Offline Trajectory Optimization of Autonomous Vehicles in a Single Intersection

Jeroen van Riel

January 2025

Contents

1 Problem analysis

This document considers the offline trajectory optimization problem for a single intersection. Recall that *offline* meant that all future arrivals to the system are known beforehand and that we assume that routes are fixed to avoid having to address some kind of dynamic routing problem. In this case, we can consider the longitudinal position $x_i(t)$ of each vehicle i along its route, for which we use the well-known *double integrator* model

$$\begin{aligned} \dot{x}_i(t) &= v_i(t), \\ \dot{v}_i(t) &= u_i(t), \\ 0 \leq v_{\max} &\leq v_{\max}, \\ |u_i(t)| &\leq a_{\max}, \end{aligned} \tag{1}$$

where $v_i(t)$ is the vehicle's velocity and $u_i(t)$ its acceleration, which is set by a single central controller. Let $D_i(s_{i,0})$ denote the set of all trajectories $x_i(t)$ satisfying these dynamics, given some initial state $s_{i,0} = (x_i(0), v_i(0))$.

Consider the single intersection illustrated in Figure 1. Assume there are two incoming lanes, identified by indices $\mathcal{R} = \{1, 2\}$. The corresponding two routes are crossing

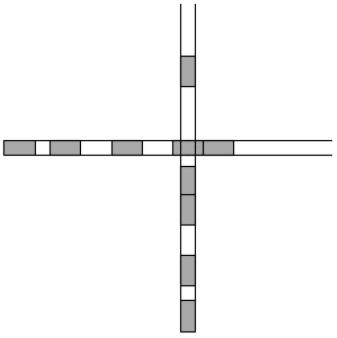

Figure 1: Illustration of a single intersection with vehicles drawn as grey rectangles. Vehicles approach the intersection from the east and from the south and cross it without turning. Note that the first two waiting vehicles on the south lane kept some distance before the intersection, such that they are able to reach full speed whenever they cross.

the intersection from south to north and crossing from west to east. We identify vehicles by their route and by their relative order on this route, by defining the vehicle index set

$$\mathcal{N} = \{(r,k) : k \in \{1,\dots,n_r\}, r \in \mathcal{R}\}, \tag{2}$$

where n_r denotes the number of vehicles following route r. Smaller values of k correspond to reaching the intersection earlier. Given vehicle index $i = (r,k) \in \mathcal{N}$, we also use the notation $r(i) = r$ and $k(i) = k$. We assume that each vehicle is represented as a rectangle of length L and width W and that its position $x_i(t)$ is measured as the distance between its front bumper and the start of the lane. In order to maintain a safe distance between consecutive vehicle on the same lane, vehicle trajectories need to satisfy

$$x_i(t) - x_j(t) \geq L, \tag{3}$$

for all t and all pairs of indices $i,j \in \mathcal{N}$ such that $r(i) = r(j), k(i)+1 = k(j)$. Let $\mathcal{C}$ denote the set of such ordered pairs of indices. Note that these constraints restrict vehicle from overtaking each other, so the initial relative order is always maintained. For each $i \in \mathcal{N}$, let $\mathcal{E}_i = (B_i, E_i)$ denote the open interval such that vehicle i occupies the intersection's conflict area if and only if $B_i < x_i(t) < E_i$. Using this notation, collision avoidance at the intersection is achieved by requiring

$$(x_i(t), x_j(t)) \notin \mathcal{E}_i \times \mathcal{E}_j, \tag{4}$$

for all t and for all pairs of indices $i,j \in \mathcal{N}$ with $r(i) \neq r(j)$, which we collect in the set $\mathcal{D}$. Suppose we have some performance criterion $J(x_i)$ that takes into account travel time and energy efficiency of the trajectory of vehicle i, then the offline trajectory optimization problem for a single intersection can be compactly written as

$$\begin{aligned}
\min_{\mathbf{x}(t)} \quad & \sum_{i\in\mathcal{N}} J(x_i) && && \text{(5a)}\\
\text{s.t.} \quad & x_i \in D_i(s_{i,0}), && \text{for all } i \in \mathcal{N}, && \text{(5b)}\\
& x_i(t) - x_j(t) \geq L, && \text{for all } (i,j) \in \mathcal{C}, && \text{(5c)}\\
& (x_i(t), x_j(t)) \notin \mathcal{E}_i \times \mathcal{E}_j, && \text{for all } \{i,j\} \in \mathcal{D}, && \text{(5d)}
\end{aligned}$$

where $\mathbf{x}(t) = [\, x_i(t) : i \in \mathcal{N}\,]$ and constraints are for all t.

1.1 Direct transcription

Although computationally demanding, problem (5) can be numerically solved by direct transcription to a non-convex mixed-integer linear program by discretization on a uniform time grid. Let K denote the number of discrete time steps and let Δt denote the time step size. Using the forward Euler integration scheme, we have

$$\begin{aligned} x_i(t + \Delta t) &= x_i(t) + v_i(t)\Delta t, \\ v_i(t + \Delta t) &= v_i(t) + u_i(t)\Delta t, \end{aligned}$$

for each $t \in \{0, \Delta t, \ldots, K\Delta t\}$. Following the approach in [1], the collision-avoidance constraints between lanes can be formulated using the well-known big-M technique by the constraints

$$\begin{aligned} x_i(t) &\leq B_i + \delta_i(t)M, \\ E_i - \gamma_i(t)M &\leq x_i(t), \\ \delta_i(t) + \delta_j(t) + \gamma_i(t) + \gamma_j(t) &\leq 3, \end{aligned}$$

where $\delta_i(t), \gamma_i(t) \in \{0, 1\}$ for all $i \in \mathcal{N}$ and M is a sufficiently large number. Finally, the follow constraints can simply be added as

$$x_i(t) - x_j(t) \geq L,$$

for each $t \in \{0, \Delta t, \ldots, K\Delta t\}$ and each pair of consecutive vehicles $(i, j) \in \mathcal{C}$ on the same lane. For example, consider the objective functional

$$J(x_i) = \int_{t=0}^{t_f} \left((v_d - v_i(t))^2 + u_i(t)^2 \right) dt,$$

where v_d is some reference velocity and t_f denotes the final time, then the optimal trajectories are shown in Figure 2.

i	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)
$x_i(0)$	15	10	0	10	0
$v_i(0)$	10	10	10	10	10

Table 1: Example initial conditions $s_{i,0} = (x_i(0), v_i(0))$ for problem (5).

1.2 General decomposition

For the case where only a single vehicle is approaching the intersection for each route, so $n_r = 1$ for each route $r \in \mathcal{R}$, it has been shown that problem (5) can be decomposed into two coupled optimization problems, see Theorem 1 in [1]. Roughly speaking, the *upper-level problem* optimizes the time slots during which vehicles occupy the intersection, while the *lower-level problem* produces optimal safe trajectories that respect these time slots. When allowing multiple vehicles per lane, we show without proof that a similar decomposition is possible. Given $x_i(t)$, the *crossing time* of vehicle i, when the vehicle first enters the intersection, and the corresponding *exit time* are respectively

$$\inf\{t : x_i(t) \in \mathcal{E}_i\} \quad \text{and} \quad \sup\{t : x_i(t) \in \mathcal{E}_i\}. \tag{6}$$

The upper-level problem is to find a set of feasible occupancy timeslots, for which the lower-level problem generates trajectories. We will use decision variable $y(i)$ for the

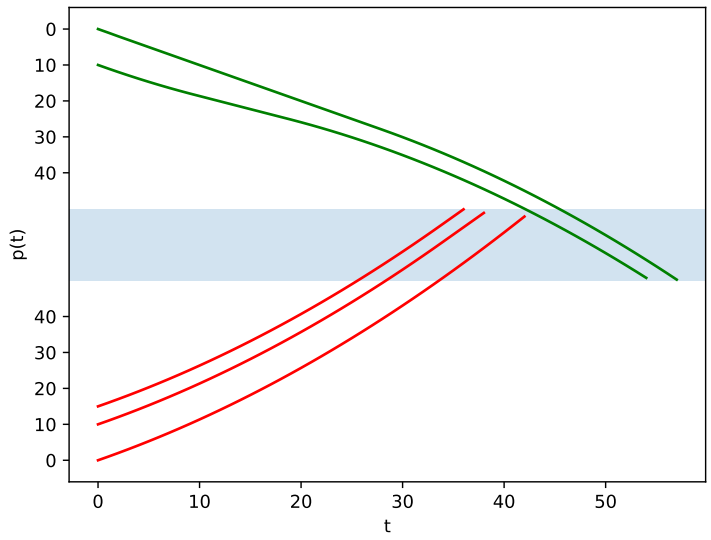

Figure 2: Example of optimal trajectories obtained using the direct transcription method with $L = 5$, $\mathcal{E}_i \equiv \mathcal{E} = [50, 70]$, $v_d = 20$, $T = 120$, $\Delta t = 0.1$ and initial conditions as given in Table 1. The y-axis is split such that each part corresponds to one of the two lanes and the trajectories are inverted accordingly and drawn with separate colors. The intersection area $\mathcal{E}$ is drawn as a shaded region. Whenever a vehicle has left the intersection, we stop drawing its trajectory for clarity.

crossing time and write $y(i) + \sigma(i)$ for the exit time. It turns out that trajectories can be generated separately for each route, which yields the decomposition

$$\begin{aligned}
\min_{y,\sigma} \quad & \sum_{r \in \mathcal{R}} F(y_r, \sigma_r) && && \text{(7a)}\\
\text{s.t.} \quad & y(i) + \sigma(i) \le y(j) \text{ or } y(j) + \sigma(j) \le y(i), && \text{for all } (i,j) \in \mathcal{D}, && \text{(7b)}\\
& (y_r, \sigma_r) \in \mathcal{S}_r, && \text{for all } r \in \mathcal{R}, && \text{(7c)}
\end{aligned}$$

where $F(y_r, \sigma_r)$ and $\mathcal{S}_r$ are the value function and set of feasible parameters, respectively, of the lower-level *route trajectory optimization* problem

$$\begin{aligned}
F(y_r, \sigma_r) = \min_{x_r} \quad & \sum_{i \in \mathcal{N}(r)} J(x_i) && && \text{(8a)}\\
\text{s.t.} \quad & x_i \in D_i(s_{i,0}), && \text{for all } i \in \mathcal{N}_r, && \text{(8b)}\\
& x_i(y(i)) = B_i, && \text{for all } i \in \mathcal{N}_r, && \text{(8c)}\\
& x_i(y(i) + \sigma(i)) = E_i, && \text{for all } i \in \mathcal{N}_r, && \text{(8d)}\\
& x_i(t) - x_j(t) \ge L, && \text{for all } (i,j) \in \mathcal{C} \cap \mathcal{N}_r, && \text{(8e)}
\end{aligned}$$

where we used $\mathcal{N}_r = \{i \in \mathcal{N} : r(i) = r\}$ and similarly for x_r, y_r and σ_r to group variables according to route. Note that the set of feasible parameters $\mathcal{S}_r$ implicitly depends on the initial states s_r and system parameters.

1.3 Decomposition under delay objective

Assume that the trajectory performance criterion is exactly the crossing time, so $J(x_i) = \inf\{t : x_i(t) \in \mathcal{E}_i\}$. This assumption makes the problem significantly eas-

ier, because we have

$$F(y_r, \sigma_r) \equiv F(y_r) = \sum_{i \in \mathcal{N}_r} y(i). \tag{9}$$

Furthermore, we assume that vehicles enter the network and cross the intersection at full speed, so $v_i(0) = v_i(y(i)) = v_{\max}$, such that we have

$$\sigma(i) \equiv \sigma = (L + W)/v_{\max}, \quad \text{for all } i \in \mathcal{N}. \tag{10}$$

Therefore, we ignore the part related to σ in the set of feasible parameters $\mathcal{S}_r$, which can be shown that to have a particularly simple structure under these assumptions. Observe that $r_i = (B_i - x_i(0))/v_{\max}$ is the earliest time at which vehicle i can enter the intersection. Let $\rho = L/v_{\max}$ be such that $y(i) + \rho$ is the time at which the rear bumper of a crossing vehicle reaches the start line of the intersection, then it can be shown that $y_r \in \mathcal{S}_r$ whenever

$$r_i \leq y(i), \text{ for all } i \in \mathcal{N}_r, \tag{11a}$$
$$y(i) + \rho \leq y(j), \text{ for all } (i,j) \in \mathcal{C} \cap \mathcal{N}_r. \tag{11b}$$

Therefore, under the stated assumptions, problem (5) reduces to the following *crossing time scheduling* problem

$$\min_{y} \quad \sum_{i \in \mathcal{N}} y(i) \tag{12a}$$
$$\text{s.t.} \quad r_i \leq y(i), \qquad \text{for all } i \in \mathcal{N}, \tag{12b}$$
$$y(i) + \rho \leq y(j), \qquad \text{for all } (i,j) \in \mathcal{C}, \tag{12c}$$
$$y(i) + \sigma \leq y(j) \text{ or } y(j) + \sigma \leq y(i), \qquad \text{for all } (i,j) \in \mathcal{D}, \tag{12d}$$

which can be solved using off-the-shelf mixed-integer linear program solvers, after encoding the *disjunctive constraints* (12d) using the big-M technique, which we will demonstrate in Section 2.1. Given optimal crossing time schedule y^*, any set of trajectories $[x_i(t) : i \in \mathcal{N}]$ that satisfies

$$x_i \in D_i(s_{i,0}), \quad \text{for all } i \in \mathcal{N}, \tag{13a}$$
$$x_i(y^*(i)) = B_i, \quad \text{for all } i \in \mathcal{N}, \tag{13b}$$
$$x_i(y^*(i) + \sigma) = E_i, \quad \text{for all } i \in \mathcal{N}, \tag{13c}$$
$$x_i(t) - x_j(t) \geq L, \quad \text{for all } (i,j) \in \mathcal{C}, \tag{13d}$$

forms a valid solution. These trajectories can be computed with an efficient direct transcription method. First of all, note that each route may be considered separately. Therefore, trajectories can be computed in a sequential fashion by repeatedly solving the optimal control problem

$$\begin{aligned}
&\texttt{MotionSynthesize}(\tau, B, s_0, x') := \\
&\qquad \arg\min_{x:[0,\tau]\to\mathbb{R}} \int_0^\tau |x(t)|dt \\
&\qquad\qquad \text{s.t. } \ddot{x}(t) = u(t), && \text{for all } t \in [0,\tau], \\
&\qquad\qquad\quad |u(t)| \leq a_{\max}, && \text{for all } t \in [0,\tau], \\
&\qquad\qquad\quad 0 \leq \dot{x}(t) \leq v_{\max}, && \text{for all } t \in [0,\tau], \\
&\qquad\qquad\quad x'(t) - x(t) \geq L, && \text{for all } t \in [0,\tau], \\
&\qquad\qquad\quad (x(0), \dot{x}(0)) = s_0, \\
&\qquad\qquad\quad (x(\tau), \dot{x}(\tau)) = (B, v_{\max}),
\end{aligned}$$

where τ is set to the required crossing time, B denotes the distance to the intersection, s_0 is the initial state of the vehicle and x' denotes the trajectory of the vehicle preceding the current vehicle.

2 Crossing time scheduling

Given a crossing time schedule y, trajectories can be efficiently computed using a direct transcription method. Hence, we focus on solving the crossing time scheduling problem 12. Before we start discussing various solution techniques, let us first introduce an alternative way of representing instances of 12 by means of a graph. Once we extend the current model to networks of intersection, this encoding will be particularly helpful.

Instances and solutions of the crossing time optimization problem (12) can be represented by their *disjunctive graph*: let $(\mathcal{N}, \mathcal{C}, \mathcal{O})$ be a directed graph with nodes $\mathcal{N}$ and the following two types of arcs. The *conjunctive arcs* encode the fixed order of vehicles driving on the same lane. For each $(i, j) \in \mathcal{C}$, an arc from i to j means that vehicle i reaches the intersection before j due to the follow constraints (12c). The *disjunctive arcs* are used to encode the decisions regarding the ordering of vehicles from distinct lanes, corresponding to constraints (12d). For each pair $\{i, j\} \in \mathcal{D}$, at most one of the arcs (i, j) and (j, i) can be present in $\mathcal{O}$.

When $\mathcal{O} = \varnothing$, we say the disjunctive graph is *empty*. Each feasible schedule satisfies exactly one of the two constraints in (12d). When $\mathcal{O}$ contains exactly one arc from every pair of opposite disjunctive arcs, we say the disjunctive graph is *complete*. Note that such graph is acyclic and induces a unique topological ordering π of its nodes. Conversely, every ordering π of nodes $\mathcal{N}$ corresponds to a unique complete disjunctive graph, which we denote by $G(\pi) = (\mathcal{N}, \mathcal{C}, \mathcal{O}(\pi))$.

We define weights for every possible arc in a disjunctive graph. Every conjunctive arc $(i, j) \in \mathcal{C}$ gets weight $w(i, j) = \rho_i$ and every disjunctive arc $(i, j) \in \mathcal{O}$ gets weight $w(i, j) = \sigma_i$. Given some vehicle ordering π, for every $j \in \mathcal{N}$, we recursively define the lower bound

$$\mathrm{LB}_\pi(j) = \max\{r_j, \max_{i \in N_\pi^-(j)} \mathrm{LB}_\pi(i) + w(i, j)\}, \tag{14}$$

where $N_\pi^-(j)$ denotes the set of in-neighbors of node j in $G(\pi)$. Observe that this quantity is a lower bound on the crossing time, i.e., every feasible schedule y with ordering π must satisfy $y_i \geq \mathrm{LB}_\pi(i)$ for all $i \in \mathcal{N}$. As the following result shows, it turns out that this lower bound is actually tight for optimal schedules, which allows us to calculate the optimal crossing times y^* once we know an optimal ordering π^* of vehicles, so we can concentrate on finding the latter.

Proposition 1. *If y is an optimal schedule for (12) with ordering π, then*

$$y_i = \mathrm{LB}_\pi(i) \quad \text{for all } i \in \mathcal{N}. \tag{15}$$

Under the condition that $\rho_i = \rho$ and $\sigma_i = \sigma > \rho$ for all $i \in \mathcal{N}$, it turns out that some properties of an optimal ordering can be immediately computed from the problem specification.

Proposition 2. *Consider an instance of (12) with $\rho_i = \rho$ and $\sigma_i = \sigma > \rho$ for all $i \in \mathcal{N}$. Suppose y is an optimal schedule with $y_{i^*} + \rho \geq r_{j^*}$, for some $(i^*, j^*) \in \mathcal{C}$, then j^* follows immediately after i^*, so $y_{i^*} + \rho = y_{j^*}$.*

2.1 Branch-and-cut

Optimization problem 12 can be turned into a Mixed-Integer Linear Program (MILP) by rewriting the disjunctive constraints using the well-known big-M method. We introduce a binary decision variable γ_{ij} for every disjunctive pair $\{i,j\} \in \mathcal{D}$. To avoid redundant variables, we first impose some arbitrary ordering of the disjunctive pairs by defining

$$\bar{\mathcal{D}} = \{(i,j) : \{i,j\} \in \mathcal{D},\ l(i) < l(j)\},$$

such that for every $(i,j) \in \bar{\mathcal{D}}$, setting $\gamma_{ij} = 0$ corresponds to choosing disjunctive arc $i \to j$ and $\gamma_{ij} = 1$ corresponds to $j \to i$. This yields the following MILP formulation

$$\begin{aligned}
\min_{y} \quad & \sum_{i \in \mathcal{N}} y_i \\
\text{s.t.} \quad & r_i \le y_i && \text{for all } i \in \mathcal{N}, \\
& y_i + \rho_i \le y_j && \text{for all } (i,j) \in \mathcal{C}, \\
& y_i + \sigma_i \le y_j + \gamma_{ij} M && \text{for all } (i,j) \in \bar{\mathcal{D}}, \\
& y_j + \sigma_j \le y_i + (1 - \gamma_{ij}) M && \text{for all } (i,j) \in \bar{\mathcal{D}}, \\
& \gamma_{ij} \in \{0,1\} && \text{for all } (i,j) \in \bar{\mathcal{D}},
\end{aligned}$$

where $M > 0$ is some sufficiently large number.

Consider some disjunctive arc $(i,j) \in \bar{\mathcal{D}}$. Let $i^<$ denote the set of indices on lane $l(i)$ from which there is a conjunctive path to i. Similarly, let $j^>$ denote the set of indices on lane $l(j)$ to which there is a conjunctive path from j. Now suppose $\gamma_{ij} = 0$, so the direction of the arc is $i \to j$, then we must clearly also have

$$p \to q \equiv \gamma_{pq} = 0 \ \text{ for all } p \in i^<, q \in j^>.$$

Written in terms of the disjunctive variables, this gives us the cutting planes

$$\sum_{p \in i^<, q \in j^>} \gamma_{pq} \le \gamma_{ij} M.$$

We refer to these as the *disjunctive cutting planes* and any feasible solution must satisfy these.

Next, we consider two types of cutting planes that follow from the necessary condition for optimality in Propositon 2. Suppose y is an optimal schedule. If we have $y_i + \rho \ge r_j$ for some conjunctive pair $(i,j) \in \mathcal{C}$, we must have $y_i + \rho = y_j$ by Proposition 2. In order to model this rule, we first introduce a binary variable β_{ij} that satisfies

$$\begin{aligned}
\beta_{ij} = 0 &\iff y_i + \rho < r_j, \\
\beta_{ij} = 1 &\iff y_i + \rho \ge r_j,
\end{aligned}$$

which can be enforced by adding the constraints

$$\begin{aligned}
& y_i + \rho < r_j + \beta_{ij} M, \\
& y_i + \rho \ge r_j - (1 - \beta_{ij}) M.
\end{aligned}$$

Now observe that the rule is enforced by adding the following cutting plane

$$y_i + \rho \ge y_j - (1 - \beta_{ij}) M.$$

We refer to the above cutting planes as *type I*. We can add more cutting planes on the disjunctive decision variables, because whenever $\beta_{ij} = 1$, the directions of the disjunctive arcs $i \to k$ and $j \to k$ must be the same for every other vertex $k \in \mathcal{N}$. Therefore, consider the following constraints

$$\beta_{ij} + (1 - \gamma_{ik}) + \gamma_{jk} \leq 2,$$
$$\beta_{ij} + \gamma_{ik} + (1 - \gamma_{jk}) \leq 2,$$

for every $(i, j) \in \mathcal{C}$ and for every $k \in \mathcal{N}$ with $l(k) \neq l(i) = l(j)$. These are the *type II* cutting planes.

2.2 Numerical examples

How to define heavy-load? Formalize the situation where "there are only two very large platoons at the end of the schedule".

Multimodal interarrival time distribution to model the natural occurrence of platoons.

Analyze running time of branch-and-cut and compare performance benefits of the different cutting planes.

3 Constructive heuristics

Methods that rely on the branch-and-cut framework are guaranteed to find an optimal solution, but their running time scale very badly with increasing instance sizes. Therefore, we are interested in developing heuristics to obtain good approximations in reasonable time. A common approach for developing such heuristics in the scheduling literature is to try and construct a good schedule in a step-by-step fashion. For our crossing time scheduling problem, we will consider methods to incrementally construct a vehicle ordering, to which we will refer as *constructive heuristics*. In order to support to upcoming discussion, we first introduce some auxiliary concepts and notation.

We define partial ordering π to be a *partial permutation* of $\mathcal{N}$, which is a sequence of elements from some subset $\mathcal{N}(\pi) \subset \mathcal{N}$. Let π be a partial ordering of length n and let $i \notin \mathcal{N}(\pi)$, then we use $\pi' = \pi + i$ to denote the concatenation of sequence π with i, so $\pi'_{1:n} = \pi_{1:n}$ and $\pi'_{n+1} = i$. Furthermore, let $\pi + \pi'$ denote the concatenation of two sequences π and π'. For each partial ordering π, the corresponding disjunctive graph $G(\pi)$ is incomplete, meaning that some of the disjunctive arcs have not yet been added. Nevertheless, observe that $\mathrm{LB}_\pi(i)$ is still defined for every $i \in \mathcal{N}$.

Observe that ordering vehicles is equivalent to ordering the lanes, due to the conjunctive constraints. We will define constructive heuristics in terms of repeatedly choosing the next lane. Hence, it may be helpful to model this process as a deterministic finite-state automaton, where the set of lane indices acts as the input alphabet $\Sigma = \{1, \ldots, n\}$, where n denotes the number of lanes. Let S denote the state space and let $\delta : S \times \Sigma \to S$ denote the state-transition function. Let s denote an instance of (12). We consider s to be a fixed part of the state, so it does not change with state transitions. The other part of the state is the current partial ordering π. The transitions of the automaton are very simple. Let $(s, \pi) \in S$ denote the current state and let $l \in \Sigma$ denote the next symbol. Let $i \in \mathcal{N} \setminus \mathcal{N}(\pi)$ denote the next unscheduled vehicle on lane l, then the system transitions to $(s, \pi + i)$. If no such vehicle exists, the transition is undefined. Therefore, an input sequence η of lanes is called a *valid lane order* whenever it is of length

$$N = \sum_{l \in \Sigma} n_l$$

and contains precisely $n_l = |\{i \in \mathcal{N} : l(i) = l\}|$ occurrences of lane $l \in \Sigma$. Given problem instance s, let $y_\eta(s)$ denote the schedule corresponding to lane order η. We say that lane order η is optimal whenever $y_\eta(s)$ is optimal. Observe that an optimal lane order must exist for every instance s, since we can simply derive the lane order from an optimal vehicle order.

Instead of mapping an instance s directly to some optimal lane order, we consider a mapping $p : S \to \Sigma$ such that setting $s_0 = (s, \varnothing)$ and repeatedly evaluating

$$s_t = \delta(s_{t-1}, p(s_{t-1}))$$

yields a final state $s_N(s, \pi^*)$ with optimal schedule π^*. Observe that this mapping must exist, because given some optimal lane order η^*, we can set $p(s_t) = \eta^*_{t+1}$, for every $t \in \{0, \dots, N-1\}$.

We do not hope to find an explicit representation of p, but our aim is to find good heuristic approximations.

3.1 Threshold heuristics

Consider the following simple *threshold rule*. Let π denote a partial schedule of length n, so $i = \pi(n)$ is the last scheduled vehicle on some lane $l = l(i)$, then define

$$p_\tau(s, \pi) = \begin{cases} l & \text{if } \mathrm{LB}_\pi(i) + \rho_i + \tau \geq r_j \text{ and } (i, j) \in \mathcal{C}, \\ \texttt{next}(\pi) & \text{otherwise,} \end{cases}$$

for some threshold $\tau \geq 0$. The expression $\texttt{next}(\pi)$ represents some lane other than l with unscheduled vehicles left.

Provide some examples. Provide some intuition why this would work, using the platoon preservation theorem. Discuss threshold tuning.

3.2 Neural heuristic

Provide some more introduction, explaining the intuition behind this architecture.

We will now consider a more general class of heuristics. We model the conditional distribution $p_\theta(\eta_{t+1}|s_t)$ with model parameters θ. Consider an instance s and some optimal lane sequence η with corresponding states defined as $s_{t+1} = \delta(s_t, \eta_{t+1})$ for $t \in \{0, \dots, N-1\}$. The resulting set of pairs (s_t, η_{t+1}) can be used to learn p_θ in a supervised fashion by treating it as a classification task.

Schedules are generated by employing *greedy inference* as follows. The model p_θ provides a distribution over lanes. We ignore lanes that have no unscheduled vehicles left and take the argmax of the remaining probabilities. We will denote the corresponding complete schedule by $\hat{y}_\theta(s)$.

Next, we discuss two ways of parameterizing the model. In both cases, we first derive, for every $l \in \Sigma$, a *lane embedding* $h(s_t, l)$ based on the current non-final state $s_t = (s, \pi_t)$ of the automaton. These are then arranged into a *state embedding* $h(s_t)$ as follows. Let η_t be the lane that was chosen last, then we apply the following *lane cycling* trick in order to keep the most recent lane in the same position of the state embedding, by defining

$$h_l(s_t) = h(s_t,\ l - \eta_t \bmod |\Sigma|),$$

for every $l \in \Sigma$. This state embedding is then mapped to a probability distribution

$$p_\theta(\eta_{t+1}|s_t) = f_\theta(h(s_t)),$$

where f_θ is a fully connected neural network.

Include illustration of the horizons and arrows to lane embeddings.

3.2.1 Padded embedding

Let $k_\pi(l)$ denote the first unscheduled vehicle in lane l under the partial schedule π_t. Denote the smallest lower bound of unscheduled vehicles as

$$T_\pi = \min_{i \in \mathcal{N} \setminus \mathcal{N}(\pi)} \mathrm{LB}_\pi(i).$$

Let the *horizon* of lane l be defined as

$$h'(s_t, l) = (\mathrm{LB}_{\pi_t}(k_{\pi_t}(l)) - T_{\pi_t}, \ldots, \mathrm{LB}_{\pi_t}(n_l) - T_{\pi_t}).$$

Observe that horizons can be of arbitrary dimension. Therefore, we restrict each horizon to a fixed length Γ and use zero padding. More precisely, given a sequence $x = (x_1, \ldots, x_n)$ of length n, define the padding operator

$$\mathrm{pad}(x, \Gamma) = \begin{cases} (x_1, \ldots, x_\Gamma) & \text{if } \Gamma \leq n, \\ (x_1, \ldots, x_n) +\!\!+ (\Gamma - n) * (0) & \text{otherwise,} \end{cases}$$

where we use the notation $n * (0)$ to mean a sequence of n zeros. The lane embedding is then given by

$$h(s_t, l) = \mathrm{pad}(h'(s_t, l), \Gamma).$$

3.2.2 Recurrent embedding

To avoid the zero padding operation, which can be problematic for states that are almost done, we can employ a recurrent architecture that is agnostic to the number of remaining unscheduled vehicles. Each variable-length horizon $h'(s_t, l)$ is simply transformed into the fixed-length vector by an Elman RNN by taking the output at the last step. Need to further specify this, but the current implementation is working.

3.3 Comparison

4 Local search

The previous section showed that constructive heuristics perform reasonably well on average and sometimes even produce optimal solutions. To further increase performance, without relying on the branch-and-bound framework, we define a local search procedure.

As seen in the previous sections, vehicles of the same lane occur mostly in groups, to which we will refer as *platoons*. For example, consider example lane order $\eta = (0, 1, 1, 0, 0, 1, 1, 1, 0, 0)$. This example has 5 platoons of consecutive vehicles from the same lane. The second platoon consists of two vehicles from lane 1. In general, let $P(\eta)$ denote the total number of platoons in η.

The basic idea of our local search neighborhood is to make little changes in these platoons by moving vehicles at the start and end of a platoon to the previous and next platoon of the same lane. More precisely, we define the following two types of modifications to a lane order. A *right-shift* modification of platoon i moves the last vehicle of this platoon to the next platoon of this lane. Similarly, a *left shift* modification of platoon i moves the first vehicle of this platoon to the previous platoon of this lane.

We construct the neighborhood of a solution by performing every possible right-shift and left-shift with respect to every platoon in the lane order. As an example, we have listed a full neighborhood in Table 2.

platoon id	left-shift	right-shift
1		(1, 1, 0, 0, 0, 1, 1, 1, 0, 0)
2	(1, 0, 1, 0, 0, 1, 1, 1, 0, 0)	(0, 1, 0, 0, 1, 1, 1, 1, 0, 0)
3	(0, 0, 1, 1, 0, 1, 1, 1, 0, 0)	(0, 1, 1, 0, 1, 1, 1, 0, 0, 0)
4	(0, 1, 1, 1, 0, 0, 1, 1, 0, 0)	(0, 1, 1, 0, 0, 1, 1, 0, 0, 1)
5	(0, 1, 1, 0, 0, 0, 1, 1, 1, 0)	

Table 2: Neighborhood of lane order $\eta = (0, 1, 1, 0, 0, 1, 1, 1, 0, 0)$.

References

[1] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *2015 American Control Conference (ACC)*, (Chicago, IL, USA), pp. 763–768, IEEE, July 2015.