# Real-Time Scheduling Autonomous Vehicles at Intersections*

Shlomi Dolev
Dept. Computer Science
Ben-Gurion University of the Negev
Be'er Sheva, Israel
dolev@cs.bgu.ac.il

Ehud Gudes
Dept. Computer Science
Ben-Gurion University of the Negev
Be'er Sheva, Israel
ehud@cs.bgu.ac.il

Hannah Yair†
Dept. Computer Science
Ben-Gurion University of the Negev
Be'er Sheva, Israel
hannaya@post.bgu.ac.il

## ABSTRACT

Emergency situations involve massive movements of (both logistically and units of movements) platoons to and from focal locations. Platoons may move in different directions and block each other in junctions, causing even deadlocks. The possibility to minimize the delay in junctions, particularly non-stopping, and waiting for a (virtual) green light, may avoid the chain phenomena of cascade stopping and cascade starting to move again, when all cars wait for the car in front of them to gain enough velocity. The remote driving system is an opportunity to stream all platoons driving in different directions without stopping, by spacing vehicles to allow conflicting traffic to move in the space between vehicles.

In this work, we present an algorithm for real-time junction scheduling towards the non-stopping junction. We demonstrate the results that imply road safety as actions are remotely controlled, by using the SUMO simulator [1].

## KEYWORDS

Scheduling, Platoon, Junction, Virtual Traffic Light.

## 1 Virtual Traffic Lights

The main idea of the Virtual Traffic Lights is to synchronize the vehicles coming to the junction in such a way that vehicles do not stop at all. They will not stop to let others cross the junction or wait for others to evacuate the junction.

Under the assumption that all vehicles on the road are autonomous and computer-controlled, the solution for Virtual Traffic Lights is early handling of junction management. That is, calculating the timing at a reasonable distance *D* before the junction and maintaining a sufficient distance between the vehicles before the junction, allowing flexibility of deceleration or acceleration of the vehicles in order for them to cross the junction as safely and quickly as possible.

The timing principle is "first come, first served". The vehicles that are closer to the junction by a given distance *D* are the ones that are timed first crossing the junction. Additionally, the vehicles arriving at the junction will cross it at high speed to evacuate the junction as quickly as possible.

In other words, the order of priority between the lanes entering the junction is equal. That is why there is no starvation; this principle produces a junction in which the vehicles cross in analog to a zipper.

In *Fig. 1*, a snapshot is shown from the algorithm for vehicle synchronization running at a non-stop junction slightly describes the zipper situation, and it is apparent that the lights in all directions at the junction are green. This figure depicts the equal order of priority between all lanes entering the junction. Note that in practice, there is no need for a physical traffic light. A demonstration video of the preliminary version [2] can be found in here.

**Figure 1: Simulator snapshot**

**Related work.** A detailed overview of the related work and comparison with our novel approach can be done easily with [2].

## 2 Virtual Traffic Lights Algorithm

The algorithm idea is to utilize the junction as much as possible. Most important, it does not cause vehicles to stop before or at a junction but only cause them to move slightly slower.

### 2.1 Definitions

1. *lanes* is defined as the set of all lanes entering the junction.
2. $c(l_i)$ is defined as the group of lanes that lane $l_i$ has conflict with.
3. *D* is defined as the distance threshold from the entries of the junction along the lanes.
4. $batch_t$ is the group of vehicles that are currently present exactly in a distance *D* (or smaller) in the time-unit *t*.
5. $layer_t$ is defined as a subset of vehicles belonging to $batch_t$ that can arrive at the junction at the same time and can cross it without collisions.

With the above terms, it can be said that the objective is to maximize the size of the *layers* and create a minimum number of *layers* from a *batch*.

In *Fig. 2* we present an example of a junction with marks of the definition $c(l_i)$. Where the lanes' length is *D* (and possibly they are parts of the whole lanes), note that *size of lanes=12.*

* This study extends the preliminary version [2].
†Corresponding author.

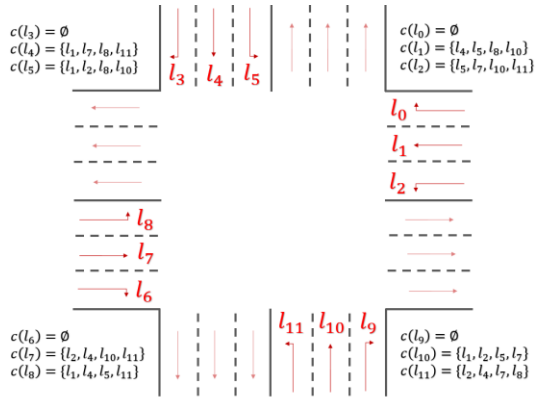


**Figure 2: Definitions example**

## 2.2 Outline of the Algorithm

The goal of the algorithm is to optimize the average waiting time of vehicles in a batch. Therefore, according to the function $c(l_i)$ that defines the group of lanes that lane $l_i$ has a conflict with.

The table $S$ is calculated, which contains the optimal separation into *layers* for every possible *batch*, where the optimal separation is the separation, which gives the minimum average waiting time. Note that the problem of finding the optimal separation into layers for a batch is NP-hard.

The calculation method of table $S$ is by dynamic programming and the way of calculating the minimum average waiting time could be by many different heuristic/approximation algorithms which give an estimated solution in a more efficient way, for example, [2-4]. However since *lanes* is small and constant, then it is better to get an inefficient but accurate solution than an estimated solution with an efficient calculation. Table $S$ is built once for each junction and then is used directly by the scheduling algorithm. The scheduling algorithm works as follows:

At each given time-unit $t$, all of the vehicles that are close to the junction at distance $D$ are collected ($batch_t$) and scheduled in the following way.

Check the possibility of joining for every vehicle in the $batch_t$ to an existing *layer* that was received by the previous *batches* and schedule their crossing time by the parameter $T$ of the $layer_T$ that they join to. After enabling vehicles from the current batch to join previous *layers*, the current batch is updated with a reduced set of vehicles. Then, for the remaining vehicles (the vehicles that cannot join to any previous *layers*) get the optimal separation from the table $S$ into new *layers*. A schedule for those *layers* is built thereafter with the necessary crossing time $T$ for each layer. For each layer in the current batch that has passed the junction, the parameter *clearIndicator* which indicates when the junction is empty, is set.

Finally, it is required to update the vehicle velocity by the distance and their calculated crossing time.

In *Table 1* we present a demonstration of three iterations of the algorithm, where the junction is according to the example presented in *Fig. 2*.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $batch_t$ | | x | | x | x | | x | x | | x | | x |
| $layer_{T1}$ | x | x | | x | | | x | x | | x | | |
| $layer_{T2}$ | | | | x | x | | x | | | x | x | |
| $layer_{T3}$ | | x | | x | | | x | | | x | | x |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $batch_{t+1}$ | x | x | | x | x | | x | x | | x | x | |
| $layer_{T4}$ | | | | | x | x | | | | | | |
| $layer_{T5}$ | | x | | | | | | x | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| $batch_{t+2}$ | | x | | x | | x | x | | x | x | x | x |
| $layer_{T6}$ | | | | | | | | | x | | x | |
| $layer_{T7}$ | | | | | | | | | | | | x |

**Table 1: Scheduling layers**

Every color in the table marks an iteration in the demonstration. The columns describe the lanes in the junction, and the rows describe batches or layers. In the $batch_{t+1}$ rows, the marks *x* describe in which lanes there are vehicles. In the $layer_{Ti}$ rows, the marks *x* describe from which lanes vehicles are joining that $layer_{Ti}$ and, the colored blank spaces describe the conflicts with the elements of the $layer_{Ti}$. The color of the marks *x* and the colored blank spaces are related to the iteration they joined and the conflicts of the elements with the same color. As seen in the table, for the first batch, layer 1 collects vehicles in lanes: 1,3,6,7,9, layer 2 collects vehicles from lane 4, and layer 3 collects vehicles from lane 11. The orange *"x"* indicates vehicles that joined from a later batch (e.g. vehicle in lane 0 in *layer* 1).

**In conclusion**, this work focuses on a real-time scheduling algorithm for autonomous vehicles in one junction, that is based on a conflicts function, separation table, and an indication parameter for emptying the junction. The algorithm works with different types of junctions and ensures smooth crossing of the junction with minimal waiting time (heuristics). This work can be extended to scheduling road traffic of platoons smoothly through one or many junctions of the road graph.

## ACKNOWLEDGMENTS

This work partially funded by the Andromeda MAGNET Consortium of the Israeli Innovation Authority, the Israeli Smart Transportation Research Center, the Lynne and William Frankel Center for Computer Science, and by Rita Altura chair in computer science.

## REFERENCES

[1] Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Fl¨otter¨od, Y.P., Hilbrich, R., L¨ucken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE (2018).

[2] S. Dolev, E. Gudes, H. Yair, Non-stopping junctions via traffic scheduling, in: Cyber Security, Cryptology, and Machine Learning: 6th International Symposium, CSCML 2022.

[3] F. Happach, L. Hellerstein, and T. Lidbetter, "A general framework for approximating min sum ordering problems," 2020.

[4] S. Dolev and A. Kesselman, "Bounded latency scheduling scheme for atm cells," Computer Networks, 2000.