

# Trajectory Optimization of Autonomous Vehicles in Networks of Intersections

Jeroen van Riel

March 2025

## Contents

<b>1 Trajectories in tandem of two intersections</b>	<b>1</b>
1.1 Single vehicle . . . . .	2
1.2 Multiple vehicles . . . . .	3
1.2.1 Ad-hoc approach . . . . .	4
1.2.2 Schedule time approach . . . . .	5
<b>2 Trajectories in networks</b>	<b>6</b>
2.1 General decomposition . . . . .	7
2.2 Decomposition for delay objective . . . . .	7
2.3 Crossing time scheduling . . . . .	8
2.4 Branch-and-bound . . . . .	8
<b>3 Constructive heuristic</b>	<b>8</b>

## 1 Trajectories in tandem of two intersections

When considering multiple vehicle driving between intersection, we can no longer ignore the issue of road capacity, because the fact that only a limited number of vehicles can drive or wait at the same time on a lane between intersections may cause network-wide effects. The capacity of lanes between intersections is intimately related to the trajectories of vehicles, which we first want to understand better. We have been using an optimal control formulation with the objective that keeps the vehicles as close as possible to the next intersection at all times (`MotionSynthesize`). This problem can be solved using direct transcription, which

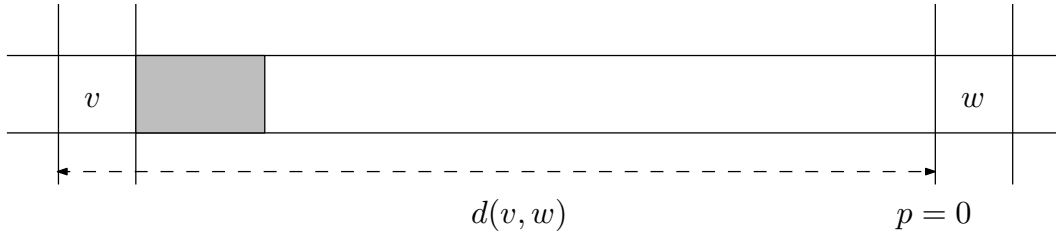


Figure 1: Tandem of two intersections  $v$  and  $w$  with lane of length  $d(v, w)$ . The grey rectangle represents some vehicle that just left intersection  $v$ . We will assume that a vehicle has maximum speed as long as it occupies an intersection, so it is only allowed to decelerate once it passes this position.

works well enough if we just want to simulate the behavior of the system. However, we believe that it is possible to explicitly formulate the optimal controller. We will explain how to compute trajectories corresponding to those obtained by direct transcription, but without using time discretization.

Before we turn to the general case of networks of intersection, we will first investigate the trajectories of vehicles in a tandem of two intersections as depicted in Figure 1. Let  $v$  denote the left intersection and  $w$  the right intersection and assume that vehicles drive from left to right. Furthermore, we will call the road segment strictly between both intersection areas the *lane*. To facilitate the following discussion, we will use  $p$  to denote the position of a vehicle to distinguish it from the state  $x = (p, v)$ , which also includes the velocity, and we fix position  $p = 0$  at the stop-line of intersection  $w$ . Let the length and width of a vehicle  $i$  be denoted by  $L_i$  and  $W_i$ , respectively. We measure the position of a vehicle at the front bumper. We will make the following assumption, that allow us to easily derive explicit expressions of these trajectories.

**Assumption 1.1.** *All vehicles have the same length  $L_i = L$  and width  $W_i = W$ . Lanes are exactly  $W$  units wide and are axis-aligned, such that intersection are squares.*

**Assumption 1.2.** *Vehicles must drive at full speed when entering an intersection and keep driving at full speed as long as they occupy an intersection.*

Now assume that some vehicle is scheduled to exit  $v$  at time  $t_0$  and to enter  $w$  at some time  $t_f$ . Let  $p_0 = -d(v, w) + W$  denote the position of the vehicle when it starts to exit  $v$ . Let  $y(t)$  denote the trajectory of the vehicle that drives in front of the current one, assuming it exists. In order to keep the vehicle as close to  $w$  as possible at every time, while respecting the double integrator vehicle dynamics  $\dot{p} = v$ ,  $\ddot{p} = u$ , we can generate a trajectory by solving the optimal control problem

$$\begin{aligned} \max_u \quad & \int_{t_0}^{t_f} p(t) dt \\ \text{s.t.} \quad & 0 \leq v(t) \leq v_{\max}, \\ & -a_{\max} \leq u(t) \leq a_{\max}, \\ & y(t) \leq p(t), \\ & p(t_0) = p_0, \quad p(t_f) = 0, \\ & v(t_0) = v_{\max}, \quad v(t_f) = v_{\max}. \end{aligned} \tag{1}$$

This problem can be solved by using a direct transcription method. After observing some example solutions, we believe that the optimal control should switch between no acceleration and either full acceleration or full deceleration, i.e., we have a control function  $u(t) := \ddot{x}(t)$  satisfies  $u(t) \in \{-a_{\max}, 0, a_{\max}\}$  and some sequence of alternating deceleration and acceleration periods, represented by some sequence of disjoint intervals

$$(D_0, A_1, D_1, \dots, A_{n-1}, D_{n-1}, A_n),$$

so that the optimal controller is given by

$$u(t) = \begin{cases} -a_{\max} & \text{if } t \in D_k \text{ for some } k, \\ a_{\max} & \text{if } t \in A_k \text{ for some } k, \\ 0 & \text{otherwise.} \end{cases}$$

## 1.1 Single vehicle

We can proof the above hypothesis and provide explicit expressions for  $D_k$  and  $A_k$  of the optimal trajectory whenever the safe following constraint  $y(t) \leq p(t)$  is never active and can essentially be ignored, either because there is no vehicle ahead or it is sufficiently far away. The proof uses a variant of the Pontryagin Maximum Principle, specifically tailored to problems with state constraints [1].

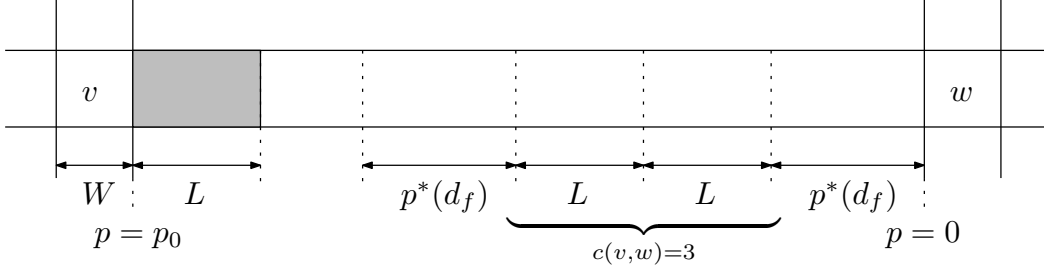


Figure 2: Tandem of intersections with distances used in the determination of the lane capacity and the waiting positions.

## 1.2 Multiple vehicles

By observing optimal trajectories generated using direct transcription, we observe that there is a lot of structure in how the trajectory of a vehicle influences the trajectories of later vehicles. In this section, we provide a direct way to explicitly calculate trajectories, without proving that these trajectories are always feasible and optimal. Instead, we verify correctness of our method numerically by comparing the output to direct transcription for various system parameters and crossing times.

We start by investigating the limit on the number of vehicles that can occupy the lane at the same time. Imagine that vehicles enter the lane until it is full and then only after all vehicles have come to a full stop, they start to leave the lane by crossing  $w$ . Without any additional constraints, it follows from the vehicle dynamics that the time it takes to fully accelerate from rest to maximum velocity is given by  $d_f = v_{\max}/a_{\max}$  and with corresponding trajectory

$$p^*(t) = a_{\max}t^2/2 \quad \text{for } 0 \leq t \leq d_f.$$

Because we assume that maximum deceleration is also  $a_{\max}$ , it also takes  $d_f$  time to fully decelerate from maximum velocity to rest. Suppose that we want to design the tandem network such that at least  $c(v, w)$  vehicles can enter and decelerate to some waiting position, from which it is also possible to accelerate again to full speed before crossing  $w$ . Vehicles are required to drive at full speed as long as they occupy any intersection. Therefore, a vehicle crossing  $v$  can only start decelerating after  $p(t) \geq W + L$ , so the earliest position where a vehicle can come to a stop is  $p = W + L + p^*(d_f)$ . Because vehicles need to gain maximum speed before reaching  $w$ , the position closest to  $w$  where a vehicle can wait is  $-p^*(d_f)$ . Hence, in order to accomodate for  $c(v, w)$  waiting vehicles, the length of the lane must satisfy

$$d(v, w) \geq W + L + 2p^*(d_f) + (c(v, w) - 1)L,$$

as illustrated in Figure 2. Conversely, given the lane length  $d(v, w)$ , the corresponding lane capacity is given by

$$c(v, w) = \text{floor} \left( \frac{d(v, w) - W - 2p^*(d_f)}{L} \right),$$

where  $\text{floor}(x)$  denotes the largest integer smaller than or equal to  $x$ .

It turns out that the fixed locations where vehicles wait in the above scenario are helpful in describing the optimal trajectories, even when vehicles never fully stop. We will denote these fixed *waiting positions* as

$$p_k = -p^*(d_f) - (c(v, w) - k)L,$$

for  $k = 1, \dots, c(v, w)$ . Furthermore, let  $p_d = p_1 - p^*(d_f)$  denote the position from which vehicles must decelerate in order to stop at the earliest waiting position  $p_1$ , which is the farthest from the destination intersection. Now consider a vehicle that moves from  $p_k$  to the next waiting position  $p_{k+1}$ , so it moves exactly distance  $L$ . We consider such a start-stop movement, without considering any safe following constraints. By symmetry of the control constraints, the vehicle moves the same distance during both acceleration and deceleration. Furthermore, the vehicle needs to be at rest at the start and end of such trajectory. Hence, it is clear that it takes the same amount of time  $d_s$  to accelerate and decelerate. We assume that  $d_s < d_f$ , which ensures that  $v_{\max}$  is never reached during the start-stop movement, which is illustrated in Figure 3. In this case, it is clear that we must have  $L = 2p^*(d_s)$ , from which we derive that  $d_s = \sqrt{L/a_{\max}}$ .

### 1.2.1 Ad-hoc approach

We will now present a method to calculate the trajectory of a vehicle based on its crossing times at  $v$  and  $w$  and the trajectory of the vehicle ahead of it. We start from a sequence of deceleration and acceleration intervals that are possibly overlapping and then merge them in pairs from left to right until they become disjoint. Specifically, let

$$t_d := t_0 + \frac{d(v, w) - W - p_d}{v_{\max}}$$

be the start of the initial deceleration interval  $D_0 := [t_d, t_d + d_f]$ , which is exactly the moment the vehicle needs to start decelerating in order to stop at the first waiting position  $p_1$ , so at time  $t_d$  the vehicle is at position  $p_d$ . Similarly, let  $t_a = t_f - d_f$  be the start of the final acceleration interval  $A_f := [t_a, t_a + d_f]$ . Now for every  $k = 1, \dots, c(v, w) - 1$ , we consider a pair of start-stop intervals  $S_k := (A_k, D_k) = ([t_k, t_k + d_s], [t_k + d_s, t_k + 2d_s])$  at some starting time  $t_k$ , moving the vehicle from  $p_k$  to  $p_{k+1}$ . We first show how to merge these intervals such that a sequence of disjoint intervals is obtained. Afterwards, we show how times  $t_k$  follow from the trajectory of the vehicle ahead.

First, we show how to merge  $D_0$  and  $S_1$ . When we have  $t_d + d_f \leq t_1$ , it is clear that both parts are already completely disjoint, so we do not have to do anything further. As  $t_d$  and  $t_1$  get closer than  $d_f$ , we need to start merging  $D_0$  and  $A_1$  as we will show. In this case, we observe that  $D_0$  gets shorter at the end by some  $\epsilon$  and the acceleration part of  $S_1$  gets shorter at the beginning, also by the same amount  $\epsilon$ , because the velocities must match. More precisely, we construct two new consecutive intervals

$$\begin{aligned} D'_0 &= [t_1 + 2\epsilon - d_f, t_1 + \epsilon], \\ A'_1 &= [t_1 + \epsilon, t_1 + d_s], \\ D_1 &= [t_1 + d_s, t_1 + 2d_s]. \end{aligned}$$

We now determine  $\epsilon$  as a function of  $t_1 - t_d$ . Because  $D'_0$  and  $A'_1$  are both  $\epsilon$  shorter, the total distance that is traversed is now  $2p^*(\epsilon)$  shorter. This means that the start of  $D'_0$  should be  $2p^*(\epsilon)/v_{\max}$  later than  $t_d$ , which gives equation

$$t_d + \frac{2p^*(\epsilon)}{v_{\max}} = t_1 + 2\epsilon - d_f,$$

which is quadratic in  $\epsilon$ . Because we must have  $\epsilon < d_f$ , we need the smallest solution

$$\epsilon = d_f - \sqrt{d_f(t_1 - t_d)}.$$

When we keep increasing  $t_d$  while keeping  $t_1$  fixed, we observe that eventually part  $A_1$  will completely disappear and  $D_0$  and  $D_1$  will become a single interval, which is easily seen to happen when  $\epsilon \geq d_s$ . Equivalently, this happens when  $t_d$  is such that  $t_d \geq t_1 - d_f + 2d_s - 2p^*(d_s)/v_{\max}$ .

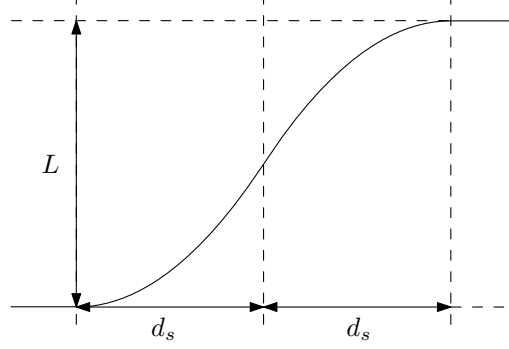


Figure 3: Shape of some start-stop trajectory of a single isolate vehicle moving forward from some current waiting position  $p_k$  to the next  $p_{k+1}$ .

We now show how two start-stop parts have to be merged if they overlap. Consider two start-stop parts  $A_k, D_k$  and  $A_{k+1}, D_{k+1}$  and suppose that  $t_k + 2d_s < t_{k+1}$  such that both parts have to be merged. The parts  $D_k$  and  $A_{k+1}$  merge by removing  $\epsilon$  on both sides, similarly as above. However, this causes the  $D_k, A_{k+1}, D_{k+1}$  parts to shift up. Therefore,  $A_k$  and  $D_k$  each need to be lengthened at the side where they meet by some  $\delta$  to match this. Hence, it turns out we have to use the intervals

$$\begin{aligned} A'_k &= [t_k, t_k + d_s + \delta], \\ D'_k &= [t_k + d_s + \delta, t_{k+1} + \epsilon], \\ A'_{k+1} &= [t_{k+1} + \epsilon, t_{k+1} + d_s], \\ D_{k+1} &= [t_{k+1}, t_{k+1} + d_s]. \end{aligned}$$

We have that  $\epsilon$  and  $\delta$  need to satisfy

$$\begin{cases} 2\delta + 2\epsilon = t_{k+1} - t_k, \\ 2p^*(d_s + \delta) - 2p^*(d_s - \epsilon) = L. \end{cases}$$

Solving this system of equations yields

$$\begin{aligned} \delta &= \frac{L/a_{\max}}{t_{k+1} - t_k} - d_s + \frac{t_{k+1} - t_k}{4}, \\ \epsilon &= (t_{k+1} - t_k)/2 - \delta. \end{aligned}$$

Finally, we consider the merge with the final acceleration bang.

Note that the above types of merging are enough to process the whole sequence of bangs, because when the first merge of  $D_0$  and  $A_1, D_1$  results in a single  $D'_1$  and there is a next  $A_2, D_2$ , we are again in the first situation.

We now show how  $t_k$  follow from the trajectory of the preceding vehicle.

### 1.2.2 Schedule time approach

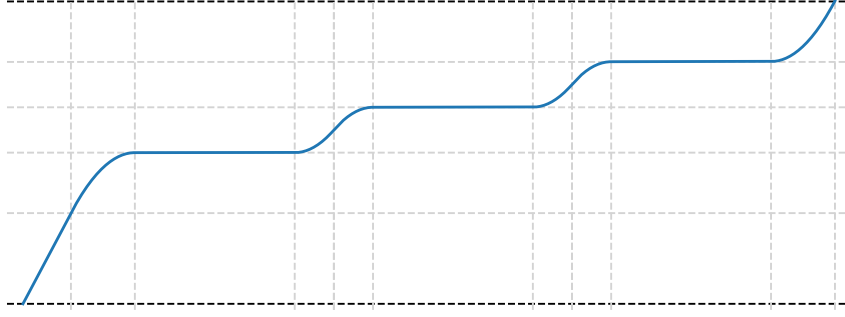


Figure 4: Sketch of vehicle trajectory in tandem with all the full start-stop parts unmerged.

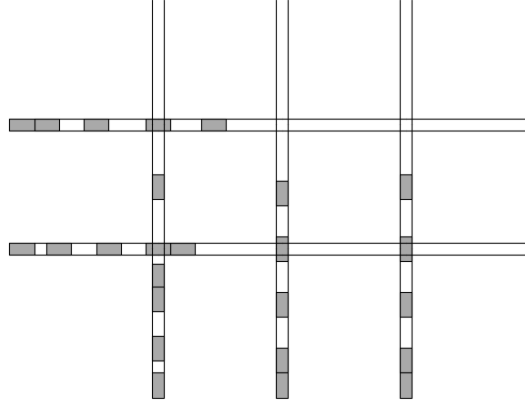


Figure 5: Illustration of some grid-like network of intersections with vehicles drawn as grey rectangles. There are five vehicle routes: two from east to west and three from south to north. Turning at intersections is not allowed.

## 2 Trajectories in networks

We now extend the single intersection model to a network of intersections without turning routes, illustrated in Figure 5. We define a directed graph  $(V, E)$  with nodes  $V$  and arcs  $E$ , representing the possible paths that vehicles can follow. Nodes of in-degree at least two are called *intersections*. Nodes with only outgoing arcs are *entrypoints* and nodes with only incoming arcs are *exitpoints*. Let  $d(v, w)$  denote the distance between nodes  $v$  and  $w$ . For each route index  $r \in \mathcal{R}$ , we let

$$\bar{V}_r = (v_r(0), v_r(1), \dots, v_r(m_r), v_r(m_{r+1}))$$

be the path that vehicles  $i \in \mathcal{N}_r$  follow through the network. We require that the first node  $v_r(0)$  is an entrypoint and that the last node  $v_r(m_{r+1})$  is an exitpoint and we write

$$V_r = \bar{V}_r \setminus \{v_r(0), v_r(m_{r+1})\}$$

to denote the path restricted to intersections. We say that some  $(v, w) \in E$  is on path  $V_r$  whenever  $v$  and  $w$  are two consecutive nodes on the path and we write  $E_r$  to denote the set of all these edges. We require that routes can only overlap at nodes by making the following assumption.

**Assumption 2.1.** *Every arc  $(v, w) \in E$  is part of at most one route  $V_r$ .*

We start by considering networks in which all roads are axis-aligned such that intersections always involve perpendicular lanes and where routes are such that no turning is required. For each  $v \in V_r$  define the conflict zone  $\mathcal{E}_r(v) = (b_r(v), e_r(v))$  and consider the union

$$\mathcal{E}_r = \bigcup_{v \in V_r} \mathcal{E}_r(v)$$

corresponding to the positions of vehicles  $i \in \mathcal{N}_r$  for which it occupies an intersection on its path  $V_r$ . By reading  $\mathcal{E}_i \equiv \mathcal{E}_r$  for  $r(i) = r$ , the single intersection problem naturally extends to the network case. Like before, the resulting problem can be numerically solved by a direct transcription method.

## 2.1 General decomposition

The general two-stage decomposition for the single intersection extends rather naturally to the present model. Let for each pair  $(i, v)$  of some vehicle  $i \in \mathcal{N}$  and an intersection  $v \in V_{r(i)}$  along its route, let

$$\inf\{t : x_i(t) \in \mathcal{E}_r(v)\} \quad \text{and} \quad \sup\{t : x_i(t) \in \mathcal{E}_r(v)\}$$

be the crossing time and exit time, which we denote by  $y(i, v)$  and  $y(i, v) + \sigma(i, v)$ , respectively. Instead of a single set of conflicts, we now define for each intersection  $v \in V$  in the network the set of conflict pairs

$$\mathcal{D}^v = \{\{i, j\} \subset \mathcal{N} : r(i) \neq r(j), v \in V_{r(i)} \cap V_{r(j)}\}.$$

Now the two-stage approach is to solve

$$\begin{aligned} \min_{y, \sigma} \quad & \sum_{r \in \mathcal{R}} F(y_r, \sigma_r) \\ \text{s.t.} \quad & y(i, v) + \sigma(i, v) \leq y(j, v) \text{ or} \\ & y(j, v) + \sigma(j, v) \leq y(i, v), \quad \text{for all } \{i, j\} \in \mathcal{D}^v \text{ and } v \in V, \\ & (y_r, \sigma_r) \in \mathcal{S}_r, \quad \text{for all } r \in \mathcal{R}, \end{aligned}$$

where  $F(y_r, \sigma_r)$  and  $\mathcal{S}_r$  are the value function and set of feasible parameters, respectively, of the parametric trajectory optimization problems

$$\begin{aligned} F(y_r, \sigma_r) = \min_{x_r} \quad & \sum_{r \in \mathcal{R}} J(x_i) \\ \text{s.t.} \quad & x_i(t) \in D_i(s_{i,0}), \quad \text{for } i \in \mathcal{N}_r, \\ & x_i(y(i, v)) = b_r(v), \quad \text{for } v \in V_r, i \in \mathcal{N}_r, \\ & x_i(y(i, v) + \sigma(i, v)) = e_r(v), \quad \text{for } v \in V_r, i \in \mathcal{N}_r, \\ & x_i(t) - x_j(t) \geq L, \quad \text{for } (i, j) \in \mathcal{C} \cap \mathcal{N}_r, \end{aligned}$$

where we again use subscript  $r$  to group variables according to their associated route.

## 2.2 Decomposition for delay objective

Suppose we use the crossing at the last intersection as performance measure, by defining the objective function as

$$J(x_i) = \inf\{t : x_i(t) \in \mathcal{E}_r(v_r(m_r))\}.$$

We show how to reduce the resulting problem to a scheduling problem, like we did in the single intersection case. It is not clear whether vehicles will always cross intersections at full

speed, but we will simply require vehicles to do so from here on. Furthermore, we will again assume that all vehicles share the same geometry. Hence, the occupation time  $\sigma \equiv \sigma(i, v)$  is the same for all vehicles and intersections. For this reason, we will write the shorthand  $y_r \in \mathcal{S}_r$ , because  $\sigma_r$  is no longer a free variable.

As a consequence of Assumption 1.1 and Assumption 1.2, each lower-level trajectory optimization problem for a given route  $r \in \mathcal{R}$  decomposes into a sequence of problems, each corresponding to two consecutive intersection along  $V_r$ . This means that  $y_r \in \mathcal{S}_r$  is equivalent to  $y_{(v,w)} \in \mathcal{S}_{(v,w)}$  for each  $(v, w) \in E_r$ , where  $y_{(v,w)}$  denotes the vector of all variables  $y(i, v)$  and  $y(i, w)$  for all  $i \in \mathcal{N}_r$  and  $\mathcal{S}_{(v,w)}$  denotes the set of values of  $y_{(v,w)}$  for which a feasible trajectory part can be found. Hence, we will now focus on a tandem of two intersections and investigate the trajectories of vehicles in this with the goal of stating sufficient conditions for  $y_{(v,w)} \in \mathcal{S}_{(v,w)}$ .

## 2.3 Crossing time scheduling

Introduce general job-shop problem and disjunctive graph. Introduce travel constraints and explain corresponding new type of arc in disjunctive graph. Introduce buffer constraints and corresponding new arc in disjunctive graph.

## 2.4 Branch-and-bound

Formulate MILP problem and investigate how the solving time scales with network size in terms of number of intersections and number of vehicles in the network. Does the single intersection cutting planes still hold? Are there any obvious cutting planes?

# 3 Constructive heuristic

Like for problems with a single intersection, we want to investigate possible constructive heuristics. In this case, the step-by-step construction can again be understood in terms of transitioning from a partial disjunctive graph to the next partial disjunctive graph by fixing the direction of some disjunctive arc. Instead of only the route index, we now also need to specify the intersection to identify a unique disjunctive arc.

We solve the instance to optimality with an exact method. Next, we use the resulting crossing time schedule to compute actions for the automaton that lead to the same schedule. However, this sequence of actions is not unique: the order in which intersections are considered does not matter for the final schedule. Therefore, we sample some intersection order and replay the sequence of actions on the automaton to generate the corresponding sequence of state-action pairs. At this point, we copy the whole disjunctive graph for each state. Alternatively, we could use some sort of masking for non-final states.

We now have the following classification task: map disjunctive graph to an action (route-intersection pair). We use a GIN to compute an embedding for each node, which is fed through an MLP and softmax to produce a probability over nodes. In Zhang et al., each action corresponds to a unique node, encoding the operations that is dispatched next. However, we only really need to provide a route-intersection pair, but how to exploit this in the policy model?

The GNN computes node embeddings, which are mapped to a score for each node. We compute the softmax over the scores of the nodes and then compute the negative log likelihood loss for backpropagation.

The model can be evaluated in two ways: we can measure how well the model fits unseen expert demonstration state-action pairs or we can measure the schedule quality when completely executing the learning policy.



## References

- [1] R. F. Hartl, S. P. Sethi, and R. G. Vickson, “A Survey of the Maximum Principles for Optimal Control Problems with State Constraints,” *SIAM Review*, vol. 37, no. 2, pp. 181–218, 1995.