2023 Design, Automation & Test in Europe Conference (DATE 2023)
Special Initiative "Autonomous Systems Design"

Reinforcement-Learning-Based Job-Shop Scheduling for Intelligent Intersection Management

Shao-Ching Huang[1], Kai-En Lin[1], Cheng-Yen Kuo[1], Li-Heng Lin[1], Muhammed O. Sayin[2], Chung-Wei Lin[1]
[1]National Taiwan University, [2]Bilkent University
{b08902049, r11922065, b08902069, b06902043}@ntu.edu.tw, sayin@ee.bilkent.edu.tr, cwlin@csie.ntu.edu.tw

***Abstract*—The goal of intersection management is to organize vehicles to pass the intersection safely and efficiently. Due to the technical advance of connected and autonomous vehicles, intersection management becomes more intelligent and potentially unsignalized. In this paper, we propose a reinforcement-learning-based methodology to train a centralized intersection manager. We define the intersection scheduling problem with a graph-based model and transform it to the job-shop scheduling problem (JSSP) with additional constraints. To utilize reinforcement learning, we model the scheduling procedure as a Markov decision process (MDP) and train the agent with the proximal policy optimization (PPO). A grouping strategy is also developed to apply the trained model to streams of vehicles. Experimental results show that the learning-based intersection manager is especially effective with high traffic densities. This paper is the first work in the literature to apply reinforcement learning on the graph-based intersection model. The proposed methodology can flexibly deal with any conflicting scenario and indicate the applicability of reinforcement learning to intelligent intersection management.**

***Index Terms*—Intelligent intersection management, job-shop scheduling, proximal policy optimization, reinforcement learning.**

I. Introduction

Intersection is a main cause of congestion. Intersection management has a long history, starting from fixed-phase signals and then dynamic-phase signals, which control the lengths of traffic lights. Due to the technical advance of connected and autonomous vehicles, intersection management becomes more intelligent and potentially unsignalized as connectivity provides more information, and autonomy provides more precise control.

Regarding intelligent intersection management, there have been many research achievements in the literature. The first-come-first-serve (FCFS) scheduling policy is the most straightforward one, and the performance is quite decent when the traffic density is low. Dresner and Stone [1] introduced the concept of reservation tiles, which are areas reserved for specific vehicles to occupy, and thus the granularity of the FCFS scheduling policy can be adjusted to trade off scheduling efficiency and system robustness. Yang *et al*. [2] proposed a bi-level optimization model, where an intersection manager integrates departure sequences and trajectory design for vehicles based on their arrival information. Lin *et al*. [3] developed a graph-based model and a divide-and-conquer scheduling policy which guarantees a deadlock-free passing order for vehicles. Guney and Raptis [4] proposed an optimization-based policy to minimize the delays of vehicles. A broad survey can be obtained in a review paper [5]. It should be mentioned that, besides centralized approaches which have intersection managers to perform scheduling, distributed approaches based on communication between vehicles and decision of each vehicle are also possible solutions to intelligent intersection management.

Recently, machine learning has been applied to intelligent intersection management. Especially, reinforcement learning can be applied to interact with the environment and adjust the corresponding scheduling policy according to a reward function. Guan *et al*. [6] proposed a centralized coordination scheme using reinforcement learning. Incorporating the reinforcement learning model into the proximal policy optimization (PPO) enhances the sample efficiency, but the computational overhead is high due to huge observation space. Haarnoja *et al*. [7] designed a learning agent following two key principles: off-policy and maximum entropy. Under the maximum entropy framework, the learning agent aims to derive the soft policy iteration which alternates between policy evaluation and policy improvement. Wu *et al*.[8] adopted distributed reinforcement learning, divided learning agents and vehicles into the corresponding coordination or independent states, and reduced the space complexity. Li *et al*. [9] proposed another distributed reinforcement learning model and utilized convolutional neural network for extracting feature vectors to deal with huge observation space.

In this paper, we propose a reinforcement-learning methodology to train a centralized intersection manager. We utilize the graph-based intersection model [3], but, different from the heuristic policy in the work, we develop a reinforcement-learning-based intersection manager. We adopt a similar learning approach which models job-shop scheduling problem (JSSP) instances as Markov decision processes (MDP) [10] and learns the best dispatch policy with proximal policy optimization (PPO) [11]. However, to apply the learning approach to intersection management, we design a transformation between the graph-based intersection scheduling problem [3] and the JSSP with *blocking*, *deadlock*, and *arrival time* constraints and apply PPO training to the *constrained*-JSSP. A grouping strategy is also developed to apply the trained model to streams of vehicles. Experimental results show that, with high traffic densities, the learning-based intersection manager can outperform an improved greedy scheduling policy. This paper is the

This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Number NTU-111V1901-5, National Science and Technology Council (NSTC) in Taiwan under Grant Number NSTC-111-2636-E-002-018, and TUBITAK BIDEB 2232-B International Fellowship for Early Stage Researchers under Grant Number 121C124.

978-3-9819263-7-8/DATE23/© 2023 EDAA

first work in the literature to apply reinforcement learning to the graph-based intersection model [3]. Different from other learning-based counterparts, the proposed methodology can flexibly deal with any conflicting scenario (*e.g.*, any shape of intersection, lane merging, lane changing), and it also indicates the applicability of reinforcement learning to intelligent intersection management.

The rest of this paper is structured as follows. Section II introduces the graph-based intersection management model and the job-shop scheduling problem. Section III presents the problem statement, the methodology, and the grouping strategy. Section IV demonstrates the experiments, and Section V concludes this paper.

II. Backgrounds

In this section, we provide the backgrounds including the graph-based intersection management model [3] and the job-shop scheduling problem.

A. Graph-Based Intersection Management Model

To achieve intelligent intersection management, how to model the safety constraints and the intersection geometric structure is a crucial issue. In the graph-based intersection management model [3], the geometric structure of an intersection is captured by a number of non-overlapping *conflict zones* which partition the whole intersection area. A conflict zone can only be occupied by at most one vehicle at any moment. With this formulation, a possible trajectory of the intersection (*e.g.*, go straight, turn left, turn right) can be viewed as a sequence of conflict zones to be passed. To model constraints in the process of intersection management, the work proposed a *timing conflict graph*. Within the graph, a vertex is a pair of a vehicle and a conflict zone, denoting the status that the vehicle occupies the conflict zone; an edge represents the timing constraint (or the passing order) between vehicles on conflict zones.

Specifically, assume that a set of vehicles X is about to be scheduled by the intersection manager. Let x_i denote the i-th vehicle and z_j denote the j-th conflict zone. The timing conflict graph $G = (V, E)$ corresponding to the intersection and the set of vehicles X is defined as follows [3]:

- There is a vertex $v_{i,j} \in V$ if z_j is on the trajectory of x_i.
- There is a Type-1 edge $(v_{i,j}, v_{i,j'}) \in E$ if both $v_{i,j}$ and $v_{i,j'}$ exist, and $z_{j'}$ is the next conflict zone of z_j on the trajectory of x_i.
- There is a Type-2 edge $(v_{i,j}, v_{i',j}) \in E$ if both $v_{i,j}$ and $v_{i',j}$ exist, x_i and $x_{i'}$ are from the same source lane, and x_i is in front of $x_{i'}$.
- There are two Type-3 edges $(v_{i,j}, v_{i',j}) \in E$ and $(v_{i',j}, v_{i,j}) \in E$ if both $v_{i,j}$ and $v_{i',j}$ exist, and x_i and $x_{i'}$ are from different source lanes.

Type-1 edges ensure that a vehicle only passes the conflict zones with the specified order. Type-2 edges ensure that vehicles from the same source lane do not overtake each other. Type-3 edges represent the undecided passing order between two vehicles entering the same conflict zone. With this definition, the scheduling problem is reduced to choosing one edge (to be removed) from each pair of Type-3 edges.

Using the graph-based intersection management model has several advantages:

- **Generality**. All kinds of intersections can be modeled as directed graphs connecting conflict zones. Even for the same intersection, different structures and granularity of conflict-zones can also be chosen to meet specific purposes, *e.g.*, higher granularity gives higher expressiveness and more detailed maneuver; lower granularity gives better computational efficiency.
- **Simplicity**. The main goal of the model is to decide the passing order between two vehicles and leave the low-level control to underlying systems.
- **Collision-Freeness**. The definition of conflict zones guarantees that no two vehicles can occupy the same conflict zone at any moment. If all vehicles follow the decided passing orders, then the collision-freeness is guaranteed. Besides, the deadlock-freeness can also be analyzed and verified.

B. Job-Shop Scheduling Problem

The job-shop scheduling problem (JSSP) is an NP-hard optimization problem. It aims to find the optimal scheduling of a set of *jobs* involving some *machines* to minimize the total processing time. A JSSP instance consists of a set of jobs and a set of machines. Each job b_i is a sequence of operations which will be processed on n_i machines sequentially. The operation sequence of a job b_i is denoted as $o_{i,1} \to o_{i,2} \to \cdots \to o_{i,n_i}$, and each operation $o_{i,j}$ $(1 \le j \le n_i)$ has its corresponding processing time $p_{i,j}$. To solve a JSSP instance, the start time $s_{i,j}$ of each operation needs to be determined, such that the overall makespan, $\max_{i,j}\{s_{i,j} + p_{i,j}\}$, is minimized.

A JSSP instance can also be modeled as a mixed graph $G' = (O, \overrightarrow{E}, \overline{E})$. O denotes the vertex set, which includes all the operations $o_{i,j}$ and one dummy operation S denoting the starting status with zero processing time. $\overrightarrow{E}$ is the set of directed edges representing the orders of consecutive operations of the same job. $\overline{E}$ is the set of undirected edges representing the undecided orders between two operations of different jobs to be processed on the same machine. Using this model, solving the JSSP instance is equivalent to fixing the direction of each undirected edge in $\overline{E}$, *i.e.*, transforming G' to a directed acyclic graph. We will refer this as the *JSSP graph-based formulation* in the following sections.

Recently, there are numerous heuristic or learning-based approaches trying to push the JSSP scheduling limits. For example, JSSP instances are modeled as Markov decision processes (MDP) and solved with a sequence of decision steps [10]. The policy is trained with the proximal policy optimization (PPO) [11], an on-policy actor-critic reinforcement learning algorithm which achieves promising results comparing with the optimal solution as well as other heuristic approaches.

III. Approach

One challenge of solving the scheduling problem with reinforcement learning is to define the problem formulation, *i.e.*, defining the state, action, and reward. Through the resemblance

to the JSSP, we are able to reuse the JSSP graph-based formulation. However, since our problem has more constraints and parameters compared with the JSSP, the existing formulation needs modifications (additional constraints). We also find that the growing number of states in the reinforcement learning needs to be handled carefully.

A. Problem Statement

We model the target intersection as a tuple $I = (Z, T)$. Z denotes the set of conflict zones. T denotes the set of all possible trajectories given the structure of the conflict zones of the intersection, and a trajectory is defined as a sequence of conflict zones. A vehicle x_i passing the intersection I is defined as a tuple $x_i = (r_i, t_i)$, where r_i denotes the earliest arrival time, and $t_i \in T$ represents the trajectory.

Given an intersection I and a set of incoming vehicles $X = \{x_1, x_2, \ldots, x_{|X|}\}$, the corresponding timing conflict graph $G = (V, E)$ can be constructed as mentioned in Section II-A. Assuming $E_k \subseteq E$ as the set of Type-k edges in E and defining $G' = (O, \overrightarrow{E}, \overline{E})$ as the JSSP graph-based formulation, we can construct G' from G as follows:

- $O \leftarrow V$. The vertex sets are the same. A vehicle x_i with trajectory $z_{i,1} \rightarrow z_{i,2} \rightarrow \cdots \rightarrow z_{i,n_i}$ is represented as n_i vertices $v_{i,z_{i,1}}, v_{i,z_{i,2}}, \ldots, v_{i,z_{i,n_i}}$ in the timing conflict graph and n_i operations $o_{i,1}, o_{i,2}, \ldots, o_{i,n_i}$ in the JSSP graph-based formulation. Note that j of $v_{i,j}$ is the index of a conflict zone, and j of $o_{i,j}$ is the index of the j-th operation of a job b_i, *i.e.*, $o_{i,j}$ is the j-th operation of b_i.
- $\overrightarrow{E} \leftarrow E_1 \cup E_2$. $\overrightarrow{E}$ contains all of the Type-1 and Type-2 edges.
- $\overline{E} \leftarrow \{(v, u) \mid (v, u) \in E_3 \wedge (u, v) \in E_3\}$, where u, v are vertices. The paired Type-3 edges are transformed into one undirected edge.
- The processing time $p_{i,j}$ of each operation $o_{i,j}$ is defined as the minimum time needed for vehicle x_i to pass the j-th conflict zone of its trajectory.

After the transformation, the scheduling problem of intelligent intersection management can be represented as a JSSP instance and solved by fixing the direction of each undirected edge. However, there are more constraints required to be considered:

- **Blocking Constraint**. Different from a JSSP instance, we cannot temporarily remove a vehicle from the latest conflict zone that it occupies. That is, if an operation involving a job and a machine is chosen to be performed, any other operation involving the machine is blocked until the job leaves the machine and moves to its next machine.
- **Deadlock Constraint**. As a result of the previous constraint, scheduling an operation may introduce deadlocks to the intersection, and thus the operation needs to be excluded until it no longer introduces deadlocks.
- **Arrival Time Constraint**. Any operation $o_{i,j}$ should be assigned with a start time $s_{i,j}$ larger than or equal to the earliest arrival time r_i.

Regarding the objective, we no longer minimize the overall makespan. Instead, we minimize the total delay time of vehicles (indeed, the overall makespan and the total delay time of vehicles are positively correlated) as follows:

$$\min \sum_{i=1}^{|X|} \left(\max_j \{s_{i,j} + p_{i,j}\} - r_i - \sum_{j=1}^{n_i} p_{i,j} \right),$$

where $\max_j \{s_{i,j} + p_{i,j}\}$ is the leaving time of vehicle x_i, r_i is the earliest arrival time of vehicle x_i, and $\sum_j p_{i,j}$ is the total passing time needed for vehicle x_i.

B. Reinforcement-Learning-Based Methodology

Following one existing approach [10], we model the procedure as an MDP. For a JSSP graph-based formulation $G' = (O, \overrightarrow{E}, \overline{E})$, we use $|O|$ steps of decision to produce a solution to the instance. We also use a similar MDP formulation as the existing approach [10] with some modifications. The details are as follows:

- **State**. For time step t, the state s_t is the current graph representation of the solution, $G'(t) = (O, \overrightarrow{E} \cup E^*(t), \overline{E} \setminus E^*(t))$. $E^*(t)$ denotes the set of directed edges originally belonging to $\overline{E}$ whose direction becomes fixed during the decision process. Besides the graph structure, each operation $o_{i,j} \in O$ also contains two features at time step t: a binary $d_{i,j}(t)$ representing if the operation $o_{i,j}$ has been scheduled, and $c_{i,j}(t)$ indicating the earliest completion time of the operation $o_{i,j}$, which can be recursively computed with previous scheduling results. Note that $d_{i,j}(0)$ is false and $c_{i,j}(0) = 0$ for each operation $o_{i,j}$.
- **Action**. The action space $A(t)$ includes all the possible operations for the current time step. At time step t, an operation $o_{i,j}$ is in $A(t)$ if (1) for each operation $o_{i',j'}$ satisfying $(o_{i',j'}, o_{i,j}) \in \overrightarrow{E}$, $d_{i',j'}$ is true, *i.e.*, all of the preceding operations of $o_{i,j}$ have been scheduled, (2) the machine of $o_{i,j}$ is not blocked by other operations (the blocking constraint), (3) scheduling $o_{i,j}$ will not introduce a deadlock (the deadlock constraint), and (4) $r_i \leq t$ (the arrival time constraint).
- **State Transition**. Given the chosen action $a_t \in A(t)$, the earliest possible time of the operation can be determined by the previous scheduling. The chosen operation is then assigned with the earliest possible time as its start time, and the graph representation $G'(t)$ is updated according to the temporal relations.
- **Reward**. We measure the reward at time step t as the extra delay produced by the chosen action a_t. Specifically, we define the reward w_t to be

$$\sum_{i=0}^{|X|} \left(c_{i,n_i}(t) - c_{i,n_i}(t-1) \right),$$

  which is the summation of the difference of the earliest completion times of each job b_i, *i.e.*, the extra delay of b_i introduced at time step t.
- **Policy**. Given a state s_t, the policy $\pi(a_t|s_t)$ outputs a distribution over the current action space A_t.

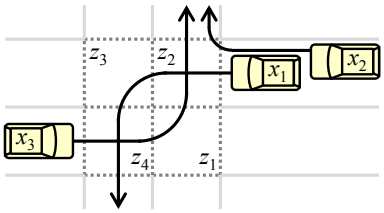

Fig. 1. An example scenario including 4 conflict zones (z_1, z_2, z_3, z_4) and 3 vehicles (x_1, x_2, x_3).

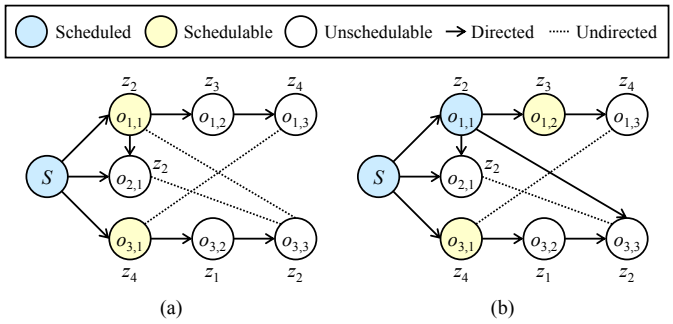

Fig. 2. Given the scenario in Figure 1, (a) shows the initial state of the scheduling, where the first conflict zone of the first vehicle of each lane is the only schedulable operations (*i.e.*, $o_{1,1}$ and $o_{3,1}$). (b) shows the state transition after the policy chooses to schedule operation $o_{1,1}$, where $o_{1,2}$ becomes schedulable since $o_{1,1}$ has been scheduled, and $o_{2,1}$ is still unschedulable since the conflict zone z_2 is still occupied (although $o_{1,1}$ has been scheduled).

Given an example scenario in Figure 1, Figure 2(a) shows the initial state of the scheduling, where the first conflict zone of the first vehicle of each lane is the only schedulable operations (*i.e.*, $o_{1,1}$ and $o_{3,1}$). Figure 2(b) shows the state transition after the policy chooses to schedule operation $o_{1,1}$, where $o_{1,2}$ becomes schedulable since $o_{1,1}$ has been scheduled, and $o_{2,1}$ is still unschedulable since the conflict zone z_2 is still occupied (although $o_{1,1}$ has been scheduled).

We follow the method in [10] to parameterize the policy, which applies the graph isomorphism network [12] to capture the features of graph-based data and has been proven to be powerful compared with other variants of graph neural networks [13]. As for the learning algorithm, we apply the PPO [11] for policy network training. The PPO is an on-policy actor-critic reinforcement learning algorithm which achieves promising results for various reinforcement learning tasks.

C. Grouping Strategy

As vehicles come as a stream, we need to group vehicles and apply the trained model to each group one by one. We also observe that, when the group size is larger, the performance of the trained PPO models decreases, and they also suffer longer training time. Therefore, we aim to group vehicles according to their arrival times. To cluster 1-dimensional data, there are two common approaches. The Jenks natural breaks optimization [14] aims to find the specified number of breaks by minimizing the variance within each class and maximizing the variance between classes simultaneously. The Kernel density estimation [15] aims to estimate the probability density function

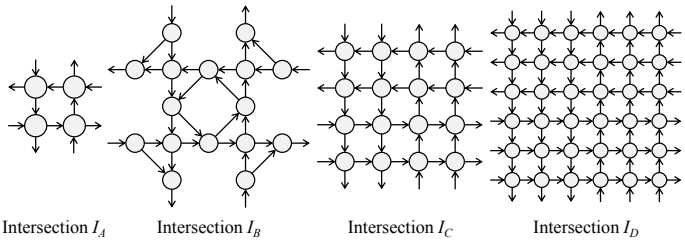

Fig. 3. Different structures of conflict zones in the experiments.

based on the given finite data samples and kernel function. Here, we choose the Jenks natural breaks optimization since the pre-determined group number makes a group size more likely to match the group size of training (the experimental results will show that this usually leads to better performance). For example, if there are 30 vehicles and the group number is set to 3, then the group size will be closer (though not always equal) to 10. Note that, it is required to solve each group of vehicle as a JSSP-based formulation. Indeed, this may lose the optimal solution from the solution space (*e.g.*, of 30 vehicles), but it allows us to train a model with a smaller group size (*e.g.*, 10 vehicles) and reduce the computational overhead in both training time and runtime.

IV. Experiments

In this section, we describe the experimental setting and demonstrate the experimental results.

A. Experimental Setting

Various environments are tested in the experiments, and they are different in three aspects:

- Different kinds of intersections, *i.e.*, different structures of conflict zones, are tested as shown in Figure 3.
- Different traffic densities, defined as the average numbers of vehicles arrived at the intersection from one lane, are tested.
- Different group sizes, defined as the numbers of vehicles being scheduled at a time, are tested.

To train our PPO model, we create our own reinforcement learning environment according to the formulation in Section III-B and apply training configuration in [10]. However, to boost the performance of PPO, we also adopt the recommendations in [16]. The modifications are as follows:

- To accommodate different environment complexities, we adjust dimensions of both policy and value networks for the different structures of conflict zones.
- We use an orthogonal initializer for network weight initialization to stabilize training.
- We find that using a higher gradient clipping threshold $\epsilon = 0.5$ gives better performance.

We compare the trained PPO models with an improved greedy (iGreedy) policy within different environments. Note that, it is challenging to compare with other approaches due to the underlying features of the graph-based intersection model and the deadlock constraint. The iGreedy policy is a greedy-based scheduling policy that schedules an operation once it

TABLE I
LIST OF ENVIRONMENTS

	Structure of Conflict Zones	Traffic Density	Group Size	Experiment		
				1	2	3
E_1	I_A	0.3	10	√		√
E_2	I_A	0.7	10	√	√	√
E_3	I_B	0.3	10	√		√
E_4	I_B	0.7	10	√		√
E_5	I_A	0.7	30		√	
E_6	I_C	0.3	10			√
E_7	I_C	0.7	10			√
E_8	I_D	0.3	10			√
E_9	I_D	0.7	10			√

is presented in the current action space $A(t)$. Also, to meet the deadlock constraint in intersection scheduling with conflict zones, the iGreedy policy first checks each operation (by the verification approach in [3]) before actually performing the operation. If an operation may produce a deadlock, then it will not be chosen, *i.e.*, the iGreedy policy chooses an operation which guarantees deadlock-freeness. By the setting, the iGreedy policy also guarantees deadlock-freeness. To evaluate the solution quality, we also use constraint programming and Google OR-Tools [17] to compute optimal solutions.

For our learning-based policy and the iGreedy policy, the experiments are performed on a machine with 8 Intel Core i7-9700 CPUs and 1 NVIDIA GeForce RTX 2080 Ti GPU. For the constraint programming for optimal solutions, the experiments are performed on a machine with Intel Xeon Gold 5218 Processor with 16 physical cores and 128 logical cores.

B. Experimental Results

In the following sections, we present the experimental results and report the gaps to the objective of an optimal solution, *i.e.*,

$$\frac{W - W^*}{W^*},$$

where W is the objective of an output solution, and W^* is the objective of an optimal solution. The training environments for the main experiments are listed in Table I, including different structures of conflict zones, different traffic densities, and different group sizes. Regarding runtime, the training of our learning-based policy takes about 2 to 6 hours to converge in most cases, depending on the complexity of training intersections. However, if we increase the group size in the training environment, the training time gets longer. For example, the training time in the environment E_5 (30 vehicles) is about 4 times more than that in the environment E_2 (10 vehicles). As for evaluation (scheduling), both of our learning-based policy and the iGreedy policy can output solutions within seconds in most cases. Even with complex structures of conflict zones like I_D, they can also output solutions within 30 seconds. On the other hand, even with a more powerful machine, the constraint programming for optimal solutions takes up to hours (for example, more than 8 hours for an environment with 4 conflict zones and 30 vehicles). We believe that the constraint programming can only be used for the evaluation of solution quality, not for real-time usage.

TABLE II
RESULTS (GAPS TO OPTIMAL OBJECTIVES) OF EXPERIMENT 1: TRAINING ENVIRONMENT AND TRAFFIC DENSITY

	E_1	E_2	E_3	E_4
Intersection	I_A	I_A	I_B	I_B
Traffic Density	0.3	0.7	0.3	0.7
Group Size	10	10	10	10
PPO(E_1)	62.4%	25.0%	43.7%	20.3%
PPO(E_2)	43.8%	**22.6%**	55.3%	27.9%
PPO(E_3)	171.8%	60.8%	23.1%	16.8%
PPO(E_4)	171.1%	61.8%	25.9%	**14.1%**
iGreedy	**29.2%**	43.8%	**12.0%**	17.5%

TABLE III
RESULTS (GAPS TO OPTIMAL OBJECTIVES) OF EXPERIMENT 2: GROUP SIZE

	E_2	E_5
Intersection	I_A	I_A
Traffic Density	0.7	0.7
Group Size	10	30
PPO(E_2)	**22.6%**	100.9%
PPO(E_5)	43.4%	82.8%
iGreedy	43.8%	83.9%
PPO(E_2) + Grouping	**22.6%**	**76.1%**

1) Experiment 1: Training Environment and Traffic Density: In this experiment, we investigate the impact of different training environments to the PPO model and consider four environments E_1, E_2, E_3, E_4 with different structures of conflict zones and different traffic densities. The results are listed in Table II which presents the gaps to the objective of an optimal solution. In the table, PPO(E_k) denotes that the PPO model is trained with the environment E_k, and, for each environment, the best result is in bold. We first focus on the best PPO models in different environments. Most of the best PPO models (among all PPO models) are trained in the same environments, except the best PPO model for E_1 is PPO(E_2). This indicates that a learning-based policy may not be generalizable to different structures of conflict zones or different traffic densities: the training environment plays an essential role and needs to be chosen carefully. We then compare the PPO models with the iGreedy approach. The PPO models have better performance and outperform the iGreedy approach when the traffic density is higher (E_2 and E_4), and the iGreedy policy has better performance when the traffic density is lower (E_1 and E_3).

2) Experiment 2: Group Size: In this experiment, we consider different group sizes. The results are listed in Table III which also presents the gaps to the objective of an optimal solution. The PPO models can outperform the iGreedy approach, but they actually do not perform well (the gap to the optimal solution is up to 82.8%) with a larger group size. Another disadvantage of training a PPO model with a larger group size is its training time. As mentioned before, PPO(E_5) takes about 4 times more training time than PPO(E_2) to complete the same number of update steps.

The results of the grouping strategy are also listed in Table III. We can improve the performance of PPO(E_2) from 100% gap to the optimal solution to 75.1% gap to the optimal solution. Especially, the performance is even better than

TABLE IV
RESULTS (GAPS TO OPTIMAL OBJECTIVES) OF EXPERIMENT 3: OVERALL COMPARISON

	E_1	E_3	E_6	E_8
Intersection	I_A	I_B	I_C	I_D
Traffic Density	0.3	0.3	0.3	0.3
Group Size	10	10	10	10
PPO	43.8%	25.9%	40.2%	**14.3%**
iGreedy	**29.2%**	**12.0%**	**26.1%**	17.9%

	E_2	E_4	E_7	E_9
Intersection	I_A	I_B	I_C	I_D
Traffic Density	0.7	0.7	0.7	0.7
Group Size	10	10	10	10
PPO	**22.6%**	**14.1%**	**28.0%**	**20.6%**
iGreedy	43.8%	17.5%	29.3%	26.1%

PPO(E_5), demonstrating that we can train a model with a smaller group size, apply the grouping strategy to a stream of vehicles, and then apply the trained model to each group one by one.

3) Experiment 3: Overall Comparison: We summarize the best results of all PPO models in Table IV. As mentioned before, the PPO models perform better when the traffic density is higher, and the iGreedy policy performs better when the traffic density is lower. The problem of intersection management is more critical when the traffic density is higher, as it creates more serious congestion. We believe that a hybrid approach, which switches from a simple rule-based policy (such as the FCFS policy or the iGreedy policy) to a learning-based policy when the traffic density becomes higher, can be a good solution in practice.

V. CONCLUSION

In this paper, we proposed a reinforcement-learning-based methodology to train a centralized intersection manager. We transformed the intersection scheduling problem to the JSSP with additional constraints. We modeled the scheduling procedure as an MDP and trained the agent with the PPO. Experimental results showed that the learning-based intersection manager is especially effective with high traffic densities. This paper is the first work in the literature to apply reinforcement learning on the graph-based intersection model. The proposed methodology can flexibly deal with any conflicting scenario and indicate the applicability of reinforcement learning to intelligent intersection management.

Potential future directions include training acceleration, other grouping strategies (preventing training with a large group size), and cross-model scheduling metrics. The first two directions are to address the issue that training may take a long time when the number of vehicles and conflict zones increases. The third direction is to achieve fair and comprehensive comparison between different intersection models. Another relevant research topic is the management and coordination of multiple intersections based on reinforcement learning.

REFERENCES

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

[2] K. Yang, S. I. Guler, and M. Menendez, "Isolated intersection control for various levels of vehicle technology: Conventional, connected, and automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 109–129, 2016.

[3] Y.-T. Lin, H. Hsu, S.-C. Lin, C.-W. Lin, I. H.-R. Jiang, and C. Liu, "Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles," *ACM Transactions on Embedded Computing Systems*, vol. 18, no. 5s, 2019.

[4] M. A. Guney and I. A. Raptis, "Scheduling-based optimization for motion coordination of autonomous vehicles at multilane intersections," *Journal of Robotics*, vol. 2020, 2020.

[5] M. Khayatian, M. Mehrabian, E. Andert, R. Dedinsky, S. Choudhary, Y. Lou, and A. Shirvastava, "A survey on intersection management of connected autonomous vehicles," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 4, pp. 1–27, 2020.

[6] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 597–12 608, 2020.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

[8] Y. Wu, H. Chen, and F. Zhu, "DCL-AIM: Decentralized coordination learning of autonomous intersection management for connected and automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 103, pp. 246–260, 2019.

[9] G. Li, J. Wu, and Y. He, "HARL: A novel hierachical adversary reinforcement learning for automoumous intersection management," 2022, arXiv:2205.02428.

[10] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to dispatch for job shop scheduling via deep reinforcement learning," in *International Conference on Neural Information Processing Systems*, 2020, pp. 1621–1632.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.

[12] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, arXiv:1810.00826.

[13] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018, arXiv:1806.01261.

[14] G. F. Jenks, "The data model concept in statistical mapping," *International Yearbook of Cartography 7*, pp. 186–190, 1967.

[15] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[16] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters in on-policy reinforcement learning? A large-scale empirical study," in *International Conference on Learning Representations*, 2020.

[17] L. Perron and V. Furnon, "OR-Tools," Google. [Online]. Available: https://developers.google.com/optimization/