Generalized Problem

Jeroen van Riel

December 2023

1 Platoon Forming Algorithms

Let us start by briefly recalling the work by Marko and Rik on the analysis of platoon forming algorithms [1]. They consider a single intersection vehicle control problem in which arriving vehicles are approaching the intersection from different lanes and need to be guided safely accross the intersection in some efficient manner (minimizing some global measure of delay and acceleration). They propose to solve this problem in two stages. In the first stage (*platoon forming algorithm*), a *polling policy* determines the order in which vehicles cross the intersection. Given this decision, precise vehicle trajectories can be computed efficiently as the second stage (*speed control algorithm*).

The motivation for studying our single intersection scheduling problem has been based on the above decomposition. Given a feasible schedule, we assume that trajectories can be efficiently computed. Up till now, we have assumed that all vehicle arrivals are known upfront, in which case the problem can be essentially reduced to an *offline* scheduling problem. When vehicle arrivals are revealed to the traffic controller over time, we are dealing with some kind of *online* problem. In this case, note that it might be beneficial to recompute (parts of) the trajectories upon new vehicle arrivals. An interesting question is whether methods based on the two-stage decomposition are still possible/sensible in the online case.

Definition 1.1 (Definition 3.1 from [1])**.** A polling policy is *regular* if an arrival in a queue does not change the *order* of service of all currently present vehicles, i.e. the new arrival is inserted somewhere in the order of service of all waiting vehicles.

The work of Marko and Rik provides a positive answer to this question under the assumption that the polling policy is *regular* (in terms of the disjunctive graph, we could say that the currently fixed disjunctive arcs are not changed upon arrival of new vehicles). However, in general, it is not clear to me whether optimal policies can always be decomposed in similar ways. To study this question further, we would first need a clearer understanding of the general problem.

2 General Traffic Control Problem

Let us provide a rough sketch of a more general vehicle control problem on a network of intersections, without assuming any structure on the solution procedure like above. Consider a very abstract problem in which vehicles represented as little dots are moving over a graph in a continuous fashion. Vehicles arrive to the graph over time and need to follow a fixed route towards a final node, where the vehicle leaves the system. The goal of the controller is to determine how vehicles move exactly along their route, while satisfying some constraints on the interaction between moving vehicles that stem from safety considerations. In some sense, this is very reminiscent of models common in train scheduling literature.

Recall our definition of the traffic network graph G with internal and external nodes. Again, let

$$R_j(0), R_j(1), \dots, R_j(n_j), R_j(n_j + 1)$$

denote the route of vehicle j and assume that vehicle j arrives at external node $R_j(0)$ at time t_j^0, assumed to be random. Let $x_j(t)$ denote the position of vehicle j on its route R_j at time $t \geq t_j^0$. We also use the notation $x_{ij}(t)$ to denote the position on the lane on the route of j leading to node i, so we have

$$x_{ij}(t) = x_j(t) - \sum_{k=0}^{i-2} d(R_j(k), R_j(k+1)). \tag{1}$$

Furthermore, let $v_j(t)$ and $a_j(t)$ denote the speed and acceleration, respectively, of vehicle j at time t.

Example 2.1. Consider the network of intersections in Figure 1. Suppose vehicle j is already in the network and assume t is such that

$$d(R_j(0), R_j(1)) < x_j(t) < d(R_j(0), R_j(1)) + d(R_j(1), R_j(2)), \tag{2}$$

then we say that vehicle j is driving on the lane between the first and second intersection on its route (recall that only the $n_j = 3$ nodes in the middle are

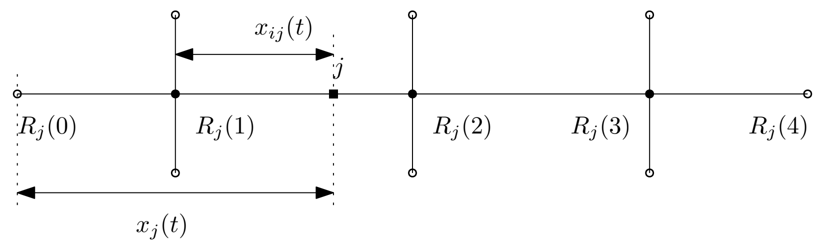

Figure 1: Illustration of a network of intersections with the position of some vehicle j given in two ways.

considered to be intersections). Furthermore, the exact position on this lane is given by

$$x_{2j}(t) = x_j(t) - d(R_j(0), R_j(1)). \tag{3}$$

□

The task of the traffic controller is to determine trajectories x_j (or equivalently either v_j or a_j) satisfying some kind of smoothness and safety constraints. For example, we may impose maximum/minimum speed constraints

$$v_{\min} \leq v_j(t) \leq v_{\max} \tag{4}$$

and acceleration constraints

$$a_{\min} \leq a_j(t) \leq a_{\max}. \tag{5}$$

Given a node i, we will use $t_{ij} = t_{ij}(x_j)$ to denote the time when vehicle j crosses i, which of course depends on the chosen trajectory x_j. Between crossing of vehicles from different lanes, we enforce a *safe lane-switching* time s_{lane}. Recall the definition of a *merge point.* At each merge point i of each path $P \in P_{jl}$ for all pairs of distinct vehicles $\{j, l\}$, we require that either

$$t_{ij} + s_{\text{lane}} \leq t_{il} \text{ or } t_{il} + s_{\text{lane}} \leq t_{ij}. \tag{6}$$

When two vehicles j and l are driving on the same lane we require some minimum *safe following distance* d_{follow}. Recall the definition of *common paths* P_{jl}. For all pairs of distinct vehicles $\{j, l\}$, for each common path $P = (i_1, \dots, i_L) \in P_{jl}$, we require for each intersection $i \in P$ that

$$x_{ij}(t) + d_{\text{follow}} \leq x_{il}(t) \text{ or } x_{il}(t) + d_{\text{follow}} \leq x_{ij}(t) \tag{7}$$

for all t in the time interval when both vehicles are driving on the common path. When we do not allow vehicles to overtake each other, exactly one of the inequalities must hold for the entire common path.

For simplicity, the two types of constraints were stated in terms of time and distance, respectively, which is not the most general approach. To illustrate, we transform constraints (7) to an equivalent version in the time-domain. Defining the inverse of x_j as

$$x_j^{-1}(x) = \inf\{t : x_j(t) = x\}, \tag{8}$$

we could instead have required

$$x_j^{-1}(x) + s_{\text{follow}} \leq x_l^{-1}(x) \text{ or } x_l^{-1}(x) + s_{\text{follow}} \leq x_j^{-1}(x), \tag{9}$$

for all positions x on the common path. Furthermore, we might want to have safe head-times that depend on the speeds of the involved vehicles.

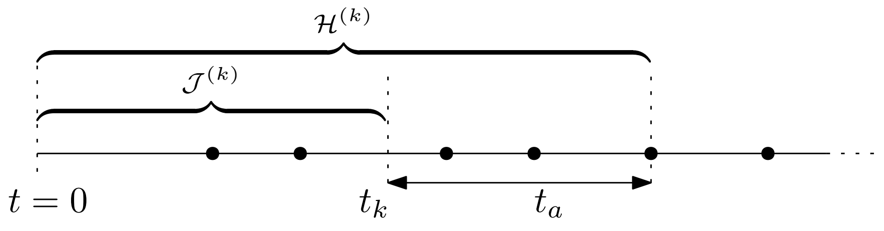

Figure 2: Illustration of some decision epoch t_k and the corresponding set of vehicles under control and the visible horizon. The dots represent vehicle arrival times t_j^0.

3 Optimization setting

Let us now consider more precisely how the controller interacts with the system defined above. First of all, we need to make some additional assumptions on the random vehicle arrivals at external nodes. Like [2] (Section 3), we assume vehicle arrivals are *safe*, which roughly means that safe following times need to be respected between consecutive arrivals. More precisely, we require that

$$|t_j^0 - t_l^0| \geq s_{\text{follow}}, \tag{10}$$

for all pairs of vehicles $\{j, l\}$ arriving to the same external node.

In general, at some time t, we assume that the controller knows the arrival times of vehicles

$$\mathcal{H}_t = \{j : t_j^0 \leq t + t_{\text{a}}\}, \tag{11}$$

for some non-negative look-ahead time t_{a}. We call this set of vehicles (with their corresponding arrival times) the *visible horizon* of the controller. Similarly, let the set of vehicles in the system at time t be denoted by

$$\mathcal{J}_t = \{j : t_j^0 \leq t\}. \tag{12}$$

Ignoring for now that vehicles eventually leave the system, we say that $\mathcal{J}_t$ are precisely the vehicles *under control* of the traffic controller.

Note that the only uncertainty in the system stems from the random arrival times of vehicles. Once trajectories are determined, the dynamics of the system are completely deterministic as long as $\mathcal{H}_t$ does not change. Therefore, the only sensible decision epochs to consider are the points in time t_k whenever $\mathcal{H}_t$ changes, i.e., when new information becomes available, see Figure 2 for an illustration. Whenever this happens, the controller needs to compute a trajectory for the newly arrived vehicles, and is free to revise the trajectories of the vehicles that are already in the system. However, these updated trajectories need to respect the trajectories that have been followed up till that point.

Let us make the sequential decision process sketched above a little bit more precise. To simplify notation, we define $\mathcal{J}^{(k)} = \mathcal{J}_{t_k}$ and $\mathcal{H}^{(k)} = \mathcal{H}_{t_k}$. Let the *state* of the system at time t_k be represented by the sets

$$s_k = (\{(t_j^0, R_j)\}_{j\in\mathcal{H}^{(k)}},\ \{(x_j^{(k-1)}(t_k), v_j^{(k-1)}(t_k), a_j^{(k-1)}(t_k))\}_{j\in\mathcal{J}^{(k)}}). \tag{13}$$

The controller needs to compute a set of trajectories

$$a_t = \{x_j^{(k)}\}_{j\in\mathcal{H}^{(k)}}, \tag{14}$$

which are functions of $t \geq t_j^0$, satisfying the constraints from the previous section and the additional constraints

$$x_j^{(k)}(t_k) = x_j^{(k-1)}(t_k), \tag{15a}$$
$$v_j^{(k)}(t_k) = v_j^{(k-1)}(t_k), \tag{15b}$$
$$a_j^{(k)}(t_k) = a_j^{(k-1)}(t_k), \tag{15c}$$

for the vehicles $j \in \mathcal{J}^{(k)}$ that were already under control, guaranteeing in some sense the *continuation* of the updated trajectories.

From the above discussion, it should be clear that the transition to the next state s_{k+1} is particularly simple. The arrivals that happen at $t_{k+1} + t_a$ are simply added to the visible horizon and the positions, speeds and accelerations follow directly from the determined trajectories a_t.

We still have to define an objective to compare the performance of different controllers. Measures of how well the controller performs on some particular sequence of arriving vehicles may be based on properties of the computed trajectories. For example, we could relate some measure of energy consumption to acceleration, or we can measure delay based on the crossing times t_{ij}. In general, the optimization objective is the expected value of this measure, where the expectation is taken over the distribution of the vehicle arrival process.

Finally, we may distinguish between a finite-horizon problem with a finite number of vehicles, or an infinite-horizon case, in which particular attention needs to be paid to the definition of the optimizaton objective, e.g., we might need to use discounting to avoid infinite sums.

4 Scheduling approach

Given the above problem formulation, we are now in a better position to explain the class of approaches that are based on explicit scheduling.

At some decision epoch t_k, the controller generates the set of trajectories as follows. First of all, one of the main assumptions is that vehicles cross intersections at maximum speed. Assuming that acceleration can change instantly, it takes

$$t_{\text{acc}} = \frac{v_{\max} - v_j(t_j^0)}{a_{\max}} \tag{16}$$

time to reach maximum speed. In this time period, simple calculation shows that the vehicle has traveled

$$x_j(t_{\text{acc}}) = t_{\text{acc}} v_j(t_j^0) + t_{\text{acc}}^2 a_{\max}/2. \tag{17}$$

To simplify notation, let

$$d_{kj} = \sum_{i=0}^{k-1} d(R_j(i), R_j(i+1)) \tag{18}$$

denote the distance of the kth intersection on the path of vehicle j. Assume that initial speeds are such that $x_j(t_{\text{acc}}) < d(R_j(0), R_j(1))$, the *earliest possible crossing time* of vehicle j for its kth intersection is simply

$$r_{kj} = t_{\text{acc}} + \frac{d_{kj} - x_j(t_{\text{acc}})}{v_{\max}}. \tag{19}$$

Assuming that vehicles arrive at maximum speed allows us to simplify this to

$$r_{kj} = \frac{d_{kj}}{v_{\max}}. \tag{20}$$

The basis of the scheduling approach is to first determine the crossing times $y_{kj} \geq r_{kj}$ for $j \in \mathcal{H}^{(k)}$ and then use this as a starting point to compute feasible trajectories that satisfy $x_j(y_{kj}) = d_{kj}$. In general, it is not clear that feasible trajectories always exist. The paper of Marko and Rik shows that it is possible in case of one intersection.

References

[1] R. W. Timmerman and M. A. A. Boon, "Platoon forming algorithms for intelligent street intersections," *Transportmetrica A: Transport Science*, vol. 17, pp. 278–307, Feb. 2021.

[2] M. Limpens, "Online Platoon Forming Algorithms for automated vehicles: A more efficient approach," Master's thesis, Eindhoven University of Technology, Sept. 2023.