Intelligent Intersection Control for Platoons of Autonomous Vehicles

William Shen
Research School of Computer Science
Australian National University
Canberra, Australia
Email: u6096655@anu.edu.au

***Abstract*—The rise in the development of autonomous vehicles brings interesting challenges especially in the coordination between such vehicles. In an environment made up entirely of autonomous vehicles, intersections should be controlled intelligently by scheduling vehicles through the intersection to reduce delays and improve travel times. We investigate potential algorithms to schedule platoons (groups) of autonomous vehicles through intersections to maximize throughput — thus reducing cost, congestion, travel time and energy consumption. We approach this as an online scheduling problem using vehicle to vehicle communication rather than vehicle to infrastructure communication to significantly reduce the costs of implementing, and to increase the potential scalability of such a solution.**

We propose an integer linear programming scheduling algorithm, and present initial findings which show that the plans produced by our scheduling algorithm can save significant amounts of time compared to traditional intersection control methods including stop signs.

Keywords*—*Autonomous Intersection Management, Platooning, Intelligent Transportation Systems, Intelligent Vehicle Systems

I. Introduction

The rise in the development of autonomous vehicles brings interesting challenges especially in the coordination between such vehicles. In an environment made up entirely of autonomous vehicles, intersections should not rely on traditional control methods including traffic lights and stop signs. These traditional control methods obstruct the flow of traffic, as vehicles must come to a standstill and wait, even if there will be no other vehicles passing through the intersection in the near future.

Instead, intersections should be controlled intelligently by determining the best set of actions when autonomous vehicles approach that intersection to minimize any potential delays and maximize throughput for that intersection.

Although there are existing solutions which can be used to solve this issue, including a first-come-first-served reservation system in [1], we believe that better solutions exist that rely primarily on vehicle to vehicle (V2V) communication rather than vehicle to infrastructure (V2I) communication, and the concept of vehicles travelling in platoons.

Existing solutions primarily rely on a 'central coordinator' at each intersection which vehicles send crossing requests to. By using V2V rather than V2I communication, we can reduce the costs of implementation whilst allowing additional communication between the vehicles themselves.

With the advent of autonomous vehicles, platooning is likely to become more widely adopted. In platoons, vehicles are grouped together such that each platoon can be considered and behave as a single vehicle itself (i.e. simultaneous acceleration and braking). This leads to several potential benefits, including reduced energy consumption through drag reduction, and decreased travel times.

Thus, we aim to investigate algorithms that intelligently control platoons of autonomous vehicles through intersections to maximize throughput. We approach this as an online scheduling problem, where we calculate the best schedule in real-time as the platoons of vehicle approach the intersection.

The Australian Bureau of Infrastructure, Transport and Regional Economics (BITRE) estimated in 2015 that the 'avoidable' social costs of traffic congestion was $16.5 billion AUD [2]. BITRE projections show that this cost could reach $30 billion AUD by 2030. Ultimately, an intelligent autonomous intersection controller could reduce travel time, congestion, energy consumption and pollution, leading to a stronger economy through a rise in efficiency.

Studies by the National Cooperative Highway Research Program (NCHRP) in the United States show that around 50% of urban crashes and 30% of rural crashes occur at intersections [3]. Thus, our intersection controller would provide all the economic benefits as aforementioned, as well as improving road safety and overall human health.

The rest of the paper is structured as follows: In Section II, we discuss current research and state of the art methods for intersection management. In Sections III, IV and V, we formally define the entities in our scheduling problem and the sub-problems which we need to consider. Next, in Sections VI and VII, we propose a naive and an optimal scheduling algorithm respectively. We then present initial findings in Section IX. Finally, in Sections X and XI, we discuss the impact of our research and future explorations on the topic.

II. Background and Related Work

A. General Traffic Management

Much research has gone into traffic signal control and signal-less intersection control. In [4], the authors explore real-time traffic signal control using platoons of incoming vehicles. [1] and [5] present centralized scheduling algorithms that rely on an intersection management agent.

As evident, these rely on a central coordinator at each intersection. Decentralized intersection controllers would be more widely implementable and cost-effective than centralized solutions. We discuss signal-less intersection control, as it has the potential to be more efficient than traditional traffic control methods.

B. *Centralized Solutions*

Centralized solutions to intersection management rely on Vehicle to Infrastructure communication. They use scheduling algorithms including first-come-first-serve (FCFS) and priority-based scheduling.

In [1], each intersection has an agent which manages control of an intersection. Vehicles send reservation requests to these agents, and wait for their request to be accepted or denied. The server processes these requests using a FCFS scheme by comparing them against an Intersection Control Policy which determines whether it is safe for the vehicle to pass through the intersection at the requested time. This approach could lead to large blockades of vehicles as the algorithm does not account for delays or travel time. Clearly, this is undesirable.

[5] proposes an improved algorithm with optimal scheduling where vehicles continuously send their status and reservations to the intersection management agent to minimize the total travel times. Although [5] represents a vast improvement over [1] as it considers travel times, it still relies on vehicles sending requests to a central scheduling agent at each intersection.

Of course, a centralized solution is not extremely feasible as the costs of having a server at each intersection is astronomical. For intuition, consider a residential area with a dense road network. It would not be ideal having a controller server at each intersection.

C. *Decentralized Solutions*

Decentralized solutions to intersection management rely on Vehicle to Vehicle communication. As there is no need for central infrastructure, the costs of implementing a decentralized solution is very low in comparison to a centralized solution.

Similar to centralized solutions, many adopt a reservation-based scheduling approach. In [6], the authors present a distributed reservation table algorithm and an unstructured peer to peer reservation algorithm. [7] discusses auction-based autonomous intersection management where each vehicle 'bids' money to be scheduled to cross the intersection sooner or later.

Evidently, these approaches are not focused on reducing potential delays or travel times. We consider an 'optimization'-based approach that schedules vehicles based on costly parameters rather than a reservation-based approach. We aim to develop an algorithm that is both decentralized and efficient. We approach this as a real-time, online scheduling problem that exploits the power of platooning.

III. Problem Overview

A. *Assumptions*

Here, we make several important assumptions about our environment:

- Every vehicle on the road is fully autonomous and supports, at minimum, the same set of operations (i.e. accelerate, brake, turn)
- Vehicles are interconnected and thus can communicate with each other (this could be through Vehicular ad-hoc Networks (VANETs))
- Inter-vehicular communication is near instantaneous
- There will always be enough vehicles so that a message may be propagated throughout our road network
- We have the ability to create or alter platoons as we see fit
- Each vehicle has a destination and an assigned route it must take (see Section V for route generation)
- The demand model of the network of vehicles is current demand – i.e. we handle vehicles as they come online.

B. *Plan*

We define a plan to be the sequence of operations that must be applied to an arbitrary number of vehicles or platoons approaching an intersection. We call a plan 'valid' if:

- No vehicle or platoon will overlap with another during the execution of the plan. That is, two vehicles or platoons cannot be in the same place at the same time, as this is a collision.

C. *Optimal Plan*

We define a plan to be optimal if it is a valid plan, and if it minimizes our objective function. We propose the total travel time for our network of vehicles or platoons to be the objective function.

D. *Vehicular Communication*

As outlined earlier, each platoon will communicate with others by using vehicle to vehicle communication. By doing so, we remove the need for a central request-processing server, thus reducing costs whilst increasing the scalability of our solution.

Vehicle to vehicle communication also allows platoons to broadcast and communicate with other platoons who may cross the same intersection in a similar time-frame ahead of time. This will allow us to calculate an optimal plan in advance and ensure disruptions are minimized.

IV. Problem Formulation

A. *Vehicle*

A vehicle has the following properties:

- $start$, the starting point of the vehicle which is used for route generation.
- end, the ending point of the vehicle which is used for route generation.
- $route$, a route which is the sequence of intersections it must travel through to reach a certain destination. This is calculated by using the road network and the $start$ and end points (see Section V).
- $next$, a reference to the next intersection the vehicle will cross.
- $heading$, in what direction the vehicle is approaching $next$ (e.g. South-East, 320°).
- d, how far the front of the vehicle is away from entering $next$, measured in m.

- l, the length of the vehicle, measured in m.
- v, the velocity of the vehicle, measured in ms^{-1}.
- a, the acceleration of the vehicle, measured in ms^{-2}.
- t_a, the estimated time of arrival to the intersection $next$, i.e. when $d = 0$.
- t_d, the estimated time of departure from the intersection $next$, i.e. when the entire vehicle has completely cleared the intersection.

We adopt the time of arrival and time of departure variables similar to [5]. To access the properties of a vehicle, we use the '.' operator. For example, a vehicle x's next intersection would be accessed using $x.next$.

B. Platoon

A platoon p consists of several vehicles, $[v_1, v_2, ..., v_n]$ in order, each containing the states described above. Here we assume list/array indexing starts at 1. We define the following properties for a platoon:

- $vehicles$, an array/list of references to the vehicles in the platoon, $[v_1, v_2, ..., v_n]$.
- $size$, the number of vehicles in the platoon, i.e. $|vehicles|$.
- d, how far the front of the platoon is away from the next intersection, i.e. $vehicles[1].d$.
- l, the length of the platoon $= \sum_{x \in vehicles} x.l$, i.e. the sum over the lengths of each vehicle in the platoon.
- v, the velocity of the platoon, we assume $\forall x \in vehicles,\ v = x.v$, i.e. each vehicle in the platoon has the same velocity.
- a, the acceleration of the platoon, we assume $\forall x \in vehicles,\ a = x.v$, i.e. each vehicle in the platoon has the same acceleration.
- t_a, the estimated time of arrival to the intersection for the platoon, i.e. $vehicles[1].t_a$.
- t_d, the estimated time of departure to the intersection for the platoon, i.e. $\max_{x \in vehicles} x.t_d$.

A platoon itself does not have a route, but each vehicle in $p.vehicles$ does. We group vehicles together into platoons if they are travelling on similar routes. From here onwards, we use the term vehicular agent (VA) to refer to a vehicle or platoon synonymously.

C. Vehicular Agent Operations

Each vehicular agent supports acceleration, braking, and turning. In our representation, we denote braking by negative acceleration.

Moreover, we may form platoons of vehicles, add a vehicle onto a platoon, or split vehicles from a platoon.

D. Intersection

Each intersection has a number of lanes, each with a width and direction in which traffic flows. This direction is also used in describing the displacement of a vehicular agent as discussed earlier.

More importantly, an intersection has a central area where the actual crossing of vehicular agents takes place. In theory, this area could take on any shape and size. This central area is where we need to check whether collisions occur or not.

For simplicity, without a loss of generality, we assume a cross intersection with one lane for each flow of traffic in a left-hand driving environment. Let the width w of each lane be constant. Then, the central area of this intersection is $(2 \times w)^2 = 4w^2$.

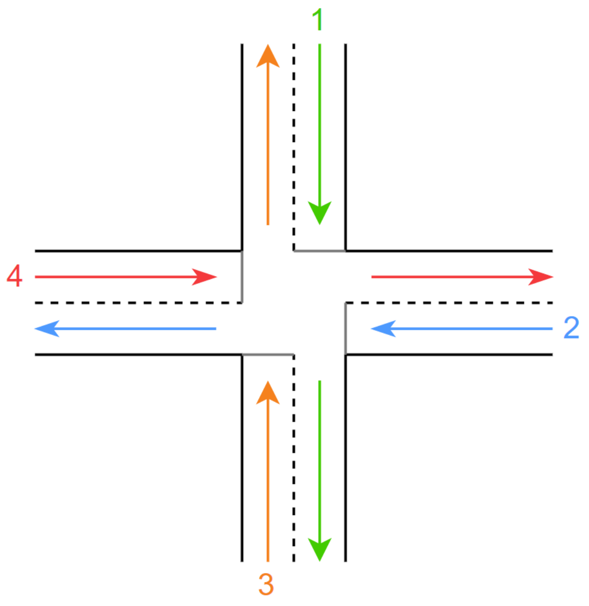


Fig. 1. Cross Intersection with one lane for each flow of traffic

In figure 1, the traffic flows in the direction of the arrow. That is, traffic in lane 1 flows South-bound, traffic in lane 2 flows West-bound, etc.

E. Trajectory

We denote each lane with a unique identification number α. In the case of figure 1, we have $\alpha \in \{1, 2, 3, 4\}$. Now, from each lane, a vehicle or platoon has the possibility of turning left, going straight, or turning right. We denote these operations by $\beta \in \{l, s, r\}$. We say a vehicle has a β intent at an intersection.

We define a trajectory to be the tuple (α, β). We can determine the trajectory of a vehicle from the structure of the road network combined with its properties $next$ and $route$.

Now, we define a pair of trajectories as being crossing if they share at least one crossing point in the intersection. For example $(3, r)$ and $(4, s)$ are crossing pairs, while $(1, s)$ and $(2, l)$ are non-crossing pairs. Evidently, we only need to check for collisions when a pair of trajectories are crossing. This ensures we remove any unnecessary computation.

F. Road Network

We may represent our road network as a weighted undirected graph $G = (V, E)$. Each vertex in V represents an intersection, and each edge in E represents the road between two intersections. Evidently, the weight on each edge represents the distance between the connecting intersections. An edge may have additional properties such as a speed limit, congestion level (traffic), or rules for what types of VA can travel on the lane (e.g. transit lane).

If we require different road lengths in each direction of travel, we may use a weighted directed graph. If we require several lanes in each direction of travel, we can use a weighted multi-directed graph.

The road network is used by each VA to keep track of its location, progress to the next intersection, etc.

V. Sub-problems

A. Route Generation

Recall that each vehicle has a starting point and an ending point. These may be considered as points on the edges of our road network.

When a vehicular agent comes online, we must assign it a route based on its desired destination. We are not concerned with how this route is assigned as we will be focusing on managing intersections.

Thus, we assume that each vehicle is automatically assigned a ‘good’ route. A ‘good’ route could take the current traffic on each edge of our graph into account (i.e. penalize high traffic edge) when generating a route.

B. Collision Detection

A plan is invalid if leads to any collision between the vehicular agents. That is, the location of a VA overlaps with the location of another.

Detecting collisions when vehicular agents are on the edges of the road network (i.e. the roads) is simple. We can simply calculate whether a vehicular agent will ‘catch-up’ with another vehicular agent ahead. Without a loss of generality, we present the conditions for collision for two vehicles:

```
// INPUT: Two vehicles travelling on the same
    road (i.e. edge)
// OUTPUT: Whether there will be a collision
boolean roadCollision(Vehicle x, Vehicle y) {
    // Check time of arrival for the end of the
        vehicle does not overlap with time of
        arrival for the front of other vehicle
    if (x.d == y.d) {
        return true;
    } else if (x.d < y.d) {
        return x.t_a + x.d / x.v < y.t_a;
    } else {
        return y.t_a + y.d / y.v < x.t_a;
    }
}
```

Detecting collisions inside the central area of an intersection is more difficult but still achievable. The most straightforward solution is the one implemented in the Autonomous Intersection Management project [1] by the University of Texas at Austin, where the area of the intersection is divided up into little grids as shown in figure 2.

If any of the grid squares are already occupied by another vehicle, then we have detected a collision and hence our plan is invalid. We use the velocity, acceleration and time of arrival of each vehicular agent to determine whether there may be a collision or not. Note that from earlier, if the trajectory of two vehicular agents are non-crossing, then we do not need to check for collisions for those agents.

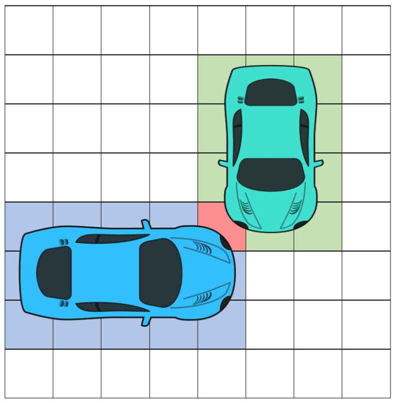


Fig. 2. Intersection Collision Detection

We present an collision detection algorithm for two vehicular agents without a loss of generality. Assume that these vehicles are on crossing trajectories.

```
// INPUT: Two vehicles bound to cross the same
    intersection
// OUTPUT: Whether there will be a collision
boolean intersectionCollision(Vehicle x, Vehicle
    y) {
    if (x.t_d <= y.t_a || y.t_d <= x.t_a)
        return false;
    if (!crossingTrajectory(x.trajectory(),
        y.trajectory())
        return false;

    // Collision table for next intersection
    CollisionTable table = new
        CollisionTable(x.next);

    // Assume time steps of 10ms
    Timestamp time = System.Time();
    while time < max(x.t_d, y.t_d) {
        if (t >= min(x.t_d, y.t_d)
            return false;

        // Progress each vehicle by 10ms and
            update table accordingly
        x.d = x.d - (x.v / 100);
        if (x.d < 0 && table.occupied(x))
            return true;
        else
            table.update(x);
        y.d = y.d - (y.v / 100);
        if (y.d < 0 && table.occupied(y))
            return true;
        else
            table.update(y);
        time = time + 10;
    }
    return false;
}
```

C. Communication Protocol

Once a valid plan for an intersection has been calculated giving the incoming vehicles, the results of executing such a plan (i.e. the next states for the VAs) will be broadcasted to all other vehicles in the nearby vicinity using vehicle to vehicle communication (V2V).

By broadcasting this information, we allow vehicular agents to determine which other vehicular agents they will encounter at future intersections, so that they can agree on a plan early on and thus minimize any potential disruptions.

Vehicles continuously broadcast their state to ensure that the plan is being followed, and to account for any new vehicles which may be joining the road network.

VI. Constraint-Based Scheduling

In the naive solution, we only care about generating a valid plan. Thus, such a plan may be generated using constraint programming.

We present the logical constraints that must be satisfied in order for a plan to be valid:

- Vehicular agents travelling on the same lane must not collide. i.e. For two vehicular agents x and y with trajectories (α_x, β_x) and (α_y, β_y) respectively, $\alpha_x = \alpha_y$ (recall α is the unique lane identification number):
    - $\neg\ roadCollision(x, y)$
    - if $x.t_a < y.t_a$. then $x.t_d < y.t_d$ (we assume no overtaking within the intersection)
- Vehicular agents with crossing trajectories must not collide. i.e. i.e. For two vehicular agents x and y with trajectories (α_x, β_x) and (α_y, β_y) respectively:
    - $\neg\ intersectionCollision(x, y)$

Recall t_a represents the time of arrival of a VA into the intersection, and t_d represents the time of departure of a VA from the intersection.

We let the decision variables be t_a and t_d for each vehicular agent, and solve the system of constraints. With these results, we calculate and make the changes in velocity and acceleration required to satisfy the planned t_a and t_d for each VA.

VII. Optimal Scheduling

In order to find an optimal solution, we adopt linear programming as we may model our scheduling problem with a linear objective function and a linear number of constraints. To formulate the problem, we firstly define the decision variables.

A. Decision Variables

For each vehicular agent v approaching the intersection, we let t_a (time of arrival) and t_d (time of departure) be the decision variables as seen in Section VI. By definition, $0 \leq t_a < t_d$.

To support platooning, we assume that the scheduler will return the t_a and t_d for each vehicle accordingly. We introduce additional constraints to ensure the consistency of vehicular agents in subsection C.

B. Objective Function

We propose that minimizing the travel time through the intersection is a sufficient objective function. That is, we aim to minimize the total time required for vehicular agents to cross an intersection.

$$\min \sum_{i=1}^{n} (v_i.t_d - v_i.t_a) \tag{1}$$

Here, n is the total number of vehicular agents currently travelling towards the intersection. v_i represents a specific vehicular agent where $i \in \{1, ..., n\}$.

In the future, we may want to investigate other objective functions, including parameters such as power consumption and delays.

C. Constraints and Schedule Calculation

We incorporate the same constraints as in Section VI on naive scheduling. Additionally, we introduce the following constraints to ensure consistencies across platoons and vehicles.

Let p represent the platoon, and $V = \{v_1, ..., v_n\}$ represent the vehicular agents that form or have been split from this platoon in order. That is, each v_i is a platoon or vehicle itself. Then:

$$p.size = \sum_{v \in V} (v.size \text{ if } v \text{ is a platoon, else } 1) \tag{2}$$

$$\forall x \in p.vehicles,\ p.v = x.v \land p.a = x.a \tag{3}$$

$$\begin{aligned} \forall x \in \{1, ..., p.size - 1\},\ & p.vehicles[x].d \\ & + p.vehicles[x].l + \gamma = p.vehicles[x+1].d \end{aligned} \tag{4}$$

In (4), $\gamma > 0$ is the 'safe' distance we must keep between each vehicle in the platoon (measured in metres).

Constraint (2) ensures that the number of vehicles in the platoon is consistent. Constraint (3) ensures that each vehicle in the platoon has the same velocity and acceleration, while constraint (4) ensures that each vehicle in the platoon is 'almost' touching only separated by γ.

Additionally, these constraints and the ones in Section VI must hold with each vehicular agent in V.

We have yet to allow for any 'clearing time' between vehicular agents. We introduce $\epsilon > 0$, the minimum time gap between the crossing of two vehicles. That is, for any two vehicular agents x and y, $|x.d - y.d| > \epsilon$.

D. Schedule Calculation

Now, with our optimization problem formulated, we may pass it to an integer programming solver as our decision variables can be broken down into integers. Similar to the naive algorithm, we receive the optimal t_a and t_d for each vehicular agent v. We calculate the changes in velocity and acceleration required to satisfy the valid plan and apply them.

For intuition, if a vehicle v's current t_a is earlier its previous t_a, it must accelerate to the corresponding velocity in order to match its new t_a and t_d.

VIII. Example Scenario

Here, we present an example intersection crossing scenario and show that our algorithm would correctly produce valid, optimal plans. For simplicity's sake, we consider a cross intersection with only two directions of travel, East and South-bound (see figure 3). Assume there is only one lane for each direction of traffic flow.

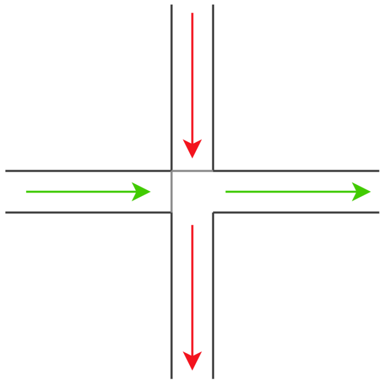


Fig. 3. Cross Intersection with one East and South-bound traffic only

We can represent the East-bound traffic with the trajectory pair $(1, s)$ (where s represents the turning intent, in this case, continuing straight) and the South-bound traffic with the trajectory pair $(2, s)$. Clearly, these trajectory pairs are crossing. Assume that the speed limit of this intersection and the connecting roads is $15ms^{-1}$.

Moreover, assume that vehicles are in platoons of a certain length, say $25m$ and are continuously flowing into the intersection at the speed limit at the same rate in both lanes. Let the width of each lane be $5m$ – that is, a VA must travel $5m$ plus its length to complete cross the intersection at the intersection start point.

For the sake of this example, we use the terms platoon and vehicle synonymously.

Then the time required for a platoon to travel through the intersection at the speed limit is $\frac{5+25}{15} = 2$ seconds. We aim to calculate the optimal gap γ between platoons on each lane such to minimize the total travel time through the intersection.

A. Optimal Formula

Assume that we have a global timestamp that starts at 0. As each platoon requires $2s$ to completely clear the intersection, and we need to ensure a 'clearing time' for safety purposes, the next platoon should enter the intersection at $(2 + \epsilon)$ time.

Allowing crossings in alternating sequence between the lanes, we derive the following formula. Assume that the vehicles in the South-bound lane are $S = \{s_1, ..., s_n\}$ and the vehicles in the East-bound lane are $E = \{e_1, ..., e_n\}$.

For S, we define:

- $s_1.t_a = 0$
- $s_1.t_d = s_1.t_a + 2 = 2$
- $\forall i \in \{2, ..., n\}, s_i.t_a = s_{i-1}.t_d + (2 + \epsilon)$

For E, we define:

- $e_1.t_a = 2 + \epsilon$
- $e_1.t_d = e_1.t_a + 2 = 4 + \epsilon$
- $\forall i \in \{2, ..., n\}, e_i.t_a = e_{i-1}.t_a + (2 + \epsilon)$

Note: We have added $(2+\epsilon)$ account for the time required for the VA crossing in the other direction's lane.

Clearly for each vehicle v in S and E, $v.t_d = v.t_a + 2$ as the vehicle takes 2 seconds to travel through the intersection at full velocity.

Notice that our plan for each vehicle follows all the constraints defined in Sections VI and VII. For the given scenario, we would expect our optimal scheduling algorithm to generate this plan.

B. Total Travel Time

We have ensured no collisions between the vehicles whilst allowing for a safe 'clearing time' ϵ. Moreover, by travelling at the speed limit of the road network we have minimized the travel time through the intersection for each vehicle.

To calculate the total travel time for n vehicles in each lane, we use the following formula:

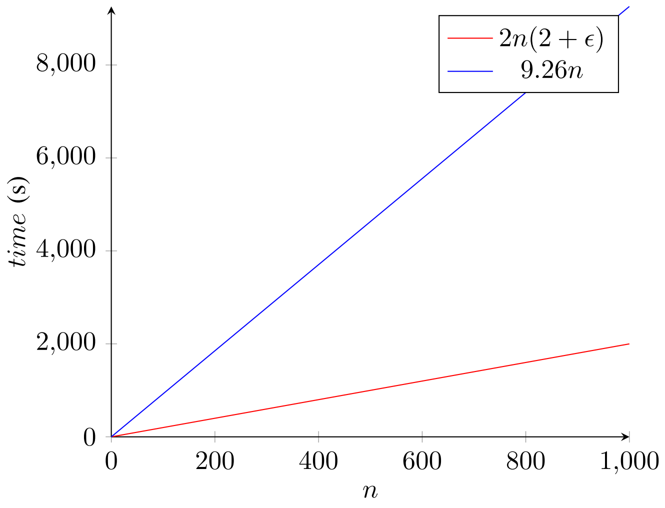
$$total\_travel\_time(n) = 2n(2 + \epsilon)$$

C. Comparison

Here, we compare the total travel time of our optimal formula as calculated above, to a modified stop sign system. For simplicity of calculation, we assume the following properties:

- The vehicles must stop once they reach the stop sign, or join the queue of vehicles waiting at the stop sign.
- The vehicles in each lane give way in alternating sequence
- The next vehicle begins crossing the intersection immediately after the previous vehicle has cleared it.
- Assuming a 'comfortable' acceleration rate of $2.8ms^{-2}$, it takes each vehicle $t = \sqrt{2s/a} = \sqrt{(2 \times (5 + 25))(2.8)} \approx 4.63s$ to cross the intersection.

Then, for n vehicles in each lane it takes a total of approximately $2n(4.63) = 9.26n$ seconds to clear all the vehicles. Clearly, $9.26n > 2n(2 + \epsilon)$.



The optimal plan clearly has significant time savings over a modified stop sign system. Of course, such a comparison is rather simple. We ideally want to experiment different scenarios with simulation software. We outline this in the future work section.

D. Speed Limits

We investigate the throughput of the intersection for the scenario presented above, but now with different speed limits v (measured in ms^{-1}). Assuming that vehicles always travel at the speed limit, the time t it takes for a vehicle to cross the intersection can be given by the following formula.

$$t = \frac{5+25}{s} = \frac{30}{v}$$

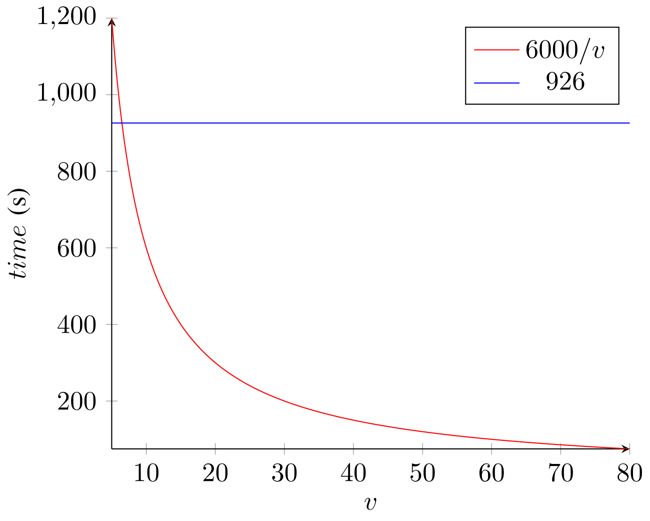
We define each t_a and t_d of a vehicle similar to part (b). Now, we derive the following total travel time formula for the optimal plan:

$$total\_travel\_time(n) = 2n(t+\epsilon) = 2n(\frac{30}{v} + \epsilon)$$

We assume that the time to cross the intersection using the modified stop sign system is unchanged. That is, for n vehicles in each lane it takes $9.26n$ seconds to clear all the vehicles.

We let the number of vehicles in each lane $n = 100$, and observe the total time required to clear them based on varying speed limits v. Then, the total travel time of our optimal plan is given by $6000/v$ (ignoring ϵ). The total travel time for the modified stop sign system is $9.26 \times 100 = 926$ seconds.

We begin the graph at $v = 5$ to show the characteristics of the optimal plan:



Note: We assume that the speed limit $v \geq 12.946ms^{-1}$, the velocity of the vehicle the moment it clears the intersection under the modified stop sign system.

We observe diminishing returns as we increase the speed limit v. With each increase in v, the marginal time saved is clearly decreasing. Thus, at least for this simple scenario, there does not seem to be an optimum speed limit. The total travel time continues decreasing as we increase the speed limit v.

IX. Results and Evaluation

We have presented initial findings for a simple example scenario and shown that the savings in travel time can be very significant. Given the limited research time-frame, we were unable to implement the algorithm, and run simulations.

Our initial findings have shown that our optimal scheduler can perform much better than traditional traffic control methods such as stop-signs.

Compared to centralized, reservation-based, first-come-first-served intersection management systems, we theorize our solution to perform better as we calculate plans for the entire network of vehicles as they approach the intersection, rather than each vehicle having to send a reservation request.

We also showed that for our example scenario, there is no optimal speed limit to maximize the throughput. We expect these results to transfer to similar intersections, but we should carefully note that we need to calculate the optimal speed limit (if any) based on an intersection by intersection basis.

X. Future Work

To understand the true effectiveness of our scheduling algorithms, simulations on a larger scale should be carried out. Given the short research time-frame, we were unable to implement our scheduling algorithm and connect it to a simulation environment such as SUMO or Aimsun. In these simulations, several factors should be measured including travel time, congestion, energy consumption, and pollution. Therefore, we should also consider different objective functions for calculating an optimal plan – that is, we could include aim to reduce congestion and pollution along with travel time.

We would evaluate the effectiveness of our algorithms against the results of simulating intersections with traditional intersection control measures (i.e. traffic lights, stop and give way signs). Moreover, we should compare the results of our scheduling algorithm against existing ones, including the reservation-based first-come-first-served system.

We also aim to better investigate rules and heuristics to form or split platoons. Although our scheduling algorithm supports such platooning by satisfying the constraints, we have not explicitly identified ways to determine how this forming or splitting occurs.

The proposed algorithm should also be investigated and formalized in a distributed setting. For example, we must ensure that no VA can 'game' the intersection manager by sending invalid state information such to increase its own performance, or introduce safety concerns.

XI. Conclusion

We have studied decentralized algorithms for scheduling platoons of autonomous vehicles through an intersection. Firstly, we described and formulated the intersection control problem. We then discussed several sub-problems including collision detection.

Next, we provided a linear-programming based scheduling algorithm that minimizes the total time required for vehicular agents to cross an intersection, and described the communication protocols so optimal plans can be agreed upon in advance through vehicle to vehicle communication (V2V).

We gave an example scenario, and showed the potential time benefits the optimal plan produced by our scheduling algorithm could have. Next, we investigated optimal speed limits for our example scenario, and concluded that there is no true speed limit at which the travel time is minimized whilst observing an example of diminishing returns.

Our initial findings show that our scheduling algorithm could have significant time gains over traditional control methods and reservation-based intersection managers. Of course, we would need to implement our algorithm and run simulations over different intersections and number of vehicles to evaluate its true benefit.

Acknowledgment

I would like to thank my supervisor, Jochen Renz, for his support, advice and encouragement throughout this research project.

References

[1] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[2] D. Cosgove, *Traffic and congestion cost trends for Australian capital cities*. Department of Infrastructure and Regional Development, Bureau of Infrastructure, Transport and Regional Economics, 2015.

[3] T. R. Neuman, *Guidance for implementation of the AASHTO strategic highway safety plan. A guide for reducing collisions at signalized intersections*, 2004.

[4] X. F. Xie, G. J. Barlow, S. F. Smith, and Z. B. Rubinstein, "Platoon-based self-scheduling for real-time traffic signal control," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2011, pp. 879–884.

[5] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth, "Advanced intersection management for connected vehicles using a multi-agent systems approach," in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 932–937.

[6] S. Adams and M. J. Rutherford, "Decentralized intersection management through peer-to-peer technology," 2015.

[7] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 529–534.

[8] A. I. M. Medina, N. van de Wouw, and H. Nijmeijer, "Automation of a t-intersection using virtual platoons of cooperative autonomous vehicles," in *IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.

[9] J. J. B. Vial, W. E. Devanny, D. Eppstein, and M. T. Goodrich, "Scheduling autonomous vehicle platoons through an unregulated intersection," *CoRR*, vol. abs/1609.04512, 2016. [Online]. Available: http://arxiv.org/abs/1609.04512

[10] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth, "Platoon-based multi-agent intersection management for connected vehicle," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 1462–1467.