







cuRL: A Generic Framework for Bi-Criteria Optimum Path-Finding Based on Deep Reinforcement Learning

Chao Chen , Lujia Li , Mingyan Li , Ruiyuan Li , Zhu Wang , Fei Wu, Chaocan Xiang 

Abstract—Traditional path-finding studies basically focus on planning the path with the shortest travel distance or the least travel time over city road networks. In recent years, with the increasing needs of diverse routing services in smart cities, the bi-criteria optimum path-finding problem (i.e., minimizing path distance and optimizing extra cost or utility according to users' preference) has drawn wide attention. For instance, in addition to distance, the previous studies further find routes with more scenery (utility) or less crime risk (cost). However, existing works are scenario-oriented which optimize specific cost or utility, ignoring that the routing planner should be universal to deal with both cost and utility in different real-life scenarios. To fill this gap, this paper proposes a generic bi-criteria optimum path-finding framework (cuRL) based on deep reinforcement learning (DRL). Specifically, we design a novel state representation and reward function for the DRL model of cuRL to overcome the challenges that 1) the cost and utility should be optimized with minimal path distance in a unified manner; 2) the diverse distributions of cost and utility in various scenarios should be well-addressed. Then, a transition preprocessing method is proposed to enable the efficient training of DRL and avoid detours. Finally, simulations are performed to verify the effectiveness of cuRL, where two criteria (i.e., solar radiation and crime risk) are modelled based on the real-world data in downtown New York. Comparing with a set of baseline algorithms, the evaluation results demonstrate the priority of the proposed framework for its generality.

Index Terms—intelligent transportation systems (ITS), route planning, cost and utility, deep reinforcement learning.

This work was supported by the National Natural Science Foundation of China under Grant 62172066, Grant 61872050, Grant 62203077, and Grant 62202070, the Natural Science Foundation of Chongqing under Grant CSTB2022NSCQ-MSX1029, the Special Research Funding for Chongqing Postdoctoral Researchers under Grant 2021XM2015, the Postdoctoral Science Foundation of China under Grant 2022M710523, and the Fundamental Research Funds for the Central Universities under Project No. 2022CDJXY-020. (Chao Chen and Lujia Li are co-first authors.) (Corresponding author: Mingyan Li.)

Chao Chen is with the State Key Laboratory of Mechanical Transmission (Chongqing University), and also with the College of Computer Science, Chongqing University, Chongqing 400044, China (Email: cshaochen@cqu.edu.cn).

Lujia Li, Mingyan Li, Ruiyuan Li, and Chaocan Xiang are with the College of Computer Science, Chongqing University, and also with the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China (Email: lilujia@cqu.edu.cn; limy2021@cqu.edu.cn; ruiyuan.li@cqu.edu.cn; xiang.chaocan@gmail.com).

Zhu Wang is with the Department of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China (Email: wangzhu@nwpu.edu.cn).

Fei Wu is with the College of Mechanical and Vehicle Engineering, Chongqing University, and also with the State Key Laboratory of Mechanical Transmission (Chongqing University), Chongqing 400044, China (Email: wufeifrank@cqu.edu.cn).

I. INTRODUCTION

PATH-FINDING over spatial road networks is one of the most fundamental yet important planning activities in smart cities [1, 2, 3]. Usually, millions of people rely on mobile GPS navigators and online trip platforms such as Google Maps to find paths to desired destinations, where the core path-finding algorithms mainly focus on minimizing the travel distance or travel time (e.g. [4, 5]). With the increasing needs of diverse routing services and the easy availability of multi-sourced data [6, 7, 8, 9], alternative routing strategies have attracted wide attention during recent years where specific two criteria are considered, including scenery and driving distance [10, 11], crime risk and driving distance [12, 13, 14], quietness and walking distance [15], and so on [16, 17]. This kind of bi-criteria optimum routing strategies plays a significant role in smart mobility and urban routing services. Depending on the application case, the alternative criteria (in addition to the common-used travel distance or travel time) can be regarded as either *cost* or *utility*. For instance, the crime risk in “safe” routes and the noise level in “quiet” routes are regarded as cost to be minimized while the scenery score in “beautiful” routes is regarded as utility to be maximized.

However, most of the existing routing literature is graph-search based and scenario-specific, which focuses on either minimizing cost or maximizing utility and lacks the capacity to optimize both of them. For instance, the traditional shortest distance algorithms (i.e., Dijkstra [4]) cannot be applied to the one searching for largest-utility paths [18, 19]. As a matter of fact, there are quite a few alternative criteria that can serve as both cost and utility in different scenarios according to users' preference. Taking the criterion of solar radiation as an example, the path with less ultraviolet radiation is normally preferred by pedestrians [20] while more solar radiation is needed when planning the path for solar-powered vehicles [21]. For the criterion of crime risk, citizens desire to find a safe path with minimal risk [12] while policemen on patrol prefer an unsafe path [10]. For this kind of criteria, applying two different optimization methods will introduce significant deployment cost and complexity. Therefore, a generic bi-criteria optimum routing framework needs to be explored to fit various criteria, especially the ones could be cost and utility in different scenarios.

Motivated by the flexible reward mechanism and powerful learning ability of deep reinforcement learning (DRL),

the DRL-based routing becomes a promising solution. Recently, DRL has been widely-utilized in numerous sequential decision-making research fields like robot control, automatic transmission, and etc. [22, 23, 24, 25]. Coincidentally, the path-finding is naturally a sequential decision-making process, where the current routing decision has a great impact on the future moves towards the destination. Therefore, the recent development of DRL has shown the possibility and capability of path planning [12, 26, 27], where the agents learn routing policies from the deep neural networks (DNN) via *trial-and-error* experiences. Better yet, DRL-based methods can work in a multi-objective way, where the reward should be elaborately designed to integrate multiple objectives for both cost and utility optimization scenarios. However, with the reward that simply combines multiple objectives (i.e., distance as well as alternative criteria), the DRL model may not converge unless the action space is fully-explored. As a result, the agent cannot reach its destination with a cyclic path. Therefore, the existing DRL-based routing methods (e.g., [12, 27]) propose their scenario-specific skills to ensure the complete path generation, that are hard to be transferred to other criteria.

This work aims to go beyond the previous DRL-based path-finding algorithms and has the goal of finding a generic bi-criteria routing framework for optimizing path distance and various alternative criteria. To this end, the following challenges should be addressed. Firstly, the cost or utility for alternative criteria as well as distance should be optimized in a unified manner. Secondly, unlike distance, some alternative criteria may lack spatial proximity over adjacent road segments and even are distributed sparsely in the city. For example, the crime risk level of road segments is influenced by a variety of factors like road conditions, public facilities, etc. [28, 29, 30]. Some roads are extremely safe without any crime reports, which makes the distribution of crime risk sparse in the whole city eventually. This heterogeneity makes various criteria hard to be represented and learned in the generic DRL framework. Thirdly, the routing strategy may cause detours when optimizing the cost or utility, which contradicts the principle objective of distance. Therefore, a balance between the distance and the target criterion should be struck.

To address the above challenges, we propose **cuRL**, a generic bi-criteria optimum path-finding framework for both cost and utility optimization based on deep Reinforcement Learning to enable diverse urban routing services. The goal of **cuRL** is to find a low-cost/high-utility path with minimal path distance from an origin to a destination. Generally, our contributions are summarized as follows:

- Unlike the previous alternative routing methods, **cuRL** introduces a novel and generic type of path generation via DRL, where a joint reward function of DRL is proposed to indicate cost and utility in a unified manner.
- Moreover, the reward function is elaborately designed to not only track the accumulative utility or cost on the path but also guide the learning agents towards destinations with minimal path distance. To this end, a novel concept of *effective distance* is introduced to ensure the arrival to destinations, and thus a trade-off mechanism between bi-criteria can be found.

- To ensure the generality, a novel state representation which not only preserves the spatial status but also provides the key information on alternative criterion is designed. This integrated state can promote the efficient training of routing policy in DRL.
- We conduct extensive experiments based on the real-world solar radiation and crime risk data in Downtown New York. Results show that **cuRL** outperforms baseline algorithms, especially for its scenario generality.

The remainder of this paper is organized as follows. Section II firstly introduces the related work. Then, Section III formally formulates the bi-criteria optimum routing problem. After that, the detail of **cuRL** is provided in Section IV and Section V presents our experimental results. Finally, Section VI concludes this paper and discusses the future work.

II. RELATED WORK

In this section, we discuss the related literature from the perspectives of routing for alternative criteria and routing with reinforcement learning, respectively.

A. Routing for Alternative Criteria

There are plenty of studies on route planning that optimize various alternative criteria [11, 13, 14, 15, 16, 31, 32, 33]. Generally, the efforts can be categorized into cost-minimizing and utility-maximizing. The metric of cost or utility in the real world embodies the beauty score, safety/risky score, happiness score, etc., which fortunately can be obtained based on multi-source data (e.g., street view images and human behaviors data) with the development of smart sensing, image processing, and GPS technology. To optimize the specific alternative criterion, these works devote to the cost/utility modelling for the path-finding procedure.

Specifically, Zheng et al. [11] model the utility of beauty based on the geospatial distribution of geotagged images, and then optimize both distance and beauty score under the framework of the Bellman-Ford algorithm [31]. Furthermore, MASSR [32] extracts more relevant information from geo-tagged images and check-ins to further quantify the utility of beauty accurately. Taking into account the actual views on specific route segments, the work [33] obtains scenic scores based on Google Street View (GSV) images and plans personalized routes according to the user's preference. Moreover, Sharker et al. [16] model the criterion of road health as utility by considering the environmental and individual factors on the routes. Concerning the works on cost optimization, SocRoutes [13] models the riskiness of roads by analyzing extremely negative sentiments inferred from Twitter data. It generates routes away from unsafe regions with a constraint detour distance. Based on the civic datasets of criminal activity and city-dwellers mobility traces, the routing strategy proposed in [14] outputs a set of paths that provide the trade-off between distance and safety for users to choose. Additionally, Quercia et al. [15] quantify the happiness score based on a crowd-sourcing platform, where two street scenes are shown for users to vote which one looks more beautiful, quiet, and happy. Based on it, a top- k list of the shortest paths is found, which

are simply re-ranked according to their total utility scores to find the optimum route.

B. Routing with Reinforcement Learning

In recent years, reinforcement learning (RL) based routing has also been widely-explored. For instance, the authors in [34, 35] deal with the vehicle routing problem (VRP) based on DRL. More specifically, Delarue et al. [34] adopt a value-based DRL where the action selection process is formulated as a mixed-integer optimization problem while Duan et al. [35] use the policy-based DRL, where routing policies are established based on graph convolution networks (GCN). Mirowski et al. [26] propose a navigation system based on DRL, where the DNN-based routing policy is trained with GSV images. The policy takes the input of the current location's GSV image as well as the destination represented by nearby landmarks, and outputs the navigation action accordingly. However, it only takes the path distance into consideration and the map size is limited due to the high-dimension of image input. Moreover, Sarker et al. [27] present a multi-criteria route recommendation system that concerns fuel consumption, travel time, and air quality. It firstly predicts these three criteria and weights them with given hyper-parameters. Then, a Q-learning based approach is utilized to learn the routing strategy. However, the state space of this traditional look-up Q-table approach is very large, which is inefficient to maintain. SafeRoute, proposed in [12], introduces a DRL approach to find safe paths in the urban environment. SafeRoute trains a DNN-based policy network which takes the embedding of both the current and destination nodes as the input. In the inference stage, it generates several optional paths and selects the one with the least crime risk.

In summary, most current literature on alternative routing criteria and RL-based routing is criterion-specific, which optimizes either cost or utility. Different from the existing works, this paper proposes a generic DRL-based routing framework, which can apply to both cost-minimizing and utility-maximizing problems, where the reward and state components of DRL are carefully designed so that the diversity of multiple criteria can be addressed.

III. PROBLEM FORMULATION

In this section, some key concepts are formally defined, followed by the problem formulation of the bi-criteria optimum path-finding.

A. Definitions

Definition 1 (Road Network). A road network is a graph $G = (N, E)$ consisting of nodes and edges, where N denotes the set of nodes including intersections and dead-ends; $E \subseteq N \times N$ denotes the set of directed edges.

Definition 2 (Edge Attribute). An edge in the road network (i.e., graph) can carry certain attribute depending on real applications. For instance, the most common edge attribute is the travel distance or the travel time. For the alternative routing criteria, an edge can carry many more kinds of attributes,

including *beautifulness, quietness, happiness, riskiness, and so on*, enabling a much wider spectrum of urban routing services.

Definition 3 (Cost and Utility, **cu**). For the edge attribute, it can be either cost or utility according to the real application scenario. For the edge cost, it is harmful and one may expect to avoid; while for the edge utility, it is enjoyable and helpful and one may expect to gain. To ease the presentation, we simply use **cu** to represent the edge attribute uniformly no matter the edge attribute serves as cost or utility. In addition, we have to emphasize that **cu** in this paper does not indicate travel distance or travel time of an edge.

Definition 4 (Path). A path τ is a sequence of connected edges in the road network from an origin node n_0 to a destination node n_k , denoted as $\tau = \langle e_{0,1}, e_{1,2}, \dots, e_{k-1,k} \rangle$, where $e_{i,i+1}$ indicates the edge from n_i to n_{i+1} .

Definition 5 (Path Distance and Path **cu**). The path distance $d(\tau)$ of a path $\tau = \langle e_{0,1}, e_{1,2}, \dots, e_{k-1,k} \rangle$ is defined as the total edge distance of all its included edges. Similarly, the path **cu** of τ is defined according to the equation: $cu(\tau) = \sum_{i=1}^k cu(e_{i-1,i})$, where $cu(e_{i-1,i})$ denotes the cost or the utility on the edge $e_{i-1,i}$. It should be noted that, unlike the common edge attribute such as the travel distance that every edge carries, **cu** can be zero in some edges, depending on the physical meaning of the specific attribute.

Definition 6 (Bi-criteria Optimum Path). For a given OD (Origin-Destination) pair, the bi-criteria optimum path is defined as the one with the minimal path distance, and the minimal path cost **c** (or the maximal path utility **u**).

Definition 7 (Routing Action). The routing action a in this paper is defined as the direction of the next road segment chosen by the pedestrian/driver at every intersection on the map. All of the directions are divided into eight classes (i.e., North, Northwest, West, Southwest, South, Southeast, East and Northeast, each covering 45 degrees in coordinate system), each of which corresponds to a type of routing action.

B. Problem Statement

In this section, we propose our Bi-Criteria Optimum Path-finding (BiCpath) problem over the road network, which is a typical routing problem as:

Given:

- 1) A road network $G(N, E)$ of the city with enriched attribute weights of criteria;
- 2) A user query with the origin and destination node pair (N_o, N_d) as well as the targeted criterion (i.e., the cost or utility) according to the user's preference.

Output: A path τ from N_o to N_d in the road network adhering to the following objectives:

- 1) *Principle objective*: minimizing the path distance $d(\tau)$;
- 2) *Sub-objective*: maximizing/minimizing the path **cu** $cu(\tau)$ based on the targeted criterion.

Formally, the **overall objective** is **maximizing** the linear combination of the above two objectives, \bar{r} , as follows:

$$\bar{r} = -d(\tau) + \alpha * cu(\tau) \quad (1)$$

where α is a hyper-parameter to weight the **cu** sub-objective. In addition, the *sign* of α indicates whether **cu** serves as cost or utility, i.e., α is set to *positive* for utility and *negative* for cost. Note that the trade-off between the principle objective and sub-objective can be achieved by adjusting α . Therefore, we propose an automatic determination method for α for both objective trade-off and framework generality, which is detailed in Section IV-C.

C. RL Formulation for BiCpath Problem

As a matter of fact, the path-finding problem is naturally a sequential decision problem, which can be solved by the RL framework [12]. Thus, the goal of BiCpath problem is transferred to finding a decision-making policy π using RL, which maximizes \bar{r} defined as Eq. 1.

The RL is formulated as a sequential Markov Decision Progress (MDP), represented by the five-tuple $\langle S, A, P, R, \gamma \rangle$. $s \in S$ denotes the state of agent in the environment. A is the action space which contains all the actions that can be taken by the agent. In our problem, A is an 8-element tuple and the routing action $a \in A$ is described in Definition 7. $p \in P$ is the transition probability for the agent to transit to the next state s_{t+1} after taking action a_t given the state s_t . $r_t \in R$ indicates the immediate reward after the agent takes an action according to the policy π . Once an action is completed, the experience corresponding to that action called the transition denoted as $Tr = (s_t, a_t, r_t, s_{t+1})$ is generated to record for learning.

In the RL-based BiCpath problem, we aim to optimize the cumulative reward G , also known as return, which accumulates the immediate reward achieved by each sequential routing action in the path. This return guarantees the long-term path optimization as formally shown in Eq. 2.

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{T-t} r_T = \sum_{k=t}^T \gamma^{k-t} r_k \quad (2)$$

where t and T refer to the current step and the terminal step, respectively; $\gamma \in [0, 1]$ is the discount rate, which is used to evaluate the importance of reward generated by future steps.

Instead of optimizing the path as shown in Eq. 1, the RL-based routing agent aims to find an optimal policy π_* that achieves the maximal long-term reward G_t . To this end, the state-action function value $Q_\pi(s, a)$, also named Q-value, is used to capture the quality of policy which takes the action of a under the state of s , denoted as,

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[G_t | s_t = s, a_t = a] \\ &= E_\pi \left[\sum_{k=t}^T \gamma^{k-t} r_k \middle| s_t = s, a_t = a \right] \end{aligned} \quad (3)$$

where $E_\pi[\cdot]$ calculates the expected value with respect to the stochastic policy π . The problem of finding optimal policy π_* is actually the problem of finding optimal Q-values $Q_*(s, a)$ because the agent will intuitively choose the optimal routing action a_* with the maximum Q-value as shown in Eq. 4. Based on the Bellman equation, the target optimal Q-value can be computed according to Eq. 5.

$$a_* = \arg \max_{a'} Q(s, a') \quad (4)$$

$$Q_*(s, a) = E \left[r_t + \gamma \max_{a'} Q_*(s_{t+1}, a') | s_t = s, a_t = a \right] \quad (5)$$

Based on this Q-value, multiple objectives (i.e., path distance and path **cu**) are combined and optimized in a unified manner. However, with this integrated Q-value, the policy may fail to converge with under-exploration, which is common in most practical scenes. As a result, the agent may cause many unnecessary detours and even cannot reach its destination. Moreover, the criteria heterogeneity should also be addressed in terms of generality. Therefore, we present the **cuRL** scheme concerning solving these challenges in the next.

IV. METHODOLOGY

In this section, we first provide the **cuRL** framework overview, then introduce the details on each component of **cuRL** and how **cuRL** fulfills the model training and inference.

A. Framework Overview

Due to the capability of handling high-dimensional input [22, 36], DRL is utilized in **cuRL** to deal with the complex routing environment. In this way, the DNN-based Q-value instead of tabular Q-value can be learned via trial-and-error experiences. The overall framework of **cuRL** is illustrated in Fig. 1. When training the DRL, we utilize a double deep Q-network (DDQN [36]) to decouple the selection and evaluation of routing actions through two networks (i.e., the Q-network and the target network). The central-agent with DDQN learns the optimal policy to maximize the accumulative reward when interacting with the environment of road network (i.e., Graph Environment). There are three main components (i.e., State Representation, Reward Computation and Transition Preprocessing) that are carefully designed for interactions between the agent and the environment, detailed as follows.

- *State Representation.* The state in MDP is determined in this component to track the spatial status of the agent on the road network. Moreover, it provides essential information on **cu** to deal with criteria heterogeneity.
- *Reward Computation.* This component computes the immediate reward through the reward function. It is designed to include two parts, i.e., the reward for guiding the agent towards the destination and the reward for optimizing the **cu** sub-objective.
- *Transition Preprocessing.* We design a transition preprocessing component for transitions generated by agents before being fed into the DDQN in the training stage. The transition preprocessing component works with two functions, namely *arrival detection* and *reward modification*. The two functions are both designed to make the training more efficient.

B. Components of **cuRL**

1) *State Representation:* The state s represents the current status of an agent in the road network. Foremost, it should contain the location information about the current node N_c and the destination node N_d , telling the agent where it is and

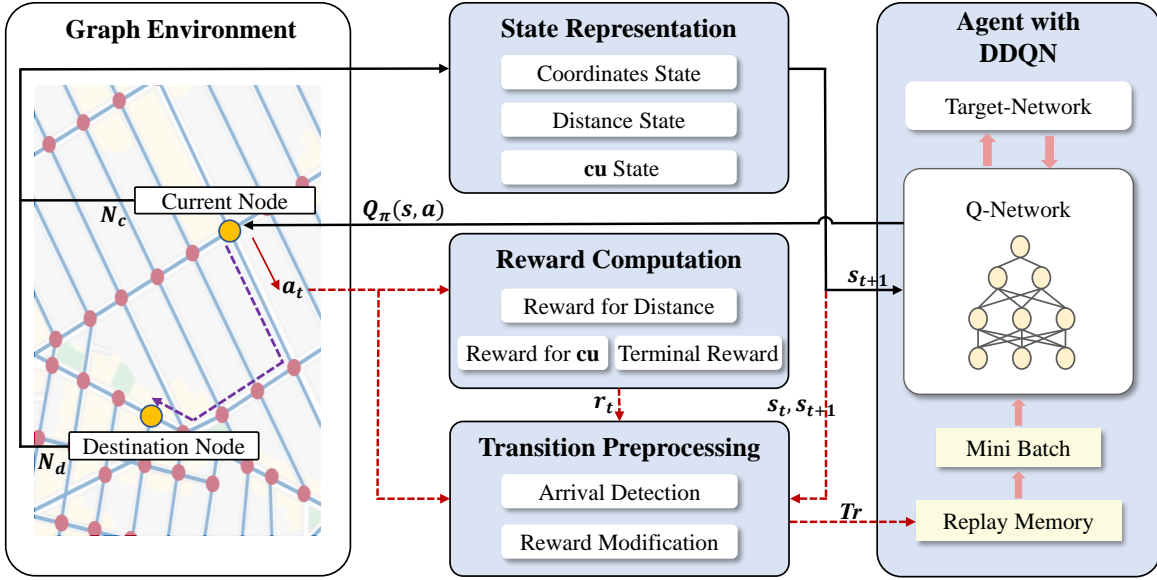


Fig. 1. The framework of **cuRL**. The arrow lines refer to the data flows in the training and inference stages (The black solid lines are used for both the training and inference stages while the red dash lines only the training stage).

where to head. Intuitively, there are two ways to represent the location of a node in the state in RL framework, i.e., one-hot encoding and latitude-longitude coordinates [37].

One-hot encoding: When the node is represented by the simple one-hot encoding [37], it is advantageous that every two nodes can be far enough in the representation space for better distinguishment by the DNN. However, it suffers from the following two major drawbacks. First, the dimension of one-hot encoding increases with the scale of the road network, making the DNN more complex with dimension disaster. Second, all geographical features in the road network are completely lost. As a result, the learning agent cannot make use of the experience of already learned samples when training new samples, which significantly slows down the policy convergence since the DDQN works in a sample-inefficient *memorizing* way, rather than a *learning* way.

Latitude-longitude coordinates: Duo to the above-mentioned drawbacks, we intend to abandon one-hot encoding and utilize the information of latitude-longitude coordinates. When using latitude-longitude coordinates to represent the state for a node, its nearby nodes in the road network will also have close representations, potentially resulting in choosing the *same best* actions for itself and its geographical neighbours. Such a result is suitable for distance-orientated path-finding problems, nevertheless, may not fit to the bi-criteria optimum path-finding problem. Taking the case of minimizing the solar radiation as the example (as shown in Fig. 2), *agent*₁ and *agent*₂ have two close origin nodes (i.e., N_1 and N_2) and the same destination node (i.e., N_d). τ_1^* and τ_2^* represent the bi-criteria optimum paths for these two OD pairs, respectively. Even though N_1 and N_2 are extremely close in the road network, it is obviously that *agent*₁ and *agent*₂ have totally different optimal action sequences because the solar radiation distributions on edges linked to them are completely different. In this case, if we simply use latitude-longitude

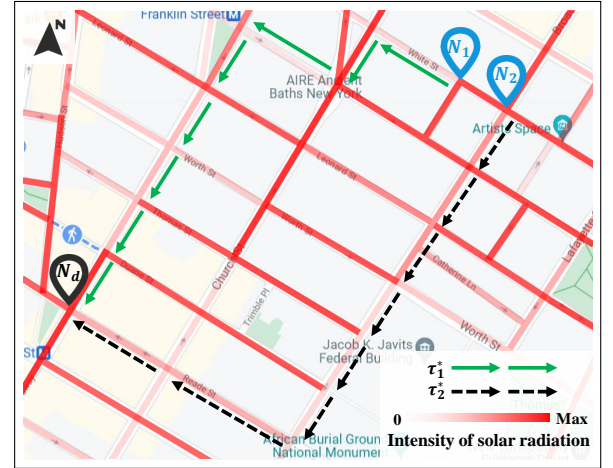


Fig. 2. An example of two optimal routes for bi-criteria optimum path-finding with the **cu** of solar radiation. With the same destination, two close original nodes lead to different optimal action sequences due to the discontinuous distribution of **cu**.

coordinates in the state representation, the differences of solar radiation distributions cannot be well-distinguished, as a result, *agent*₁ and *agent*₂ may take the *same* actions due to the closeness of the state representation. Therefore, the node state representation should be carefully designed to address the issue of **cu** distinguishment.

Based on the above analysis, we propose a novel state representation that not only provides *enough* graphical feature information for the agent to learn, but also makes *enough* discrimination among adjacent nodes. Specifically, the designed node state consists of the following three parts:

- **Coordinates State.** Latitude-longitude coordinates of the current node are still preserved because such data provide the most intuitive spatial information.
- **Distance State.** We also integrate a priori knowledge of

the shortest distance in our representation. The distance state is defined as the shortest distances from all of the current node's neighboring nodes to the destination node in the road network. Distance state also helps the agent to move to the destination with the minimal distance potentially.

- **cu State.** It is used to represent the state for the sub-objective, helping the agent to learn the spatial feature of the **cu**. We use the weights of **cu** of all edges linked to the current node to represent it.

Overall, the state s is defined by a two-tuple (X_c, X_d) , where the two elements X_c, X_d track the representation of current node N_c and destination node N_d , respectively. Note that X_c contains all the three types of states listed above while X_d omits the distance state.

Based on the current state s , the central agent chooses a routing action from action space A . If there is no linked road in the direction of the chosen action, the graph environment will select an action again until there exists a road. After the agent takes an action from a node to the next node in the road network, an immediate reward r will be returned for this action, which should be carefully designed in the next to ensure the efficient learning of **cuRL**.

2) *Reward Computation:* Recall that the objective of **cuRL** is twofold, namely path distance minimization and **cu** optimization, formulated as Eq. 1. Therefore, we design a reward function that consists of two parts, r_{dis} denoting the principle objective and r_{cu} denoting the sub-objective.

Intuitively, to optimize the two objectives, using edge distance and edge **cu** to represent r_{dis} and r_{cu} respectively may be a feasible solution. Unfortunately, simple combination of edge attributes does not have the ability to guide the agent to the destination. To solve this issue, we propose a new concept of *effective distance* to indicate the principle objective r_{dis} instead of using the original edge distance. Effective distance is designed according to a heuristic rule which helps the agent to generate paths with the approximate shortest distance. The definition of effective distance is shown as follows.

Definition 8 (Effective distance). *Effective distance (dis_{eff}) measures how far the agent can move towards the destination by an action, which is defined as the difference between the distance along the origin-destination (N_oN_d) direction and the normal direction \vec{n} of N_oN_d (i.e., $\vec{n} \perp N_oN_d$). The detailed calculation of dis_{eff} is described in Fig. 3.*

Effective distance considers both path distance and action direction. An action that introduces a bigger road segment distance and a smaller angle to the OD direction will have a bigger effective distance, which could make the agent move closer to the destination more effectively. With such heuristic design, the agent is expect to learn to generate a path to the destination with a minimal path distance.

The effective distance based reward for the principle objective (r_{dis}) is computed according to Eq. 6.

$$r_{dis} = norm_{-1,1}(dis_{eff}) \quad (6)$$

where the function $norm_{a,b}(c)$ normalizes c and makes it fall into $[a,b]$. The normalization is based on the equation

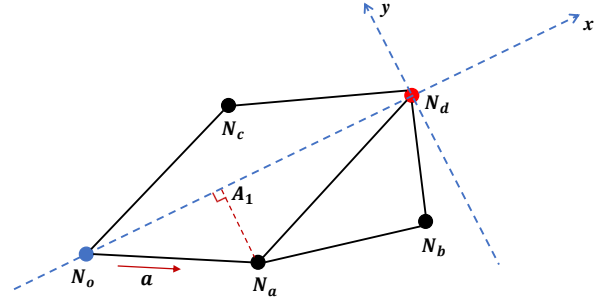


Fig. 3. An example of calculating effective distance. Suppose an agent traveling from origin N_o to destination N_d takes the action a in N_oN_d direction at node N_o on the graph. Take node N_d as the coordinate origin, the N_oN_d direction as the x -axis, and the normal direction \vec{n} as the y -axis to establish a coordinate system. Then draw vertical lines of x -axis from node N_a at point A_1 . For the action a , $dis_{eff} = (|N_oN_d| - |A_1N_d|) - |A_1N_a|$, where $|N_oN_d| - |A_1N_d|$ represents the distance of the agent moving towards N_oN_d direction and $|A_1N_a|$ evaluates the distance that deviates from N_oN_d direction.

$\frac{c_{norm}-a}{b-a} = \frac{c-c_{min}}{c_{max}-c_{min}}$ where c_{norm} is the output of the function $norm_{a,b}(c)$. c_{max} and c_{min} are the maximum and minimum one of c , respectively. a and b are set to -1 and 1, since the effective distance can be positive or negative.

Reward for the sub-objective (r_{cu}) is simply computed based on the original edge **cu**, shown in Eq. 7.

$$r_{cu} = norm_{0,1}(cu_a) \quad (7)$$

where cu_a is the **cu** weight on the edge determined by the agent's action. r_{cu} is always non-negative so it is normalized to the range from 0 to 1.

To combine r_{dis} and r_{cu} , we also use the hyper-parameter α to trade-off (i.e., $r_{dis} + \alpha * r_{cu}$), just in line with the hyper-parameter in the overall objective Eq. 1. When α is set to 0, **cu** is not taken into consideration and the BiCpath problem is degraded to the shortest-distance routing problem.

Besides, to make the DRL model converge faster, an additional terminal reward r_T with a much bigger value than r_{dis} and r_{cu} is introduced, if the agent arrives at the destination node. Consequently, the final immediate reward can be calculated according to Eq. 8.

$$r = r_{dis} + \alpha * r_{cu} + \mathbb{I} * r_T \quad (8)$$

where \mathbb{I} indicates whether the agent arrives at the destination node (set to 1) or not (set to 0).

3) *Transition Preprocessing:* There is a phenomenon that in a number of cases, the agent cannot arrived at the destination, just wandering in the road network or cycling in a loop in the model inference stage. To alleviate such issue, we design a transition preprocessing component in **cuRL**, which consists of *arrival detection* and *reward modification*.

We firstly analyzes the reason of this phenomenon. Since DDQN is a value-based RL method, the samples of one-step transitions are used, where the set of transitions $\{Tr = (s_t, a_t, r_t, s_{t+1})\}$ will be stored in the replay memory by the agent. These transitions are collected in the real path-finding trials for Q-network training, based on which the agent uses the ϵ -greedy method to select the routing action. ϵ -greedy

method chooses the optimal action with the highest neural network output and explores the other actions with a probability of ϵ (more details can be found in Section IV-C). This can guarantee the action exploration to find a more satisfactory path. However, there are some under-explored routes if the scale of the road network is large. This may lead to unstable routing policy and loop paths that cannot arrive at destinations. As a result, the one-step transitions in this kind of routes may further confuse the Q-network training.

Therefore, we propose the arrival detection function, which filters the set of transitions $\{Tr = (s_t, a_t, r_t, s_{t+1})\}$ using a buffer before they are fed into the replay memory of DDQN. Only transitions of paths that can arrive at their destinations successfully will be removed from the buffer to the replay memory to train the model. Then, the reward modification function is introduced, which is responsible for loop path detection. If all actions within the loop path are not generated with the exploration way in the ϵ -greedy method, their corresponding rewards will be modified using a negative figure. The negative reward gives a penalty to the action in a loop to avoid the agent being trapped in the loop in the later periods. Afterwards, the new transitions are stored in the replay memory. Due to the small learning rate, DDQN parameters will be updated in a smooth way without violent fluctuations even when the rewards of new transitions are negative.

C. cuRL Training and Inference

Figure 1 illustrates the framework of **cuRL** with the data flows in both training and inference stages. To make the training process clear, we further provide the corresponding pseudo-code in Alg. 1. We also recommend readers to refer to the framework figure and pseudo-code throughout this section.

In RL, the training period consists of multiple episodes and each episode corresponds to an algorithm instance. In our algorithm, an episode refers to the process of an agent from a random origin to a random destination within the preset maximum step T . If the agent cannot reach the destination after the given maximum step, it will also terminate at this step and the next episode will start. At each step in an episode, feeding with the state vector $s = (X_c, X_d)$, the agent takes an action with ϵ -greedy strategy (Lines 6~10 in Alg. 1) to balance the exploration and the exploitation. For the exploration, the agent will choose a random action in the action space; while for the exploitation the agent will take the action having the highest Q-value generated by DDQN network. The exploitation utilizes experience that the agent learnt from the history data while the exploration chooses the action randomly to avoid falling into the local optima and generate the potentially better routing policy. In the first episode, there is just no experience to exploit, thus ϵ is set to 1. As the experience accumulates in the training stage, ϵ gradually becomes smaller as iteration continues (Line 11 in Alg. 1). δ ($\delta < 1$) is a hyper-parameter adjusting the decay speed of ϵ . In the last episodes, ϵ tends to 0 and only the exploitation is utilized.

For convenience, we use t to represent the sequence ID of the current step in an episode in the rest presentation. After an agent takes an action a_t , it gets an immediate reward r_t from

Algorithm 1: Training of **cuRL** based on DDQN

```

1 Initialize replay memory  $M$ , buffer  $B$  in the transition preprocessing component, Q-network and target-network parameters  $\theta$  and  $\theta'$ .
2 for  $episode = 1 : N$  do
3   Initialize the current node  $N_c = N_o$  (the origin node), the destination node  $N_d$ , the current state  $s_t$ .
4   Set  $Flag_{train} = 0$ .
5   for  $t = 1 : T$  do
6     if The random number is less than  $\epsilon$  then
7       Select the action  $a_t$  randomly.
8     else
9       Select the action  $a_t = \arg \max_a Q(s_t, a; \theta)$ .
10    end
11    Update  $\epsilon = \epsilon * \delta$ .
12    Take action  $a_t$  in the environment, get the next node  $N_n$ , the next state  $s_{t+1}$  and the reward  $r_t$ .
13    Store transition  $Tr = (s_t, a_t, r_t, s_{t+1})$  to buffer  $B$ .
14    Update  $N_c = N_n$ ,  $s_t = s_{t+1}$ .
15    if The agent arrived at the destination node then
16      Move all transitions from  $B$  to  $M$ .
17      Set  $Flag_{train} = 1$ .
18    end
19    if The actions of transitions without exploration in  $B$  make up a loop then
20      Modify the rewards of transitions in loop.
21      Move the new transitions from  $B$  to  $M$ .
22      Set  $Flag_{train} = 1$ .
23    end
24    if  $Flag_{train} == 1$  then
25      Sample a minibatch  $M_{batch}$  from  $M$ .
26      Calculate the loss for  $M_{batch}$  with Eq. 10.
27      Update  $\theta$  with Adam [38] optimizer.
28      Synchronize  $\theta' = \theta$  every  $\Delta$  steps.
29      break
30    end
31    if  $t == T$  then
32      Empty the buffer  $B$ .
33    end
34  end
35 end

```

the environment and the state of environment transits from s_t to s_{t+1} . The transition denoted by Tr will be temporally stored in the buffer B of transition preprocessing component, then the arrival detection and reward modification functions begin to work. The transitions in the buffer B will be moved to the replay memory M in DDQN or not, according to the judgment of transition preprocessing component. M is an empty queue with limited length ($|M|$) and will be gradually filled with transitions. Once the number of stored transitions reaches $|M|$, the most earliest stored transitions will be popped out and replaced by the new transitions. The training frequency can be adjustable and here we set it to be the same to the agent's episodes, i.e., we train the agent once for each OD pair. The target-network and Q-network in DDQN have the same

structure with weight parameters θ' and θ , respectively. Target Q-value Q_{target} (Eq. 9) is used to estimate the optimal Q-value defined in Eq. 5. In each training, the Q-network parameters θ and the target-network parameters θ' will be updated, as shown in Lines 24~30 in Alg. 1.

$$Q_{target} = r_t + \gamma Q(s_{t+1}, \arg \max_a Q(s_{t+1}, a; \theta_t); \theta'_t) \quad (9)$$

$$L(\theta) = \frac{1}{|M_{batch}|} \sum_{Tr \in M_{batch}} [Q_{target} - Q(s_t, a_t; \theta)]^2 \quad (10)$$

It is necessary to note that a pre-training stage is essential since it enables the model with the basic capability to reach the destination. In the pre-training stage, the parameter α in the reward function is set to 0, indicating that only the principle objective is optimized. Moreover, the arrival detection is also disabled in the transition preprocessing component to explore the possible paths to the destination. Other settings are exactly the same to the training shown in Alg. 1. After the pre-training stage, α grows gradually to optimize the sub-objective. We set a fixed step size (e.g., 0.01) for the growth of α . For every model with different α s, their performances for optimizing **cu** (the definition of performance metrics is detailed in Section V) are tested separately. The growth rate of model performance will gradually decrease with α increases (e.g., model performance in Fig. 6). If the growth rate reaches a preset threshold (e.g., 5%), we will stop the training and the α at that time is chosen as the result parameter.

As mentioned before, the inference process is shown using black solid arrows in Fig. 1. The agent just takes the action with the highest Q-value step by step until it reaches the destination node. Although the transition preprocessing component certainly works effectively in the training stage, there are still a tiny fraction of OD pairs with which the agent would fall into a loop path. Under such circumstance, an extra operation that activates the ϵ -greedy method again for those OD pairs, is taken to help the agent to get rid of loops.

V. EVALUATION

In this section, we empirically evaluate the performance of the proposed **cuRL**. We introduce the experimental setup, baseline algorithms used for comparison, evaluation metrics, and performance results.

A. Experimental Setup

1) *Data Description*: We conducted experiments using two representative **cus** (i.e., the solar radiation and the crime risk) in the road network of the Downtown Lower Manhattan, New York, US. The road network with 998 nodes and 3326 edges is crawled via the OpenStreetMap [39] which is a well-known crowdsourced platform. The values of **cu** are obtained based on other crowdsourced data, detailed as follows:

- **Solar radiation**. The concept of solar radiation here refers to the *Global Horizontal Irradiance* [40] and the intensity of solar radiation is calculated using the method proposed in [41] which takes into account the blocking

by human facilities on both sides of the street. In different roads of a city at the same time, the solar radiation differs from each other because human facilities like trees and buildings along road sides would block the solar radiation to varying degrees. Moreover, the height, density and arrangement direction of human facilities are also different. Thus, the solar radiation differs from road to road and lacks the spatial proximity. The weights of **cu** in our experiment are assigned using the average intensity of solar radiation within an hour from 11:00 to 12:00. The spatial distribution of such average solar radiation over the road network is shown in Fig. 4(a). As can be seen, for the solar radiation attribute, each edge in the road network carries a certain value of weight.

- **Crime risk**. The crime risk is obtained through the statistics of historical crime data near an edge, reflecting how unsafe the road it is. We refer to the measurement method used in [12] to weight the crime risk of an edge, as detailed in Eq. 11.

$$risk = \frac{\sum_{j=1}^n distance(c_j, x_m)}{l} \quad (11)$$

where n is the number of historical crime reports within the radius of half length of the edge; c_j is the j -th crime report and x_m is the midpoint of the edge; $distance(c_j, x_m)$ calculates the Euclidean distance between c_j and x_m ; l represents the edge length. Crime data is collected from NYC Open Data [42], which include the time, coordinate and type of crimes, etc. The crimes with types of assault, robbery and dangerous weapons are used because we are more concerned about the street-level safety. There are no crimes on some edges occurred in history and the crime risk is set to 0 for such edges. Thus, weights of the crime risk are sparsely distributed over the road network, as shown in Fig. 4(b). As can be observed, compared to the solar radiation attribute, only a small fraction of edges carry a varying degree of the crime risk.

2) *Baseline Algorithms and Evaluation Metrics*: Both distance and **cu** are considered in the routing objective, thus they are naturally used as the evaluation metrics. An algorithm outperforms if it generates paths with shorter distance, bigger **cu** when maximizing **cu** and smaller **cu** when minimizing **cu**. As the comparison, a set of state-of-art algorithms are also adopted, listed as follows.

- **Shortest**. It only considers the distance when routing, generating paths with the shortest distance using the classical Dijkstra algorithm. It is the upper bound of optimizing the distance.
- **Least cu**. It only considers the **cu** minimization which is only for the scenarios that the **cu** serves as cost. It is the upper bound of minimizing the **cu**.
- **SPTH**. It refers to the routing algorithm proposed in [15], which considers both beauty and quietness along the route. The core idea of this algorithm includes: 1) generating the top- k shortest paths; 2) picking the one with the best sub-objective score among k paths. In our evaluation, we use **SPTH** to find the route with the minimal **cu**.

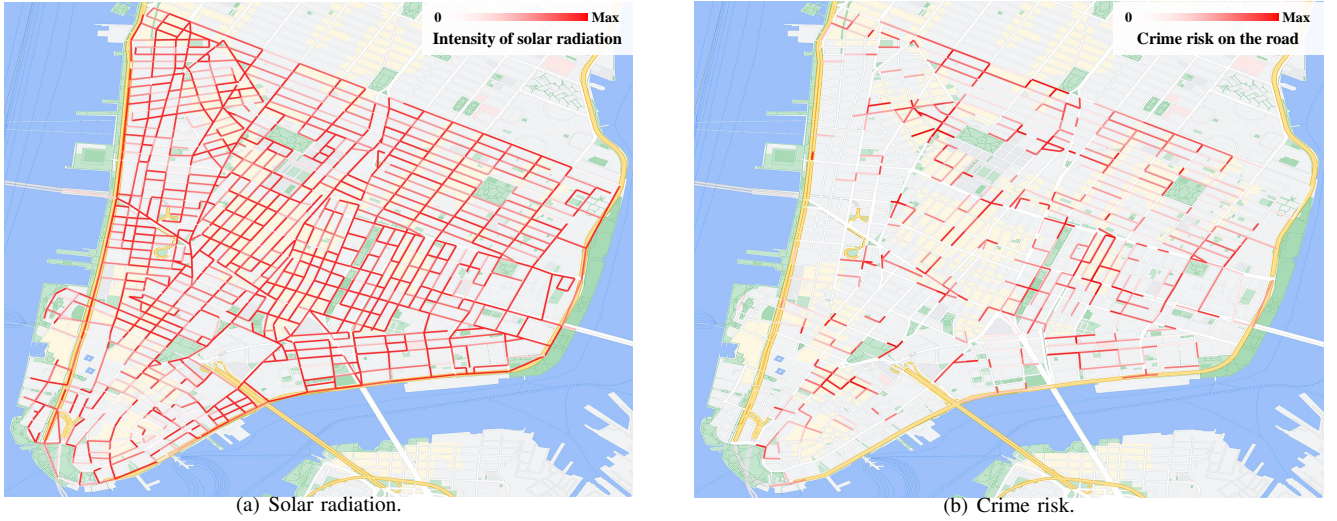


Fig. 4. Distributions of \mathbf{cu} over the road network. A darker color indicates a higher weight value for both figures.

- **2TD-AOP.** It aims to find the most beautiful path in a road network with sparse and time-dependent weights of edge attribute (i.e., beauty) [10]. Thus, it is only used as the comparison for crime risk maximization. In addition, when running **2TD-AOP**, \mathbf{cu} weights are simply set to the same at different times. In other words, the edge time-dependence characteristic is ignored in the evaluation.
- **Reversed \mathbf{cu} .** It first reverses the values of \mathbf{cu} according to $cu_{revi} = \max(\mathbf{cu}) - cu_i$, then find the path with the least value of cu_{rev} using **Least \mathbf{cu}** . We set this algorithm to validate whether the *edge attribute minimizing algorithm* could be transferred to *edge attribute maximizing algorithm* simply by reversing the weight of the edge attribute.

For different application scenarios, the dimensions of evaluation results are diverse. To compare the experimental results more comprehensively, we adopt the ratios of each algorithm's path distance $d(\tau)$ and path \mathbf{cu} $cu(\tau)$ (see Definition 5) to that of **Shortest** to measure the performance. In another word, the path distance and \mathbf{cu} obtained by **Shortest** is used as the reference. The average ratios over all tested OD pairs are regarded as evaluation metrics, called *distance ratio* and *\mathbf{cu} ratio* respectively, as defined in the following equations.

$$distance\ ratio = \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{d(\tau_i)}{d(\tau_{Shortest_i})}, \quad (12)$$

$$cu\ ratio = \frac{1}{|P|} \sum_{i=1}^{|P|} \frac{cu(\tau_i)}{cu(\tau_{Shortest_i})} \quad (13)$$

where P is the set of all tested OD pairs; τ_i is the path obtained by the tested algorithm for the i -th OD pair; and $\tau_{Shortest_i}$ refers to the path for the i -th OD pair obtained by the **Shortest** algorithm. It is obvious that both path distance and \mathbf{cu} ratios found by **Shortest** equal 1. Moreover, the algorithm with the distance ratio much closer to 1 and the \mathbf{cu} ratio farther way from 1 implies the better performance.

According to the distribution of \mathbf{cu} and the features of baseline algorithms, we choose **Shortest**, **Least \mathbf{cu}** and **SPTH**

as baselines when minimizing \mathbf{cu} (both the solar radiation and the crime risk), **Shortest** and **Reversed \mathbf{cu}** when maximizing the solar radiation, and **Shortest**, **Reversed \mathbf{cu}** and **2TD-AOP** when maximizing the crime risk.

3) *Evaluation Environment and Parameter Settings:* All the evaluation experiments are programmed using Python 3.7 with TensorFlow-1.4 and Keras-2.3, and running on a PC with 4 NVIDIA GeForce RTX 2080 Ti GPUs and 192 GB RAM.

In total, 5000 OD pairs with random origin and destination nodes are generated in the road network (i.e., $|P|$ equals 5000). In the training stage, the number of episodes $N = 50000$, the max steps in an episode $T = 100$, the ϵ ranges from 0.5 to 0 during the training process, the discount factor $\gamma = 0.96$. In the DDQN network, the replay memory size $|M| = 200000$, and minibatch size $M_{batch} = 64$, the neural networks of Q-network and target-network are implement using fully connected layers with the hidden layer size of 1024, 512, 128, the synchronize frequency for target-network $\Delta = 300$. In the learning process of neural networks, the learning rate $lr = 0.00001$.

B. Overall Results

By aggregating the evaluation results of different application scenarios, the overall results are shown in Fig. 5. When minimizing the solar radiation and the crime risk, Fig. 5(a) and Fig. 5(b) show that **cuRL** is generally more capable of trading the distance against \mathbf{cu} , compared with the baseline algorithms. Not surprisingly, **Shortest** obtains the path with the best performance in terms of the distance ratio, while **Least \mathbf{cu}** obtains the best \mathbf{cu} ratio, since they aim to optimize either the distance or \mathbf{cu} . To be more specific, when minimizing the solar radiation, **cuRL** finds paths with the distance ratio of 1.04 and the \mathbf{cu} ratio of 0.95, while **Least \mathbf{cu}** obtains paths with the distance ratio of 1.11 and the \mathbf{cu} ratio of 0.86. One unanticipated finding is that **SPTH** ($k = 10$) performs slightly better than **cuRL** in both terms of the distance ratio and the \mathbf{cu} ratio, with the value of 1.01 and 0.94 respectively. One explainable reason for such phenomenon is that the top- k

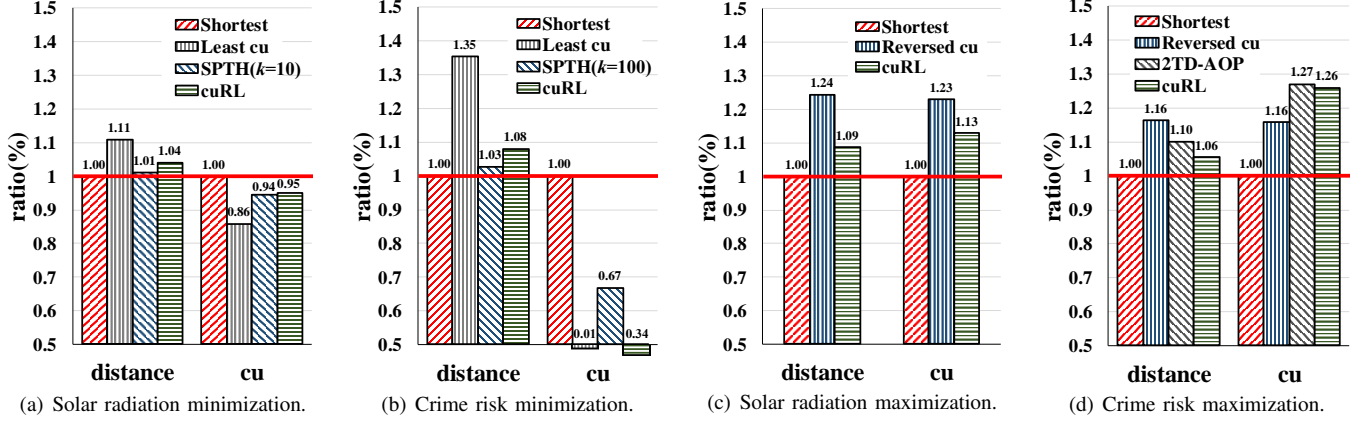


Fig. 5. Overall results of **cuRL** comparing with the baseline algorithms. (a), (b) show the results when minimizing **cu** and (c), (d) show the results when maximizing **cu**.

shortest paths obtained by **SPTH** also include the one with the minimal **cu** by accident. To verify such conclusion, we further provide the result of minimizing the crime risk in Fig. 5(b), in which the crime risk attribute distributes quite *unevenly and irregularly* across the road network. If the value of k is larger, **SPTH** will have more potential to generate paths with less **cu**. However, no matter how to adjust the value of k , **SPTH** cannot outperform **cuRL** both in distance and **cu**. Fig. 5(b) shows the results when k in **SPTH** equals 100, where **SPTH** obtains the path with the distance ratio and the **cu** ratio of 1.03 and 0.67, while **cuRL** achieves much better performance, with the value of 1.08 and 0.34 in distance ratio and **cu** ratio.

When maximizing the solar radiation and the crime risk, as evidenced by results shown in Fig. 5(c) and Fig. 5(d), again, **cuRL** outperforms other baseline algorithms in most of cases. As mentioned before, results obtained by **2TD-AOP** are only compared when maximizing the crime risk. More specifically, when maximizing the solar radiation, **cuRL** achieves the distance ratio of 1.09 and the **cu** ratio of 1.13, much better than that of **Reversed cu** in distance ratio and slightly worse in **cu** ratio. This is because that **Reversed cu** is proposed to maximize **cu** but ignores path distance optimization. We further examine whether **Reversed cu** can still obtain better **cu** ratio than **cuRL** when maximizing the crime risk, with the results shown in Fig. 5(d). As can be observed, **Reversed cu** obtains the paths with the **cu** ratio of 1.16, performing the worst compared to that of **2TD-AOP** and **cuRL**. It is thus safe to draw the conclusion that it is not always feasible to transfer the minimizing problem to a maximizing one by the simple weight-reversing in path-finding. In terms of the **cu** ratio, it is expected that **2TD-AOP** performs the best, with a value of 1.27, however, it fails to obtain a satisfactory result in distance ratio. Worse still, it can only work for the case of sparse edge attribute. In contrast, **cuRL** obtains the bi-criteria optimum paths since it optimizes the distance and **cu** simultaneously.

In conclusion, although **cuRL** is inferior to some comparison algorithms sometimes, it is more generic due to it is more capable of minimizing cost or maximizing utility while maintaining an optimized distance, and working stably under the road network with either dense or sparse edge attributes.

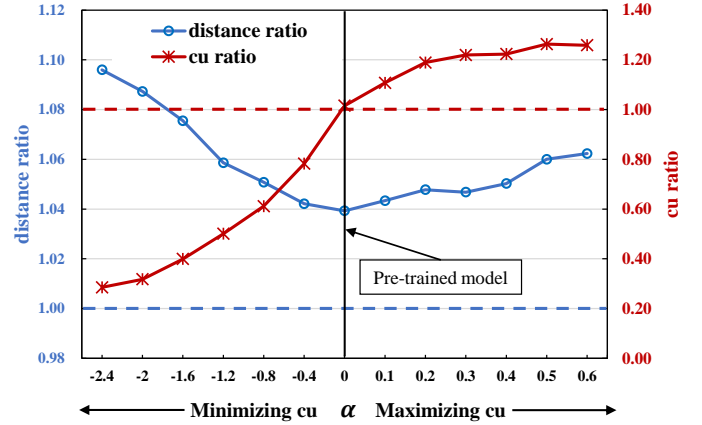


Fig. 6. Evaluation results of crime risk optimization with **cuRL** under different α s in the reward function.

C. Effectiveness of Reward Design

To verify the effectiveness of the reward design in **cuRL**, we examine the performance of the model under different α s during the training process using the example of optimizing the crime risk. As discussed in Section IV-C, the pre-training when setting α to 0 is necessary. As can be observed in Fig. 6, the distance ratio fluctuates within a range when α changes and reaches a minimum under $\alpha = 0$ with a value of 1.04, indicating the pre-trained model is effective and achieves a quite similar path distance to that of **Shortest**. In this case, the model can be regarded as the shortest-path-finding one, since only the principle objective is included in the reward function. In addition, models with $\alpha < 0$ correspond to **cu** minimization while models with $\alpha > 0$ correspond to **cu** maximization, with the detailed analysis as follows.

When concerning the crime risk maximization scenario, as can be seen in the right panel of Fig. 6, both distance and **cu** ratios climb gradually as α increases. Compared to the **cu** ratio, the distance ratio shows a smaller fluctuation, suggesting the **cu** maximization is achieved without much extra cost in path distance. Moreover, when α becomes larger than 0.5, the **cu** ratio stops growing. Worse yet, the distance ratio still

increases on the contrary. A possible explanation is that the agent tries to earn more r_{cu} but does not move towards the destination node, resulting in unnecessary detours when path-finding. In a word, the ability of maximizing **cu** has reached its limit when setting α to 0.5. Thus, it is meaningless to continue to increase α and the training can be terminated in this case. Similar conclusions can be also drawn when concerning the crime risk minimization scenario, with the results shown in the left panel of Fig. 6. In this case, the **cu** ratio is always smaller than 1. Similarly, the distance ratio fluctuates within a small range under different negative α s, suggesting the **cu** minimization is also accomplished at little cost of path distance increase. What is more, the **cu** minimization model reaches its limit when setting α to -2.

D. Effectiveness of State Representation

In **cuRL** framework, the state representation is carefully designed so that both the principle objective and the sub-objective can be efficiently learned and optimized. As discussed in Section IV, the state representation includes three parts, namely *coordinate state*, *distance state*, and *cu state*. To evaluate the effectiveness of each state type, an ablation study is conducted. In addition, we further compare the performance of state representations using latitude-longitude coordinates and one-hot encoding.

In summary, all the compared state representation methods are listed as follows.

- **Coord.** The state representation only includes the latitude-longitude coordinate state.
- **Onehot.** The state is only represented by one-hot encoding.
- **cuRL-coord.** The state representation includes the distance and **cu** states.
- **cuRL-dis.** The state representation includes the coordinate and **cu** states.
- **cuRL-cu.** The state representation includes the coordinate and distance states.

We adopt the scenario of *minimizing the solar radiation* to evaluate different state representation methods. The overall results in terms of the distance ratio and **cu** ratio are presented in Tab. I. It should be noted that all results are obtained when the parameter α in the reward function has been carefully tuned. From the results shown in Tab. I, we can draw the following conclusions:

- **Coord can optimize neither path distance nor path cu.** As shown in Tab. I, both distance and **cu** ratios of **Coord** are bigger than 1. This is because that **Coord** only considers latitude-longitude coordinates and fails to optimize the path **cu**. Furthermore, from the respective of distance optimization, both **Onehot** and **cuRL** outperform **Coord**. This verified our concern that latitude-longitude coordinates are *insufficient* for agents to optimize the path distance.
- **Compared to Coord, Onehot achieves a better result.** Although **Onehot** does not explicitly encode **cu** and coordinate information in the state representation, the nodes can still be well-distinguished. Specifically,

TABLE I
EVALUATION RESULTS OF DIFFERENT STATE REPRESENTATION METHODS.

	cu ratio	distance ratio	dimension
Coord	1.061	1.062	4
Onehot	1.054	0.975	1996
cuRL-coord	1.032	1.010	24
cuRL-dis	1.062	0.976	20
cuRL-cu	1.033	1.013	12
cuRL	1.033	0.968	28

it *memorizes* an optimum action for each state without *learning* any routing knowledge. Thus, it can be observed that **Onehot** outperforms **Coord** and can optimize **cu**. However, the efficient learning of **Onehot** without a significant slowdown in convergence is computationally impractical, due to the curse of dimensionality. In more detail, the dimension of the state vector represented by **Onehot** is extremely high, by up to 1996, while it is only 4 for **Coord**.

- **Both coordinate and distance states are essential.** On one hand, the distance ratio of **cuRL-dis** is quite bigger than 1 when the *distance state* is excluded. On the other hand, the **cu** ratio of **cuRL-coord** is also bigger than 1, indicating that the path **cu** is actually not optimized.
- **The cu state can effectively improve cu optimization.** Comparing the results of **cuRL-cu** and **cuRL**, we can easily find that they obtain an identical distance ratio (i.e., 1.033), and moreover, the **cu** ratio of **cuRL** is much smaller and less than 1 while that of **cuRL-cu** is bigger than 1. Again, the solar radiation is not minimized since the corresponding **cu** ratio is bigger than 1.

VI. CONCLUSIONS AND FUTURE WORK

In the paper, we investigated the routing problem with alternative optimization criteria and proposed a generic bi-criteria routing framework **cuRL** based on DRL that optimizes both path distance and path cost/utility. Contrast experiments and ablation study have validated the effectiveness of **cuRL**, as well as its generality in various scenarios. Besides the criteria mentioned in our evaluation, the proposed framework can be applied to other optimization scenarios given the targeted attribute of road segments. Therefore, **cuRL** could be an ideal solution to provide diverse routing services, which is an emerging trend for navigation platforms.

In the future, we plan to extend this work from several aspects. Firstly, we aim to develop a routing platform on mobile devices like mobile phones or vehicle-mounted intelligent navigators to collect the datasets of multiple criteria from the real world, and then deploy the **cuRL** framework on it. Secondly, we will further explore the potential of graph representation learning and graph convolution networks [43, 44, 45] serving as more effective state representation methods. Thirdly, considering that some of the alternative criteria are time-dependent (e.g., the solar radiation, the level of traffic congestion), this poses new challenges to the bi-criteria optimum routing. Since DRL also shows its capability

of modelling time-dependent characteristics [46, 23], we intend to further extend **cuRL** to suit the optimization of time-dependent criteria.

REFERENCES

- [1] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "Tripplanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2015.
- [2] Q. Wang, B. Guo, Y. Liu, Q. Han, T. Xin, and Z. Yu, "Crowdnavi: Last-mile outdoor navigation for pedestrians using mobile crowdsensing," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, 2018.
- [3] J. Yuan, Y. Zheng, C. Zhang, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, p. 99–108.
- [4] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [6] P. S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan, "From taxi GPS traces to social and community dynamics: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–34, 2013.
- [7] D. Zhang, B. Guo, and Z. Yu, "The emergence of social and community intelligence," *Computer*, vol. 44, no. 7, pp. 21–28, 2011.
- [8] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.
- [9] C. Chen, D. Zhang, Y. Wang, and H. Huang, *Enabling Smart Urban Services with GPS Trajectory Data*. Springer, 2021.
- [10] C. Chen, L. Gao, X. Xie, L. Feng, and Y. Wang, "2TD Path-Planner: Towards a more realistic path planning system over two-fold time-dependent road networks," *IEEE Computational Intelligence Magazine*, vol. 16, no. 2, pp. 78–98, 2021.
- [11] Y.-T. Zheng, S. Yan, Z.-J. Zha, Y. Li, X. Zhou, T.-S. Chua, and R. Jain, "GPSView: A scenic driving route planner," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1, feb 2013.
- [12] S. Levy, W. Xiong, E. Belding, and W. Y. Wang, "SafeRoute: Learning to Navigate Streets Safely in an Urban Environment," *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 6, pp. 1–17, 2020.
- [13] J. Kim, M. Cha, and T. Sandholm, "Socroutes: Safe routes based on tweet sentiments," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, p. 179–182.
- [14] E. Galbrun, K. Pelechris, and E. Terzi, "Urban navigation beyond shortest route: The case of safe paths," *Information Systems*, vol. 57, pp. 160–171, 2016.
- [15] D. Quercia, R. Schifanella, and L. M. Aiello, "The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city," in *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, 2014, p. 116–125.
- [16] M. H. Shaker, H. A. Karimi, and J. C. Zgibor, "Health-optimal routing in pedestrian navigation services," in *Proceedings of the First ACM SIGSPATIAL International Workshop on Use of GIS in Public Health*, 2012, p. 1–10.
- [17] Y. Li, S. George, C. Apfelbeck, A. M. Hendawi, D. Hazel, A. Teredesai, and M. Ali, "Routing service with real world severe weather," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, p. 585–588.
- [18] K. Ishima, S. Tsukiyama, and S. Shinoda, "An algorithm to detect positive cycles in a constraint graph for layout compaction," in *1990 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1990, pp. 2853–2856 vol.4.
- [19] R. Uehara and Y. Uno, "Efficient algorithms for the longest path problem," in *Algorithms and Computation*, R. Fleischer and G. Trippen, Eds., 2005, pp. 871–883.
- [20] D. P. Igoe, N. J. Downs, A. V. Parisi, and A. Amar, "Evaluation of shade profiles while walking in urban environments: A case study from inner suburban sydney, australia," *Building and Environment*, vol. 177, p. 106873, 2020.
- [21] P. Zhou, C. Wang, and Y. Yang, "Design and optimization of solar-powered shared electric autonomous vehicle system for smart cities," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.
- [22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [23] S. Sun, X. Zhao, Q. Li, and M. Tan, "Inverse reinforcement learning-based time-dependent a* planner for human-aware robot navigation with local vision," *Advanced Robotics*, vol. 34, no. 13, pp. 888–901, 2020.
- [24] Y. Ding, B. Guo, L. Zheng, M. Lu, D. Zhang, S. Wang, S. H. Son, and T. He, "A city-wide crowdsourcing delivery system with reinforcement learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 3, sep 2021.
- [25] C. Liu, C.-X. Chen, and C. Chen, "META: A city-wide taxi repositioning framework based on multi-agent reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13 890–13 895, 2022.
- [26] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to navigate in cities without a map," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, p. 2424–2435.
- [27] A. Sarker, H. Shen, and K. Kowsari, "A data-driven reinforcement learning based multi-objective route recommendation system," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2020, pp. 103–111.
- [28] R. Steinbach, C. Perkins, L. Tompson, S. Johnson, B. Armstrong, J. Green, C. Grundy, P. Wilkinson, and P. Edwards, "The effect of reduced street lighting on road casualties and crime in england and wales: controlled interrupted time series analysis," *Journal of Epidemiology & Community Health*, vol. 69, no. 11, pp. 1118–1124, 2015.
- [29] G. S. Armstrong and D. L. MacKenzie, "Private versus public juvenile correctional facilities: Do differences in environmental quality exist?" *Crime & Delinquency*, vol. 49, no. 4, pp. 542–563, 2003.
- [30] A. Loukaitou-Sideris, R. Liggett, H. Iseki, and W. Thurlow, "Measuring the effects of built environment on bus stop crime," *Environment and Planning B: Planning and Design*, vol. 28, no. 2, pp. 255–280, 2001.
- [31] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.
- [32] C. Chen, X. Chen, L. Wang, X. Ma, Z. Wang, K. Liu, B. Guo, and Z. Zhou, "MA-SSR: A memetic algorithm for skyline scenic routes planning leveraging heterogeneous user-generated digital footprints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 5723–5736, 2017.
- [33] N. Runge, P. Samsonov, D. Degraen, and J. Schöning, "No more autobahn! scenic route generation using googles street view," in *Proceedings of the 21st International Conference on Intelligent User Interfaces*. Association for Computing Machinery, 2016, p. 147–151.
- [34] A. Delarue, R. Anderson, and C. Tjandraatmadja, "Reinforcement learning with combinatorial actions: An application to vehicle routing," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 609–620.
- [35] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu, "Efficiently solving the practical vehicle routing problem: A novel joint learning approach," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, p. 3054–3063.
- [36] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, p. 2094–2100.
- [37] Z. Shen, W. Du, X. Zhao, and J. Zou, "DMM: fast map matching for cellular data," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, London United Kingdom, 2020, pp. 1–14.
- [38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (Poster)*, 2015.
- [39] O. contributors. (2021, mar) Openstreetmap data. [Online]. Available: <https://www.openstreetmap.org>
- [40] P. Fu and P. M. Rich, "Design and implementation of the solar analyst: an arcview extension for modeling solar radiation at landscape scales," in *Proceedings of the nineteenth annual ESRI user conference*, vol. 1, 1999, pp. 1–31.
- [41] M. Deng, W. Yang, C. Chen, Z. Wu, Y. Liu, and C. Xiang, "Street-level solar radiation mapping and patterns profiling using Baidu Street View images," *Sustainable Cities and Society*, vol. 75, p. 103289, Dec. 2021.
- [42] N. O. D. contributors. (2021, Mar.) Nyc data. [Online]. Available: <https://opendata.cityofnewyork.us/>
- [43] J. Jain, V. Bagadia, S. Manchanda, and S. Ranu, "Neuomlr: Robust & reliable route recommendation on road networks," *Advances in Neural*

Information Processing Systems, vol. 34, 2021.

- [44] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering a* search algorithms with neural networks for personalized route recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 539–547.
- [45] C. Chen, Q. Liu, X. Wang, C. Liao, and D. Zhang, "semi-traj2graph: Identifying fine-grained driving style with GPS trajectory data via multi-task learning," *IEEE Transactions on Big Data*, pp. 1–1, 2021.
- [46] C. Mao and Z. Shen, "A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 179–197, 2018.



Chao Chen is a Full Professor at College of Computer Science, Chongqing University, Chongqing, China. He obtained his Ph.D. degree from Pierre and Marie Curie University and Institut Mines-Télécom/Télécom SudParis, France in 2014. He received the B.Sc. and M.Sc. degrees in control science and control engineering from Northwestern Polytechnical University, Xi'an, China, in 2007 and 2010, respectively. Dr. Chen got published over 80 papers including 20 ACM/IEEE Transactions. His work on taxi trajectory data mining was featured

by IEEE SPECTRUM in 2011, 2016, and 2020 respectively. He was also the recipient of the Best Paper Runner-Up Award at MobiQuitous 2011. His research interests include pervasive computing, mobile computing, urban logistics, data mining from large-scale GPS trajectory data, and big data analytics for smart cities. He is a senior member of IEEE.



Lujia Li is pursuing his M.Sc. degree at the College of Computer Science, Chongqing University, China. He received his B.Sc degree from the College of Computer Science, Chongqing University, China, in 2020. His research interests include intelligent transportation systems, spatio-temporal trajectory data mining, and reinforcement learning.



Mingyan Li received the B. E. degree in telecommunication engineering from Jinlin University, Changchun, China, in 2015, and the Ph.D. degree at the Department of Electronic Engineering, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2021. She joined the College of Computer Science, Chongqing University in 2021 and now works as a postdoctoral research associate. She was a Visiting Professor in University of Waterloo, Canada (2019–2020). Her current research interests include

industrial wireless networks and application in industrial automation, joint design of communication and control in industrial cyber-physical systems, software-defined networking and network slicing, ultra-reliable low-latency communication, and time-sensitive networks for industrial internet.



ment and Urban Computing.

Ruiyuan Li is an Associate Professor with Chongqing University, China. He received the B.E. and M.S. degrees from Wuhan University, China in 2013 and 2016, respectively, and the Ph.D. degree from Xidian University, China in 2020. He was the Head of Spatio-Temporal Data Group in JD Intelligent Cities Research, leading the research and development of JUST (JD Urban Spatio-Temporal data engine). Before joining JD, he had interned in Microsoft Research Asia from 2014 to 2017. His research focuses on Spatio-temporal Data Manage-



Zhu Wang is currently an Associate Professor of computer science with the Northwestern Polytechnical University, Xi'an, China. He received the Ph.D. degree from Northwestern Polytechnical University, Xi'an, China, in 2013. His research interests include pervasive computing, mobile social network analysis, and mobile healthcare.



control theory, prediction and control of dynamic characteristics of intelligent unmanned equipment.

Fei Wu is an Assistant Professor at College of Mechanical and Vehicle Engineering, Chongqing University, Chongqing, China. He obtained his Ph.D. degree from Hunan University in 2015. Dr. Wu published over 40 papers in the Journal of Comput. Method. App. L. M (IF 6.76), Appl. Math. Model (IF 5.12), Compos. Struct (IF 5.04), Appl. Phys. Lett (IF 3.71), J. Soud. Vib (IF 3.66), Comput. Mech (IF 4.04), Int. J. Numer. Meth. Eng (IF 3.47) and Mechanical Engineering. His research interests include intelligent unmanned vehicle decision and



Chaocan Xiang is an Associate Professor at the College of Computer Science, Chongqing University, Chongqing, China. He received the BS and Ph.D. degrees in computer science and engineering from the Nanjing Institute of Communication Engineering, China, in 2009 and 2014, respectively. He studied at the University of Michigan-Ann Arbor in 2017. His current research interests include wireless sensor networks, crowd-sensing networks, and IoT.