Vehicle dynamics

In control theory, it is common to model motion dynamics of a system in terms of a state vector $s(t) \in \mathbb{R}^n$ and a control input vector $u(t) \in \mathbb{R}^m$, which result in a scalar position $y(t)$ via the equations

$$\dot{s}(t) = As(t) + Bu(t), \tag{1a}$$
$$y(t) = Cs(t). \tag{1b}$$

Furthermore, the state and control trajectories are often restricted by imposing linear constraints of the form

$$Gs(t) \leq b, \tag{2a}$$
$$Fu(t) \leq d. \tag{2b}$$

In the discussion that follows, each vehicle is modeled as a *double integrator*, with $s(t) = (p(t), v(t))$, where $p(t)$ and $v(t)$ are the scalar position along a predefined path and corresponding velocity, respectively. The three matrices are chosen such that

$$\begin{aligned} \dot{s}(t) &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} s(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t), \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} s(t), \end{aligned}$$

which may simply be rewritten as

$$\dot{p}(t) = v(t), \quad \dot{v}(t) = u(t), \quad y(t) = p(t), \tag{3}$$

where we recognize that the control input $u(t)$ corresponds directly to the acceleration of the vehicle.

Intersection model

Consider an intersection with n incoming lanes. We define the index set

$$\mathcal{N} = \{(l, k) : k \in \{1, \dots, n_l\},\ l \in \{1, \dots n\}\},$$

where n_l denotes the number of vehicles of lane l. To further help with notation, given vehicle index $i = (r, s) \in \mathcal{N}$, we define $l(i) = r$ and $k(i) = s$.

We assume that the position $p_i(t)$ of some vehicle $i \in \mathcal{N}$ corresponds to the physical front of the vehicle. In order to model a safe distance between vehicles on the same lane, we require that

$$p_i(t) - p_j(t) \geq P_i,$$

for all t and all pairs of indices $i, j \in \mathcal{N}$ such that $l(i) = l(j),\ k(i) + 1 = k(j)$, with $P_i \geq 0$. Let $\mathcal{C}$ denote the set of such ordered pairs of indices. Note that these constraints restrict vehicles from overtaking each other. Furthermore, in order to model collision avoidance, we say that a vehicle *occupies the intersection* whenever $p_i(t) \in [L, H_i] = \mathcal{E}_i$. The collision avoidance constraints are given by

$$(p_i(t), p_j(t)) \notin \mathcal{E}_i \times \mathcal{E}_j,$$

for all t and for all pairs of indices $i, j \in \mathcal{N}$ with $l(i) \neq l(j)$, which we collect in the set $\mathcal{D}$. Note that the length of a vehicle can be modeled by choosing H_i and P_i appropriately. Let $D_i(s_{i,0})$ denote the set of feasible trajectories $x_i(t) = (s_i(t), u_i(t))$ given some initial state $s_{i,0} = (p_i(0), v_i(0))$ and satisfying the vehicle dynamics given by equations (3). Given some performance criterion $J(x_i)$, the type of coordination problem we want to study is of the form

$$\begin{aligned}
\min_{\mathbf{x}(t)} \quad & \sum_{i \in \mathcal{N}} J(x_i) && && \text{(4a)} \\
\text{s.t.} \quad & x_i \in D_i(s_{i,0}), && \text{for all } i \in \mathcal{N}, && \text{(4b)} \\
& p_i(t) - p_j(t) \geq P_i, && \text{for all } (i,j) \in \mathcal{C}, && \text{(4c)} \\
& (p_i(t), p_j(t)) \notin \mathcal{E}_i \times \mathcal{E}_j, && \text{for all } \{i,j\} \in \mathcal{D}, && \text{(4d)}
\end{aligned}$$

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

Direct transcription

Optimization problem (4) can be transcribed directly into 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

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

for each $t \in (0, \Delta t, \dots, K\Delta t)$. The disjunctive constraints are formulated using the big-M technique by the constraints

$$\begin{aligned}
p_i(t) &\leq L + \delta_i(t) M, \\
H - \gamma_i(t) M &\leq p_i(t), \\
\delta_i(t) + \delta_j(t) + \gamma_i(t) + \gamma_j(t) &\leq 3,
\end{aligned}$$

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

$$p_i(t) - p_j(t) \geq P_i,$$

for each $t \in (0, \Delta t, \dots, K\Delta t)$ and each pair of consecutive vehicles $(i, j) \in \mathcal{C}$.

For example, consider the objective functional

$$J(x_i) = \int_{t=0}^{t_f} \left((v_d - v_i(t))^2 + u_i(t)^2 \right) dt, \tag{7}$$

where v_d is some reference velocity and t_f denotes the final time. For example, see the optimal trajectories in Figure 1.

i	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)
p_i	15	10	0	10	0
v_i	10	10	10	10	10

Table 1: Example initial conditions for problem (4).

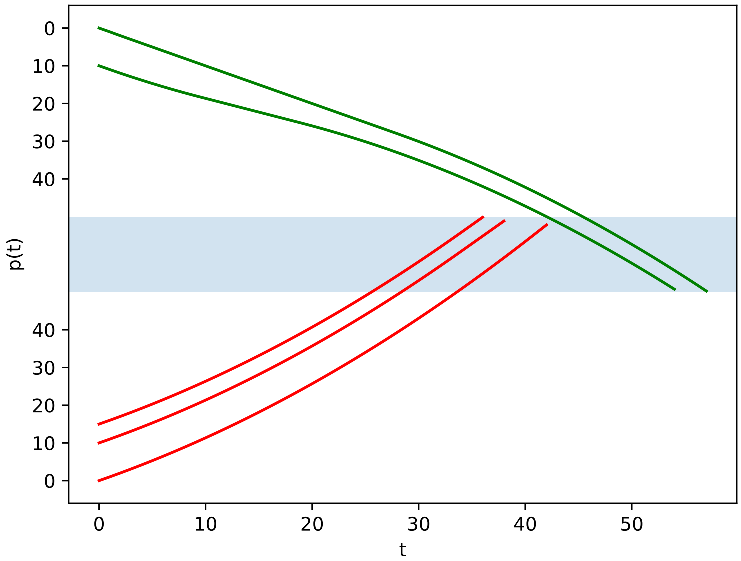


Figure 1: Example of optimal trajectories obtained using the direct transcription method with $P_i = P = 5, \mathcal{E}_i = \mathcal{E} = [50, 70], v_d = 20, T = 120, \Delta t = 0.1$ and initial conditions as given in Table 1. The y-axis is split such that each part corresponds to one of the two lanes 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 criterion

We start by considering a subclass of problems that allow us to almost ignore the vehicle dynamics. As performance criterion, we consider the *crossing time*

$$J(x_i) = \inf_t \{t : p_i(t) = L\}. \tag{8}$$

Furthermore, we impose a maximum speed for every vehicle, so

$$v_i(t) \leq v_{\max}, \tag{9}$$

for every t. We do not define any linear constraints on the control input, so we assume *instantaneous acceleration* is possible. For the purpose of the following discussion, it is not necessary to rigorously define what we mean by that. Define the *earliest crossing time* of vehicle i as

$$\begin{aligned} r_i = \inf_{x_i} \; & J(x_i) \\ \text{s.t.} \; & x_i \in D_i(s_{i,0}) \end{aligned}$$

It is not hard to see that we must have $r_i = (L - p_i(0))/v_{\max}$. Instead of optimizing in terms of trajectories $\mathbf{x}$, we consider first finding a *schedule* y for the crossing times by solving

$$\begin{aligned}
\min_{y} \quad & \sum_{i\in\mathcal{N}} y_i && && \text{(10a)}\\
\text{s.t.} \quad & r_i \le y_i && \text{for all } i \in \mathcal{N}, && \text{(10b)}\\
& y_i + \rho_i \le y_j && \text{for all } (i,j) \in \mathcal{C}, && \text{(10c)}\\
& y_i + \sigma_i \le y_j \text{ or } y_j + \sigma_j \le y_i && \text{for all } \{i,j\} \in \mathcal{D}, && \text{(10d)}
\end{aligned}$$

where $\rho_i = P_i/v_{\max}$ and $\sigma_i = (H_i - L)/v_{\max}$. By using the well-known big-M method, (10) can be turned into a mixed-integer program, for which solvers are readily available. Finally, note that an instance s of (10) is completely characterized by the tuple

$$s = (\mathcal{N}, \rho, \sigma, r).$$

Proposition 1. *The coordination problem* (4) *with performance criterion* (8) *and maximum speed constraints* (9) *is equivalent with* (10)*.*

Proof. We show that any feasible solution can be translated to a feasible solution to the other problem without changing the objective value.

Consider a set of trajectories $\mathbf{x}(t)$. Consider some arbitrary vehicle $i \in \mathcal{N}$. It follows directly from the definition of r_i that we must have $J(x_i) \ge r_i$. Consider a pair of consecutive vehicles $(i,j) \in \mathcal{C}$ on the same lane. For every $t \ge J(x_i)$, trajectory x_i must satisfy

$$p_i(t) \le L + (t - J(x_i))v_{\max}$$

and by the constraint (4c), trajectory x_j must satisfy

$$p_j(t) \le L + (t - J(x_i))v_{\max} - P_i.$$

Hence, we have $p_j(t) \le L$ if and only if $t \le J(x_i) + P_i/v_{\max}$, which implies that $J(x_j) \ge J(x_i) + \rho_i$. Consider a pair of vehicles $\{i,j\} \in \mathcal{D}$ on distinct lanes. By a similar reasoning, constraint (4d) implies that we have either $J(x_i) + \sigma_i \le J(x_j)$ or $J(x_j) + \sigma_j \le J(x_i)$. This shows that $y_i = J(x_i)$ is a feasible schedule for (10).

Now consider a feasible schedule y_i. For every $i \in \mathcal{N}$, we construct a trajectory x_i such that $J(x_i) = y_i$ by setting $p_i(t) = p_i(0) + t(L - p_i(0))/y_i$ for $0 \le t < y_i$ and $p_i(t) = L + (t - y_i)v_{\max}$ for $t \ge y_i$, so instantaneous acceleration is happening at $t = 0$ and $t = y_i$. □

Ordering vehicles

Instances and solutions of the crossing time optimization problem (10) can be represented very clearly by their *disjunctive graph*, which we define next. 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 (10c). The *disjunctive arcs* are used to encode the decisions regarding the ordering of vehicles from distinct lanes, corresponding to constraints (10d). For each pair $\{i,j\} \in \mathcal{D}$, at most one of the arcs (i,j) or (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 (10d). 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

$$\mathrm{LB}_\pi(j) = \max\{r_j, \max_{i \in N_\pi^-(j)} \mathrm{LB}_\pi(i) + w(i, j)\}, \tag{11}$$

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 $i \in \mathcal{N}$. Next, we show 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.

Proposition 2. *If y is an optimal schedule for* (10) *with ordering π, then*

$$y_i = \mathrm{LB}_\pi(i) \quad \textit{for all } i \in \mathcal{N}. \tag{12}$$

Proof. Suppose y is an optimal schedule with ordering π. We write $\pi(k)$ for the kth element in the ordering, which is a permuation of $\mathcal{N}$. Consider the smallest $k \in \{1, \dots, |\mathcal{N}|\}$ such that vehicle $j = \pi(k)$ satisfies $y_j > \mathrm{LB}_\pi(j)$. If no such k exists, y already satisfies (12). Otherwise, we construct a schedule y' by setting $y'_i = y_i$ for every $i \in \mathcal{N}, i \neq j$ and $y'_j = \mathrm{LB}_\pi(j)$.

We now argue that y' is stil a feasible schedule. Due to their direction, we only have to verify the inequalities in (10) corresponding to incoming arcs $(i, j) = (\pi(r), \pi(k))$ with $r < k$. For these nodes i, we have $y_i = \mathrm{LB}_\pi(i)$ by definition of k. From the definition of LB then follows that

$$y'_j = \mathrm{LB}_\pi(j) \geq \mathrm{LB}_\pi(i) + w(i, j) = y'_i + w(i, j),$$

which shows that all inequalities still hold.

The new schedule has strictly better objective $\sum_{i \in \mathcal{N}} y'_i < \sum_{i \in \mathcal{N}} y_i$, which contradicts the assumption that y is optimal. $\square$

The previous result shows that we can concentrate on finding an optimal ordering π. 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. Before we present this rule, we first prove the following lemma that provides an easier expression for calculating the lower bounds under these assumptions.

Lemma 1. *Let π be some permutation of $\mathcal{N}$. Assume that $\sigma_i = \rho_i + s$, for every $i \in \mathcal{N}$, with $s > 0$. Consider a pair $i, j \in \mathcal{N}$ such that i is the immediate predecessor of j in π, so $\pi^{-1}(i) + 1 = \pi^{-1}(j)$, then*

$$\mathrm{LB}_\pi(j) = \max\{r_j, \mathrm{LB}_\pi(i) + w(i, j)\}. \tag{13}$$

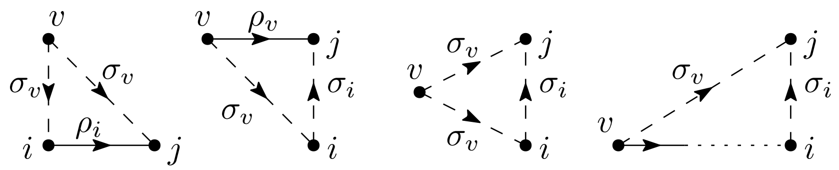


Figure 2: Sketch of the four cases distinguished in the proof of Lemma 1. Arc weights are given and disjunctive arcs $\mathcal{O}(\pi)$ are drawn with a dashed line.

Proof. Suppose $(i, j) \in \mathcal{C}$, see Figure 2, then the incoming disjunctive arcs of j are $N_\pi^-(j) \setminus \{i\} \subset N_\pi^-(i)$. Therefore, we have

$$\max_{v \in N_\pi^-(j) \setminus \{i\}} \mathrm{LB}_\pi(v) + \sigma_v \leq \mathrm{LB}_\pi(i),$$

so that $\mathrm{LB}_\pi(v) + w(v, j) \leq \mathrm{LB}_\pi(i) + w(i, j)$ for all $v \in N_\pi^-(j)$.

Otherwise, we have $(i, j) \in \mathcal{O}(\pi)$. Let $v \in \mathcal{N}$ such that (v, j) is an arc. If $(v, j) \in \mathcal{C}$, then we have

$$\mathrm{LB}_\pi(v) + w(v, j) = \mathrm{LB}_\pi(v) + \rho_v \leq \mathrm{LB}_\pi(v) + \sigma_v + \sigma_i \leq \mathrm{LB}_\pi(i) + w(i, j),$$

where the second inequality follows from $(v, i) \in \mathcal{O}(\pi)$. If $(v, j) \in \mathcal{O}(\pi)$ with $l(v) \neq l(i)$, then $(v, i) \in \mathcal{O}(\pi)$, so

$$\mathrm{LB}_\pi(v) + w(v, j) = \mathrm{LB}_\pi(v) + w(v, i) \leq \mathrm{LB}_\pi(i) \leq \mathrm{LB}_\pi(i) + w(i, j).$$

If $(v, j) \in \mathcal{O}(\pi)$ with $l(v) = l(i)$, then there is a path of conjunctive arcs between v and i, so we must have $\mathrm{LB}_\pi(v) + \rho_v \leq \mathrm{LB}_\pi(i)$. Furthermore, from $w(v, j) = \sigma_v = \rho_v + s$ follows that

$$\mathrm{LB}_\pi(v) + w(v, j) = \mathrm{LB}_\pi(v) + \rho_v + s \leq \mathrm{LB}_\pi(i) + s \leq \mathrm{LB}_\pi(i) + w(i, j).$$

To conclude, we have shown that $\mathrm{LB}_\pi(v) + w(v, j) \leq \mathrm{LB}_\pi(i) + w(i, j)$ for any $v \in N_\pi^-(j)$, from which statement (13) follows. $\square$

Proposition 3. *If an instance of* (10) *is such that $\rho_i = \rho$ and $\sigma_i = \sigma > \rho$ for all $i \in \mathcal{N}$ and $r_{i^*} + \rho \geq r_{j^*}$ for some $(i^*, j^*) \in \mathcal{C}$, then j^* follows immediately after i^* in any optimal ordering π.*

Proof. Suppose π is an optimal ordering with $\pi^{-1}(i^*) + 1 < \pi^{-1}(j^*)$ and let y denote the corresponding schedule. Let $\mathcal{I}(i, j) = \{i, \pi(\pi^{-1}(i) + 1), \dots, j\}$ be the set of vehicles between i and j. Let $f = \pi(1)$ and $e = \pi(|\mathcal{N}|)$ be the first and last vehicles, respectively, and set $u = \pi^{-1}(i^*) + 1$ and $v = \pi^{-1}(j^*) - 1$, see also Figure 3. Construct new ordering π' by moving vehicle j^* forward by $|\mathcal{I}(u, v)|$ places and let y' denote the corresponding schedule. We have $y_i = y'_i$ for all $i \in \mathcal{I}(f, i^*)$. Using Lemma 1, we compute

$$\begin{aligned}
y'_{j^*} &= \max\{r_{j^*}, y_{i^*} + \rho\} = y_{i^*} + \rho, \\
y_u &= \max\{r_u, y_{i^*} + \sigma\}, \\
y'_u &= \max\{r_u, y_{i^*} + \rho + \sigma\},
\end{aligned}$$

where we used that $y_{i^*} + \rho \geq r_{i^*} + \rho \geq r_{j^*}$ by assumption. Note that we have $y_{i^*} + \sigma + (|\mathcal{I}(u,v)| - 1)\rho \leq y_v$, regardless of the type of arcs between consecutive vehicles in $\mathcal{I}(u,v)$. Therefore,

$$y_{j^*} - y'_{j^*} \geq y_v + \sigma - y_{i^*} - \rho \geq 2\sigma + (|\mathcal{I}(u,v)| - 2)\rho.$$

We now show that $y'_k \geq y_k$ and $y'_k - y'_{j^*} \leq y_k - y_{i^*}$ for every $k \in \mathcal{I}(u,v)$. For $k = u$, it is clear that $y'_u \geq y_u$ and

$$y'_u - y'_{j^*} = \max\{r_u - (y_{i^*} + \rho), \sigma\} \leq \max\{r_u - y_{i^*}, \sigma\} = y_u - y_{i^*}.$$

Now proceed by induction and let x be the immediate predecessor of k for which the inequalities hold, then

$$y'_k = \max\{r_k, y'_x + w(x,k)\} \geq \max\{r_k, y_x + w(x,k)\} = y_k$$

and the second inequality follows from

$$\begin{aligned}(y'_k - y'_x) + (y'_x - y'_{j^*}) &= \max\{r_k - y'_x, w(x,k)\} + (y'_x - y'_{j^*}) \\ &\leq \max\{r_k - y_x, w(x,k)\} + (y_x - y_{i^*}) \\ &= (y_k - y_x) + (y_x - y_{i^*}).\end{aligned}$$

Let l denote the immediate successor of j^*, if there is one. Regardless of whether j^* and l are in the same lane, we have $y_{j^*} + \rho \leq y_l$. We derive

$$y'_v = y'_v - y'_{j^*} + y'_{j^*} \leq y_v - y_{i^*} + y'_{j^*} = y_v + \rho \leq y_{j^*} - \sigma + \rho,$$

from which follows that $y'_v + \sigma \leq y_l$, which means that $y_i \geq y'_i$ for $i \in \mathcal{I}(l,e)$.

We can now compare the objectives by putting everything together

$$\begin{aligned}\sum_{i \in \mathcal{N}} y_i - y'_i &= y_{j^*} - y'_{j^*} + \sum_{i \in \mathcal{I}(u,v)} y_i - y'_i + \sum_{i \in \mathcal{I}(l,e)} y_i - y'_i \\ &\geq 2\sigma + (|\mathcal{I}(u,v)| - 2)\rho + \sum_{i \in \mathcal{I}(u,v)} (y_i - y_{i^*}) - (y'_i - y'_{j^*}) \\ &\qquad + |\mathcal{I}(u,v)|(y'_{j^*} - y_{i^*}) \\ &\geq 2\sigma - 2\rho > 0\end{aligned}$$

which contradicts the assumption that π was optimal. □

Partial ordering

Finding an optimal schedule by solving the mixed-integer linear program boils down to systematically evaluating all different schedules. This method does not scale well to larger instances, so we are interested in good heuristics. A common approach in the scheduling literature is to incrementally construct a schedule by fixing one job starting time (vehicle crossing time) at each step. We will consider incrementally constructing a vehicle ordering. Therefore, 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

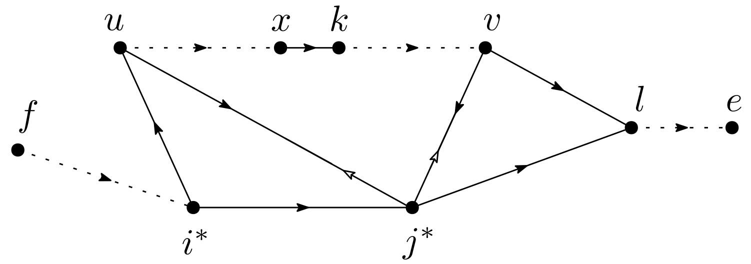


Figure 3: Sketch of the nodes and most important arcs used in the proof of Proposition 3. Dashed arcs represent chains of unspecified length. The two open arrows indicate the new direction of their arc under ordering π'.

sequence π with i, so $\pi'_{1:n} = \pi_{1:n}$ and $\pi'_{n+1} = i$. Furthermore, recursively define the concatenation of two sequences by $\pi +\!\!+\, \pi' = (\pi +\!\!+\, \pi'_1) +\!\!+\, \pi'_{2:m}$, where m is the length of 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 $i \in \mathcal{N}$. Now consider the following rule for partial orderings, which is a generalization of Proposition 3, because $\mathrm{LB}_{\pi'}(i^*) = r_{i^*}$ for the empty ordering $\pi = \varnothing$.

Proposition 4. *Consider an instance of* (10) *with $\rho_i = \rho$ and $\sigma_i = \sigma > \rho$ for all $i \in \mathcal{N}$. Let π be a partial ordering of length n. If π is such that $LB_\pi(i^*) + \rho_{i^*} \geq r_{j^*}$ for some $(i^*, j^*) \in \mathcal{C}$ with $i^*, j^* \notin \pi$, then j^* follows immediately after i^* in any complete optimal ordering $\pi +\!\!+\, \pi'$.*

Proof. Consider a new problem instance s' on the unscheduled vehicles $\mathcal{N}' = \mathcal{N} \setminus \mathcal{N}(\pi)$, with $r'_i = \mathrm{LB}_\pi(i)$ for all $i \in \mathcal{N}'$. It follows from Proposition 3 that j^* follows i^* immediately in any optimal ordering π' of the remaining vehicles. $\square$

Lane ordering

Observe that ordering vehicles is equivalent to ordering the lanes, due to the conjunctive constraints. We propose a method based on repeatedly choosing the next lane. This may be modeled as a deterministic finite-state automaton, where the set of lane indices acts as the input alphabet $\Sigma = \{1, \dots, n\}$, 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 (10). 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 $(s, \pi \mathbin{+\!\!\!+} i)$. If no such vehicle exists, the transition is undefined. Observe that some lane sequence η is valid 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 l, for each $l \in \Sigma$. Let $\delta(s, \eta)$ denote the state that we obtain after applying sequence η to the automaton with initial state $s_0 = (s, \varnothing)$, which generalizes the single step transition function by recursively defining

$$\delta(s, \eta_{1:t}) = \delta(\delta(s, \eta_{1:t-1}), \eta_t).$$

Behavioral cloning

It is now clear that task of finding an optimal schedule has been reduced to finding an optimal lane sequence. Therefore, our goal is now to find a mapping from problem instances to optimal lane sequences. This mapping is of course very complex, so our aim is to find a good approximation. Instead of formulating a direct mapping, we model the conditional distribution $p_\theta(\eta|s)$ of the optimal lane sequence given a problem instance s and we factorize it as

$$p_\theta(\eta \mid s) = \prod_{t=1}^{N} p_\theta(\eta_t \mid \delta(s, \eta_{1:t-1})),$$

where θ denotes the model parameters. We aim to learn this conditional distribution from a set of instances with corresponding optimal schedules. Given problem instance s, let $y_\eta(s)$ denote the schedule obtained from lane order η. We first compute an optimal schedule $y^*(s)$. Next, we compute η such that $y^*(s) = y_\eta(s)$ to obtain the corresponding trajectory of states $s_t = \delta(s, \eta_{1:t})$. The resulting set of pairs (s_t, η_t) can be used to learn p_θ in a supervised fashion by treating it as a classification task.

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

Parameterization

We will now discuss two ways of parameterizing the model. In both cases, we first derive, for every $l \in \Sigma$, a *lane embedding* $h_l(s_t)$ based on the current nonterminal state $s_t = (s, \pi_t)$ of the automaton. We then apply the following trick, which we call *lane cycling*. Let η_t

These embeddings are 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.

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

$$h_l(s_t) = (\mathrm{LB}_{\pi_t}(k_{\pi_t}(l)) - T_{\pi_t}, \dots, \mathrm{LB}_{\pi_t}(n_l) - T_{\pi_t}).$$

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 $x = (x_1, \dots, x_n)$ of length n, define the padding operator

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

where we use the notation $n * (0)$ to mean a sequence of n zeros. The full observation is now given by

$$h(s_t) = (\mathrm{pad}(h_1(s_t), \Gamma), \dots, \mathrm{pad}(h_{|\Sigma|}(s_t), \Gamma)).$$

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.

Experiments

We consider an intersection with 2 approaching lanes. Instances are generated by sampling g_i and ρ_i and setting

$$r_i = \sum_{k=1}^{k(i)} g_i + \sum_{k=1}^{k(i)-1} \rho_i,$$

for each $i \in \mathcal{N}$. Let $\mathcal{X}$ denote the training data, consisting of pairs of (s_t, η_t) as obtained from exact solutions as explained above.

We interpret $p_\theta(s_t)$ as the probability of choosing lane 1. We use the binary cross entropy loss

$$-\frac{1}{|\mathcal{X}|}\sum_{(s_t,\eta_t)\in\mathcal{X}} \mathbb{1}\{\eta_t = 1\}\log(p_\theta(s_t)) + \mathbb{1}\{\eta_t = 2\}\log(1 - p_\theta(s_t)),$$

where we use $\mathbb{1}(\cdot)$ to denote the indicator function. We use learning rate 10^{-3} and the Adam optimizer. Let $\mathcal{Y}$ denote a set of test instances. Let $\mathrm{obj}(y)$ denote the objective of schedule y, then we report on the average approximation ratio defined as

$$\hat{y}_\theta/y^* = \frac{1}{|\mathcal{Y}|}\sum_{s\in\mathcal{Y}} \mathrm{obj}(\hat{y}_\theta(s)) \;/\; \mathrm{obj}(y^*(s)).$$

We are particularly interested in the ability of the model to generalize to instances with more vehicles, because this is where the MILP solving time begins to become prohibitive.

Discussion

It might be insightful to compare the model probability of the computed greedy schedule to the model probability of the optimal solution computed using mixed-integer linear programming. This provides us an indication of model fit and whether greedy inference is good enough or methods like beam search might be necessary.

It might be interesting to analyze the feature attribution of the neural network using a method like Integrated Gradients.

Implementation details

Recall that instances of problem (10) are completely characterized by

$$s = (\mathcal{N}, \rho, \sigma, r).$$

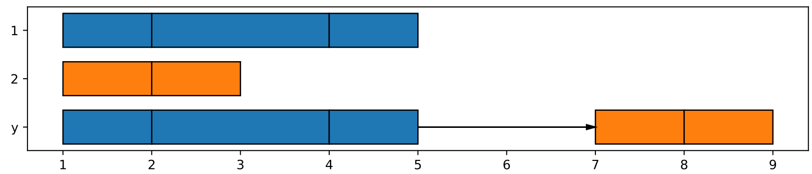
Throughout the code, it is assumed that

$$\sigma_i = \rho_i + s,$$

because this allows us to use the result from Lemma 1. Therefore, instances are represented by specifying earliest crossing time r_i, length ρ_i and *switch-over* time s. We will refer to earliest crossing time as the *release time* of the vehicle. Instances are represented in the code as basic dictionaries of the form

```
instance = {
    'release': [[1, 2, 4], [1, 2]],
    'length':  [[1, 2, 1], [1, 1]],
    'switch':  2
}
```

Instances and (partial) schedules can be visualized using `util.plot_schedule()`. For example, instance above together with the optimal solution is given in the following figure.



Each vehicle is drawn as a rectangle, whose width represents ρ_i. The color of each rectangle corresponds to its lane. The first two rows of the figure visualize the instance specification and the last row visualizes the optimal schedule y. The arrow visualizes the switch-over time s.

The automaton keeps track of $\mathrm{LB}_\pi(i)$ for each node i, given the current partial ordering. It provides these values as basic observations. We transform these into the desired observations for training our heuristic. The method

`Automaton.exhaustive(lane)`

returns whether the rule of Proposition 4 applies to the given lane.

Bibliographical notes

The disjunctive graph is a common formalism used for job shop scheduling problems (see Chapter 7 of [1]), to which our crossing time scheduling problem is related. Furthermore, in scheduling theory terminology, the result of Proposition 2 says that optimal schedules are necessarily *semi-active schedules*, see Definition 2.3.5 in [1]. The rule for optimal orderings (Proposition 3) is equivalent to the *Platoon Preservation Theorem* of Limpens [2].

Machine learning has been used extensively to solve combinatorial problems, see for example the seminal paper [3] and surveys [4, 5].

References

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems.* Cham: Springer International Publishing, 2016.

[2] M. Limpens, *Online Platoon Forming Algorithms for Automated Vehicles: A More Efficient Approach.* Bachelor, Eindhoven University of Technology, Sept. 2023.

[3] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural Combinatorial Optimization with Reinforcement Learning,” Jan. 2017.

[4] Y. Bengio, A. Lodi, and A. Prouvost, “Machine Learning for Combinatorial Optimization: A Methodological Tour d’Horizon,” Mar. 2020.

[5] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, “Reinforcement Learning for Combinatorial Optimization: A Survey,” Dec. 2020.