# Analysis of Optimal Solutions to Robot Coordination Problems to Improve Autonomous Intersection Management Policies

Florent Altché, Arnaud de la Fortelle

# Analysis of Optimal Solutions to Robot Coordination Problems to Improve Autonomous Intersection Management Policies

Florent Altché[2,1] and Arnaud de La Fortelle[1]

*Abstract*— **The deployment of Cooperative Intelligent Transportation Systems (C-ITS) raises the question of future traffic management systems, which will be operating with an increasing amount of information and control over the infrastructure and the vehicles. This topic of research shares some similarities with robot coordination problems, inspiring our research on autonomous intersection management. In this article, we use a mixed-integer linear programming formulation for time-optimal robots coordination along specified paths and apply it to intersection management for autonomous vehicles. Our formulation allows to simultaneously solve a discrete optimal vehicle ordering problem, and a (discretized) continuous optimal velocity planning problem taking into account kinodynamics constraints. This allows faster pruning of the decision tree for the discrete problem, thus reducing computation time. A possible application for ITS is to evaluate the efficiency loss from a given vehicle ordering policy, or dynamically adapt policies to improve their efficiency. Moreover, any intermediary solution found by the solver can be used as a heuristically good policy, with proved bounds on sub-optimality.**

## I. INTRODUCTION

Cooperative Intelligent Transportation Systems (C-ITS) and particularly autonomous vehicles will likely have important implications for future traffic management systems. With advanced communication systems and increased control on both the vehicles and the infrastructure, it becomes imaginable to "perfectly" coordinate vehicular traffic flows. Although this is certainly overoptimistic, as there are many hypotheses to refine and a lot of engineering to do, this thought gives a direction for our research. Provided we have reasonably good information sharing and control, how can we design the best intersection management system for autonomous vehicles? Should we keep traffic lights? Should we only allow vehicles to enter the intersection at full speed to minimize occupation time? Several methods have been proposed [1]–[4], as well as several metrics to evaluate such systems [5]. However, finding the optimal solution is ultimately NP-hard so that it is difficult to evaluate methods.

In this paper, we focus on the analysis of such optimal solutions. We measure optimality in terms of traffic flow, by minimizing the average sojourn time of the vehicles in the intersection, which is equivalent to maximizing intersection throughput. We present a simplified model combining two important and difficult properties: the optimality of the discrete choice of a vehicle ordering (with relative *priorities* between conflicting vehicles, which encodes a homotopy class of trajectories), and the optimality of individual vehicle trajectories respecting this ordering, which is a continuous problem. Most previous work mostly considered the optimization of the continuous component, leaving the discrete optimization up to exhaustive enumeration. Our objective is to show that treating both problems simultaneously allows to find good heuristics for near-optimal priority-assignment policies.

This paper is articulated as follows. In Section II, we present a review of related work in the field of autonomous intersection management and multi-robot coordination. In Section III, we briefly describe the mixed-integer linear programming framework allowing to solve the time-optimal coordination problem for multiple moving robots with kinodynamic constraints. In Section IV, we illustrate on a simple example how this framework can help evaluate intersection management policies. In Section V, we show how near-optimal policies could be designed for real-time applications. Finally, Section VI gives some perspectives.

## II. RELATED WORK

Traffic lights are the most widespread method used to regulate heavy-traffic intersection; in this setting, right-of-way is alternated between lanes to avoid possible collisions between vehicles. A good timing of the phases is critical to the performance of the regulated intersection, and a lot of work has focused on finding good or optimal timings [6]–[9] off-line. However, such systems are suboptimal by nature, as they do not adapt to real-time changes in the volume of traffic. More recently, several authors have proposed using sensors [10], [11] or communication [12] to adapt traffic lights phases and timing in real-time. Other authors [13], [14] have proposed methods based on queuing theory to demonstrate asymptotic properties for a certain class of traffic control algorithms. Although all of these systems are shown to improve intersections throughput by allowing a finer control of traffic, the need of using phases during which vehicles in a lane are indiscriminately allowed to pass does not guarantee optimality, and it is difficult to measure this loss of efficiency.

Naumann et al. [15] are the first to propose using the foreseen abilities of autonomous vehicles to improve intersection management a step further. They introduce a distributed technique based on a reservation mechanism, where vehicles reserve an area of the intersection for a certain duration, and only the vehicles having a valid reservation can enter at a

[1] MINES ParisTech, PSL Research University, Centre for robotics, 60 Bd St Michel 75006 Paris, France [florent.altche, arnaud.de_la_fortelle] @mines-paristech.fr

[2] École des Ponts ParisTech, Cité Descartes, 6-8 Av Blaise Pascal, 77455 Champs-sur-Marne, France

This research was supported by the European FP7 project AutoNet2030 (Grant Agreement NO. 610542).

given time. This approach has been perfected by Dresner et al. [1] to accommodate a larger number of vehicles. In such systems, the order in which reservations are granted can obviously have a huge influence on the efficiency of the autonomous intersection. In [1], reservations are granted in a simple first-come, first-served (FCFS) policy, which is known [16] to be *less* efficient than pre-timed traffic lights for higher levels of traffic.

In [3], a reservation-based intersection management method is proposed, along with a heuristically efficient reservation-granting method. Reference [17] further studies improvements to this method. These techniques, however, are based on a set of general rules, such as having the fastest vehicle go first, and give no guarantee regarding optimality.

We argue that the problem of autonomous intersection management is closely related to that of (optimal) multi-robot coordination, where the robots are the autonomous vehicles, and that the study of the latter can help find improved methods for the former. Peng et al. [18] are among the first to apply optimization techniques to solve a multi-robot coordination problem, taking into account kinodynamic constraints. In their approach, robots evolve along fixed paths divided in segments of predetermined length, and collision avoidance is ensured by requiring that two robots cannot simultaneously be in conflicting segments. The coordination problem is formulated as a mixed-integer nonlinear programming problem, which is very difficult to solve. Using approximations of the vehicle dynamics, the exact problem is translated into two easier mixed-integer linear problems. Digani et al. [19] use a minimum-velocity hypothesis to formulate a (non-convex) quadratic programming problem for multi-robot coordination with kinematic constraints. A shared issue with those two approaches is the difficulty of handling the dynamic constraints, which need to be either approximated or ignored. The foundation of this paper is a new formulation for the multi-robot coordination problem along fixed paths using the concept of priorities [20], which has already been used for intersection management [21]. Using time discretization, our method allows a resolution-exact modeling of second-order dynamics, presented in [22].

## III. MILP-BASED OPTIMAL ROBOTS COORDINATION

In this section, we briefly present our mixed-integer linear programming (MILP) formulation for multi-robots coordination under kinodynamic constraints. A more detailed presentation can be found in [22].

We consider a set of $N$ robots evolving on predetermined paths inside a bounded *coordination region*. For automated driving where coordination is mostly needed in the middle of road intersections, the coordination region would be chosen as the center of the intersection including a portion of the roads leading to, and exiting from this center, as illustrated in Fig. 1.

We assume that each robot $i$ follows a predetermined path $\gamma_i$ inside the coordination region, and we only consider the longitudinal behavior of the robots. The dynamics of robot $i$ are described as a double integrator:

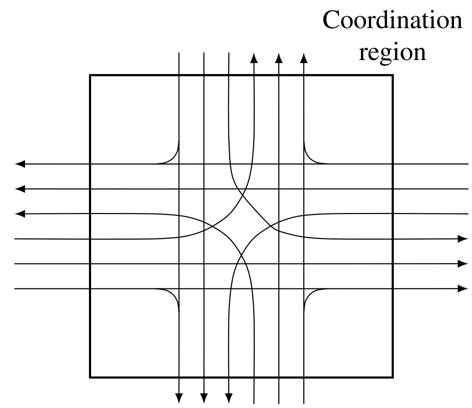



Fig. 1. Example of paths inside and outside the coordination region for a three-lanes road intersection.

$$\dot{s}_i = v_i, \tag{1}$$
$$\dot{v}_i = a_i, \tag{2}$$

where $s_i$ the curvilinear position of robot $i$ along its path, $v_i$ its longitudinal velocity and $a_i$ its acceleration. This fixed-paths hypothesis is well-suited for road intersections, as vehicles generally stay inside circulation lanes.

The origin of $s_i$ is chosen so that $s_i = 0$ when the front of the robot enters the coordination region, and $s_i = s_i^{out} > 0$ when it fully exits. The velocity is non-negative bounded such that $v_i \in [0, \overline{v}_i]$, and we assume that the longitudinal acceleration $a_i$ of robot $i$ is bounded to an interval $\left[\underline{a_i}, \overline{a_i}\right]$ with $\underline{a_i} < 0 < \overline{a_i}$.

We assume that robot $i$ enters the coordination region at time $t_i^{in}$ with speed $v_i^{in} \in [0, \overline{v}_i]$, and is required to leave the coordination region with speed $v_i^{out}$. We let $t_i^{out}$ denote the corresponding exit time. Note that $v_i^{out}$ should be properly chosen to avoid collisions outside of the coordination region.

For a pair of distinct robots $i$ and $j$, we call *collision set* between $i$ and $j$, noted $\mathcal{C}_{ij} \subset [0, S_i^{out}] \times \left[0, S_j^{out}\right]$ the set of positions $(s_i, s_j)$ inside the coordination region where $i$ and $j$ would collide; we say that $i$ and $j$ are *conflicting* if $\mathcal{C}_{ij} \neq \emptyset$. To simplify this presentation, we assume that collision sets are connected, which amounts to assuming that two robots can cross paths at most once. Moreover, we approximate the exact collision sets with minimally bounding polygons, with edges either parallel to the coordinate axes or to the first diagonal $s_i = s_j$.

A key remark to be made is that, for two conflicting robots $i$ and $j$, one necessarily clears a conflict area before the other can enter. Previous work [20] has shown that the relative order of passage between conflicting robots can be interpreted as *priorities*, and that the given of priority relations for all conflicting robots define discrete classes of solutions to the coordination problem. In general, there can exist up to $2^{N(N-1)/2}$ such classes.

Building from these results, we show in [22] that complying with assigned priorities is equivalent to respecting a set of linear constraints involving binary variables. Indeed, under our hypotheses, four possible types of conflicts can exist: robots can follow one another throughout the coordination region (a), or have crossing (b), merging (c) or diverging (d) paths, as illustrated in Fig. 2. In these diagrams, priority "$i$ goes before $j$" (noted $i \succ j$) corresponds to cases where the trajectory lies on the right-hand side of the collision set $\mathcal{C}_{ij}$. Note that the additional integer variables are needed to deal with the fact that the boundary of these sets are piecewise-linear rather than linear.

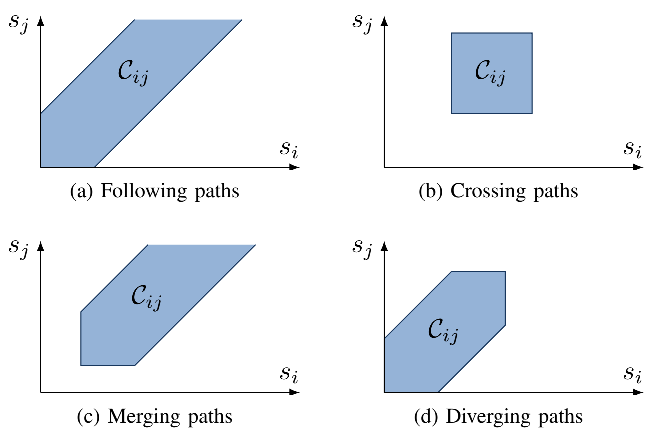



Fig. 2. Possible bounding polygons for the collision set between two conflicting robots $i$ and $j$. The edges of $\mathcal{C}_{ij}$ are straight lines with equations $s_i = A$, $s_j = B$ or $|s_i - s_j| = C$ where $A, B, C$ are constants.

Moreover, we show that using temporal discretization allows an exact linear modeling of second-order kinodynamic constraints for constant accelerations, unlike the previous approaches relying on spatial discretization which require approximating robots dynamics. More specifically, with a time step duration of $\tau$ and noting $s_i^k$ and $v_i^k$ the position and speed of vehicle $i$ at step $k$, constant acceleration second-order dynamics are enforced as:

$$s_i^{k+1} - s_i^k - \frac{1}{2}\left(v_i^{k+1} + v_i^k\right) = 0 \tag{3}$$

$$v_i^{k+1} - v_i^k \leq \overline{a_i}\tau \tag{4}$$

$$v_i^{k+1} - v_i^k \geq \underline{a_i}\tau \tag{5}$$

Since time is not directly a variable in this formulation, we also introduce a binary variable at each time step and for each vehicle, $\sigma_i^k$, which indicates if robot $i$ has exited the coordination region at step $k$. To minimize the average sojourn time of robots, we simply can maximize the sum $\sum_{i,k} \sigma_i^k$. Interestingly, this is equivalent to maximizing the throughput of the intersection.

This formulation, which can be handled by many widely available solvers, allows to simultaneously solve two problems: the discrete problem of choosing the best priority assignment, which is the aim of an automated intersection management policy, and the continuous (but discretized) problem of finding optimal trajectories respecting those priorities under kinodynamic constraints.

## IV. Policy evaluation

In this section, we show how this mixed-integer linear programming formulation can be used to evaluate an intersection management policy on the example of autonomous vehicles at an intersection. The simulation is based on the free traffic modeling tool SUMO [23] and uses its path generation algorithm to compute collisions sets. Vehicles are added using Poisson arrival times and normally-distributed entry speeds (truncated to a minimum speed, and a maximum speed). The optimization problem is solved using the commercial MILP solver Gurobi [24], using its Python interface to generate the constraints. Lastly, if the problem is feasible, the solution trajectories are simulated in SUMO using the TraCI interface to verify that they do not generate collisions. Vehicles are modeled as rectangles of $5\,\mathrm{m}$ length by $2\,\mathrm{m}$ width, with $[\underline{a_i}, \overline{a_i}] = [-3\,\mathrm{m\,s^{-2}}, 4\,\mathrm{m\,s^{-2}}]$. The exit speed for all vehicle $i$ is set as $v_i^{out} = \bar{v} = 15\,\mathrm{m\,s^{-1}}$ and $v_i^{in}$ is normally distributed with average $12\,\mathrm{m\,s^{-1}}$ and standard deviation $3\,\mathrm{m\,s^{-1}}$, truncated to $[10\,\mathrm{m\,s^{-1}}, 15\,\mathrm{m\,s^{-1}}]$.

Simulations are performed on a personal computer running on an Intel Core i7-4790 CPU (clocked at 3.60 GHz) with 16 GB of RAM.

We present the results from three distinct sets of simulations; our objective is to evaluate the performance loss caused by the choice of sub-optimal priority assignments. In order to do so, we compare the optimal assignment found using the problem of Section III with a first-come, first-served (FCFS) ordering of the vehicles as proposed, *e.g.*, in [1] which allows simple computation of optimal trajectories.

### A. Microscopic simulation

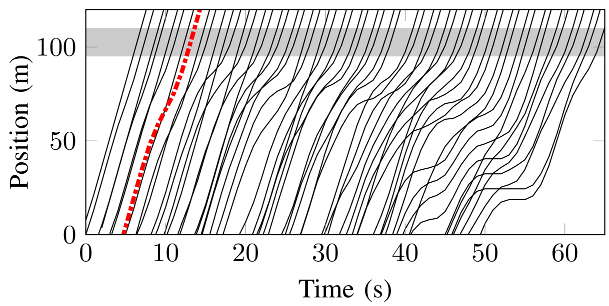
The first simulation compares the optimal trajectories with those resulting from first-come, first-served (FCFS) ordering. Figure 3 shows these trajectories, for a given set of 57 vehicles on a single-lane, four-roads intersection. For illustration purposes, vehicles only travel from south to north, and from east to west in this example. It can be seen that FCFS ordering causes a higher average sojourn time for vehicles, and in fact saturates the intersection, as can be seen in the increasing time vehicles remain stopped. By contrast, an optimal ordering allows the same vehicles to pass without stopping. Note that this does not guarantee that every vehicle is better served by an optimal ordering. For instance, the vehicle with trajectory highlighted in red exits the intersection later in the optimal ordering than in the FCFS ordering.

The attached video submission[1] shows two simulations run on the intersection presented in Fig. 1, with optimal and first-come, first-served ordering.

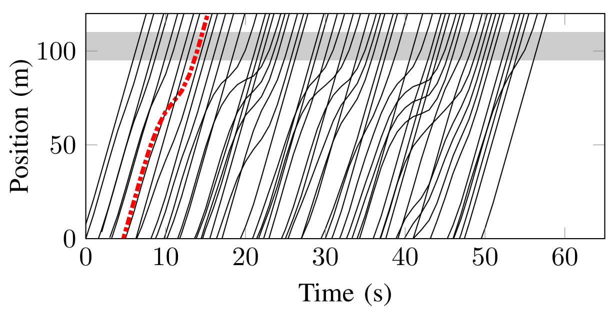
### B. Number of vehicles

It is known that first-come, first-served policies are often efficient for lower volumes of traffic, whereas they lose efficiency when the number of vehicles increases [16]. Fig. 4

[1]Also available at `https://youtu.be/U7I7x09Hkeo`




(a) First-come, first-served ordering




(b) Optimal ordering

Fig. 3. Trajectories of the vehicles in a single-lane intersection for FCFS and optimal ordering of the vehicles. The gray areas correspond to the center of the intersection. Notice that trajectories do not intersect in the gray areas.

shows the influence of the number of vehicles $N$ on the average relative time loss for the optimal priority assignment and FCFS ordering. This indicator is computed as $\frac{t_i^{policy} - t_i^{alone}}{t_i^{alone}}$, where $t_i^{alone}$ is the time vehicle $i$ would have needed to exit the coordination region if there were no other vehicle, and $t_i^{policy}$ the sojourn time of the vehicle in the given policy. These data have been computed over a set of 20 initial configurations on the intersection presented in Fig. 1.

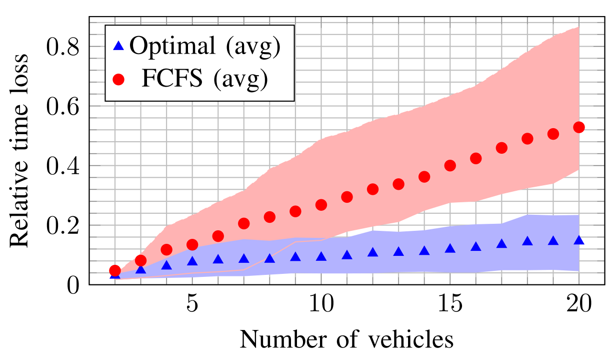



Fig. 4. Comparison of average normalized delays for optimal and FCFS ordering for an increasing number of robots. The shaded areas correspond to the $[50\%, 75\%]$ percentiles of vehicles for each policy.

### C. Influence of time step duration

The discretization time step has a double influence on the solution: first, the safety constraints force vehicles to have passed the conflict area one time step before another vehicle is admitted. Second, we assume constant acceleration during one time step. As the duration of a time step decreases, the average normalized delay is expected to decrease for the same initial configuration.

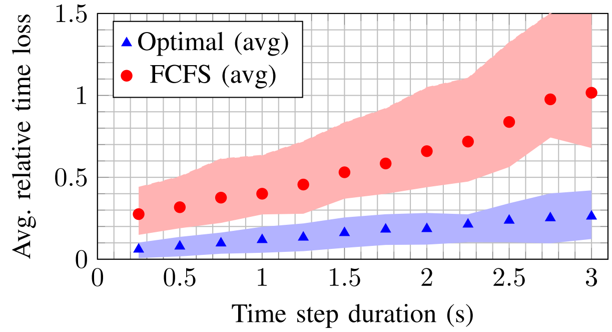



Fig. 5. Influence of the time step duration on the average normalized delay for optimal and first-come, first-served ordering. The shaded areas correspond to the $[50\%, 75\%]$ percentiles of vehicles for each policy.

To measure the effect of the choice of time step duration on solution quality, we show the average relative time loss for various values of the time step, for both optimal and first-come, first-served (FCFS) ordering in Fig. 5. These data have been computed over a set of 20 initial configurations comprising 15 vehicles. The shaded areas correspond to the $[50\%, 75\%]$ percentiles. Note that, due to the relatively small number of vehicles in the simulations, the first quartile (corresponding to the first $25\%$ of the vehicles) have a near-zero time loss for both optimal and FCFS orderings. It can be seen that the average delay in the optimal ordering is always below that of FCFS ordering. This result emphasizes the fact that optimizing vehicles trajectory for a given ordering is less effective than choosing a good ordering with lower-quality trajectories. In the remainder of this article, a time step duration of $1\,\mathrm{s}$ is used.

## V. Design of near-optimal policies

In this prospective section, we describe how using the presented formulation could allow to design near-optimal policies for real-time applications. A first possibility for reducing computation time with bounded optimality loss is to use less strict termination criteria for the MILP solver. Indeed, branch-and-bound algorithms used in most solvers keep exploring the decision tree until the current solution is very close to the best known bound for the objective function, usually with tolerated relative difference (or gap) of less than $10^{-4}$. In our particular formulation, we observe that the solver generally finds relatively quickly a reasonably good solution, then spends a lot of time improving this solution to gain a few percents of optimality. For real-time applications, a possible approach to decrease computation time is to increase the maximum tolerated gap, leading to a potentially sub-optimal solution in a much shorter time while ensuring an upper bound on the loss of optimality. In Fig. 6 (red and blue curves), we compare the average computation times for a gap tolerance of $10^{-4}$ and $10^{-1}$ over 10 instances for a time step duration of $1\,\mathrm{s}$; in our test instances, the actual

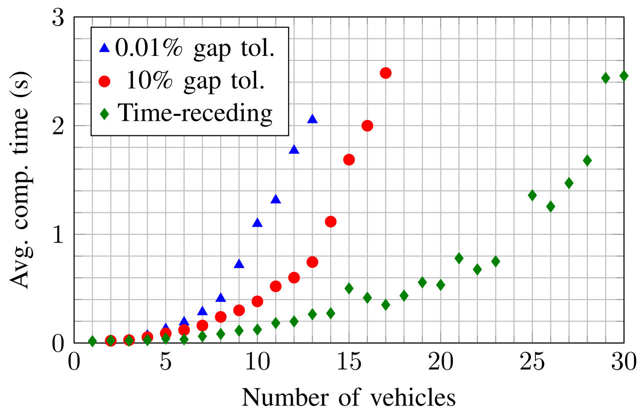



Fig. 6. Computation time, depending on the number of vehicles, for a gap tolerance value of $10^{-4}$ and $10^{-1}$, and for the time-receding approach with tolerance value of $10^{-1}$.

average optimality loss from increasing gap tolerance to $10\%$ is approximately $5\%$. Using this method, up to $14$ vehicles can be treated in less than a second, whereas only $10$ vehicles can be treated in the same time span with a gap tolerance of $10^{-4}$.

Another possibility for real-time applications is to use a receding time horizon approach. At each time step, the optimization problem can be solved only considering the vehicles currently inside the coordination region, taking their position and speed at the beginning of the time step as initial conditions. Vehicles inside the coordination region are tasked to adjust their control to comply with the optimal trajectory. This process can then be repeated at the next time step. Note that in this approach, priorities assigned at a given time step can still be modified in the following steps, which should have a very limited impact on optimality. The green curve in Fig. 6 shows the average computation time needed to converge (with a $10\%$ gap tolerance and $1\,\mathrm{s}$ time steps), as a function of the number of vehicles simultaneously present inside the coordination region. Using this approach, up to $24$ simultaneous vehicles can be treated in less than a second, which is reasonable for real world applications. For an even higher number of vehicles, choosing a longer time step duration could also be a possibility, since the proposed formulation ensures that such solutions remain feasible.

Note that vehicles are not required to follow the resolution-optimal trajectory found by the solver, as long as they respect their assigned relative priorities. For reasonable control laws with time as optimization criterion, the above-computed near-optimal policies will likely remain efficient although no optimality guarantee could be given. Notably, [21] has demonstrated a decentralized model predictive control scheme allowing vehicles to cross an intersection under assigned priorities.

## VI. Conclusion and Perspectives

In this paper, we have used a simplified autonomous intersection model: there are no unexpected events, no pedestrian, no human-driven vehicles, perfect communication, etc. However, this simplified model allows to reduce the complexity — although the problem remains NP-hard — and we can get optimal solutions for small instances of traffic flows. This is a first lesson: very clearly, such a time-optimal algorithm can be used to manage simple intersections where the traffic flow is limited. Using the rather standard average normalized delays criteria, this allows a significant decrease in the waiting times (from up to 40% to around 10%) comparing with a First-Come First-Served policy where all vehicles also optimize their speed profile. We must emphasize this is true for fluid traffic patterns. However, it shows that theoretically, one can compute an optimal scheduling in some cases. Moreover, we know from previous work that our priority framework allows easily to take into account unexpected events [25] or even distributed schemes [21], which could allow to relax some hypotheses.

What we need to do now it to get a better idea of the optimal patterns. We know from previous studies that FCFS is not optimal for dense traffic situations, where gated or exhaustive policies (in the sense of queuing theory, similar to adaptive traffic lights) are better, and may be optimal. This study is another confirmation FCFS is not optimal, though for very small traffic it is closer to optimality. Very clearly, we must change our methods for intersections subjected to large traffic flows since real-time computation of an optimal solution, even for our simplified model, is out of reach.

Moreover, we also showed that the branch-and-bound algorithm applied to this particular problem converges much faster than it terminates, meaning we can get optimal (or slightly suboptimal) solutions long before we have the proof of its optimality. This may lead to some approximated optimization that could still be operational for large flows. This is clearly an interesting avenue of research, as very few results exist on what a very good policy would be like for an intersection with dense to saturated traffic.

## References


[1] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004*. IEEE Computer Society, July 2004, pp. 530–537.

[2] O. Mehani and A. de La Fortelle, "Trajectory planning in a crossroads for a fleet of driverless vehicles," in *EUROCAST*, ser. Lecture Notes in Computer Science, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., vol. 4739. Springer, February 2007, pp. 1159–1166.

[3] A. de La Fortelle, "Analysis of reservation algorithms for cooperative planning at intersections," in *Proceedings of the 13th International IEEE Annual Conference on Intelligent Transportation Systems*, September 2010.

[4] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *Automatic Control, IEEE Transactions on*, vol. 60, no. 6, pp. 1515–1527, June 2015.

[5] A. de La Fortelle and X. Qian, "Autonomous driving at intersections: combining theoretical and practical approaches," in *ITS World Congress 2015 Proceedings*, 2015.

[6] P. Hunt, D. Robertson, R. Bretherton, and M. C. Royle, "The scoot on-line traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, 1982.

[7] P. Lowrie, "The sydney coordinated adaptive traffic system-principles, methodology, algorithms," in *International Conference on Road Traffic Signalling, 1982, London, United Kingdom*, no. 207, 1982.

[8] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 823–839, 2013.

[9] Y. Yin, "Robust optimal traffic signal timing," *Transportation Research Part B: Methodological*, vol. 42, no. 10, pp. 911–924, 2008.

[10] D. McKenney and T. White, "Distributed and adaptive traffic signal control within a realistic traffic simulation," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 574–583, 2013.

[11] S. Faye, C. Chaudet, and I. Demeure, "A distributed algorithm for adaptive traffic lights control," *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1572–1577, 2012.

[12] C. C.-N. Pandit Kartik, Ghosal Dipak, "Adaptive Traffic Signal ControlWith Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1459–1470, 2013.

[13] J. Gregoire, E. Frazzoli, A. de La Fortelle, and T. Wongpiromsarn, "Back-pressure traffic signal control with unknown routing rates," *CoRR*, vol. abs/1401.3357, 2014.

[14] J. Gregoire, X. Qian, E. Frazzoli, A. de La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, June 2015.

[15] R. Naumann, R. Rasche, and J. Tacken, "Managing autonomous vehicles at intersections," *IEEE Intelligent Systems*, vol. 13, no. 3, pp. 82–86, may/june 1998.

[16] M. Vasirani and S. Ossowski, "Evaluating Policies for Reservation-based Intersection Control," in *Proceedings of the 14th Portuguese Conference on Artificial Intelligence*, 2009, pp. 39–50.

[17] A. R. Bill, "A Next-Generation Intersection Control Algorithm for Autonomous Vehicles," in *Transportation Research Board 92nd Annual Meeting*, 2013.

[18] J. Peng, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, apr 2005.

[19] V. Digani, M. A. Hsieh, L. Sabattini, and C. Secchi, "A Quadratic Programming approach for coordinating multi-AGV systems," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, aug 2015, pp. 600–605.

[20] J. Gregoire, "Priority-based coordination of mobile robots," *arXiv preprint*, vol. arXiv:1306.0785, 2013.

[21] X. Qian, J. Gregoire, A. de La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *Control Conference (ECC), 2015 European*, July 2015, pp. 3452–3458.

[22] F. Altché, X. Qian, and A. de La Fortelle, "Time-optimal Coordination of Mobile Robots along Specified Paths," *arXiv preprint*, vol. arXiv:1603.04610, 2016.

[23] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[24] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[25] J. Gregoire, S. Bonnabel, and A. De La Fortelle, "Priority-based intersection management with kinodynamic constraints," in *European Control Conference 2014*, Oct. 2014, european Control Conference 2014.