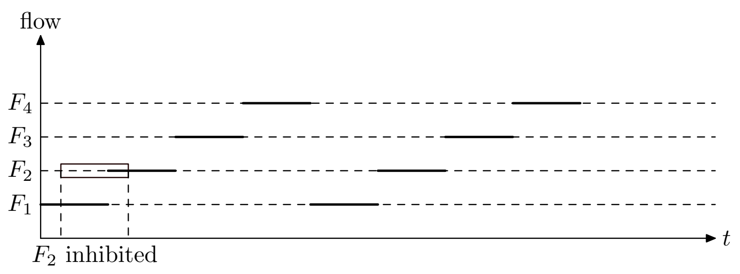

Figure 5: The naïve algorithm introduces simple inhibition and prevents the opening of a stream during the streetcar transit. The rectangular box represents a streetcar.

```
#define interference= \
  ((active_lane==inhibited_lane) && \
  (next_streetcar<next_change) && (next_streetcar+streetcar_duration>t))

t=0; // time
active_lane=0;
lane_open=20; // Delta T
next_streetcar=0;
streetcar_duration=20;
inhibited_lane=0; // lane that might be inhibited

while (t<100000) { //  max time
  next_change=t+lane_open; // schedule next (normal) lane change
  if interference
    inhibit_start=next_streetcar;
  else
    inhibit_start=-1; // no inhibition
  add_cars_to_all_lanes(t,next_change);
  // have cars pass in active lane until the next lane change
  //    or until the arrival of the streetcar:
  let_cars_pass(active_lane,t,next_change,inhibit_start);
  <schedule next streetcar, if necessary>
  active_lane=(active_lane+1) % nlanes; //change lanes
  t=next_change;
}
```

Figure 6: The naïve algorithm.

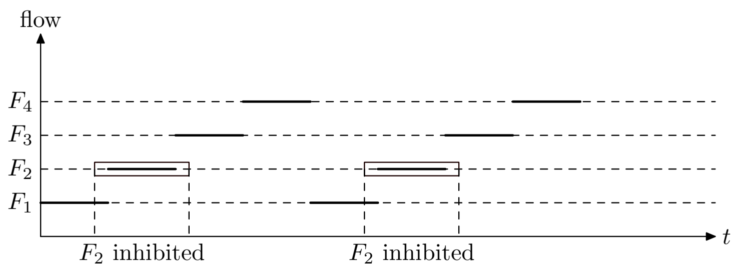

Figure 7: A case where the naïve algorithm can lead to a long wait for stream F_2.

It is the same average wait as in the free algorithm, because lanes 1 and 2 are not influenced by lane 0 or the streetcar.

Other runs show the same behavior:

lane	longest wait	average wait	number of cars
0	640.398	312.928	15088
1	223.148	198.499	23082
2	222.798	198.418	23081

lane	longest wait	average wait	number of cars
0	750.984	302.199	15593
1	223.008	198.740	23056
2	223.598	198.321	23089

This algorithm is obviously *unfair* and has a considerable impact on the streetcar-interfering lane. Not enough time is alloted to lane 0 for all cars to pass in the time it takes for cars in other lanes to pass through the crossing. This results in an accumulation of delays. Sometimes, lanes are inhibited for several rounds, which is especially irritating (figure 7).

5.2 The naïve algorithm improved

A first improvement over the previous algorithm is to try to correct the time allocation problem. This is known as “compensation.” We can try to avoid to break a lane slot, except when the streetcar-interfering lane can still be open a certain amount of time before the arrival of the streetcar. This doesn’t ensure that every lane is open the same amount of time, but it is a compromise between alloting each lane the same amount of time, and delaying the lanes too much.

There are therefore two cases, the one exhibited in figure 8 (case 1), and the first one exhibited in figure 9 (case 2).

Naturally, we would expect all waits to become longer, except for the streetcar-interfering lane which should have a less congested traffic. Paradoxically, the average and longest waits may all become better. In this algorithm, there is both a positive factor for non-interfering lanes, namely the second case where we switch lanes immediately, and a negative factor, namely the first case, where lanes are delayed. Depending on the choice of parameters, the combination of these two factors can lead to improvements such as those observed here.

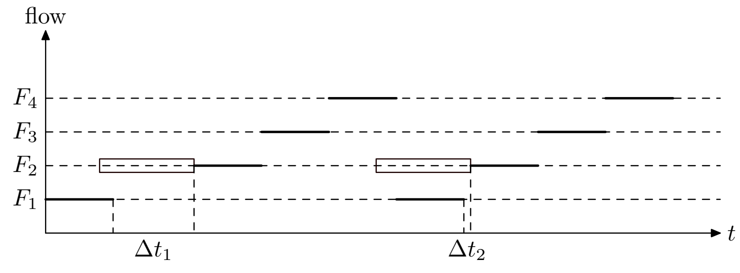

Figure 8: The naïve algorithm improved. Stream F_2 (and the others) are delayed until the streetcar has passed.

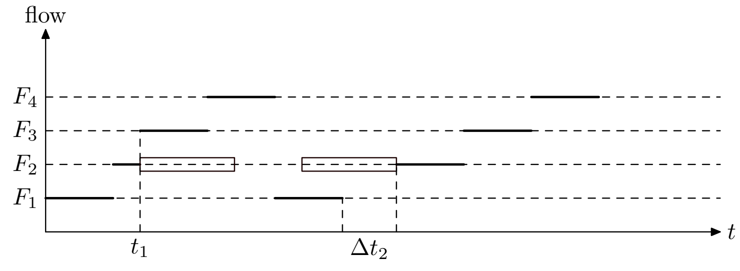

Figure 9: Naïve algorithm improved. Here, when the first streetcar arrives at t_1, traffic switches from lane 2 to lane 3 (a so-called “green recall” is performed), but when the second streetcar arrives, traffic on lane 2 is first delayed. These are the two interference cases of the algorithm.

Typical runs are:

lane	longest wait	average wait	number of cars
0	419.020	259.589	18057
1	221.703	177.156	25603
2	221.242	177.372	25615

```
t=0; // time
active_lane=0;
lane_open=20; // Delta T
next_streetcar=0;
streetcar_duration=20;
inhibited_lane=0; // lane that may be inhibited

while (t<100000) { //  max time
  // naïve improved (changing lanes as soon as a streetcar arrives)
  next_change=t+lane_open; // schedule next (normal) lane change
  if interference {
     if (next_streetcar>=t)
       next_change=next_streetcar; // (case 1)
     else
       next_change=t; // i.e., now (case 2)
  }
  add_cars_to_all_lanes(t,next_change);
  // have cars pass in active lane until the next lane change:
  let_cars_pass(active_lane,t,next_change);
  <schedule next streetcar, if necessary>
  active_lane=(active_lane+1) % nlanes; //change lanes
  t=next_change;
}
```

Figure 10: The naïve algorithm improved.

lane	longest wait	average wait	number of cars
0	387.398	256.748	18226
1	220.836	177.800	25523
2	224.211	177.602	25588

lane	longest wait	average wait	number of cars
0	416.180	258.197	18134
1	221.387	177.003	25626
2	222.100	177.788	25564

This algorithm is naturally *unfair*, but less than the previous one, as we try not to waste time with no traffic.

5.3 Algorithm 4

This algorithm is actually only a partial improvement over the previous one. Here we try to stick to an homogeneous time allotment by ignoring the second case in the previous algorithm. An example is given in figure 11.

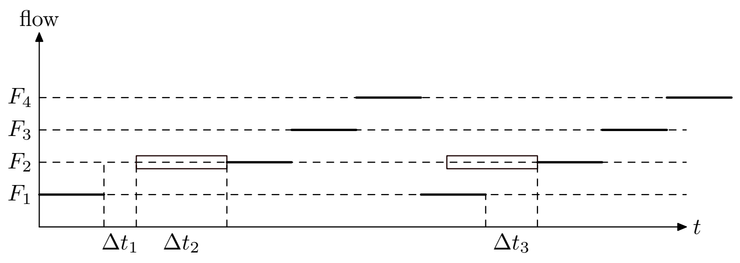

Figure 11: Algorithm 4.

```
t=0; // time
active_lane=0;
lane_open=20; // Delta T
next_streetcar=0;
streetcar_duration=20;
inhibited_lane=0; // lane that may be inhibited

while (t<100000) { //  max time
  // schedule the next (normal) lane change:
  next_change=t+lane_open;
  if interference {
    next_change=next_streetcar+streetcar_duration+lane_open;
    add_cars_to_all_lanes(t,next_change);
    t=next_streetcar+streetcar_duration;
    <schedule next streetcar>
  } else
    add_cars_to_all_lanes(t,next_change);
  // have cars pass in active lane until the next lane change:
  let_cars_pass(active_lane,t,next_change);
  <schedule next streetcar, if necessary>
  active_lane=(active_lane+1) % c.nl; // change lanes
  t=next_change;
}
```

Figure 12: Algorithm 4.

The algorithm is given in figure 12.

Typical runs become:

lane	longest wait	average wait	number of cars
0	353.155	235.692	19687
1	355.388	234.877	19757
2	356.890	235.545	19710

lane	longest wait	average wait	number of cars
0	343.375	233.698	19846
1	343.586	232.821	19918
2	344.211	231.640	20001

lane	longest wait	average wait	number of cars
0	361.828	234.968	19745
1	360.973	235.777	19690
2	363.004	235.107	19746

Each lane has now as much time as the others.

This algorithm is *fair*, but time is wasted everytime a streetcar comes, as everything is freezed. We can do better!

5.4 Algorithm 5

This is the first algorithm where we resort to changes in the scheduling sequence. Up to now, lane 0 was followed by lane 1, which was followed by lane 2, etc., except that there could be delays between the end of a lane and the opening of the next one.

The new algorithm is based on the previous one, but we try to take advantage of the long "frozen time" to schedule a non-interfering lane for exactly that time, before reverting to the interfering lane. We assume it is always the same non-interfering lane which is scheduled. An example is given in figure 13 where lane 3 is scheduled during the time Δt_1 where lane 2 is either waiting, or inhibited. Lane 3 gets a so-called "green extension."

The algorithm is given in figure 14. An obvious consequence of this algorithm is that another lane gets a great extra share of time, and its performance will be greatly enhanced, although it is never inhibited by the streetcar. The running figures are therefore not surprising. The interfering lane and the other lanes, except the lane which gets extra time, all have similar longest and average waits.

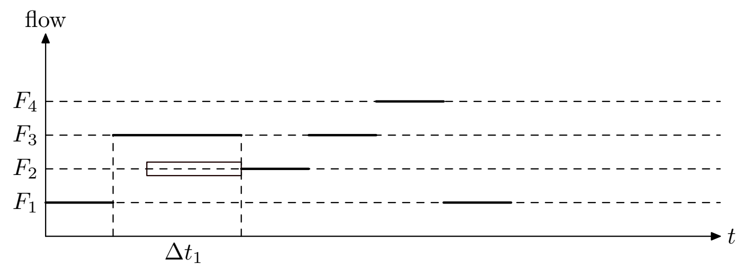

Figure 13: Algorithm 5.

```
t=0; // time
active_lane=0;
lane_open=20; // Delta T
next_streetcar=0;
streetcar_duration=20;
inhibited_lane=0; // lane that may be inhibited

while (t<100000) { //  max time
  next_change=t+lane_open; // schedule next (normal) lane change
  if interference {
    next_change=next_streetcar+streetcar_duration;
    add_cars_to_all_lanes(t,next_change);
    active_lane=(active_lane+1) % c.nl; // change the active lane
    lane_exception=1; // we set a lane exception
    let_cars_pass(active_lane,
                  t,next_streetcar+streetcar_duration);
    <schedule next streetcar>
  } else { // no interference
    add_cars_to_all_lanes(t,next_change);
    // have cars pass in active lane until the next lane change:
    let_cars_pass(active_lane,t,next_change);
  }
  <schedule next streetcar, if necessary>
  // change lanes:
  if (lane_exception) { // return to previous lane
    active_lane=(active_lane-1+c.nl) % c.nl;
    lane_exception=0;
  } else
  active_lane=(active_lane+1) % c.nl;
  t=next_change;
}
```

Figure 14: Algorithm 5.

lane	longest wait	average wait	number of cars
0	362.602	236.733	19609
1	222.534	146.655	30008
2	362.344	236.580	19635

lane	longest wait	average wait	number of cars
0	353.722	237.322	19564
1	221.578	145.124	30220
2	360.167	238.312	19489

lane	longest wait	average wait	number of cars
0	352.453	238.627	19469
1	221.113	144.888	30303
2	349.562	238.189	19514

This algorithm is again *unfair*, but we managed to greatly improve the average and longest waiting time of lane 1, without altering significantly the two other lanes.

Lane 1 gets significantly better times because it is the lane to which traffic switches when a streetcar arrives on the interfering lane, when it is its turn.

5.5 Algorithm 6

This algorithm is based on the previous one, and a different lane is also scheduled when the interfering lane has to wait. However, we have seen that it led to an extra allocation of time to the lane following the one interfering. The previous algorithm reaches one of the goals, namely to avoid wasted or "frozen" time, but time is not allotted fairly.

This last algorithm meets both the first goal (no wasted time), and the second goal (each lane gets the same share). This is merely done by having lanes keeping track of their allotted time and returning it to other lanes when necessary.

More precisely, if a lane is allowed to be open, say 20 seconds, more than it was supposed to, it means that this lane is ahead by 20 seconds on *all* other lanes. Returning these 20 seconds amounts to allot 20 extra seconds to all other lanes.

This, however, is not always possible, or at least it isn't so simple. In the case of the interfering lane, traffic may be bounded by the occurrence of a streetcar, and if we assume that streetcars have the highest priority, the time that the interfering lane receives from another lane may need to be broken in two or more pieces. Figure 15 shows how this is done. When lane 2 was supposed to be scheduled, it couldn't, because of the impending streetcar, and lane 3 was scheduled instead, until the streetcar was gone through. The extra time received by lane 3 is α, and it must be returned to lanes 1, 2 and 4. The next time lanes 1 and 4 are scheduled, their time slot is indeed increased by α. But lane 2 can't run its traffic for more than α_1. The remaining part, $\alpha_2 = \alpha - \alpha_1$, is added the next time lane 2 is scheduled.

Formally, each lane has a "credit" which represents an amount of time it can use to lengthen its traffic. Every time a lane uses its credit time, it decreases it. A special treatment is given to the lane receiving the initial extra credit, because it has to return it after it used it. The corresponding algorithm is given in figure 16.

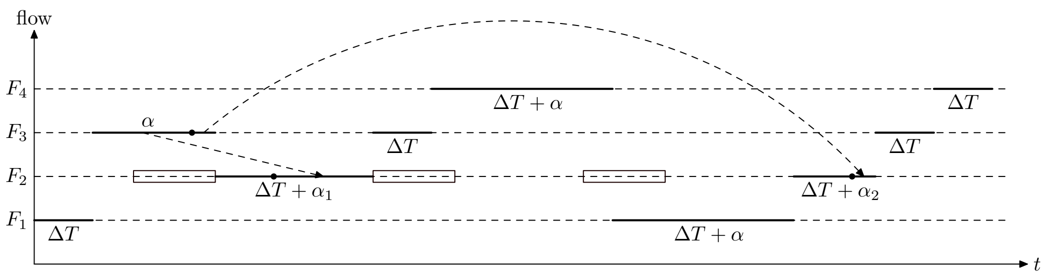

Figure 15: Algorithm 6. In this example, although lane 2 was inhibited, and although lane 3 got extra time (green extension), in the end each lane received $2\Delta T + \alpha$ time (green compensations), and traffic was never frozen.

Typical runs are now:

lane	longest wait	average wait	number of cars
0	289.508	188.748	22779
1	288.299	196.383	23134
2	315.475	188.207	22839

lane	longest wait	average wait	number of cars
0	287.869	187.663	22855
1	290.949	196.723	23096
2	335.988	188.271	22797

lane	longest wait	average wait	number of cars
0	303.766	187.317	22878
1	287.789	195.989	23159
2	313.148	187.843	22806

```
t=0; // time
active_lane=0;     inhibited_lane=0; // lane that may be inhibited
next_streetcar=0;  lane_open=20; streetcar_duration=20;

while (t<100000) { //  max time
  next_change=t+lane_open; // schedule next (normal) lane change
  if interference {
     next_change=next_streetcar+streetcar_duration;
     add_cars_to_all_lanes(t,next_change);
     active_lane=(active_lane+1) % nlanes; // change the active lane
     // we add to the new lane's credit the time it has to give back
     time_credit[active_lane]
          =time_credit[active_lane]+(next_streetcar+streetcar_duration-t);
     lane_exception=1; // we set a lane exception
     let_cars_pass(active_lane,t,next_streetcar+streetcar_duration);
     <schedule next streetcar>
     for (j=1;j<c.nl;j++) {  // we add credits to the other lanes
       l=(active_lane+j) % c.nl;
       time_credit[l]=time_credit[l]+time_credit[active_lane];
     }
     time_credit[active_lane]=0; // all the credit was used
  } else {
     // no streetcar interferes, we must use our credits now!
     if ((active_lane==inhibited_lane) &&
       (next_change+time_credit[active_lane]>next_streetcar)) {
      // avoid a new interference
      dt=next_streetcar-next_change;
      next_change=next_change+dt;
      time_credit[active_lane]=time_credit[active_lane]-dt;
     } else { // no new interference
      next_change=next_change+time_credit[active_lane];
      time_credit[active_lane]=0; // reset
     }
     add_cars_to_all_lanes(t,next_change);
     // have all cars pass in active lane until the next lane change:
     let_cars_pass(active_lane,t,next_change);
  }
  <schedule next streetcar, if necessary>
  // change lanes:
  if (lane_exception) { // return to previous lane
    active_lane=(active_lane-1+c.nl) % c.nl;
    lane_exception=0;
  } else active_lane=(active_lane+1) % c.nl;
  t=next_change;
}
```

Figure 16: Algorithm 6.

This algorithm comes very close to fairness, although there seems to be a bias towards lane 1 in average, and towards the last lane in the worst case. We haven't fully investigated these anomalies, which could be artefacts of the finiteness of our implementation.

It should be noticed that compared to the free algorithm, the longest waits are all worse, but the average waits are better! By avoiding the streetcars, we don't make things necessarily worse, since we are merely reshuffling the schedules. In spite of a lane interacting with a streetcar, this algorithm produces average results which level the influence of the streetcar.

6 Conclusion

Our study has taken a few very simple assumptions and has investigated the practical influence of several choices. It turned out that streetcars can be integrated to traffic without being too much of a noisance. Of course, real cases are more complex, and we can have a variety of special circumstances, such as traffic jams, interactions with pedestrians, self interference, failures, etc. which are outside the scope of this study. But for regular, smooth traffic, we believe that our approach shows, by comparison with real observed cases, that there is still a lot of room for improvement in the quality of a driver's journey through a city.

7 Related work

Traffic control, and in particular traffic-light control, is a vast domain for which groundbreaking work was done in the 1960s, but which is now expanding more and more, especially in the realm of intelligent vehicles.

Non preemptive traffic has received the most attention and traffic-light control has been examined and solved in various cases, given certain reasonable assumptions [17].

In particular, a whole branch is devoted to simulation and simulation tools, using various models. For examples of intersections modelling and their simulation, see Daum, Klee and others [6, 13]. However, the simulation tools ease the test of algorithms, but algorithms are seldom given or studied. Among the different models which have been used, we mention (timed) Petri nets [2].

Optimal signal-control strategies have only received scarce attention, but this is quickly changing. For a general background on traffic-flow theory, see Lieu [21]. Traffic scheduling algorithms are a vital part of congestion control schemes. It would seem at first that traffic lights are not needed, but this would naturally enhance accidents, as the drivers would never be able to apply scheduling algorithms without errors. A scheduler is needed, and it is important that the scheduler isn't too conspicuous. If traffic lights were switching too often, the consequence would again be chaos.

Conflicts are receiving more and more attention. In recent work, different alleys have been explored. An early approach made use of graph theory in order to cover general scheduling with conflicts [9].

Conflicts are usually resolved by priority, but numerous strategies can be applied, and it doesn't seem that enough experience has yet been gathered to decide which strategy is best, as there are many factors, including human factors, involved. For instance, Jones, Hallworth and Fox examine a variety of strategies that were deployed, but without simulating them or measuring their effect [12]. For a real example, see for instance [7]. Some work has also been done in order to analyze the effect of different types of vehicles in an intersection [14].

Most transit priority projects have only been deployed in the US within the past few years, and so far, many transportation authorities have only issued guidelines. For instance, in August 2003, the Virginia Tech Transportation Institute issued guidelines for the planning and deployment of emergency vehicle preemption and transit priority strategies. ITS America also issued guidelines in 2003 [20] that show in particular the strategies deployed in various cities (Portland, Seattle, Chicago, etc.) and the measured impact on traffic. But all these guidelines let appear that there is yet no clear homogeneous measure which could decide of the best strategy.

Interesting work has recently been done on the simulation of different classes of vehicles [11], and scheduling has even been investigated for airline traffic. At landing, a plane from airline A could for instance have a greater priority than another plane from airline B, although the second plane arrived first [3]. However, these strategies are applied on an individual basis, and do not seem easily applicable to urban traffic. In particular, it isn't clear if there is an airline equivalent of traffic lights which would apply to several planes. It seems that such strategies also entail a risk.

Then, there is of course also the problem of modelling and simulating. In many cases, modelling is done in order to evaluate real-time traffic control schemes. See for instance Palm [19]. Models can also be fine or coarse-grained. Our model, for instance, is a microscopic one and we keep track of every vehicle (see [15]). Other models are macroscopic, for instance when based on the principles of hydrodynamics [4]. Finally, recent work made use of fuzzy logic [5].

Traffic scheduling is also related to scheduling in a network router, although there usually isn't an equivalent to traffic lights. Paquets may have some priority, but as soon as a packet arrives, it is usually eligible for transmission. Such a transmission is therefore more related to traffic scheduling with stop signs, than with traffic lights.

Compared to all these studies, our work has only considered that streetcars have the highest priority. In our model, they are never interrupted by traffic. But in reality, there are cases where it makes sense to stop streetcars, for instance when they are ahead of time. Many cities actually consider deploying flexible preemption, where the priority varies and where it can be on request by the streetcar driver or the emergency vehicle driver.

References

[1] Jeffrey Archer and Iisakki Kosonen. The potential of micro-simulation modelling in relation to traffic safety assessment. In *ESS Conference Proceedings, Hamburg*, 2000.

[2] Francesco Basile, Ciro Carbone, Pasquale Chiacchio, René K. Boel, and Camelia C. Avram. A Hybrid Model for Urban Traffic Control. In *Proceedings of the International Conference on Systems, Man and Cybernetics, October 10–13 2004 The Hague, The Netherlands*, 2004.

[3] Gregory C. Carr, Heinz Erzberger, and Frank Neuman. Airline arrival prioritization in sequencing and scheduling. In *2nd USA/Europe Air Traffic Management R&D Seminar, Orlando*, 1998.

[4] Ramakrishna Casturi. *A macroscopic model for evaluating the impact of emergency vehicle signal preemption on traffic.* Master of science, Faculty of Virginia Polytechnic Institute and State University, 2000.

[5] Yu-Chiun Chiou, Ming-Te Wang, and Lawrence W. Lan. Coordinated transit-preemption signal along an arterial: iterative genetic fuzzy-logic controller (GFLC) method. *Journal of the Eastern Asia Society for Transportation Studies*, 6:2321–2336, 2005.

[6] Thorsten Daum. An HCFG Model of a traffic intersection specified using Himass-j. In S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 158–165.

[7] Kevin Fehon, Jim Jarzab, Casey Emoto, and Deborah Dagang. Transit Signal Priority for Silicon Valley Bus Rapid Transit. Technical report, 2003.

[8] B. Han and S. Yagar. Traffic with Bus and Streetcar Interactions. *Road Traffic Monitoring*, pages 108–, 1992.

[9] Sandy Irani and Vitus Leung. Scheduling with conflicts, and applications to traffic signal control. In *Proceedings of the seventh annual ACM-SIAM symposium on discrete algorithms*, pages 85–94, Atlanta, Georgia, US, 1996.

[10] J. Jacobson and Y. Sheffi. Analytical Model of Traffic Delays under Bus Signal Preemption: Theory and Application. In *Transportation Research B, vol 15B*. 1981.

[11] R. Jayakrishnan, Cristián E. Cortés, Riju Lavanya, and Laia Pagès. Simulation of Urban Transportation Networks with Multiple Vehicle Classes and Services: Classifications, Functional Requirements and General-Purpose Modeling Schemes. Technical Report UCI-ITS-WP-02-16, Institute of Transportation Studies, University of California, Irvine, 2002.

[12] S. Jones, M. Hallworth, and K. Fox. State of the Art and User Needs for Selected Vehicle Priority. Technical Report UTMC 01 Deliverable 1, 1998.

[13] Harold Klee. Microscopic Car Modeling for Intelligent Traffic and Scenario Generation in the UCF Driving Simulator. Technical report, University of Central Florida, School of Electrical Engineering and Computing Science, (no date).

[14] Kara M. Kockelman and Raheel A. Shabih. Effect of vehicle type on the capacity of signalized intersections: The Case of Light-Duty Trucks. *Journal of Transportation Engineering*, 126(6):506–512, 2000.

[15] Roger V. Lindgren and Sutti Tantiyanugulchai. Microscopic Simulation of Traffic at a Suburban Interchange. In *Proceedings of the 2003 ITE Annual Meeting, Seattle, USA*, 2003.

[16] L. Y. Ma. A mathematical model for optimisation of traffic operations performance. In L. J. Sucharov, editor, *Urban Transport V*. 1999.

[17] Elina Mancinelli, Guy Cohen, Stéphane Gaubert, Jean-Pierre Quadrat, and Edmundo Rofman. On Traffic Light Control of Regular Towns. Technical Report 4276, INRIA, 2001.

[18] Nan Ni and Laxmi Narayan Bhuyan. Fair Scheduling in Internet Routers. *IEEE Transactions on Computers*, 51(6):686–701, June 2002.

[19] William J. Palm. Modeling for real-time traffic control in the Rhode Island Intelligent Road. Technical report, University of Rhode Island Transportation Center, 2002.

[20] R. J. Baker and J. J. Dale and others. An Overview of Transit Signal Priority. Technical report, ITS America, Washington, D.C., 2003.

[21] Nagui Rouphail, Andrzej Tarko, and Jing Li. Traffic flow at signalized intersections. In Henry Lieu, editor, *Traffic-flow theory*, chapter 9. 1999.