IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 June 2024, accepted 29 July 2024, date of publication 29 July 2024, date of current version 6 August 2024.
10.1109/ACCESS.2024.3434552
Digital Object Identifier 10.1109/ACCESS.2024.3434552

RESEARCH ARTICLE

Reinforcement Learning Aided Sequential Optimization for Unsignalized Intersection Management of Robot Traffic

机器人交通无信号交叉路口管理的强化学习辅助序贯优化

NISHCHAL HOYSAL G [1], (Student Member, IEEE), AND PAVANKUMAR TALLAPRAGADA [1,2], (Member, IEEE)

[1]Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science at Bengaluru, Bengaluru 560012, India
[2]Department of Electrical Engineering, Indian Institute of Science at Bengaluru, Bengaluru 560012, India

Corresponding author: Nishchal Hoysal G (nishchalg@iisc.ac.in)

This work was supported in part by the Robert Bosch Centre for Cyber Physical Systems (RCPS), Indian Institute of Science (IISc), Bengaluru, India.

ABSTRACT We consider the problem of optimal unsignalized intersection management, where it is desired to find a safe and optimal sequence in which robots (modelled as double integrators with control input constraints) can cross the intersection. This problem involves repeatedly solving a mixed integer program (with robot acceleration trajectories as decision variables) with different parameters, for which the computation time using a naive optimization algorithm scales exponentially with the number of robots and lanes. Hence such an approach is not suitable for real-time implementation. In this paper, we propose a solution framework that combines learning and sequential optimization. In particular, we propose an algorithm for learning a shared policy that given the traffic state information, determines the crossing order of the robots. Then, we optimize the trajectories of the robots sequentially according to that crossing order. This approach inherently guarantees safety at all times. We illustrate the performance of this approach, using extensive simulations. Our approach, on average, significantly outperforms the heuristics from the literature in various metrics like objective function, weighted average of crossing times and computation time. For example, in some scenarios, we have observed that our approach offers up to 150% improvement in objective value over the first come first serve heuristic. Even on untrained scenarios, our approach shows a consistent improvement (in objective value) of more than 30% over all heuristics under consideration. We also show through simulations that the computation time for our approach scales linearly with the number of robots (assuming all other factors are constant). We further implement the learnt policies on physical robots with a few modifications to the solution framework to address real-world challenges and establish its real-time implementability.

INDEX TERMS Robot coordination, deep reinforcement learning, autonomous intersection management, warehouse automation.

I. INTRODUCTION

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information see https://creativecommons.org/licenses/by-nc-nd/4.0/

Note that optimal trajectories inherently determine an optimal crossing order. Solution methods to such a problem inherently involve repeatedly solving mixed integer programs, for which the computational complexity of naive optimization methods scales very badly with the number of robots and lanes. As a result, they are not suitable for real-time implementation. In this work, we propose a learning based solution framework to get safe, near-optimal solutions in real-time, a combination that is not addressed in any single framework in the literature.

A. RELATED WORK

Several researchers have been working on problems related to optimal control and automated vehicles [redacted] and using different methods and tools over the years. Surveys in [1], [2], [10], [11], and [12] give a detailed description of the recent literature. Here we focus on the optimal unsignalized intersection management problem, which is a very common formulation in the area. While some formulate this problem as a mixed integer linear program [3], [13], [14], [15], others consider a model predictive control approach [18] and some formulate it as a nonlinear problem, and solve using genetic algorithms [redacted] [5], [6], [redacted], while typically the optimization goal is to minimize the cumulative travel times or maximize the cumulative distance covered by all the robots, recent works like [7], [16], [17], and [18] also take energy consumption into account while computing the optimal trajectories. Works like [9] and [19] discretize the space and use A^* like algorithms to get collision free optimal trajectories for the participating agents.

As a naive or generic optimization method for [redacted], these problems scale very badly with the number of robots and lanes. There is also [redacted] proposed [redacted] simpler solution methods with [redacted], [22], [redacted] to reduce the computational effort. To achieve computational simplicity, [23] describe a general idea of a priority based, motion planning for coordination among multiple agents, but assuming the knowledge [redacted] some pre-assigned priorities to each agent. Similarly, [redacted] with prioritized planning idea, some other works, such as [24] and [25], use model based heuristic to decide the [redacted] sequence and solve the [redacted] optimization problem, precisely that order. [redacted] the trajectories involved robots in an intersection. Work [redacted] uses a similar approach for the ramp merging problem. While some works like [26], [27] consider finding optimal and safe trajectories to robots assuming a priority (e.g. first-in first-out) rule, some like [28], [29] propose an optimization problem to get such priority rule and solve another optimization problem (in receding horizon control framework) to get trajectories. Work [30] proposes a two-stage optimization method combining the discrete tile and conflict point methods. In [31], optimization based methods [redacted] proposed [redacted] location uncertainties of agents. Other approaches to the problem include auction based methods [32], [33] and first come first serve based reservation of regions against time [19]. Work in [33] proposes a game-theoretic social-welfare optimal auction strategy to decide the crossing order. Others [4], [34] construct/use a priority graph to get feasible solutions and conflict resolution. Work [35] proposes an algorithm which improves [redacted] solutions. Work [36] [redacted] control problem and computes the priority/crossing order using a set of features associated with the robots. Then the robots solve an optimal control problem sequentially to obtain their trajectories.

Some works like [redacted] and [38] take a new approach and use gaussian process to get multi-agent trajectories and motion plans. Specifically, [36] and [37] use factor graph methods to improve computation time.

The [redacted] papers [redacted] consider only a first order kinematic model for the robots and the learnt policy gives the position trajectories of the robots. Works like [46], [45], and [47] propose multi-agent learning methods to approach the intersection management problem. Reference [38] proposes a learning based solution for optimizing robots trajectories [redacted] deviations of robots from some nominal trajectories. Work [48] [redacted] platoons of vehicles and make them pass through the intersection. In other multi-robot applications, there are works like [49] and [50] that learn path planning policies for navigation through narrow passages or hallways. Other works such as [51] present RL algorithms for general multi-robot trajectory coordination for purely first order kinematic robots. The core idea of our work has similarities to [52], [53], [54], and [55], which use RL methods to improve the efficiency of solving a parametrized combinatorial or mixed-integer optimization problem, for which the parameters are revealed online.

Our [redacted] is [redacted] warehouse automation. Prioritized/sequential planning seems to be a popular approach since it is both scalable and gives near-optimal solutions. Some works like [23], [56], [57], [58], [redacted] and [59] assume the knowledge of some priorities over agents negotiating a path conflict (at an intersection), while [60] uses first come first serve policy for conflict resolution.

B. DRAWBACKS OF CURRENT LITERATURE AND MOTIVATION

For computational scalability reasons, [redacted] in the [redacted] in that order (prioritized/sequential planning). Scheduling is done using various techniques like auctions [32], job scheduling mechanisms [21] or some heuristics [14], [24],

[redacted]. The problem with this approach is that the scheduling is dissociated from the trajectory generation and may lead to sub-optimal trajectories. Moreover, optimal policy for scheduling may be highly dependent on the type of intersection and other settings. Our past work [18] has a similar solution framework as in this paper. However, [18] contains no method or learning algorithm for obtaining a policy for determining crossing orders.

To the best of our knowledge, there are few works like [28], [29], and [61], which make a systematic attempt at assigning priorities to the involved agents. Existing approaches to find a good set of priorities include iterative search algorithms [61] or solving an optimization problem [28], [29], which themselves are time consuming.

Although [redacted] RL to directly generate trajectories of the robots, the approach inherently cannot guarantee safety in a deterministic way. Further, these works only consider first order kinematic models of the robots, whereas we consider a double integrator model for the robots. Our proposed approach is very similar in spirit to [52], [53], [54], [55], which use learning methods to solve combinatorial optimization problems. However, these papers are in very different applications. Additionally, all these works provide algorithms that learn policies for a single agent rather than multiple agents.

1) MOTIVATION

Scalability, provable safety and efficiency are important aspects for multi-robot systems like in autonomous warehouses since they directly affect the capital/operational cost. In light of the drawbacks of the existing literature, it is important to address all these three aspects in a combined manner. In this work, we address scalability, provable safety and efficiency together by proposing a way to combine learning and optimization methods. In particular, we use the optimization framework to ensure provable safety and use learning framework to improve scalability, while ensuring near-optimality.

C. CONTRIBUTIONS

The following are the main contributions of this paper.

- We propose an algorithm, which [redacted] policy for determining the order in which the robots cross an isolated intersection given certain features of the traffic. The algorithm learns a policy that is shared among all the robots. We use this policy over a solution framework that combines learning and online optimization for sequential optimization of trajectories for each robot in the sequence as constraints. This way, we ensure scalability as well as safety at all times, both during training and deployment. The framework implicitly handles continual stream of robot traffic.

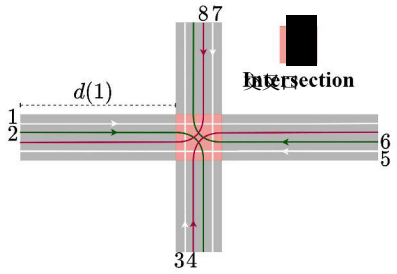


FIGURE 1. A schematic of an example intersection and the region of interest (RoI) with 8 lanes.

- Through extensive simulations, we establish that the proposed framework solves the intersection management problem in real-time and provides near-optimal solutions. In general, the performance of the trained policies is significantly superior compared to many heuristics from the literature. For example, in some scenarios, the learnt policies on average provide up to 40% improvement in terms of the considered objective function and up to 20% improvement in terms of the average time to cross when compared against the first come first serve heuristic. Also, in many empirical simulations, the trained RL policy outperforms all the compared heuristics by 40% in terms of the objective function and by 20% in terms of average time to cross weighted by robot priorities for a considerable range of arrival rates.
- We propose some adaptations to the underlying solution framework to address real-world implementation challenges like tracking errors and delays associated with communication and computation. The framework with these adaptations is deployed on physical robots to establish the real-time implementability of the framework i.e., communication and computations delays in the deployed framework are smaller than the spare-time available in the framework.

II. UNSIGNALIZED INTERSECTION MANAGEMENT AND PROBLEM SETUP

In this section we first describe the problem, then discuss the intersection management problem and discuss some challenges in solving it in real-time. Then, we pose a problem of learning computationally efficient and near-optimal policies that can be utilized for safe intersection management in real-time. We first describe the intersection geometry, robots and notation.

We consider an isolated region of interest (RoI), denoted as $\mathcal{R} \subset \mathbb{R}^2$, consisting of M number of fixed lanes. Let $S(l)$ for $l \in \{1,\dots,M\}$ be the set of points in RoI $\mathcal{R}$ that form the lth lane. With a slight abuse of notation, we assume that each lane $S(l)$, for all $l \in \{1,\dots,M\}$, is a [redacted] set. The *intersection* in the RoI $\mathcal{R}$ is [redacted]

[redacted]

即 R 中至少属于两条车道的所有 RoI 的集合。我们假设交叉口 Z 是一个连通集，并且每条车道以进入、经过并离开交叉口 Z 一次。

Figure 1 shows one such configuration, where the intersection Z with the RoIs. 对于车道 l，我们将其通往交叉口的长度记为 $d(l)$。

A. DEFINITIONS OF ROBOT RELATED VARIABLES AND PARAMETERS

We assume that the robots travel only along the fixed lanes inside the RoI, i.e., they do not change lanes. We denote the set of robots under consideration by V. We denote the length of robot $i \in V$ by L_i. Further, we denote the time of entry into the RoI and the time of exit from the RoI of robot i by $t_i^{\mathcal{E}}$ and $t_i^{\mathcal{X}}$, respectively. We use $x_i(t)$ and $v_i(t)$ to denote the longitudinal position (specified by the front of the robot) along its lane and longitudinal velocity of robot i, respectively, at time $t \geq t_i^{\mathcal{E}}$. We use $\dot{x}_i(t)$ and $\dot{v}_i(t)$ to denote the time derivative of $x_i(t)$ and $v_i(t)$ evaluated at time t, respectively. On each lane, we set up the coordinates such that for a robot $i \in V$ we have, $x_i(t_i^{\mathcal{A}}) = -d(q_i)$ and $x_i(t_i^{\mathcal{E}}) = 0$. To capture if two robots $i, j \in V$ are on lanes that intersect in the intersection area, we define a variable $c(i, j)$ as

$$c(i,j) = \begin{cases} \quad \\ \quad \end{cases} \tag{1}$$

where % denotes the empty set, $c(i, j)$ is equal to 0, 1 and −1 if and only if the robots i and j are on the same lane, on different lanes that do not intersect and on different lanes that intersect, respectively. See Table 1 for a comprehensive list of important notations and their definitions used in this paper.

B. CONSTRAINTS

We assume that the robots follow the double integrator dynamics with longitudinal acceleration as their input, i.e.,

$$\quad \tag{2}$$

where $x_i(t)$ and $v_i(t)$ are the time derivatives of position and velocity of robot i at time t, $u_i(t)$ is the acceleration input to the robot i at time t. $t_i^{\mathcal{A}}$ is the arrival time of robot i.

We assume that the robots' velocities and accelerations are bounded, i.e.,

$$\quad \tag{3}$$

where $\underline{u} < 0$, $\bar{u} > 0$ and $\bar{v} > 0$, $\forall i \in V$.

To avoid collisions inside the intersection between robots on pairs of conflicting or incompatible lanes (pairs of lanes which intersect), we impose the constraint

$$\quad \tag{4}$$

where $t_i^{\mathcal{E}}$ and $t_i^{\mathcal{X}}$ are the entry and exit times of robot $i \in V$ into the intersection, respectively. Note that the constraint (4) is combinatorial.

Next, we formulate the rear-end safety constraint. In intersection management literature, it is common to enforce that any two robots on a lane are separated by at least a fixed distance. However, note that in situations like warehouses, there is always a chance of robot or communication or coordination failure. Hence we propose a conservative rear-end safety constraint, which ensures the existence at all times, of a feasible input (acceleration) trajectory for a robot to come to a stop avoiding collision with its predecessor. Following is the definition of the safe-following distance between two robots.

Definition 1 (Safe-following distance): For a pair of robots $i, j \in V$ on the same lane with j following i, i.e., $c(i, j) = 0$ and $t_i^{\mathcal{A}} > t_j^{\mathcal{A}}$, a quantity $\mathcal{D}(i, j, t)$ is a safe-following distance at time t if $x_i(\tau) - x_j(\tau) \geq \mathcal{D}(i, j, t), \forall \tau \geq t$, where $x_i(\tau)$ and $x_j(\tau)$ are the positions of robots i and j at time τ following the maximum braking input (MBM) from time t onwards.

Lemma 1 (Safe-following distance): For a pair of robots $i, j \in V$ on the same lane with j following i, i.e., $c(i, j) = 0$ and $t_i^{\mathcal{A}} > t_j^{\mathcal{A}}$. Then, the continuous function

$$\quad \tag{5}$$

provides a safe-following distance for the pair of robots i and j.

Using Lemma 1, we formulate the rear-end safety constraints

$$\quad \tag{6}$$

C. CONTROL OBJECTIVE

Consider a set of robots V arriving into the RoI at different times during the time interval of interest. Since the robots arrive at different times, the RoI serves different numbers of robots in different time intervals. For the robots in V, we want to minimize their cumulative time taken to cross the intersection weighted by their priorities, i.e.,

$$\quad \tag{7}$$

where $r_i > 0$ is a weight, or priority, for robot i. Note that $u_i(\cdot)$, for any $i \in V$, is the decision variable

TABLE 1. Table of some important notations.

Notation	Definition
	RoI and intersection geometry related
$\mathcal{R}$	Region of interest (RoI)
M	Number of lanes
$\mathcal{P}_m(\cdot)$	Set of points in RoI $\mathcal{R}$, parameterized by m^{th} lane
$\mathcal{I}$	Intersection
$\ell(\cdot)$	Length of the approach or the lane in RoI
	Robot related
q_i	Lane of robot i
$x_i(t), v_i(t), u_i(t)$	Position, velocity and acceleration (input), respectively, of robot i along its lane at time t
$\dot{x}_i(t), \dot{v}_i(t)$	Time derivatives of position and velocity, respectively, of robot i along its lane at time t
$t_i^{\mathcal{A}}, t_i^{\mathcal{E}}, t_i^{\mathcal{X}}$	Times of entry into RoI, entry into intersection and exit from intersection, respectively, of robot i
$\bar{v}$	Upper bound on robot velocities
$\underline{u}, \bar{u}$	Lower and upper bounds, respectively, on the (input) acceleration for the robots
$c(i,j)$	Indicator variable to represent lane compatibility of robots i and j, see (1)

机器人 i 的车道。机器人 i 沿其车道在时刻 t 的位置和速度（输入）的时间导数，分别为 $x_i(t), v_i(t), u_i(t)$。机器人 i 进入 RoI、进入交叉口和离开交叉口的时刻，分别为 $\dot{x}_i(t), \dot{v}_i(t)$。机器人速度的上界 $\bar{v}$。机器人加速度（输入）的下界和上界，分别为 $\underline{u}, \bar{u}$。用于表示机器人 i 和 j 的车道兼容性的指示变量，见 (1)

$t_i^{\mathcal{A}}, t_i^{\mathcal{E}}, t_i^{\mathcal{X}}$

$\bar{v}$

$\underline{u}, \bar{u}$

$c(i,j)$

$i \in V$。因此，问题 (7) 是一个时变最优控制问题，其中 V 中的每个机器人都有不同的时间范围。这对于处理在不同时间到达的机器人流尤其困难。因此，我们针对机器人流 V 的交叉口管理，构建如下代理最优控制问题。

Hence, Problem (7) is a time-varying optimal control problem, with each robot in V having a different time horizon. This is particularly difficult to handle for a stream of robots that arrive at different times. Hence, we formulate the following proxy optimal control problem for intersection management for a stream of robots V.

t_i

Here, T_h is a sufficiently long time horizon. We assume that the arrival times, $t_i^{\mathcal{A}}$, of robots $i \in V$ are known, but the number of robots in V is unknown, before their arrival. Note that Problem (8) is a proxy for Problem (7), and we use the formulation (8) as it has the advantage of a fixed time horizon T_h for each robot.

Remark 1 (Challenges in Solving Problem (8)): There are several challenges in solving Problem (8). They mainly stem from the following factors.

(i) At any given time, only finitely many robots are present in the RoI, and the future arrival of robots is not available.

(ii) Constraint (4) is combinatorial, which makes Problem (8) a mixed integer program. A naive optimization approach scales exponentially with the number of robots and lanes. This is problematic because intersection management problem is both time and safety critical.

In order to counter these challenges, we use a data-driven approach similar to the one in [18], wherein data obtained through simulations was used to manually tune the policy. In the current work, we propose algorithms that learn near-optimal policies to Problem (8).

D. INFORMAL PROBLEM STATEMENT

In this work, we aim to develop learning algorithms that approximate near-optimal policies for Problem (8). While we allow the learning to be centralized, implementation of learnt policies should be distributed, i.e., the policy for each robot should depend on only the data that can be easily obtained using robot-to-robot and robot-to-infrastructure communication. The proposed solution should be applicable to a continual stream of robots arriving randomly into the RoI. Note that *any feasible solution to Problem (8) implicitly guarantees safety.*

III. SOLUTION FRAMEWORK TO ADDRESS RANDOM ARRIVAL TIMES

In this section, we present an overall framework/algorithm for solving the intersection management problem. The broad solution framework is a modified version of the one in [18]. We recap the main aspects of this framework.

As the robots arrive randomly in time, and their synchronization and coordination between robots requires planning for groups of robots at a time, we split the trajectory of each robot into two phases - *provisional phase* and *coordination phase*. Provisional phase of a robot begins when it arrives at the RoI and ends when its coordination phase begins. A coordination phase algorithm runs every T_c seconds and assigns safe and efficient trajectories for crossing the intersection to the robots in their provisional phase.

Before we discuss the specifics of this framework, we introduce some notation. Let $t_i^{\mathcal{C}}$ denote the time at which the coordination phase of robot i begins. Note that $t_i^{\mathcal{C}} = t_i^{\mathcal{A}}$ if robot i does not have a provisional phase. Let $k \in \mathbb{Z}_{\geq 1}$ denote the kth coordination phase algorithm, starting at time kT_c. Let $V(k) := \{i \in V : t_i^{\mathcal{A}} < kT_c \leq t_i^{\mathcal{C}}\}$ be the set of robots which entered the RoI before kT_c, $V_s(k) := \{i \in V(k) : t_i^{\mathcal{C}} < kT_c\}$ be the robots that entered coordinated phase before kT_c, $V_p(k) := V(k) \setminus V_s(k)$,

at the set of robots that need to be allocated coordinated phase trajectories at kT_c.

A. PROVISIONAL PHASE

Consider a robot $i \in V_r(k) \setminus V_s(k)$. To ensure that the robot does not enter the intersection before it enters the coordinated phase, we impose the constraint

$$\|p_i[t] - p_j[t]\| \geq 2r_{\text{safe}}, \quad \forall j \in V_p(k), \ t \in [kT_c, \max\{A_i(k-1)T_c\}, kT_c]$$

Let $u_i[t], t \in [\max\{A_i(k-1)T_c\}, kT_c]$ be an optimal solution of the following problem.

$$\min_{u_i[t]} \int_{\max\{A_i(k-1)T_c\}}^{kT_c} \|u_i[t]\|^2 dt$$

$$\text{s.t. } \dot{p}_i[t] = u_i[t], \quad \|p_i[t] - p_j[t]\| \geq 2r_{\text{safe}}, \ \forall j \in V_p(k), \ t \in [\max\{A_i(k-1)T_c\}, kT_c]$$

B. COORDINATED PHASE

At the time kT_c, for each $k \in \mathbb{N}$, some or all the robots in $V_r(k)$ are assigned their coordinated phase trajectories, which they start executing immediately. Ideally, we would like the coordinated phase control input trajectories for the robots in $V_p(k)$ to be an optimal solution of Problem (11), which we call as the *combined optimization* problem.

$$\min_{u_i[t]} \sum_{i \in V_p(k)} \int_{kT_c}^{A_i(k)T_c} \|u_i[t]\|^2 dt$$

$$\text{s.t. } \dot{p}_i[t] = u_i[t], \quad \|p_i[t] - p_j[t]\| \geq 2r_{\text{safe}}, \ \forall i, j \in V_p(k), \ t \in [kT_c, \max_{i \in V_p(k)}\{A_i(k)T_c\}]$$

Note that the robots in $V_s(k)$ would also appear in the constraints of Problem (11). Intersection safety constraint in Problem (11) is still combinatorial. Solving Problem (11) inherently involves picking a feasible solution with highest objective value among all the feasible crossing orders. As a result, this formulation scales exponentially with the number of robots and is not suitable for real-time implementation. We illustrate this exponential scaling of computation times through simulations in Figure 5.

Algorithm 1 presents the overall framework. It describes how, by invoking the subroutine provisional_phase, it determines the robots for which provisional phase trajectories are computed and when they start executing them. Algorithm 1 also describes when coordinated phase trajectories are computed and when they start executing them. Algorithm 1 perpetually checks for new robots entering the RoI and assigns them provisional phase trajectories as and when they arrive. Periodically, with period T_c, the algorithm computes coordinated phase trajectories for all the robots in their provisional phase. Among these robots, those that can cross the intersection with the computed trajectory start executing them and enter their coordinated phase. On the other hand, the robots that cannot cross the intersection with the newly computed trajectories, continue in provisional phase with updated provisional phase trajectories, computed as per (10).

Remark 2 (Provisional Phase Trajectories for a Robot): In busy intersections, there might be cases where the solution to Problem (11) gives a trajectory for a robot

Algorithm 1 Overall_algorithm

Inputs: Coordination Time Period T_c; Time Period T_c.

Initialize $k = 1$, $V_p(1) = \emptyset$, $V_s(1) = \emptyset$

while True **do**

  if a robot enters the RoI **then**

    Run provisional_phase for robot i

    Add robot i to $V_p(k)$

  end

  if time $= kT_c$ **then**

    Compute coordinated_phase on $V_p(k)$

    Set $V_{nc}(k)$ as the set of robots among $V_p(k)$ for which trajectories are computed for the coordinated phase do not cross the intersection

    Set $V_s(k+1) = V_s(k) \cup V_p(k) \setminus V_{nc}(k)$ and $V_p(k+1) = \emptyset$

    for $i \in V_{nc}(k)$ **do**

      Run provisional_phase for i

      Add i to $V_p(k+1)$

    end

    Set $k = k + 1$

  end

end

$i \in V_r(k)$ such that the robot does not exit the intersection by $kT_c + T_c$. This can potentially cause infeasibility of Problem (11) in subsequent coordinated phases. In order to avoid such a situation, such affected robots (robot i and its successors in the crossing order) go through the provisional phase again for the interval $[kT_c, (k+1)T_c)$. This is the reason why the time horizon for robot $i \in V_p(k)$ is $[\max\{A_i(k-1)T_c\}, kT_c]$ in Problem (10). We emphasize that the difference between provisional phase computations and coordinated phase computations is only in the computed trajectories itself (using the computed trajectories). Specifically, in Algorithm 1, the robots in $V_p(k)$ are going through coordinated phase computation and at the end of kth iteration in Algorithm 1, only the robots in $V_s(k+1) \setminus V_s(k)$ start their coordinated phase. Robots in $V_{nc}(k)$ will go through a provisional phase again. •

C. SEQUENTIAL OPTIMIZATION FOR COORDINATED PHASE

To solve the combined optimization problem in real-time, we propose a modified version of sequential optimization from [18], in Algorithm 2. The algorithm takes as input a set of quantities called *precedence indices*, $(p_i \in \mathbb{R}, \ \forall i \in V_p(k))$, which determine the crossing order. Each robot in $V_p(k)$ obtains its coordinated phase trajectory sequentially, as per the crossing order, by solving the optimization problem in (12).

In Algorithm 2, while the precedence indices of each robot in $V_p(k)$ are distinct, it is not necessary. In Step 2, $\mathcal{F}$ is the set of robots in $V_p(k)$ that do not have a coordinated phase trajectory yet. In Step 3, i^* is

Algorithm 2 Sequential_optimization

Inputs: V_Q, $\{x_i(k), v_i(k), u_i(k), p_i(k) : i \in V_Q\}$, $p_p(k)$

1 $V_Q \leftarrow V_p(k)$
2 **if** $V_Q \neq \emptyset$ **then**
3 For every $i \in V_Q$, find $\mathcal{F}_i \leftarrow \{j \in V_Q \text{ s.t. } q_j = q_i\}$
$i^* \leftarrow \arg\max_{i \in V_Q} \{p_j\}$
4 Solve (10) for i^* to obtain $u_{i^*}(\cdot)$
5 **if** i^* crosses the intersection by time $kT_c + T_h$ when executing $u_{i^*}(\cdot)$, **then**
$t_{i^*}^{\mathcal{C}} \leftarrow kT_c$
i^* starts executing u_{i^*} at $t_{i^*}^{\mathcal{C}}$
Remove i^* from V_Q
Return to line 2
6 **end**
7 **end**

Algorithm 2 Sequential_optimization

Inputs: V_Q, $\{x_i(k), v_i(k), u_i(k), p_i(k) : i \in V_Q\}$, $p_p(k)$

1 $V_Q \leftarrow V_p(k)$
2 **if** $V_Q \neq \emptyset$ **then**
3 For every $i \in V_Q$, find $\mathcal{F}_i \leftarrow \{j \in V_Q \text{ s.t. } q_j = q_i\}$
$i^* \leftarrow \arg\max_{i \in V_Q} \{p_j\}$
4 Solve (10) for i^* to obtain $u_{i^*}(\cdot)$
5 **if** i^* crosses the intersection by time $kT_c + T_h$ when executing $u_{i^*}(\cdot)$, **then**
$t_{i^*}^{\mathcal{C}} \leftarrow kT_c$
i^* starts executing u_{i^*} at $t_{i^*}^{\mathcal{C}}$
Remove i^* from V_Q
Return to line 2
6 **end**
7 **end**

V_Q 中所有车道上距离交叉口最近的机器人集合，在步骤 4 中，任意破平局后，我们从 V_Q 中选出优先级指标 p_i 最大的机器人 i^*。在步骤 5 中，$V_i^{\mathcal{F}}$ 是所有在机器人 i^* 之前已被分配协调相位轨迹的机器人的集合。如果步骤 6 中计算出的 $u_{i^*}^*(\cdot)$ 能使机器人在 $kT_c + T_h$ 之前穿过交叉口，那么机器人 i^* 就从 $t_{i^*}^{\mathcal{C}} = kT_c$ 开始执行 $u_{i^*}^*(\cdot)$，将其作为协调相位控制轨迹。随后，从 V_Q 中移除 i^*，循环继续。另一方面，如果 i^* 无法通过控制轨迹 $u_{i^*}^*(\cdot)$ 穿过交叉口，则跳出循环，i^* 及 V_Q 中其余机器人将进入另一个临时阶段，如注释 2 所述。

In Step 6 of Algorithm 2, we solve the following optimization problem.

$$\min_{u_i(\cdot)} \int_{t_i^{\mathcal{A}}}^{t_i^{\mathcal{C}}} u_i(t)^2 \, dt$$

where $t_i^{\mathcal{C}}$ is the minimum wait time, τ_i, given by,

$$t_i^{\mathcal{C}} = \min_{t \geq t_i^{\mathcal{A}}} \{t : \text{constraint (4) is satisfied}\}$$

Remark 3 (算法2的特征): Algorithm 2 has several good features, which help in achieving the design goals of scalability and real-time implementation. In a variety of simulations, we have consistently observed that the computation for Algorithm 2 scales linearly with the number of robots. We illustrate this feature in Figure 5. Algorithm 2 is an efficient version of the DD-SWA algorithm proposed in [18], as the precedence indices need not be recomputed after the trajectory optimization for each robot. The solution framework, including Algorithm 2, can be implemented in a distributed manner, as the only information required is that of the robots in $V_p(k)$ and their trajectories. We refer readers to [18] for relevant discussions.

The solution framework, including Algorithm 2, is flexible and can be used with any precedence ordering policy. In particular, we propose a reinforcement learning algorithm in the next section, which learns a policy that gives out "near-optimal" precedence indices, given some information about the traffic state.

IV. LEARNING A POLICY THAT GIVES NEAR-OPTIMAL CROSSING ORDERS

Recall that at kT_c, the beginning of the kth coordinated phase, $V_p(k)$ is the set of robots that are in the provisional phase, and V_Q is the set of robots that are in the provisional phase and have not crossed the intersection. Let $n = |V_Q|$. Algorithm 2 takes as input the precedence indices of the robots in $V_p(k)$, and tries to find the optimal solution to the combined optimization problem (11) and more generally to the original optimization problem (8). In particular, we propose a reinforcement learning algorithm for learning a policy, which

(i) can be implemented in real time, and

(ii) is shared, i.e., the same policy is used by all the robots.

Algorithm 2 的步骤 6 中，通过求解以下优化问题得到 $u_i(\cdot)$。

其中，$t_i^{\mathcal{C}}$ 是最小等待时间，计算如下，

Remark 3 (算法2的特征): 算法2具有多项良好特征，有助于实现可扩展性和实时实现的设计目标。在各种仿真中，我们始终观察到，算法2的计算量随机器人数量线性增长。我们在图5中展示了这一特征。算法2是 [18] 中提出的 DD-SWA 算法的高效版本，因为每个机器人的轨迹优化完成后，无需重新计算优先级索引。包括算法2在内的解决方案框架可以以分布式方式实现，因为所需的信息仅是 $V_p(k)$ 中机器人及其轨迹的相关信息。我们将读者参考 [18] 以获取相关讨论。

The solution framework, including Algorithm 2, is flexible and can be used with any precedence ordering policy. In particular, we propose a reinforcement learning algorithm in the next section, which learns a policy that gives out "near-optimal" precedence indices, given some information about the traffic state.

为使解决方案框架完整，我们仍需说明如何选择优先级索引 p_i。下一节将提出一种算法，用于学习一种策略，该策略根据有关交通状态的一些信息给出"近似最优"的优先级索引。

IV. 学习能给出近似最优通行顺序的策略

Theorem 1 (系统整体避免碰撞): If every robot $i \in V$ satisfies the rear-end safety constraint (6) at the time of its arrival, and its initial velocity is such that $v_i(t_i^{\mathcal{A}}) \leq \min(\bar{v}_i, \sqrt{2\underline{u}x_i(t_i^{\mathcal{A}})})$, feasibility of problems (10), (11) and (12) is guaranteed. Consequently, safety of all the robots is also guaranteed for all time under Algorithm 1.

Proof. Recall that each robot is allowed to decelerate, i.e., $\underline{u}_i < 0$. Hence due to the assumption that robot i satisfies the rear-end safety constraint (6) at $t_i^{\mathcal{A}}$, it is feasible to apply the input $u_i(t) = \underline{u}$ for all $t \geq t_i^{\mathcal{A}}$ until the robot comes to a stop before entering the intersection, making Problem (10) feasible.

Now consider Problem (11). If it is feasible, then the trajectory $u_i^*(\cdot)$ computed in Step 6 of Algorithm 2 enables the robot to cross the intersection safely (i.e., it satisfies the rear-end safety constraint (6), in addition to the other constraints), and hence Problem (11) is feasible. Otherwise, Problem (11) is infeasible, and the robot comes to a stop before entering the intersection, making Problem (12) feasible.

Note that, if Problem (10) is infeasible, then the trajectory $u_i(t) = \underline{u}$ for all $t \geq t_i^{\mathcal{A}}$ is the provisional phase trajectory for robot i, that when it goes through the coordinated phase computations, say at $t_i^{\mathcal{C}}$, the intersection safety constraint (4) is satisfied at $t_i^{\mathcal{C}}$. This inherently ensures that $t_i^{\mathcal{C}} \geq \tau_i$ is feasible (recall τ_i is the minimum wait time for i). This guarantees feasibility of problems (11) and (12). □

Theorem 1 (系统整体避免碰撞): 如果每个机器人 $i \in V$ 在其到达时刻 $t_i^{\mathcal{A}}$ 满足追尾安全约束 (6)，且其初始速度为使得 $v_i(t_i^{\mathcal{A}}) \leq \min(\bar{v}_i, \sqrt{2\underline{u}x_i(t_i^{\mathcal{A}})})$，则问题 (10)、(11) 和 (12) 的可行性得到保证。因此，在算法1下，所有机器人的安全性也将在所有时刻得到保证。

证明. 回顾一下，每个机器人都允许减速，即 $\underline{u}_i < 0$。因此，根据机器人 i 在 $t_i^{\mathcal{A}}$ 时刻满足追尾安全约束 (6) 的假设，问题 (10) 不存在可行解的唯一可能原因是违反了速度上限约束 $v_i(t) \leq \bar{v}_i$。在这种情况下，施加输入 $u_i(t) = \underline{u}$ 对所有 $t \geq t_i^{\mathcal{A}}$ 都是可行的，因为它满足追尾安全约束 (6)，并且不会违反速度上限约束 (10)。此外，假设 $v_i(t_i^{\mathcal{A}}) \leq \sqrt{2\underline{u}x_i(t_i^{\mathcal{A}})}$，确保存在可行控制轨迹，使机器人在进入交叉口前停下来，从而使问题 (12) 可行。

请注意，如果问题 (10) 不可行，则机器人 i 的协调阶段的轨迹 $u_i(t) = \underline{u}$ 对于 $t \geq t_i^{\mathcal{A}}$ 是机器人 i 的临时协调相位轨迹，当其进行协调相位计算（如在 re 时）时，它将通过交叉口而不违反交叉口安全约束 (4)。这自然确保 $t_i^{\mathcal{C}} \geq \tau_i$ 是可行（回顾一下，τ_i 是 i 的最小等待时间）。这保证了问题 (11) 和 (12) 的可行性。□

(iii) Its own attributes, i.e., 其自身的特性，即策略的输入是机器人本地可获取的信息，或通过与其相邻机器人通信获得的信息。
information available to a robot locally or through communication with its neighbouring robots.

(iv) Its own position and velocity, and the position and velocity of all robots in $V_c(k)$.

We denote the shared policy by the function $g(\cdot)$. For robot i, the input to the policy is the *feature vector* f_i that captures the state of the traffic relevant to robot i. Then, the precedence indices are

$$[\quad\quad\quad

2) ONLINE AND OFFLINE LEARNING APPROACHES

Algorithm 3 is the online learning approach to the proposed learning algorithm. We generate streams of robots using a poisson process for determining the arrival times of robots and choosing their initial velocities randomly. All these robots go through provisional and coordinated phases as described in Section III, where Algorithm 2 is used for obtaining coordinated phase trajectories. Since in Algorithm 2, we only care about the relative order of the precedence indices, we have a softmax layer (only during training) as the last layer in the RL actor network and use the $\tilde{V}$ values as the precedence or corresponding robots. We observed that having this layer leads to quicker learning since it limits the possibility of different actions leading to same crossing order. We add the corresponding state, action, reward and next state tuple to the replay buffer and update the critic, actor and the target networks as described above. Note that because of the softmax layer, the implementation of this algorithm cannot be distributed in a truly distributed manner. We use a modified Ornstein-Uhlenbeck process (with decaying variance in added Gaussian noise) as the exploration noise. Algorithm 3 can be run in an online loop continuously, gathering data and learning from it. A schematic flow of Algorithm 3 is presented in Figure 2.

Algorithm 3 Online_learning

Inputs: $V_p(k)$, $V_s(k)$, W_i, $i \in \mathcal{V}(k)$, Shared Policy g, Replay-Buffer

Initialize $\mathcal{R} = \{s, a, R, s'\}$, $\mathcal{V}(k)$, $\tilde{V}(k)$, $\mathcal{P}(k)$, $\mathcal{P}_p(k)$, $\mathcal{P}_s(k)$, $\mathcal{P}_{ps}(k)$, $\mathcal{P}_{sp}(k)$, $\mathcal{P}_{ss}(k)$, $\mathcal{P}_{pp}(k)$, $\mathcal{P}_{\tilde{V}}(k)$, $\mathcal{P}_{\tilde{V}\mathcal{P}}(k)$, $\mathcal{P}_{\mathcal{P}\tilde{V}}(k)$, $\mathcal{P}_{\tilde{V}\tilde{V}}(k)$, $\mathcal{P}_{\mathcal{P}\mathcal{P}}(k)$, $\mathcal{P}_{\mathcal{P}}(k)$, $\mathcal{P}_{\tilde{V

Remark 5 (Different [illegible] DDPG [64], MADDPG [63] and MAJA-DDPG): While DDPG, MADDPG and MAJA-DDPG are all centralized learning algorithms, DDPG learns a policy for a single agent. Both MADDPG and MAJA-DDPG learn a shared policy for multiple agents. The main difference between MADDPG and our proposed MAJA-DDPG is in the computation of the sampled gradient $\nabla_\theta J(\theta)$ (16). MADDPG uses the gradient of the critic function Q with respect to the shared actor parameters θ through the joint action (concatenated actions of all the agents), whereas MADDPG estimates the gradient of Q with respect to the shared actor parameters θ through the action of a randomly selected agent in the corresponding computation. •

Remark 6 (Features of a Robot): We consider the following quantities as features of a robot $i \in V_a(k)$: its distance from the entry to RoI $(x_i(kT_c) - d(q_i))$, its current velocity $(v_i(kT_c))$, its priority (r_i), its lane identifier (q_i), upper-bound on its velocity (v_i), upper-bound on acceleration $(\bar{a}_i)$, time since its arrival into RoI $(kT_c - t_i^a)$, its estimated minimum wait time $(\tau_i = \max\{t_j^* : j \in V_s(k)\})$, the number of robots following on its lane, and the average distance between the robots following in its lane. As can be seen, most of the features are directly measured or computed by robot i, whereas the others can be obtained by communicating with neighbouring robots. •

Remark 7 (Shared Policy): We use a neural network to capture the shared policy $g(\cdot)$, which gives the precedence index n_i of a robot $i \in V$. Figure 3 presents a schematic of this neural network. The input to this network is the features, f_i of the robot $i \in V$ ($f_i \in \mathbb{R}^{10}$ since we use 10 features for a robot as listed in Remark 6). This input is processed through two hidden layers. The first hidden layer has 4 units with ReLU activation and the other hidden layer has 2 units with linear activation. A linear combination of the outputs at the last hidden layer is taken as the output of the policy (the precedence index). •

Remark 8 (Centralized Implementation of the Shared Policy): Notice that once a policy is trained, it can be implemented either in a centralized or distributed manner, with the help of robot to infrastructure (R2I) and robot to robot (R2R) communication. For centralized implementation, a central intersection manager collects and maintains the up-to-date information about the state of all the robots in the RoI. The provisional phase for new robots and coordinated phases for existing robots can be computed in a centralized way (following Algorithm 2). Further, the updated phase, τ_i, of the vehicles through the R2I communication. For a distributed implementation, notice that all the features of a robot can be constructed using local information and can be acquired by a chain of R2R and R2I links. Also, the softmax activation in computing precedence indices can be ignored. Moreover, trajectories for a robot in the coordinated phase can also be computed locally on that robot, after all the required information for that robot-specific optimization problem being communicated using R2R and R2I communication. We assume that all communications, i.e., instantaneous broadcasts, are reliable. Discussions on required bandwidth, specific communication protocols to be used etc. are outside the scope of this work. A more detailed discussion of a distributed implementation can also be found in [18]. •

V. SIMULATION RESULTS

In this section, we present the results of simulations comparing the proposed learning-based sequential optimization algorithm against combined optimization and some other policies from literature.

The source code to reproduce the simulations is available at the link: https://github.com/Control-Network-Systems-Group-IISc/InMan-SeqOpt-Learn.

A. SIMULATION SETUP

1) SIMULATION PARAMETERS

We consider a four-way intersection with 8 lanes meeting in the intersection centre, each [illegible] m, and an approach length of 2.8m. The width of each lane is 0.7m. Hence, the RoI is a square of side 2.8m. Each robot is modelled as a square of side 0.7m. The length of the inter-robot gap is 0.75m and upper and lower bounds on its acceleration are $2m/s^2$ and $-2m/s^2$, respectively. The initial velocity of robot i is sampled from a uniform distribution on $[0, \bar{v}_i]$. We carried out simulations for streams of randomly arriving robots with various *average arrival rates*. In particular, streams were generated by choosing tentative arrival times of robots into the lanes according to a poisson process, with a specified average arrival rate. The actual arrival time of each robot is delayed till the rear-end-safety condition (6) is satisfied. We set the coordinated phase computation interval to 6s, i.e., $T_c = 6s$.

When the arrival rates to all the lanes are the same, we say that the simulation has static homogeneous traffic, otherwise. If the (lane dependent) average arrival rate remains same throughout the duration of simulation, we say that the simulation has static traffic and time-varying traffic. We choose the average arrival rates on different lanes to be 0.13, 0.18, 0.08, 0.15, 0.19, 0.09, 0.05, and 0.16 robots/lane/s for lanes 1, 2, 3, 4, 5, 6, 7, and 8 respectively. We set the average arrival rate to be the same, 0.13 robots/lane/s, for all the lanes in the case of static homogeneous traffic. For the case of static heterogeneous traffic, we choose the average arrival rates on different lanes to be 0.13, 0.18, 0.08, 0.15, 0.19, 0.09, 0.05, and 0.16 robots/lane/s for lanes 1, 2, 3, 4, 5, 6, 7, and 8 respectively. We set the average arrival rate to be the same, 0.13 robots/lane/s, for all the lanes in the case of static homogeneous traffic. For the case of static heterogeneous traffic, we choose the average arrival rates on different lanes to be 0.13, 0.18, 0.08, 0.15, 0.19, 0.09, 0.05, and 0.16 robots/lane/s for lanes 1, 2, 3, 4, 5, 6, 7, and 8 respectively. To capture the time-varying traffic, we choose the average arrival rate for each lane uniformly from {0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15} every 100s. In case of burst-mode-time-varying traffic, every 30s, we choose to set average arrival rate to be 0.15 robots/lane/s for the first 10s and 0.05 robots/lane/s for the remaining 20s.

A robot i is said to have homogeneous parameters if all the robots in $V_a(k)$ have the same priority and upper bound on velocity i.e., $r_i = 1$ and $\bar{v}_i = 1.5$m/s

$\forall i \in \mathcal{V}$。另一方面，如果不同机器人的优先级和速度上限各不相同，则称该仿真的参数为异质参数。具体而言，我们将机器人进入感兴趣区域时的优先级设为随机值，从集合 $\{1,2,4,5\}$ 中采样，各值的概率分别为 0.5、0.3、0.15 和 0.05。我们还设置了与车道相关的速度上限：车道 1、4、5 和 8 上的机器人速度上限为 1.5m/s，车道 2、3、6 和 7 上的机器人速度上限为 1m/s。

On the other hand, the simulation is said to have heterogeneous parameters if the priorities and velocity upper bounds are different for different robots. Specifically, we choose to set the robot priorities randomly by sampling from the set $\{1,2,4,5\}$ with probability 0.5, 0.3, 0.15 and 0.05 respectively at the time of its entry into RoI. We choose to set lane dependent velocity upper bounds where robots on lanes 1, 4, 5 and 8 have velocity upper bound as 1.5m/s and those on lanes 2, 3, 6 and 7 have velocity upper bound as 1m/s.

仿真时间步长设置为 0.1 秒。表 3 总结了我们为评估所提出方法的性能而设置的各种仿真场景。每个标示为同质和静态交通的单元格，也显示了相应训练或测试仿真所采用的机器人/车道/秒到达率。

Different simulation scenarios are set up to evaluate the performance of the proposed approach is presented in Table 3. Each cell that shows homogeneous and static traffic, also shows the set of arrival rates in robots/lane/s for which the corresponding training or testing simulations were done.

2) CML 训练

为训练策略，我们收集了 10 个独立的经验回放缓冲区（replay buffers），每个缓冲区对应一个平均到达率，对于在同质交通环境下训练的情形，每个缓冲区对应一个平均到达率设置，对于在异质交通下训练的情形，则对应 10 个相同的平均到达率设置），每个缓冲区都包含从随机流中采集的 5000 个协调阶段的数据（随机流按照算法3演化）。随后将这些独立的经验回放缓冲区合并为一个合并经验回放缓冲区。接着，我们使用该合并经验回放缓冲区，通过 CML 方法训练一组包含 10 个策略（网络初始化各不相同）的所有训练仿真一样，我们对折扣因子使用 0.99 的折扣因子。

2) CML TRAINING

For training policies, we collect 10 independent replay buffers, one for each average arrival rate, for cases with training on homogeneous traffic setting and 10 of the same average arrival rate setting for cases with training on heterogeneous traffic setting), each containing data of 5000 coordinated phases collected from random streams (evolving according to Algorithm 3). These individual replay buffers are merged to form a single merged replay buffer. This merged replay buffer is then used to train, using the CML method, a set of 10 policies (with different network initializations). We use a discount factor of 0.99 for all training simulations.

B. 次优性与计算时间

B. SUB-OPTIMALITY AND COMPUTATION TIMES OF SEQUENTIAL OPTIMIZATION METHOD

在本小节中，我们首先说明，一般而言，序贯优化能够得到近似最优解。为此，我们枚举所有可能的通行顺序，对联合优化与序贯优化的性能进行比较，并通过穷举搜索选择最佳通行顺序，我们将其称为 BESTSEQ。我们还说明，序贯优化在计算方面具有巨大优势；由 CML 策略确定优先级索引时，其计算性能优于联合优化和 BESTSEQ。

In this subsection, we first demonstrate that, in general, sequential optimization can give near optimal solutions. For this, we compare the performance of combined optimization against sequential optimization for all possible crossing orders and choosing the one with the best crossing order by exhaustive search, which we call as BESTSEQ. We also illustrate that there is a tremendous computational advantage of sequential optimization, with a CML policy determining the precedence indices over combined optimization and BESTSEQ.

为进行这三项比较，我们从 3 个各为 500 秒的流中收集了协调阶段问题实例（$V_i(.)$ 和 $V_j(.)$），其平均到达率为 0.08 机器人/车道/秒，并采用 BESTSEQ（同质、静态交通以及异质参数设置，其中包含 7[illegible] 个）。对于每个协调阶段问题实例，我们还使用联合优化计算了最优解。

For these comparisons, we collect coordinated phase problem instances ($V_i(.)$ and $V_j(.)$) from 3 streams of 500s each, with an average arrival rate of 0.08 robots/lane/s with BESTSEQ (homogeneous, static traffic and heterogeneous parameter setting with T_h = 30s). For each coordinated phase instance, we also computed the optimum solution using combined optimization.

对于给定的协调阶段问题实例 $\{\mathcal{V}_i(.), \mathcal{V}_j(.)\}$，令 C_1 和 C_2 分别表示通过联合优化和 BESTSEQ 得到的问题（1）的目标值。由于联合优化的计算成本较高，我们仅考虑满足 $|\mathcal{V}_p(.)| \leq 6$ 的协调阶段，共计 217 个协调阶段实例。表 2 列出了问题实例数（实例数）

For a given coordinated phase problem instance $\{\mathcal{V}_i(.), \mathcal{V}_j(.)\}$, let C_1 and C_2 denote the objective values of Problem (1) obtained using combined optimization and BESTSEQ respectively. Since combined optimization is computationally expensive, we consider only those coordinated phases with $|\mathcal{V}_p(.)| \leq 6$, making a total of 217 coordinated phase instances. Table 2 presents the number of problem instances (No. of inst.),

TABLE 2. Sub-Optimality of BESTSEQ.

$\lvert\mathcal{V}_p(.)\rvert$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
No. of inst.	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
opt gap (%)	0	[illegible]	[illegible]	[illegible]	[illegible]	0.05
95(%) 90(%)	90	[illegible]	[illegible]	[illegible]	[illegible]	3.29

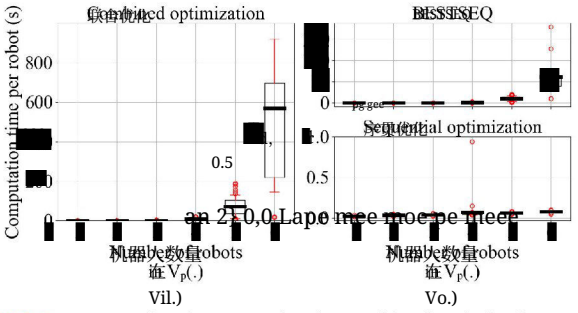


FIGURE 5. Computation time per robot for combined optimization, BESTSEQ and CML trained sequential optimization.

[illegible] 95th percentile (90th) of optimality gaps,

[illegible]

We see that, on average, the sub-optimality of BESTSEQ is within an acceptable range (∼ 2%).

对于同一组问题实例，图5对比了联合优化、BESTSEQ 和采用 CML 训练策略的序贯优化的每机器人计算时间。在图中，黑色粗线表示均值，箱体表示第一和第三四分位数之间的数值范围，须线表示第10和第90百分位数，红色圆圈表示离群值。我们看到，尽管存在一定的次优性，BESTSEQ 的计算时间仍远低于联合优化。采用 CML 训练策略的序贯优化，其每机器人计算时间基本不随机器人数量变化，[illegible] BESTSEQ 和联合优化的计算时间则随机器人数量较多时。因此，我们所提出的框架更适合实时实现。

For the same set of problem instances, computation time per robot for combined optimization, BESTSEQ and sequential optimization with a CML trained policy are compared in Figure 5. In this figure, bold black lines represent the mean, boxes represent the range of values between the first and third quartile, the whiskers represent the 10th and 90th percentile and red circles are outliers. We see that, with some sub-optimality, BESTSEQ incurs far less computation time. The BESTSEQ and the combined optimization times per robot for sequential optimization with a CML trained policy, is essentially constant with the number of robots and several orders of magnitude lower than that of BESTSEQ and combined optimization - specially for higher number of robots. Thus, our proposed framework is far more suitable for real-time implementation.

C. 与其他策略的比较

C. COMPARISON OF CML TRAINED POLICIES AGAINST OTHER POLICIES

接下来，我们将比较 CML 训练策略与 CML 训练策略的性能。由于除交通到达率极低的情况外，联合优化的计算时间都难以承受，因此我们在这些比较中不考虑联合优化。虽然 BESTSEQ 的计算并没有联合优化那么大，但其成本也高得难以承受。因此我们只报告低交通流条件下 BESTSEQ 的数据，鉴于其相对于联合优化的最优性差距较小，在计算时间可接受的情况下，它也可作为合理的基准。

Next we compare the CML trained policies against some heuristics from literature. Since the computation time for combined optimization is prohibitive except for very low traffic arrival rates, we skip it from these comparisons. Though BESTSEQ is not as computationally intensive as combined optimization, it too is prohibitively costly and we report data from BESTSEQ only for low traffic conditions. Given its small optimality gap with respect to combined optimization, it also serves as a reasonable benchmark where its computation time is manageable.

TABLE 3. Traffic parameters and other parameters used in different simulations. Acronyms: Hm.: Homogeneous, Ht.: Heterogeneous, Param.: Parameters. Note: For Sim-1 to Sim-5, we train separate CML policies for each simulation. But for testing Sim-6 to Sim-9 we use the policies trained in Sim-2.

	Training				Testing			
	Traffic	Param.	T_h (s)	T_r (s)	Traffic	Param.	T_h (s)
Sim-1	Hm., Static $\{0.01, 0.02, ..., 0.1\}$		30	20	Hm., Static $\{0.01, 0.02, ..., 0.1\}$		30
Sim-2	Hm., Static $\{0.11, 0.12, ..., 0.2\}$		60	30	Hm., Static $\{0.11, 0.12, ..., 0.2\}$		60
Sim-3	Hm., Static $\{0.01, 0.02, ..., 0.1\}$		30	20	Hm., Static $\{0.01, 0.02, ..., 0.1\}$		30
Sim-4	Hm., Static $\{0.11, 0.12, ..., 0.2\}$		60	30	Hm., Static $\{0.11, 0.12, ..., 0.2\}$		60
Sim-5	Ht., Static				Ht., Static		
Sim-6	Hm., Static $\{0.11, 0.12, ..., 0.2\}$				Hm., Static $\{0.01, 0.02, ..., 0.1\}$		
Sim-7					Hm., Static $\{0.125, 0.175, 0.225, ..., 0.375\}$		
Sim-8					Ht., bursty inside time-varying		30
Sim-9					Ht., random time-varying		

In this section, we describe the overall solution approach for determining the crossing order in sequential optimization, Algorithm 2. We compare the policies generated by CML against the following policies for various situations.

(i) BFSTSEQ: Sequential optimization with a best-first-search (BFS) algorithm determining the best crossing order, which is determined with an exhaustive search.

(ii) CMLCEN: Sequential optimization with a central (not-shared) RL policy deciding the precedence indices for each involved robot as a function of features of all the involved robots and pseudo-robots.

(iii) FCFS [5], [26]: First come first serve. Complete trajectory for a robot, say $i \in V$, to cross the intersection is obtained as and when it enters the RoI (at t_i^A) considering $V_p(.) = \{i\}$ and $V_s(.) = \{j : j \in V \wedge t_j^A < t_i^A\}$.

(iv) TTR [24]: Sequential optimization with negative of the ratio of distance to intersection and current velocity of a robot as its precedence index.

(v) PDT [24]: Sequential optimization with negative of the product of distance to intersection and time-to-react of a robot as its precedence index.

(vi) CDT [24]: Sequential optimization with a convex combination of distance to intersection and time-to-react with the convex combination parameter 0.5.

(vii) OCP [28]: Sequential optimization with order in time for virtual intersection entry and exit times, detailed in Section II(a) of [28]. Then the steps 1 to 5 proposed in Section III(a) of [28] are followed to decide the crossing order.

We compare the performance of the proposed CML-based policy approach against the above-mentioned policies for various traffic and parameter settings, for e.g., homogeneous traffic (Sim-1 through Sim-4) and heterogeneous traffic (Sim-5). For Sim-1 through Sim-5, we train different policies for each simulation. We also compare the performance of policies trained on some set of average arrival rates against heuristics on a test set of different average arrival rates (Sim-6 and Sim-7), while varying average arrival rates (Sim-8 and Sim-9). For testing Sim-6 to Sim-9, we use the policies trained in Sim-2. These serve as a test for learnt policy generalization. Table 3 presents a comprehensive list of training and testing traffic and other parameters used in different simulations.

We compare the performance of the following heuristics against the learnt CML policy:

We run each of the heuristic policies on 100 randomly generated streams each of 300s long, for each average arrival rate. For the learnt policies from CML approach, we run each of the 10 trained policies on 10 randomly generated streams (hence 100 random streams), for each average arrival rate in the simulation (refer to Table 3). We remove the data of first 90s in each stream to neglect transient traffic behaviour and compute the average (over 100 streams) of the following values (values of the objective function J_j of Problem (48) with the first T_j seconds of a robot's trajectory data), from generated streams. We call this the *average performance* of policy p, denoted by J_p, for a given arrival rate. We then compare the average performance of the learnt CML policy against some policy p using the quantity

$$\frac{\hat{J}_{CML} - \hat{J}_p}{\hat{J}_p}$$

for each case of the arrival rate and the 10 trained CML policies against the heuristics using the same 100 streams over which $\hat{J}_p$ is computed. This is done to ensure that the variance in $\hat{J}_{CML} - \hat{J}_p$ is not affected by the randomness in the streams. For this purpose, we also store the values of $\hat{J}_p$ for each case of the arrival rate and each stream in a file. Further, we compare

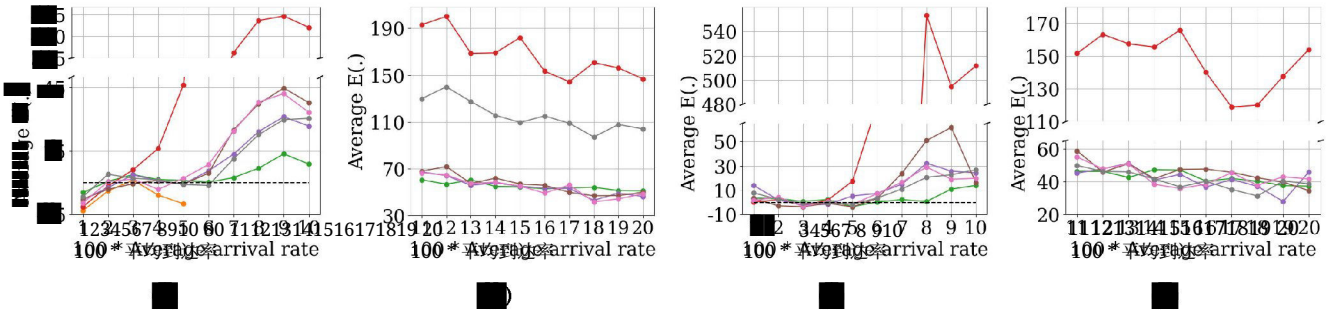


FIGURE 6. Percentage improvement in average performance of CML trained policies over that of different heuristics (average $E(.)$, see (17)), averaged over 100 random streams for various heuristics and average arrival rates (Figures (a), (b), (c) and (d) for Sim-1, Sim-2, Sim-3 and Sim-4 respectively). Dashed black line is 0%. See Figure 9 for legend.

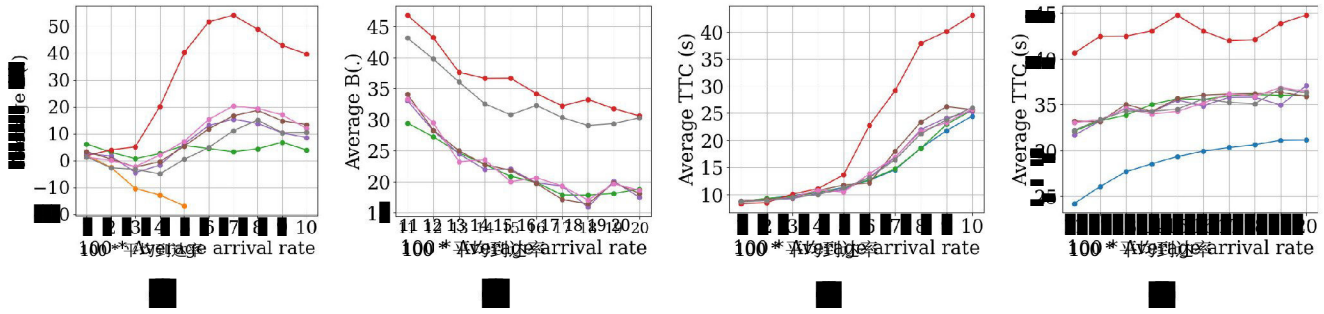


FIGURE 7. Percentage reduction in weighted average of TTC for CML when compared with various heuristics ($B(.)$, see (18)) and Average time to cross (TTC) for different policies/heuristics averaged over 100 random streams (Figures (a), (b), (c) and (d) for Sim-1, Sim-2, Sim-3 and Sim-4 respectively). See Figure 9 for legend.

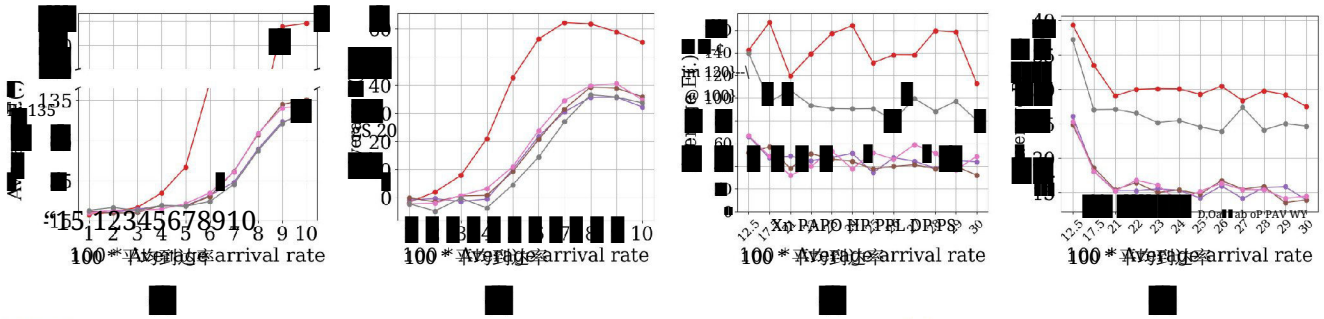


FIGURE 8. Percentage improvement in average performance of CML trained policies (average $E(.)$, see (17)) over that of various heuristics averaged over 100 random streams ((a) for Sim-6 and (c) for Sim-7). Comparison of weighted average time to cross (TTC) against CML for various heuristics (average $B(.)$, see (18)) ((b) for Sim-6 and (d) for Sim-7). See Figure 9 for legend.



FIGURE 9. Legend for Figures 6, 7, 8, 17a and 17b.

on the fly during the implementation, communication delays and computation times made during the formulation in Section II. In the following, we describe how we have relaxed these assumptions to make the framework fit for practical implementation.

We limit the scope of this paper by assuming that a low level trajectory tracking controller is available on each robot which tracks the given reference trajectory faithfully

Tracking errors:

Suppose the tracking error is bounded by b units. The tracking errors are handled by considering $\hat{L}_i = L_i + 2b$ as the length of robot $i \in V$ for all computations in the algorithm. Notice that this affects both rear-end and intersection safety constraints.

Communication and computation delays:

Suppose the communication delay is Δ units, computation time per robot is δ units, bounds on computation time per robot are Δ units and A_+ units for provisional phase and coordinated phase, respectively. Computation time and communication delays are handled by pre-computing the trajectories. That is if a

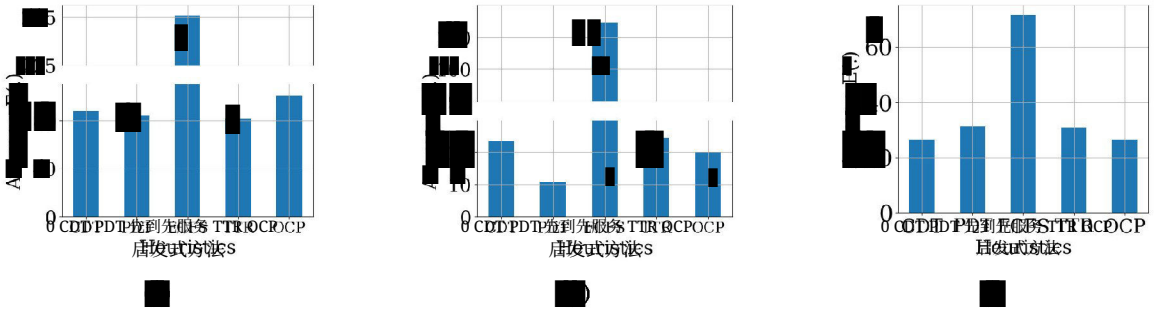


FIGURE 10. CML: Percentage improvement in average performance of CML trained policies over that of different heuristics (E_ℓ), see (47)) averaged over 100 random streams for various heuristics and average arrival rates (Figures (a), (b) and (c) for Sim-5, Sim-8 and Sim-9 respectively).

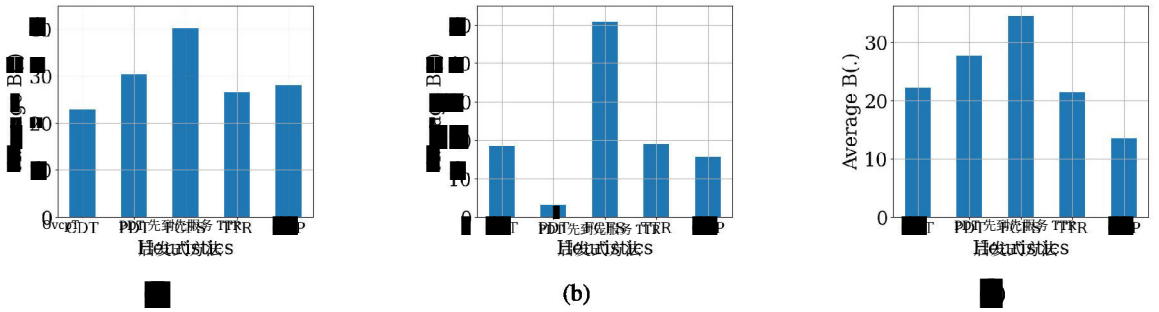


FIGURE 11. Comparison of weighted average time to cross (TTC) against CML for various heuristics (average $B(.)$, see (48)) (Figures (a), (b) and (c) for Sim-5, Sim-8 and Sim-9 respectively).

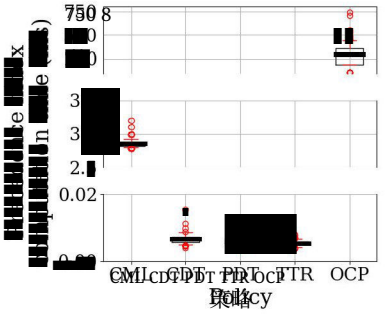


FIGURE 12. Comparison of computation times required to get precedence indices (crossing order) for different policies/heuristics.

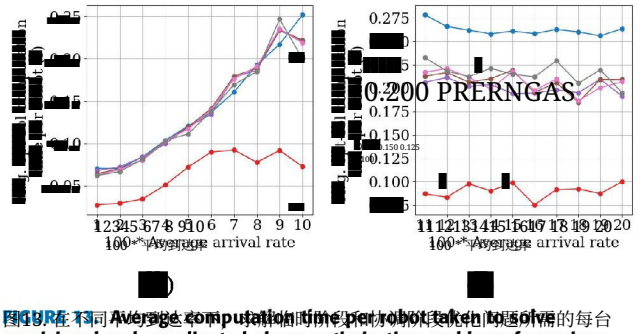


FIGURE 13. Average computation time per robot taken to solve provisional and coordinated phase optimization problems for various average arrival rates. Figures (a) and (b) are with simulation parameters corresponding to Sim-1 and Sim-2 respectively. See Figure 9 for legend.

robot is added/deleted, the RoI gets updated. Robots in the RoI will compute their $\mathcal{A}$ and $\mathcal{D}$ sets, velocity of entry and initiate its provisional phase trajectory computations at time $t = \mathcal{A}_p - \delta$ and $t = \mathcal{D}_p - \delta$ respectively. Similarly, we initiate the computations for k^{th} coordinated phase at $kT_c - N_p\Delta_c - \delta$ for the robots in $\mathcal{V}_p$ at that time, where N_p is an appropriate upper bound on the number of robots in $\mathcal{V}_p$. The computations are made assuming perfect prediction of positions and velocities of robots at kT_c.

If the communication delay is known, then the errors due to such delays can be merged with tracking errors by adding the maximum possible distance a robot $j \in \mathcal{V}_p$ can cover during the communication delay period which is $\bar{v}_j\delta$. In this case, trajectory pre-computation times may not involve the communication delay term δ. Since the communication delays are small in our experiments, we follow this method.

Note that, the above discussion is for the ideal scenario without these adaptations. These adaptations to the algorithm are made only during implementation.

A. SIMULATION STUDY ON EFFECT OF ADAPTATIONS

Since we have not included these strategies in the theoretical framework, an interesting question is to see how the performance of the policy in the adapted framework compares with that in the ideal framework. Since the adaptations are incorporated in the implementation, it is natural to expect some degradation in the performance of the policy in the adapted framework setting. In this regard we compare the performance with and without adaptations for setting in Sim-1 and Sim-2.

Recall that, due to the computation delays, the actual coordinated phase times are handled by pre-computations. Notice that, even with pre-computations, the coordinated phase occurs periodically in the adapted framework with the same period as the ideal

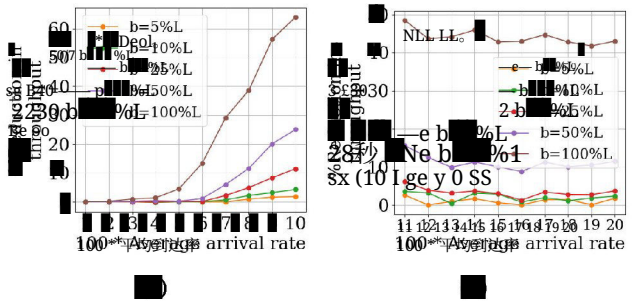


FIGURE 14. Percentage reduction in throughput (number of robots crossed per unit time) for various average arrival rates and various values of buffer b when compared against $b = 0$. Figures (a) and (b) are with simulation parameters corresponding to Sim-1 and Sim-2 respectively. L denotes the length of a robot.

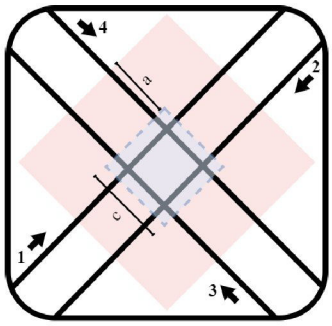


FIGURE 15. A schematic of the layout used for implementation on robots. The black lines represent the paths for the robots to follow. The union of red and blue shaded regions is the RoI and the blue shaded region is the intersection. The robots flow in the direction of the arrows depicted along with lane numbers. $a = 0.8$m and $c = 0.53$m.

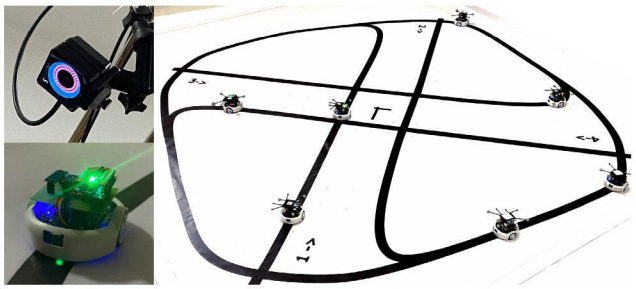


FIGURE 16. Setup for hardware implementation in our lab. The bottom-left picture is a close-up of a robot used, the top-left picture is one of the cameras of OptiTrack motion capture system. The picture on the right depicts an ongoing experiment.

delay, we do not consider it in the following numerical study. Instead, we model any error in prediction of robot state for pre-computations as tracking error.

As mentioned earlier, we add a buffer of $2b$ units to the length of the robot in the constraints. In Figure 14, we study the effect of tracking errors on the average throughput (number of robots crossed per unit time averaged over 10 policies and 10 simulations per policy) by comparing the ideal scenario ($b = 0$) against various values of b for Sim-1 and Sim-2. In all these simulations, we use the CML policies trained on the ideal scenario with $b = 0$. In order to reduce the randomness in the comparisons, we fix the same number of robots and the same random seed for their arrival times for all simulations. The actual stream may be different for different values of b due to the effect of the buffer value. In Figure 14, we observe that the average throughput consistently reduces with increase in the buffer value. Also, it is interesting to note that for small buffer values, the reduction in throughput is reasonably small.

VII. IMPLEMENTATION IN A LAB SETTING

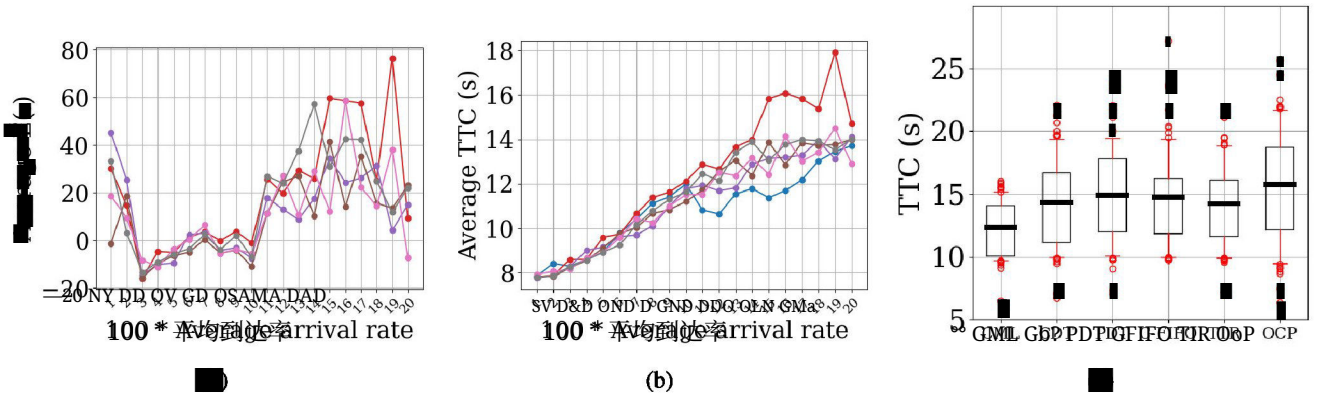
We implemented the proposed algorithm on a collection of line following robots, specifically 3pi+ 32U4 Turtle edition robots manufactured by Pololu Robotics and Electronics [65]. Due to the lack of significant computational and communication capabilities on these robots, we run the algorithm on a computer, store the computed trajectories and use XBee S6B WiFi modules to communicate the trajectories to the robots at each time-step. We use OptiTrack motion capture system [66] to track these robots. We create a 4-way intersection as shown in Figure 15. The blue shaded lines represent the lanes for the robots. The robots follow the lines to cross the intersection and loop around on the perimeter square and re-enter the RoI on a randomly chosen lane. This process continues repeatedly so that we have a continual robot streams.

A. PARAMETERS FOR EXPERIMENTS

The approach length for each lane is $l = 0.8$m, $\forall l \in \{1, 2, 3, 4\}$ (see Figure 15) and the width of the lane is $a = 0.8$m, $\forall l \in \{1, 2, 3, 4\}$. The RoI is a square of side $c = 0.53$m (depicted in Figure 15). The length of each robot is 0.1m. The bound on inherent tracking errors and tracking errors due to communication delay was measured to be $b = 0.075$m. To allow for better trajectory tracking performance, we set $v_l = 0.25$m/s $\forall l \in V$, even though the robots are capable of speeds up to 0.4m/s. Given these bounds, we chose $L_l = 0.15$m for buffer to address communication delays and tracking errors. We set $T_s = 2$s. We implemented PID controllers for low-level trajectory tracking and used XBee modules for communication. We use a PID position tracking controller for low-level trajectory tracking, which faithfully tracks the trajectory with bounded error. Figure 16 shows a picture of the actual lab setup where we ran these experiments.

B. INDICATIVE RESULTS FROM EXPERIMENTS

We used CML approach to learn policies in the ideal simulations (without tracking errors, communication and computation delays and restricting parameters only to the RoI) using CML approach. We trained a set of 10 policies each with different network parameter initialization on the set of average arrival rates $\{0.11, 0.12, \ldots, 0.2\}$ robots/lane/s. We tested these 10 learnt policies in ideal simulation environment and chose the policy with the highest sum of average throughput values for the average arrival rates in the set $\{0.01, 0.02, \ldots, 0.2\}$ robots/lane/s, over 10 test simulations for each average arrival rate and compare its



(a) (b)

FIGURE 17. Figure (a): Percentage improvement in average performance of CML trained policy over that of different policies and Figure (b) Average time to cross (TTC) for different policies. Both the plots are under ideal simulation environment for parameters used in experiments. See Figure 9 for legend. Figure (c): Box plot of TTC values for 3 runs of 3 different initial positions each for different policies. The box represent the range of values between 1st and 3rd quartile, whiskers represent range of values between 10th and 90th percentile and bold black line represents the mean of the TTC values for each policy.

性能与启发式方法进行比较。这些仿真结果的图示见图 17a 和图 17b。我们观察到，对于实验中所考虑的参数，CML 策略只有在平均到达率超过 0.1 机器人/车道/秒后，才开始优于其他启发式方法。

the results of these simulations is presented in figures 17a and 17b. We observe that, for the parameters considered in the experiments, the CML policies start to outperform the other heuristics only after an average arrival rate of 0.1 robots/lane/s.

However, due to the limitations imposed in our lab setting (e.g. number of robots, approach length for each lane, etc.), we do not present results from our experiments with longer continual streams of robots. A short video of our lab implementation with continual streams negotiating the intersection safely can be seen using the link https://youtu.be/Io4DxmpJPaI. This stands as a proof of concept for real-time implementability of the methods proposed in this work. We leave the larger scale implementation for future work.

然而，由于实验室设置的限制（例如机器人数量、每条道的接近段长度等），我们不展示长时间连续机器人流的实验结果。点击链接 https://youtu.be/Io4DxmpJPaI 可观看一段简短视频，展示了我们在实验室中实现的机器人连续车流安全通过交叉口的协商过程。这可以证明本文所提方法具备实时可实施性。大规模实现留待未来工作。

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have considered a multi-robot coordination problem in the context of autonomous intersection management. We have proposed a reinforcement learning based approach to obtain a near-optimal solution for multi-robot unsignalized intersection management, which can be implemented in real-time. The proposed solution gives a policy that is shared by all the robots and can be deployed in a distributed manner. With extensive simulations, we have established that such a policy outperforms the major heuristics proposed in the literature, over a range of average traffic arrival rates. We also illustrate the usefulness of the proposed solution approach in the presence of practical challenges like tracking errors, communication and computation delays and have implemented the learnt policies on robots in a lab setting with the proposed adaptations. The proposed method is flexible so that, with fresh training, policies for different kinds of intersections can be learnt. As already seen, such a training will need some basic

VIII. 结论与未来工作

本文研究了自主交叉口管理中的多机器人协调问题。我们提出了一种基于强化学习的近似最优解决方案，该方案可实时实现。所提出的解决方案给出了一种由所有机器人共享的策略，且可采用分布式方式部署。通过大量仿真，我们证明，在一系列平均交通到达率下，该策略优于文献中提出的主要启发式方法。我们还展示了，相较于针对交叉路口管理问题的现有优化方法，所提出的解决方案在计算时间方面有了显著改善。我们还对一些实际挑战（如跟踪误差、通信延迟和计算延迟等）进行了说明，并在实验室环境中对机器人实施了所学习的策略，同时采用了所提出的适应措施。所提出的方法具有灵活性，只需重新训练，就能学习适用于不同类型交叉路口的策略。如前所述，此类训练需要一些基本的

information about the intersection and the vehicles. However, expert knowledge is necessary (as in the case of some heuristics and analytical methods). One future research direction is to use transfer learning methods to reduce training time when a policy learnt for one intersection setting is to be adapted to another setting. It is also interesting to see if a single policy can be learnt to work for a set of intersections and robots with different parameters like lane and RoI dimensions, robot velocity and acceleration limits, etc. Other future directions include implementation fine-tuning, extensions to allow lane changes, to handle disturbances, dynamic obstacles and for a network of intersections.

ACKNOWLEDGMENT

The authors would like to thank Ayush Das, Rithvik Mahajan, Nadh Sobh, and Vedant Patil for their help in collecting data and implementing the algorithm on robots in the lab setting.

REFERENCES

[1] C. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[2] M. Khayatian, M. Mehrabian, E. Andert, R. Dedinsky, S. Choudhary, Y. Lou, and A. Shrivastava, "A survey on intersection management of connected autonomous vehicles," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 4, pp. 1–27, Oct. 2020.

[3] M. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of automated vehicles at intersections: Turn-based vs. time-based approach," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2019, pp. 2510–2225.

[4] Y. Qian, J. Gregoire, A. de La Fortelle, and F. Moutarde, "Decentralized model predictive control for smooth coordination of automated vehicles at intersection," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 3452–3458.

[5] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 81–90, Mar. 2012.

[6] I. Papamichail and M. Papageorgiou, "Traffic-responsive linked ramp-metering control," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 111–121, Mar. 2008.

[7] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[8] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1066–1077, May 2017.

[9] A. Gholamhosseinian and J. Seitz, "A comprehensive survey on cooperative intersection management for heterogeneous connected vehicles," *IEEE Access*, vol. 10, pp. 7993–7972, 2022.

[10] Z. Qin, A. Tal, Z. Sun, G. Wu, P. Hao, and X. Liao, "Game theoretic application to intersection management: A literature review," *IEEE Trans. Intell. Vehicles*, early access, Mar. 21, 2024, doi: 10.1109/TIV.2024.3379986.

[11] S. A. Fayazi and A. Vahidi, "Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 3, pp. 287–299, Sep. 2018.

[12] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 1, pp. 8–21, Spring 2017.

[13] S. A. Fayazi, A. Vahidi, A. Luckow, "通过混合整数线性规划的自动驾驶车辆交叉路口协调," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4920–4925.

[14] S. A. Fayazi, A. Vahidi, A. Luckow, "基于混合整数线性规划的自动驾驶车辆交叉路口协调," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4920–4925.

[15] A. A. Malikopoulos, C. G. Cassandras 和 Y. J. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, Jul. 2018.

[16] A. A. Malikopoulos, C. G. Cassandras 和 Y. J. Zhang, "一种用于互联自动驾驶车辆的分散式能量最优控制框架," *Automatica*, 第93卷, 第244—256页, 2018年7月.

[17] A. A. Malikopoulos, L. Beaver 和 I. V. Chremos, "网联自动驾驶车辆的最优时间轨迹与协调," *Automatica*, 第125卷, 2021年3月, 文章编号: 109469. [Online]. Available: https://doi.org/10.1016/j.automatica.2020.109469

[18] 面向网联自动驾驶车辆的时间最优轨迹与交叉路口协调," 载于第16届IEEE智能交通系统国际会议论文集, 2013年10月, 第529—534页.

[19] M. Hausknecht, T. C. Au, and P. Stone, "Autonomous intersection management: Multi-intersection optimization," in *Proc. RSJ Int. Intell. Robots Syst.*, 2011, pp. 4581–4586.

[20] P. Tallapragada and J. Cortés, "Coordinated dynamic spacing of platoons of autonomous vehicles at signal-free intersections," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 2, pp. 1002–1016, Jun. 2020.

[21] P. Tallapragada and J. Cortés, "Hierarchical-distributed optimized coordination of intersection traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 100–116, Jan. 2020.

[22] B. Chalaki and A. A. Malikopoulos, "A priority-aware replanning and resequencing framework for coordination of connected and automated vehicles," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 487–492, Apr. 2021.

[23] M. Kneissl, A. Molin, H. Esen, and M. Jokic, "Sampling based decentralized optimal motion planning for multiple nonholonomic vehicles," in *Proc. 58th IEEE Conf. Decis. Control (CDC)*, Dec. 2019, pp. 2985–2990.

[24] G. R. de Campos, P. Falcone, and J. Sjöberg, "Autonomous cooperative driving: A velocity-based negotiation approach for intersection crossing," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 1456–1461.

[25] A. Suriyarachchi, I. M. Tariq, C. Mavridis, and J. S. Baras, "Real-time priority-based cooperative highway merging for heterogeneous autonomous traffic," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2022, pp. 2019–2026.

[26] H. Xu, Y. Xiao, C. G. Cassandras, Y. Zhang, and L. Li, "A general framework for decentralized safe optimal control of connected and automated vehicles in multi-lane signal-free intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17382–17396, Oct. 2022.

[27] S. Chalaki, W. Turneberg, and M. Kim, "A safe and robust autonomous intersection management system using a hierarchical control strategy," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 50–61, Jan. 2021.

[28] X. Pan, B. Chen, S. Timotheou, and S. A. Evangelou, "A convex optimal control framework for autonomous vehicle intersection crossing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 163–177, Jan. 2023.

[29] X. Pan, B. Chen, L. Dai, S. Timotheou, and S. A. Evangelou, "A hierarchical robust control strategy for decentralized signal-free intersection management," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 5, pp. 2011–2026, Sep. 2023.

[30] S. Dai, H. Xiao, B. Chen, and S. A. Evangelou, "Real-time optimal scheduling and control for autonomous vehicle intersection coordination," in *Proc. 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 2601–2606.

[31] C. Vitale, P. Kolios, and G. Ellinas, "Autonomous intersection crossing with vehicle location uncertainty," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17346–17361, Oct. 2022.

[32] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 529–534.

[33] N. Suriyarachchi, R. Chandra, J. S. Baras, and D. Manocha, "GAMEOPT: Optimal real-time multi-agent planning and control for dynamic intersections," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 2599–2606.

[34] C. Liu, C.-W. Lin, S. Shiraishi, and M. Tomizuka, "Distributed conflict resolution for connected autonomous vehicles," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 1, pp. 18–29, Mar. 2018.

[35] X. Chen, M. Hu, B. Xu, Y. Bian, and H. Qin, "Improved reservation-based method with controllable gap strategy for vehicle coordination at non-signalized intersections," *Phys. A, Stat. Mech. Appl.*, vol. 604, Oct. 2022, Art. no. 127953.

[37] T. Regev and V. Indelman, "Decentralized multi-robot belief space planning in unknown environments via efficient re-evaluation of impacted paths," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5591–5598.

[38] B. Chalaki, L. Beaver, B. Remer, K. Jang, E. Vinitsky, K. Wu, G. Anagnostopoulos, A. Kotsialos, F. Zhao, G. Gunter, J. Gopalakrishnan, A. Tanwani, R. Fernandez, J. Sprinkle, and A. A. Malikopoulos, "Experimental validation of a real-time optimal controller for coordination of cavs in a multi-lane roundabout," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3314–3321.

[39] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1671–1678.

[40] L. Kästner, V. Shcherbyna, H. Laarabi, M. Hafez, J. Lambrecht, and D. Meyer, "Towards collision-free, efficient and socially-polite navigation of mobile robots via deep reinforcement learning," in *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2021, pp. 1–6.

[41] J. Cui, W. Liu, A. Kumar, J. Li, and J. Biswas, "Coverage path planning using heterogeneous unmanned aerial vehicles," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8900–8906.

[42] Y. Wu, H. Chen, and F. Zhu, "DCL-AIM: Decentralized coordination learning of autonomous intersection management," *arXiv preprint arXiv:2007.06519*, 2020.

[43] L. Wu, M. Jiang, and J. Zhang, "Cooperative multiagent deep deterministic policy gradient (CoMADDPG) for intelligent connected transportation with unsignalized intersection," *Math. Problems Eng.*, vol. 2020, pp. 1–12, Jul. 2020.

[44] G. Rodriguez-Gil, J. Del Ser, S. Gil-Lopez, I. Laña, and I. Olabarrieta, "Urban traffic flow forecasting by means of a double-layer ensemble of regression trees," in *Proc. Int. Conf. Model. Simulation Appl. Optim. (ICMSAO)*, Apr. 2019, pp. 1–6.

[45] G. P. Antonio and C. Maria-Dolores, "Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7033–7043, Jul. 2022.

[46] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[47] J. Xue, B. Li, and R. Zhang, "Multi-agent reinforcement learning-based autonomous intersection management protocol with attention mechanism," in *Proc. IEEE 25th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2022, pp. 1182–1187.

[48] Z. Pan, Y. Hu, Y. Cheng, T. Fan, W. Liao, C. Wu, Liu, and H. Zhang, "COOR-PLT: A hierarchical control model for coordinating adaptive platoons of connected and autonomous vehicles at signal-free intersections based on deep reinforcement learning," *Transp. Res. C, Emerg. Technol.*, vol. 146, Jan. 2023, Art. no. 103933.

[49] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2666–2673, Apr. 2021.

[50] N. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 9839–9849.

[51] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6252–6259.

[52] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Oper. Res.*, vol. 134, Oct. 2021, Art. no. 105400.

[53] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.

[54] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 9839–9849.

[55] D. Bertsimas and B. Stellato, "Online mixed-integer optimization in milliseconds," *INFORMS J. Comput.*, vol. 34, no. 4, pp. 2229–2248, Jul. 2022.

[56] W. Ding, L. Sabattini, C. Secchi, and C. Fantuzzi, "Formation control of multi-AGV systems with collision avoidance and deadlock prevention," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 1322–1327.

[57] L. Draganjac, D. Miklic, Z. Kovacic, G. Vasiljevic, and S. Bogdan, "Decentralized control of multi-AGV systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016.

[58] X. Chen, Z. Xing, L. Feng, T. Zhang, W. Wu, and X. Hu, "Avoiding deadlocks for multi-AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 1364–1377, Apr. 2023.

[59] S. Dergachev and K. Yakovlev, "Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path planning," in *Proc. Int. Conf. Artificial Intelligence (ICAI)*, Jan. 2021, pp. 1489–1494.

[60] F. Pratissoli, R. Bonuccini, N. Battilani, and L. Sabattini, "Hierarchical traffic management of multi-AGV systems with deadlock prevention applied to industrial environments," *IEEE Trans. Autom. Sci. Eng.*, early access, May 24, 2023, doi: 10.1109/TASE.2023.3276233.

[61] J. P. van den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2005, pp. 430–435.

[62] P. Tallapragada and J. Cortés, "On decentralized feedback stabilization of unicycle formations in the presence of input constraints," in *Proc. Amer. Control Conf.*, Jul. 2016, pp. 6105–6110.

[63] X. Chu and H. Ye, "Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning," 2017, *arXiv:1710.00336*.

[64] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, "Robot-centric elevation mapping with uncertainty estimates," in *Proc. Int. Conf. Climbing Walking Robots (CLAWAR)*, 2014, pp. 433–440.

[65] *3pi+ 32U4 Robot—Turtle Edition (75:1 LP Motors), Assembled*. Accessed: Aug. 12, 2023. [Online]. Available: https://www.pololu.com/product/3738

[66] NaturalPoint. *Motion Capture Systems-OptiTrack Webpage*. Accessed: Aug. 12, 2023. [Online]. Available: https://optitrack.com



NISHCHAL HOYSAL C. (Student Member, IEEE) received the B.E. degree in mechanical engineering from the B.M.S. College of Engineering, Bengaluru, India, in 2016. He is currently pursuing the Ph.D. degree with the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science at Bengaluru.

His research interests include multi-agent learning methods for multi-agent planning and coordination.



PAVANKUMAR TALLAPRAGADA (Member, IEEE) received the B.E. degree in instrumentation engineering from the S.G.G.S. Institute of Engineering and Technology, Nanded, India, in 2005, the M.Sc. (Engg.) degree in instrumentation from Indian Institute of Science at Bengaluru, in 2007, and the Ph.D. degree in mechanical engineering from The University of Maryland, College Park, in 2013. He was a Postdoctoral Scholar with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, from 2014 to 2017. He is currently an Assistant Professor with the Department of Electrical Engineering and the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science at Bengaluru. His research interests include networked control systems, distributed control, multi-agent systems, and dynamics in social networks.