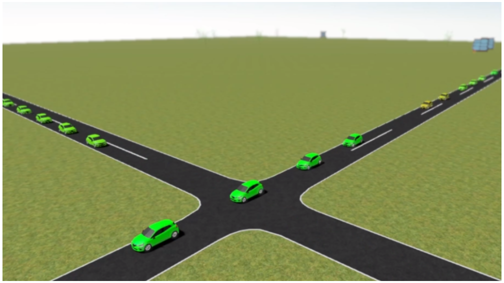
Coordination of autonomous vehicles

Jeroen van Riel

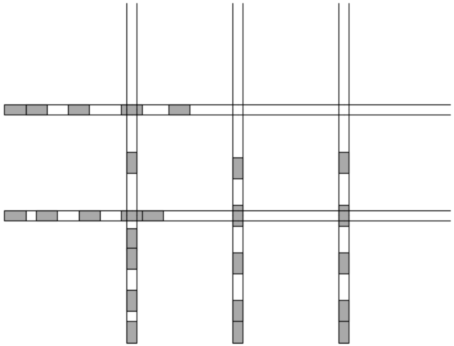
January 2025

Coordination of autonomous vehicles



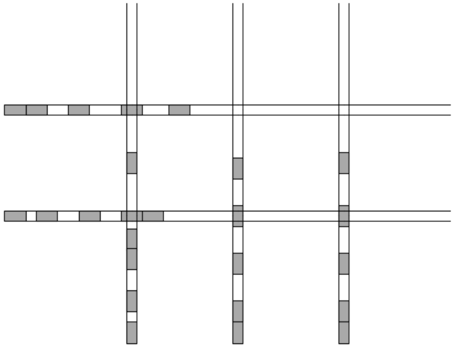
- Less human intervention (less traffic lights required)
- Better guarantees on safety
- Potentially reduce economic costs

. . . as optimal control problem



- Multi-agent optimal control problem
    - Minimize total travel time
    - Avoid collisions

. . . as optimal control problem



- Some simplifying assumptions
    - Central control with perfect communication
    - Fixed routes
    - All future arrivals known

How to solve it?

- Direct transcription methods
    - Provide optimal trajectories
    - Computationally very demanding
- How to solve large instances?
    - Need for approximation
    - Exploit problem structure (decomposition)
    - Automatically derive heuristics (learning)

Research questions

Q1: Exploit problem structure (decomposition)

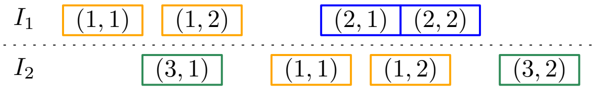
How can we model the offline trajectory optimization problem, modeling coordination of autonomous vehicles in networks of unsignalized intersections, as a variant of job-shop scheduling to effectively reduce the dimensionality of the problem, while guaranteeing the generation of collision-free trajectories?

Q2: Automatically derive heuristics (learning)

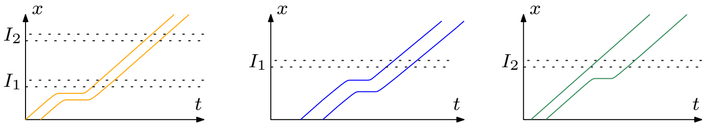
Can recent deep reinforcement learning formulations for combinatorial optimization problems, like job-shop scheduling, be adapted to the offline trajectory optimization problem, formulated in terms of job-shop scheduling, in order to provide a scalable optimization algorithm that automatically learns to exploit hidden structures in problem instances?

Decomposition

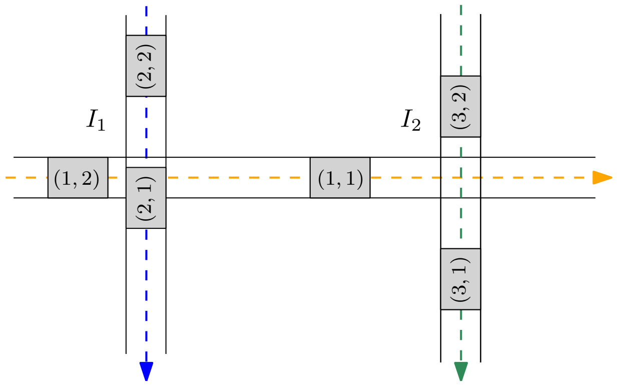
- Upper-level crossing time scheduling
  - Mixed-Integer Linear Programming (MILP)

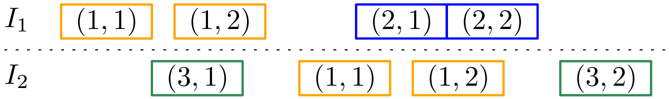


- Lower-level trajectory optimization problem
  - Direct transcription $\rightarrow$ linear programming

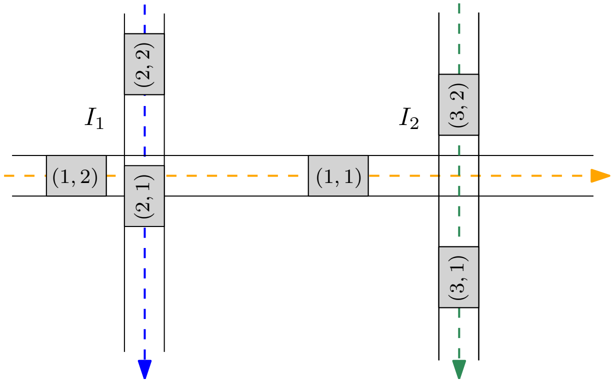


Determine crossing times



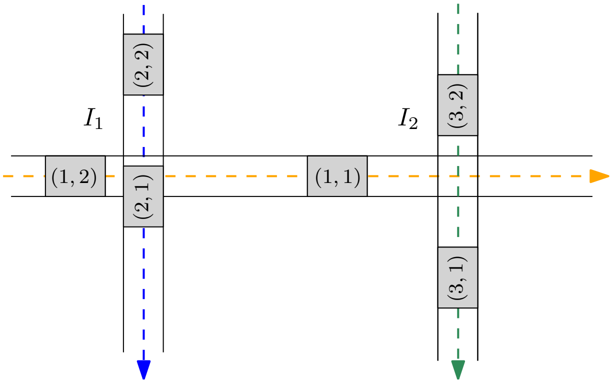


Determine crossing order



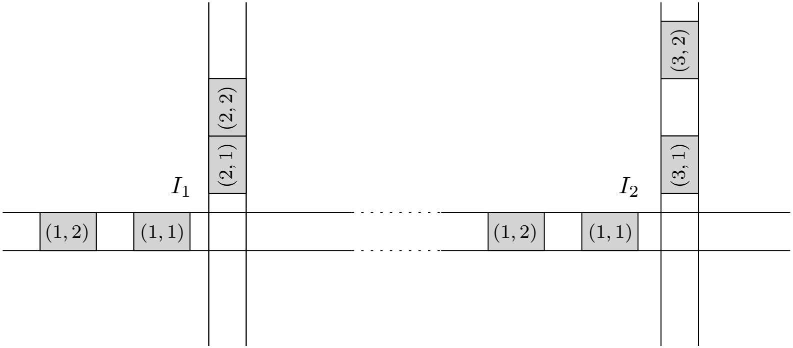
- Crossing times follow from crossing order

Determine crossing order

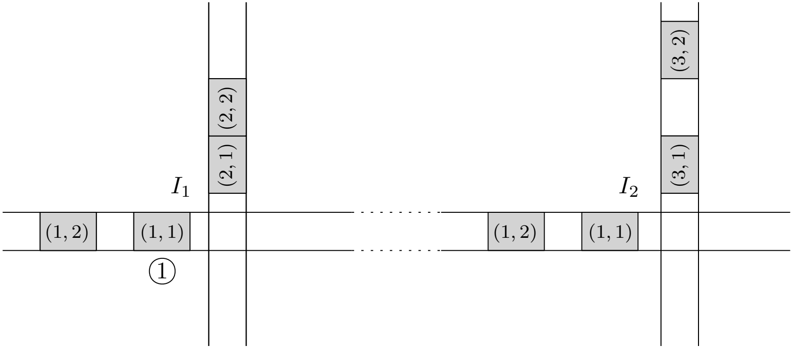


- Map instance to optimal crossing order
- Use step-by-step construction. . .

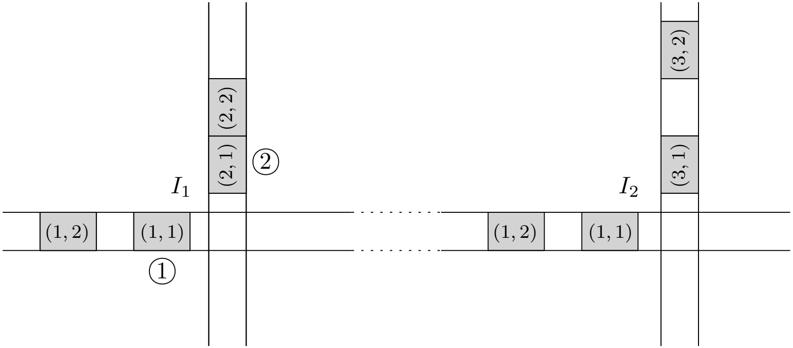
Determine crossing order



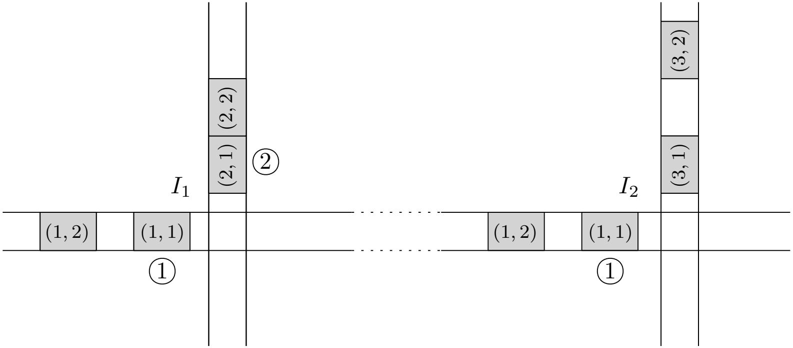
Determine crossing order



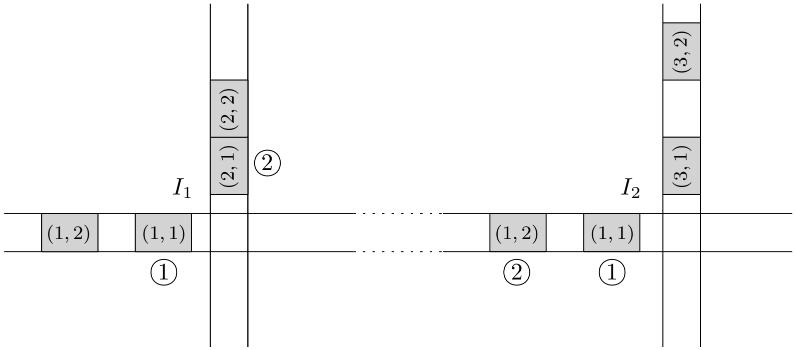
Determine crossing order



Determine crossing order



Determine crossing order



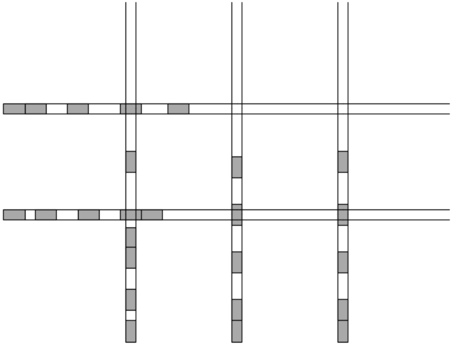
Learn crossing order

- ~~Map instance to optimal crossing order~~
- Map partial order to next partial order (policy)

- We can learn this policy from examples!
  - Imitation learning from optimal MILP solutions
  - Reinforcement learning with dense delay reward

Overview of project plan

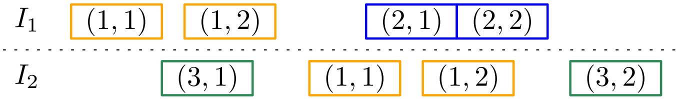
- Coordination as optimal control problem
- Vehicle scheduling + trajectory optimization
- Sequentially construct crossing order
- Learn from examples

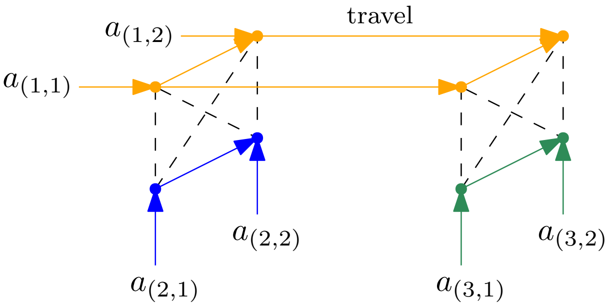


Appendix: Disjunctive graph

Disjunctive graph

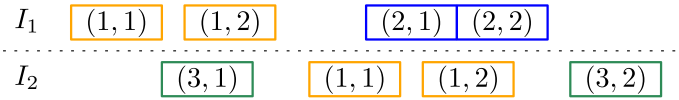
- Partial solutions encoded as disjunctive graph augmented with lower bounds on crossing times
- Parameterize ordering policy based on graph neural network embedding of augmented disjunctive graph

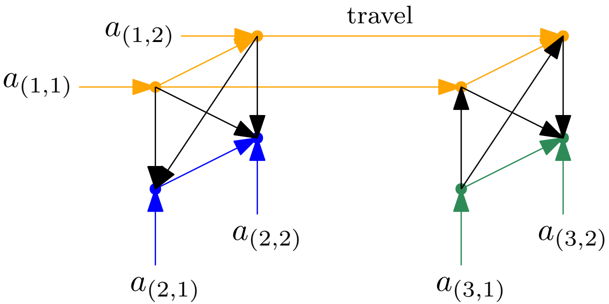




Disjunctive graph

- Partial solutions encoded as disjunctive graph augmented with lower bounds on crossing times - Parameterize ordering policy based on graph neural network embedding of augmented disjunctive graph



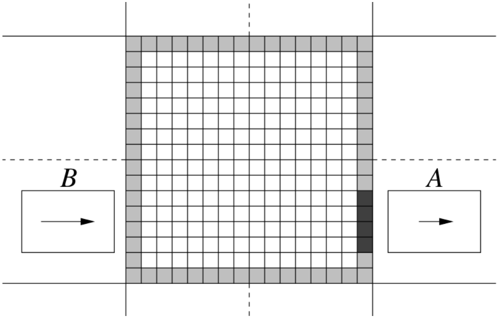


Appendix: Related literature

- Autonomous intersections
- Neural combinatorial optimization

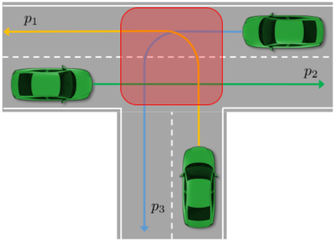
Autonomous intersections

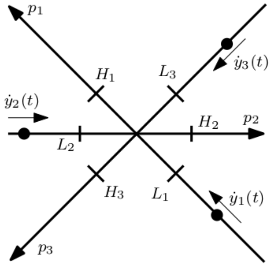
- "Autonomous Intersection Control" (Dresner & Stone)
  - Single intersection
  - Time slot reservation-based protocol
  - Central intersection manager



Autonomous intersections

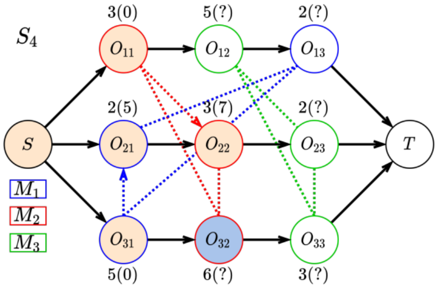
- "Approximate Optimal Coordination" (Hult et al.)
  - Single intersection
  - Single vehicle per lane
  - Explicit collision-avoidance constraints



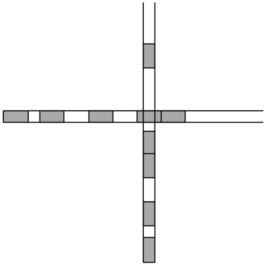


Neural combinatorial optimization

- "Learn to dispatch" (Zhang et al.)
  - Job-shop scheduling problem
  - Dispatch next operation
  - Policy using Graph Isomorphism Network (GIN)



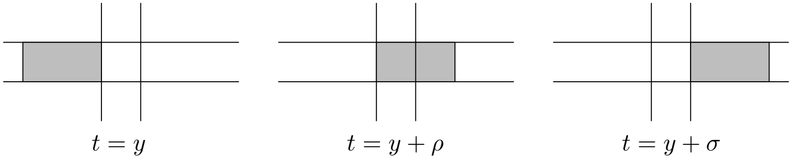
Appendix: Single intersection



- Notation
- Upper-level crossing time scheduling
- Lower bound on starting times
- Imitation learning with neural policy
- Lower-level trajectory optimization

Notation

- vehicle indices $\mathcal{N}$
- $y(i)$ is crossing time of vehicle i
- r_i earliest crossing time of vehicle i



- i and j same lane: $y(i) + \rho \leq y(j)$
- i and j distinct lanes: $y(i) + \sigma \leq y(j)$ or $y(j) + \sigma \leq y(i)$

Upper-level crossing time scheduling

- conjunctive constraints $\mathcal{C}$
- disjunctive (conflict) constraints $\mathcal{D}$

$$\begin{aligned}
\min_{y} \quad & \sum_{i \in \mathcal{N}} y(i) & \\
\text{s.t.} \quad & r_i \leq y(i), & \text{for all } i \in \mathcal{N}, \\
& y(i) + \rho \leq y(j), & \text{for all } (i,j) \in \mathcal{C}, \\
& y(i) + \sigma \leq y(j) \text{ or } y(j) + \sigma \leq y(i), & \text{for all } (i,j) \in \mathcal{D}
\end{aligned}$$

Upper-level crossing time scheduling

- Formulate as mixed-integer linear program (MILP)
- Introduce binary decision variables γ_{ij}
- Use big-M technique

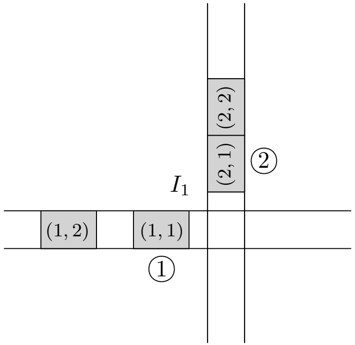
$$\begin{aligned}
\min_{y} \quad & \sum_{i \in \mathcal{N}} y_i & \\
\text{s.t.} \quad & r_i \leq y_i, & \text{for all } i \in \mathcal{N}, \\
& y_i + \rho_i \leq y_j, & \text{for all } (i,j) \in \mathcal{C}, \\
& y_i + \sigma_i \leq y_j + \gamma_{ij} M, & \text{for all } (i,j) \in \mathcal{D}, \\
&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

Lower bounds on starting times

- Disjunctive graph given current order π
- Nodes are vehicle indices $\mathcal{N}$
- Edges $i \xrightarrow{w(i,j)} j$
  - Conjunctive edges $i \overset{\rho}{\to} j$
  - Disjunctive edges $i \overset{\sigma}{\to} j$ or $j \overset{\sigma}{\to} i$
- Lower bounds LB_π on starting times given current order π

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

Imitation learning with neural policy



- crossing order $\pi = ((1,1),(2,1))$ of vehicles
- step-by-step construction of this order
  - 1. choose $(1,1)$
  - 2. choose $(2,1)$
  - 3. $\dots$

Imitation learning with neural policy

- get optimal trajectories from MILP solver
- parameterize policy based on LB_π
  - only consider $\mathrm{LB}_\pi(j)$ for unscheduled j
  - recurrent embedding of $\mathrm{LB}_\pi(j)$ per lane
  - alternatively, use zero padding
- fit policy parameters to expert transitions

Lower-level trajectory optimization

- position x, velocity v, control input u
- position of vehicle in front x', follow distance L
- position of intersection B, crossing time τ

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