2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Multi-agent Path Finding for Cooperative Autonomous Driving

Zhongxia Yan, Han Zheng, Cathy Wu

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10611649

***Abstract*— Anticipating possible future deployment of connected and automated vehicles (CAVs), cooperative autonomous driving at intersections has been studied by many works in control theory and intelligent transportation across decades. Simultaneously, recent parallel works in robotics have devised efficient algorithms for multi-agent path finding (MAPF), though often in environments with simplified kinematics. In this work, we hybridize insights and algorithms from MAPF with the structure and heuristics of optimizing the crossing order of CAVs at signal-free intersections. We devise an optimal and complete algorithm, Order-based Search with Kinematics Arrival Time Scheduling (OBS-KATS), which significantly outperforms existing algorithms, fixed heuristics, and prioritized planning with KATS. The performance is maintained under different vehicle arrival rates, lane lengths, crossing speeds, and control horizon. Through ablations and dissections, we offer insight on the contributing factors to OBS-KATS's performance. Our work is directly applicable to many similarly scaled traffic and multi-robot scenarios with directed lanes.**

I. Introduction

The development of autonomous driving technology raises the possibility of intelligent coordination of connected and automated vehicles (CAVs) towards societal objectives, such as reducing congestion and fuel consumption, as well as improving safety. Therefore, many works on intelligent transportation systems [4], [27], [28], [30] have studied potential positive impacts of cooperative driving of CAVs. In particular, signal-free intersections are regions where coordination of CAVs is critical to safety and efficiency. These intersections are not restricted to intelligent transportation systems, but also are commonly found in real-world robotic warehouses at crossings between directed lanes [16]. In this work, we adapt insights and algorithms from multi-agent path finding (MAPF) for the coordination of a cooperative driving intersection with rich vehicle kinematics. Like [28], we divide the overall task of coordinating CAVs into two sequential phases, first optimizing the CAV crossing order then computing order-conditioned vehicle trajectories. To define the crossing order, we divide the intersection into a reservation system where the arrival and departure times at *subzones* are planned by our low-level Kinematic Arrival Time Scheduling (KATS), which substitutes for a low-level path planning algorithm. While existing works optimize the crossing order with First-In-First-Out (FIFO) heuristics [4] and Monte Carlo Tree Search (MCTS) [29], we demonstrate that our MAPF-inspired high-level prioritized planning [5] and Order-based Search (OBS) algorithms obtain significantly superior solution quality. We obtain order-conditioned vehicles trajectories with trajectory optimization rather than single agent path planning algorithms like A* search [6] or SIPP [21], allowing us to bypass kinematics limitations.

In summary, our main contributions are:

1) Incorporating insights from MAPF, we design an algorithm for ordering vehicle crossings at a signal-free intersection and translating the crossing order to vehicle trajectories within a kinematic bicycle model.
2) We empirically characterize the OBS-KATS's significant improvement of vehicle delays over baselines under a wide range of intersection settings.
3) We prove the soundness, completeness, and optimality of OBS-KATS for finding vehicle crossing orders.

We provide full source code for reproducibility on GitHub.

II. Related Work

A. Cooperative Driving at Intersections

Cooperative driving of connected and automated vehicles (CAVs) has been studied in intelligent transportation settings ranging from adaptive cruise control [26] to traffic networks with diverse structures [30]. In particular, several recent cooperative driving strategies have been proposed for optimizing the crossing order of CAVs at signal-free intersections [28]. The First-In-First-Out (FIFO) strategy has been studied by [4] as a heuristic crossing order. Given an existing crossing order, the Dynamic Resequencing method [32] inserts a newly arriving vehicles into a suitable position, but keeps the rest of the order unchanged. On the other hand, [29] demonstrates that Monte Carlo Tree Search (MCTS) can be used to obtain a more optimal crossing order by periodically replanning the existing order. Our work significantly improves upon these previous methods in the cooperative driving setting by leveraging insights and algorithms from multi-agent path finding.

B. Multi-agent Path Finding

The classical multi-agent path finding (MAPF) problem [25] is a NP-hard [31] problem which seeks to find the shortest collision-avoiding paths for a set of agents in a discrete graph. Since the space of joint agent trajectories is intractably large to consider [23], nearly all MAPF algorithms rely on repeatedly calling single-agent path planner such as A* search [6] or SIPP [21], while holding paths of some set of other agents as constraints.

Prioritized planning (PP) [5], [24] plans one agent trajectory at a time in random agent order while avoiding

This work was supported by the National Science Foundation (NSF) CAREER award 2239566 and the MIT Amazon Science Hub.

Zhongxia Yan, Han Zheng and Cathy Wu are with the Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Email:{zxyan,hanzheng,cathywu}@mit.edu

collisions with all previously planned trajectories. Conflict-based Search (CBS) [23] is a seminal solver which relies on backtracking tree-search to resolve pairwise agent collisions, and Priority-based Search (PBS) [17] is a scalable extension of CBS, albeit suboptimal and incomplete. We derive significant algorithmic insights from these works.

Recent methods have aimed at improving the solution quality [8], [13] and completeness [14], [18], [20] under large-scale settings with up to thousands of agents.

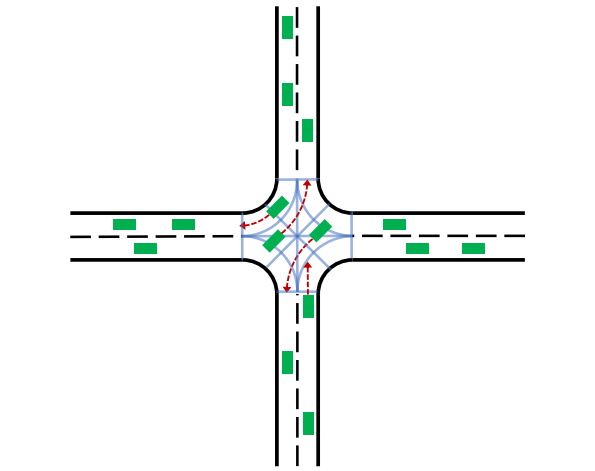
C. Continuous MAPF and Multi-robot Motion Planning

As classical MAPF is discrete time and space, continuous settings may be discretized for application of MAPF algorithms [7]. Recent continuous MAPF works have investigated planning with continuous time directly [1], [2], [9], gut require simplified agent kinematics such as constant speed along graph edges. Relatedly, [15] applies MAPF to intersection traffic settings with unbounded acceleration. Finally, works in multi-robot motion planning [11], [19] have applied sampling based methods like probabilistic roadmaps [10] to plan over settings with continuous 2D space and time. As traffic systems typically contain well-defined lanes, formulating our problem with continuous 2D space is unnecessary.

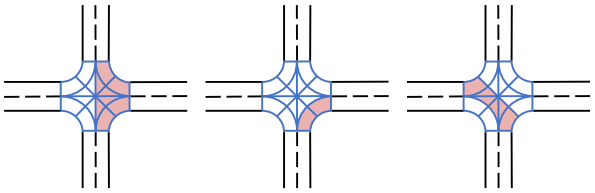
III. Problem Formulation

We formulate the cooperative driving problem at a single intersection, though this formulation is applicable to any single-junction traffic scenario (e.g. highway merging [30]). Consider a four-way intersection with directions $i \in \mathcal{D} = \{1, 2, 3, 4\}$; along each direction, a single *entering* lane (towards intersection) and *exiting* lane has length ℓ_{lane}. Vehicle routes $r = (i, j) \in \mathcal{R} = \mathcal{D}^2$ are considered, and a vehicle k with length ℓ_k traveling along route r passes the intersection from direction i to direction j, either heading straight or making a left- or right-turn. If space is available, vehicle k enters the system from an entering lane at a deterministic rate λ_i (veh/hr/lane) with initial speed v_0, its route is sampled according to $r_k \sim P(r = (i, \cdot))$ to account for different turn probabilities. Towards collision avoidance, we design a division of the intersection into 16 reservation subzones $z \in \mathcal{Z}$ (Fig. 1), which may only be occupied by one vehicle at a given time, based on geometries of crossing vehicle routes. While the subzone design in [28], [29] does not permit simultaneous left turns, our design permits four turning vehicles (two left-turn and two right-turn) vehicles to pass the intersection simultaneously. Longitudinal position along route r is defined in the range $[0, \ell_r]$. The position, speed, and acceleration of a vehicle k at step t is denoted as $x_k(t)$, $v_k(t)$, and $a_k(t)$, respectively. The start and end positions of each subzone z along each passing route r are denoted as $x_{z,r,0}$ and $x_{z,r,1}$, respectively. Vehicles are subjected to maximum straight speed $\overline{v}$ and maximum turning speed $\overline{v}_{r,z} \leq \overline{v}$ in a subzone, as well as acceleration limits $[\underline{a}, \overline{a}]$. We assume perfect sensing, inter-vehicle communication, and control.

Like works before us [28], the objective at each planning step is to find the ordering of vehicles crossing the intersection which minimizes total vehicle delay, which is



(a) Intersection with reservation subzones



(b) Subzones for straight, right-, and left-turn routes

Fig. 1: **Geometry of our studied intersection.** Our algorithms are applicable to junctions in general, *e.g.* merging, as the exact geometry is encoded by the start and end positions of subzones along vehicle routes, $x_{z,r}$.

defined as the difference between travel time $\sum_k t(x_k \geq \ell_{r_k})$ and minimum travel time $\sum_k \underline{t}(x_k \geq \ell_{r_k})$ absent of other vehicles; we use the notation $t(x_k \geq x)$ to denote the first time such that $x_k \geq x$. The crossing order is a partial ordering which defines precedence relationships for vehicles whose routes cross the same reservation subzone, but not vehicles whose routes do not overlap. For vehicles k and k', let $k \prec k'$ denotes that k precedes k' in the crossing order. For a vehicle k, let $\prec k$ denote the set of all vehicles preceding k. Vehicles already passing through or moving away from the intersection do not need to be ordered.

A. Kinematic Bicycle Model

While we plan with the longitudinal 1D model of vehicles along their routes, all control inputs are translated to and executed on a kinematic bicycle model [22]. Here, the front wheels and rear wheels of the vehicle are laterally aggregated into two wheels. The control inputs are acceleration a and front wheel steering angle δ. The distance from the center of gravity to front and rear wheels is half of vehicle length ℓ_k. ψ denotes the heading. β denotes the slip angle.

$$\dot{x}_x = v\cos(\psi + \beta) \qquad \dot{x}_y = v\sin(\psi + \beta) \qquad \dot{v} = a$$
$$\dot{\psi} = \frac{v\cos(\beta)}{\ell_k}\tan(\delta) \qquad \beta = \tan^{-1}\left(\frac{\tan\delta}{2}\right) \tag{1}$$

IV. Optimal Crossing Order Search with MAPF

We extract elements of previous works on cooperative intersection crossing [28] and design algorithms for finding the optimal crossing order from a MAPF perspective: for the high-level crossing order search, our PP and OBS algorithms integrate traffic structures. For low-level subzone reservation (akin to single-agent path planning), our KATS technique schedules arrival and departure times at reservations subzones. We use the computed crossing order to plan vehicle trajectories sequentially, with trajectory optimization. We sketch our overall method in Algorithm 1.

A. Kinematic Arrival Time Scheduling (KATS)

MAPF algorithms rely on numerous calls to a single-agent path planner (typically A* or SIPP, which are fast but often require models with limited kinematics, like constant speed [9]). On the other hand, general mixed-integer trajectory optimization is expressive but cannot be directly used as a single-agent path planner due to the high computational overhead. Therefore, we refine the arrival times scheduling technique sketched by [28] into a proxy for a single-agent path planner: Kinematic Arrival Time Scheduling (KATS). KATS can be efficiently invoked by high-level planners for computing an optimal crossing order.

KATS plans the subzone arrival and departure times for a vehicle k on route r. Let $t_d(\prec k, z)$ be the latest time that vehicles preceding k occupy subzone z. Let $t_a(k,z) = t(x_k \geq x_{z,r,0})$ be the arrival time of vehicle k at subzone $z \in \mathcal{Z}(r) \subset \mathcal{Z}$ and $t_d(k,z) = t(x_k - \ell_k \geq x_{z,r,1})$ be the departure time. KATS computes the interval $[t_a(k,z), t_d(k,z)]$ for all $z \in \mathcal{Z}(r)$. The arrival time at the first subzone z_0 along route r is computed by

$$t_a(k,z_0) = \max\left\{\underline{t_a}(k,z_0),\ \max_{z\in\mathcal{Z}(r)}\{t_d(\prec k,z) - \delta t(z_0,z)\}\right\} \tag{2}$$

where the first term $\underline{t_a}(k,z_0)$ is the minimum arrival time to z_0 (independent of other vehicles) and the second term is the earliest crossing start time such that the vehicle travels at *constant* speed within the intersection and reaches every subzone after it becomes available. $\delta t(z_0,z) = \frac{x_{z,r,0} - x_{z_0,r,0}}{v_{z_0}}$ is the travel time from z_0 to z at the *maximum attainable crossing speed* $v_{z_0} \leq \overline{v}_{r,z}$. To achieve the minimum time to enter subzone z_0, the vehicle accelerates at $\overline{a}$ for as long as feasible, then travels at maximum speed $\overline{v}$ if feasible, then decelerates at $\underline{a}$ if needed to $\overline{v}_{r,z}$. A crossing order is *infeasible* if some vehicle has insufficient distance to decelerate to $v \leq \overline{v}_{r,z}$. While KATS enforces collision-free subzones, it does not detect rear-end collisions with other vehicles along the approaching and departing lanes, and thus may be overly optimistic, as discussed in Section IV-D. Thus, a crossing ordering giving a following vehicle precedence over a leading vehicle may be *feasible* but is unlikely to be optimal and will be pruned by heuristics below.

Theorem 1: If a crossing order is feasible, calling KATS in this order obtains the optimal constant-speed crossing times for all vehicles consistent with the crossing order.

Proof: (Sketch) Consider the first vehicle k in the crossing order. By construction, no other acceleration strategy besides the one above allows k to arrive at z_0 earlier than $\underline{t_a}(k,z_0)$ or with greater speed than v_{z_0} above. Therefore, arriving earlier than $t_a(k,z_0)$ either contradicts the minimum arrival time or enters some z before $\prec k$ has departed. Thus k achieves the optimal delay. By induction on crossing order, KATS obtains optimal delay for all vehicles. ■

B. Prioritized Planning (PP) with Traffic Heuristics

Naively optimizing the crossing order with PP [5] simply samples n_{orders} random orders, evaluates the total delay of each order with KATS, then return the best crossing order. As naive PP does not leverage traffic structures and performs poorly, we augment naive PP with the two pruning heuristics introduced by [28] for their MCTS-based method: **1)** When sampling a random crossing order for PP, we constrain every vehicle k to be sampled *after* its leader (vehicle in front of k) in the lane. Sampling one by one from a space κ of $\leq |\mathcal{D}| = 4$ vehicles at a time, the overall search space reduces from $O(|K|!)$ to $O(4^{|K|})$ orderings. **2)** We select $k \in \kappa$ if its minimum arrival time $\underline{t_a}(k,z)$ at each subzone $z \in \mathcal{Z}(r_k)$ is earlier than that of all other vehicles; if no vehicle satisfies this condition, we uniformly randomly sample a vehicle whose minimum arrival time is not later than all other vehicles at all subzones. We apply these intuitions to design our order-based search next.

C. Order-based Search (OBS) with Traffic Heuristics

Inspired by the PBS algorithm [17] in MAPF, we design the OBS algorithm (Algorithm 1) for searching for crossing orders. While PBS searches the space of all partial orderings, we search the space of all partial orderings *consistent with a total ordering of vehicles crossing each subzone*.

Each node of the OBS depth-first search tree corresponds to a set of vehicles K which are yet to be ordered and an ordering $\prec$ across all vehicles. We define $\kappa \subseteq K$ as the set of vehicles with no preceding vehicles in K. For two vehicles k and k', we define the operator $k \ll k'$ to denote the following property: the subzone departure times of k and all vehicles preceding k is less than the subzone arrival times of k' and all vehicles succeeding k' for every subzone $z \in (\mathcal{Z}(r_k) \cup \mathcal{Z}(r_{\prec k})) \cap (\mathcal{Z}(r_{k'}) \cup \mathcal{Z}(r_{\prec k'}))$. Intuitively, if $k \ll k'$ and $\kappa = \{k, k'\}$, then $k \prec k'$ because k and $\prec k$ crossing earlier does not delay k' or $\succ k'$. If $k \not\ll k'$, even if k departs all subzones earlier than k' arrives, we cannot let $k \prec k'$ because some vehicle preceding k departs some subzone later than some vehicle succeeding k'. If $\exists k \in \kappa\ \forall k' \in \kappa_{\neq k}$ such that $k \ll k'$, we assign precedences $k \prec \kappa_{\neq k}$, remove k from K, and update κ with the new K. Otherwise, as in PBS, we branch over the precedence of two vehicles in κ. If κ is empty, we read the crossing order from $\prec$.

We apply similar traffic heuristics to OBS as described for PP. To control the search duration, we limit the number of orders found to n_{orders} total by distributing a budget of $\lceil \frac{n_{\text{orders}}}{2} \rceil$ orders to the first child and the remaining to the second child. This strategy allows exploration to be focused

Algorithm 1 OBS-KATS

procedure COOPERATIVEDRIVING
  for $h = 0$ to H **do**
    $\mathcal{K} \leftarrow$ set of all current vehicles
    for each newly entered vehicle k, with $\mathcal{K} \prec k$ **do**
      arrival/departure times $\leftarrow$ KATS(k, $t_d(\prec k, \mathcal{Z})$)
      TRAJOPT(k, $\prec k$, arrival/departure times)
    execute next step along vehicle trajectories
    if $h \mod H_c = 0$ **then**
      vehicles on entering lanes $K \subseteq \mathcal{K}$
      crossing order $\leftarrow$ OBS(K, $t_d(\prec K, \mathcal{Z})$, n_{orders})
      for each vehicle k in crossing order **do**
        arr./depart. times $\leftarrow$ KATS(k, $t_d(\prec k, \mathcal{Z})$)
        TRAJOPT(k, $\prec k$, arrival/departure times)

procedure KATS(k, $t_d(\prec k, \mathcal{Z})$) # Section IV-A
 # k: vehicle k to plan
 # $t_d(\prec k, \mathcal{Z})$: latest subzone departure times of $\prec k$
  $t_a(k, z_0) \leftarrow$ apply Equation 2
  for $z \in \mathcal{Z}(r_k)$ **do**
    $t_a(k, z) \leftarrow t_a(k, z_0) + \delta t(z_0, z)$
    $t_d(k, z) \leftarrow t_a(k, z_0) + \frac{x_{z,r_k,1} + \ell_k - x_{z_0,r_k,0}}{\bar{v}_{r_k,z}}$
  return $\{[t_a(k, z), t_d(k, z)] \mid z \in \mathcal{Z}(r_k)\}$

procedure OBS(K, $t_d(\prec K, \mathcal{Z})$, n_{orders}) # Section IV-C
 # K: set of vehicles to obtain a crossing order for
 # $t_d(\prec K, \mathcal{Z})$: latest subzone departure times of $\prec K$
 # n_{orders}: number of orders to obtain
   # heuristic rule 1
  $\prec \leftarrow$ initial ordering of vehicles along lanes
  orders $\leftarrow$ empty list
  procedure EXPAND(K, $\prec$, n_{orders})
    # expand a search node...
    $\kappa \leftarrow \{k \in K \mid \forall k' \in K \ k' \not\prec k\}$
    while $\exists k \in \kappa \ \forall k' \in \kappa_{\neq k} \ (k \ll k')$ **do**
      $\prec \leftarrow \prec \cup \{k \prec k' \ \forall k' \in \kappa_{\neq k}\}$
      let $K \leftarrow K \setminus \{k\}$
      update $\kappa \leftarrow \{k \in K \mid \forall k' \in K \ k' \not\prec k\}$
    if $\kappa = \emptyset$ **then**
      construct order from $\prec$, append to orders
      compute delay(order)
      return 1
    $n \leftarrow 0$
    # heuristic rule 2: k is closer to the intersection
    $k, k' \leftarrow$ two vehicles $\in \kappa$ s.t. $k \not\ll k'$ and $k' \not\ll k$
    # 1st child
    run KATS on k' and any necessary $k'' \succ k'$
    if schedules for k' and any k'' are feasible **then**
      n += EXPAND(K, $\prec \cup \{k \prec k'\}$, $\lceil \frac{n_{\text{orders}}}{2} \rceil$)
    if $n = n_{\text{orders}}$ **return** n
    # 2nd child: will be skipped if $n_{orders} = 1$
    run KATS on k and any necessary $k'' \succ k$
    if schedules for k and any k'' are feasible **then**
      n += EXPAND(K, $\prec \cup \{k' \prec k\}$, $n_{\text{orders}} - n$)
    return n
  EXPAND(K, $\prec$, n_{orders})
  return $\arg\min_{\text{order} \in \text{orders}}$ delay(order)

on the shallower nodes in the tree search, where decisions are more influential than decisions deeper in the tree search.

We now prove several properties about OBS.

Theorem 2: All orders found by OBS are crossing orders, *i.e.* OBS is sound.

Proof: First note that precedence is only ever assigned between $k, k' \in \kappa$, whose members contain no precedence over each other by definition. Therefore, OBS never assigns an inconsistent precedence and is consistent with any initial precedence relations provided. As more precedences are assigned, some vehicle k must be removed from K eventually, allowing some vehicle $k' \succ k$ to join κ eventually. By induction, *every* vehicle in K must eventually be added to κ, and thus be removed from K eventually. Each vehicle removed from K has precedence over all remaining vehicles in K. Therefore, the removal order from K is a valid total ordering. At the leaf node, $\kappa = \emptyset$, so $K = \emptyset$ and all vehicles must be present in the total ordering returned. ■

Theorem 3: Given that a crossing order exists, OBS with $n_{\text{orders}} = \infty$ finds the optimal constant-speed crossing order, *i.e.* OBS is asymptotically optimal and complete.

Proof: We show that some branch of the OBS tree must reach an optimal crossing order, if one exists. A node in the OBS tree must add an optimal precedence relation along some branch. There are two cases:

1) $k \prec k'$ is added $\forall k' \in \kappa_{\neq k}$. In this case, all vehicles $k'' \in K \setminus \kappa$ are already preceded by k or preceded by some $k' \in \kappa_{\neq k}$. For the former, k must cross earlier than k'' by definition. For the latter, $k \ll k' \prec k''$ implies that, at every subzone z, the latest subzone departure time of k and all vehicles preceding k is already earlier than the earliest subzone arrival time of k' and k''. Thus, assigning k to precede all other vehicles $k'' \in K_{\neq k}$ does not delay the crossing of any k'', and giving k precedence is optimal.

2) $k \prec k'$ is added to one child branch and $k' \prec k$ to the other. This case must be optimal because either $k \prec k'$ or $k' \prec k$ is consistent with the optimal crossing order. Without loss of generality, assume that $k \prec k'$ is consistent with optimal, then KATS must find that replanning arrivals times for k' and k'' is feasible because $\prec$ only has a subset of the precedence constraints of the optimal crossing order.

As every non-leaf OBS node adds at least one optimal precedence along some branch, OBS must reach the optimal leaf node because there are at most $|K|^2$ possible precedence relations total. The leaf node corresponds to a sound crossing order, as shown earlier. Thus OBS always finds the optimal crossing order and is complete. ■

D. Trajectory Optimization

With the total crossing ordering of vehicles, we obtain trajectories for each vehicle one-by-one, accounting for the positions of all previously planned vehicles and obeying the scheduled arrival times at subzones. KATS may be overly optimistic and inconsistent with trajectory optimization as KATS does not prevent collision between vehicles outside the intersection. Thus, following the scheduled times precisely may be infeasible. To ease infeasibility, we **1)** incrementally

delay the scheduled time constraint until feasible **2)** allow all vehicles to exceed the turning speed limit except at the midpoint of a turn, which allows a vehicle to decelerate into a turn and accelerate out of a turn. Given planning horizon T_p, trajectory optimization for each vehicle is formulated as follows and optimized with a discretization $\mathrm{d}t$:

$$\begin{aligned} &\max_{x(t),v(t)} \int_0^{T_p} v(t)\mathrm{d}t \quad \text{s.t.} \\ x(0) = 0 \qquad & v(0) = v_0 \qquad 0 \le v(t) \le \overline{v} \\ & \underline{a} \le \frac{v(t) + v(t+\mathrm{d}t)}{\mathrm{d}t} \le \overline{a} \\ & x(t+\mathrm{d}t) - x(t) = \frac{v(t) + v(t+\mathrm{d}t)}{2} \\ \underline{x} \le x(t) \le \overline{x} \qquad & v(t_{mid}) \le \overline{v}_{r,z} \end{aligned} \tag{3}$$

where $v(t_{mid})$ is the speed at the midpoint crossing time, $\overline{x}$ is the maximum safe position of a vehicle given its subzone arrival times and leading vehicles trajectories on both the entering and exiting lane, and the minimum position constraint $\underline{x}$ ensures that the vehicle departs a subzone on schedule. To obtain the steering angles along a route, we utilize a PID controller tracking the center of the route.

E. Why is Crossing Order Useful?

We acknowledge that the optimal arrival times *consistent with an optimal crossing order* does not necessarily imply optimal arrival times *in general* for minimizing delay. Indeed, similar to observed by [17], optimal arrival times may not be consistent with any crossing order. An example can be obtained by manipulating our vehicle and subzone geometries. Let the intersection be a 10 by 10 grid of square subzones, and let each vehicle be the size of one subzone. One vehicle approaches the intersection along each of the four directions, symmetrically. Clearly, the optimal arrival times is obtained by simultaneously allowing all four vehicles pass the intersection. However, these arrival times are not consistent with any crossing order, because each vehicle enters some subzone before another vehicle. With an optimal crossing order of [*up*, *right*, *down*, *left*], only the first three vehicles can enter at the same time, and *left* waits for *up* to finish crossing before entering their shared subzone.

Nevertheless, since using trajectory optimization as a single-agent path planner is not practical, MAPF algorithms tend to use path planners on simplified kinematics instead, as we do with KATS, resulting in a mismatch between the trajectories planned with simplified kinematics and ones planned with trajectory optimization. Obtaining a crossing order allows us to plan trajectories with complex kinematics according to the crossing order, adding delays when necessary to ease infeasibility due to the mismatch before planning subsequent vehicles. On the other hand, while a classical MAPF algorithm may find the optimal symmetric solution for the described example in simplified kinematics, a mismatch with trajectory optimization may occur resulting in infeasibility, which cannot be resolved by adding delays as doing so may conflict with other vehicles' trajectories. Therefore, crossing orders may be more robust to model mismatch between the kinematics used in MAPF and the kinematics used in trajectory optimization.

V. Default Experimental Setup

We modify HighwayEnv [12] to simulate the system with discretization $\mathrm{d}t = 0.1$s for $H = 1000$ timesteps. Fig. 1 illustrates subzone geometries. We set arrival rate $\lambda = 1500$veh/hr/lane with initial speed $v_0 = 5$m/s. Crossing order computation occurs every $H_c = 100$ steps. Each vehicle is planned for a horizon T_p which is sufficient for it to reach the end of its route. Maximum speed is $\overline{v} = 13$m/s, with $\overline{v}_{r,z} = 6.5$m/s on left turns and $\overline{v}_{r,z} = 4.5$m/s on right turns. A vehicle goes straight, turns left, and turns right with 60%, 20%, and 20% chance, respectively. Each lane has width $w_{\text{lane}} = 4.5$m and length $\ell_{\text{lane}} = 250$m. Each vehicle has length $\ell_k = 5$m and width 2m. Each intersection is a square with edge length $5w_{\text{lane}}$. The left turn radius is $3w_{\text{lane}}$ and the right turn radius is $2w_{\text{lane}}$. Vehicles collide when their bounding boxes overlap; for verifying algorithmic correctness, we do not add any temporal or spatial padding around each vehicle. We run all settings on 100 environment seeds, where we quantify the 95% confidence interval of the mean with bootstrap sampling. All methods are implemented in Python since KATS is very fast (around 10000 calls/s), unlike single-agent path planners for classical MAPF settings, which are often implemented in C++ for efficiency. Trajectory optimization uses CVXPY [3].

VI. Experimental Results

We demonstrate the effectiveness of OBS against the FIFO order [4], MCTS [28], [29], and our own PP on various intersection configurations. All methods use KATS. As no code was provided, we implement MCTS to the best of our abilities, with the same traffic heuristics as PP and OBS.

A. Delay vs Crossing Order Computation Overhead

In Fig. 2, we measure the average vehicle delay as a function of the computation overhead of $n_{\text{orders}} \in [2^0, 2^{14}]$ for PP, $n_{\text{orders}} \in [2^0, 2^{13}]$ for OBS, and $n_{\text{simulations}} \in [2^0, 2^9]$ for MCTS. We observe that OBS is significantly stronger than PP, which is still significantly stronger than MCTS. We note that 10s per crossing order computation is a very long computation time and much longer than practical for deployment; the previous work in cooperative driving [28] plans for around 0.1s, albeit with C++. With 10s of computation, the corresponding throughputs for the FIFO, MCTS, PP, and OBS configurations are 1740, 2050, 2080, and 2160veh/hr with confidence interval of ±20veh/hr.

Interestingly, though the same traffic heuristics are used, the best solution quality of PP and MCTS is similar to the worst solution quality for OBS, obtained with $n_{\text{orders}} = 1$ and orders of magnitude less computation. We initially conjectured that the early plateauing performance of PP and MCTS may be due to the use of traffic heuristics, which may prevent finding the optimal solution. As such,

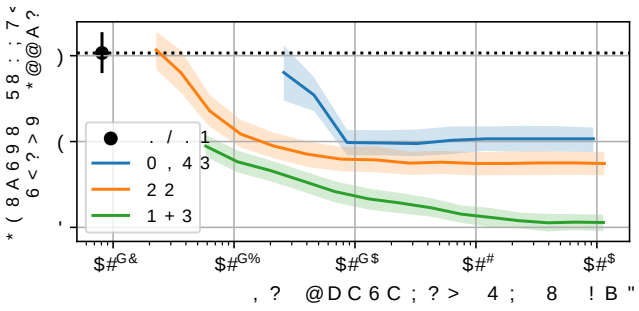


Fig. 2: **Delay vs computation time** per crossing order replan.

we attempt to disable heuristic rule 2 for selecting the first 10 vehicles of the ordering; however, we find that the performance is significantly worse. For example, doing so for PP with computation times of 0.1s and 2s per crossing order result in average delays of 8.4s and 6.0s, respectively, both significantly worse than PP with heuristics always enabled. Thus, the difference in performance of the methods is not due to the traffic heuristics used. Rather, OBS seems to have a significant algorithmic advantage by gracefully handling partial orderings rather than searching for total orderings.

B. Robustness to Intersection Configurations

In Table I, we probe the robustness of our method under different conditions by varying the arrival rates λ, lane lengths ℓ_{lane}, turning speeds $\overline{v}_{r,z}$ for [straight, right- and left-turn], and control horizon H_c. All methods are run for similar time, around 0.1s per crossing order. We find that the relative performance of all methods are consistent across configurations. The effects of arrival rate and turning speed are intuitive, so we focus on the other configurations.

For the short lane length $\ell_{\text{lane}} = 50$m, we observe very little gap between MCTS, PP, and OBS. This is likely due to the much smaller search space, as a 50m lane typically contains around 3 to 4 vehicles per lane, while longer lanes contain significantly more vehicles. We see that gaps between different methods increases with the problem complexity.

Regarding the control horizon H_c, we observe that more frequent replans is actually slightly harmful for FIFO. At each replan, the crossing order stays constant for FIFO, but the arrival times are updated by KATS and the trajectories replanned. Since there is a mismatch between KATS and trajectory optimization, arrival times planned by KATS at step h deviates from those at step $h - H_c$, shifting the constraints for trajectory optimization. This mismatch affects search-based methods as well, resulting in lowest delay at $H_c = 100$; for $H_c = 200$, delay is increased due to insufficient replans.

C. Delay vs Crossing Geometry

In Fig. 3, we examine the delay for each crossing geometry (left-turn, straight, and right-turn). While OBS significantly reduces delay for routes with all geometries, it especially reduces the delay (compared to other methods) for the straight route through the intersection, which permits the highest crossing speed. For example, while the right-turn delay is higher than the straight delay for other methods, the straight delay is lower for OBS. While less apparent, similar effect can be seen for the left-turn delay. We conjecture that this OBS behavior may be due to two reasons: **1)** the delay of a vehicle going straight is the greatest if the vehicle is forced to wait at the intersection, so OBS may prioritize straight-moving vehicles **2)** a straight crossing takes the least amount of time and is less disruptive.

TABLE I: **Average delay (s) vs intersection configurations**

Arrival Rate λ (veh/hr)	**FIFO**	**MCTS**	**PP**	**OBS**
1000	6.6	4.0	3.6	**3.2**
1500	9.6	6.0	5.6	**4.7**
2000	11	7.5	7.0	**6.0**
[500, 2000, 1000, 1200]	6.9	4.3	4.0	**3.3**
Lane Length ℓ_{lane} (m)				
50	3.4	2.8	2.7	**2.7**
100	7.5	4.8	4.6	**4.1**
250	9.6	6.0	5.6	**4.7**
500	16	10	9.0	**7.8**
Crossing Speed $\overline{v}_{r,z}$ (m/s)				
[13, 6.5, 4.5]	9.6	6.0	5.6	**4.7**
[13, 13, 13]	3.4	2	1.8	**1.6**
Control Horizon H_c (dt=0.1s)				
25	12	8.1	7.4	**6.3**
50	10	6.7	6.1	**5.3**
100	9.6	6.0	5.6	**4.7**
200	8.7	6.5	6.4	**5.8**

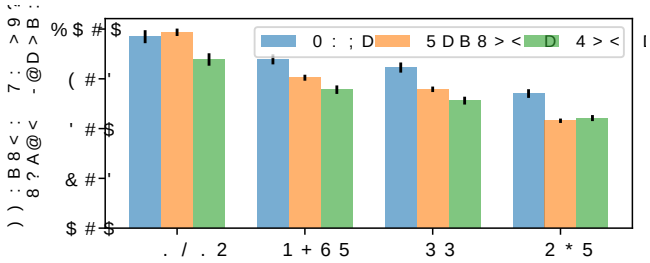


Fig. 3: **Delay along entire route vs crossing geometry**

VII. Conclusions

In this work, we seek to bridge the gap between the robotics community and the control / intelligent transportation communities. Future directions could identify other settings in traffic and robotics where the crossing order may be helpful, as well as extending the proposed algorithm to mixed traffic settings where the stochasticity of human driving behavior must be addressed. We also hope that additional insights from the robotics community may guide future algorithms for coordinating CAVs in large-scale and general traffic scenarios.

Acknowledgment

This work was supported by the National Science Foundation (NSF) CAREER award (#2239566) and the MIT Amazon Science Hub.

REFERENCES

[1] A. Andreychuk, K. Yakovlev, E. Boyarski, and R. Stern, "Improving continuous-time conflict based search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 220–11 227.

[2] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

[3] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[4] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[5] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 477–521, 1987.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[7] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[8] T. Huang, J. Li, S. Koenig, and B. Dilkina, "Anytime multi-agent path finding via machine learning-guided large neighborhood search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 9368–9376.

[9] K. Kasaura, M. Nishimura, and R. Yonetani, "Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2d roadmaps," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 494–10 501, 2022.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[11] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 494–13 499.

[12] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[13] J. Li, Z. Chen, D. Harabor, P. Stuckey, and S. Koenig, "Anytime multi-agent path finding via large neighborhood search," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[14] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "Mapf-lns2: fast repairing for multi-agent path finding via large neighborhood search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 10 256–10 265.

[15] J. Li, E. Lin, H. L. Vu, S. Koenig *et al.*, "Intersection coordination with priority-based search for autonomous vehicles," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 578–11 585.

[16] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 272–11 281.

[17] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7643–7650.

[18] K. Okumura, "Improving lacam for scalable eventually optimal multi-agent pathfinding," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[19] K. Okumura and X. Défago, "Quick multi-robot motion planning by combining sampling and search," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[20] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[21] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5628–5635.

[22] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[23] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[24] D. Silver, "Cooperative pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 1, 2005, pp. 117–122.

[25] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[26] B. Van Arem, C. J. Van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, pp. 429–436, 2006.

[27] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for mixed autonomy traffic," *IEEE Transactions on Robotics*, 2021.

[28] H. Xu, C. G. Cassandras, L. Li, and Y. Zhang, "Comparison of cooperative driving strategies for cavs at signal-free intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7614–7627, 2022.

[29] H. Xu, Y. Zhang, L. Li, and W. Li, "Cooperative driving at unsignalized intersections using tree search," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4563–4571, 2019.

[30] Z. Yan, A. R. Kreidieh, E. Vinitsky, A. M. Bayen, and C. Wu, "Unified automatic control of vehicular systems with reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 789–804, 2022.

[31] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 1443–1449.

[32] Y. Zhang and C. G. Cassandras, "A decentralized optimal control framework for connected automated vehicles at urban intersections with dynamic resequencing," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 217–222.