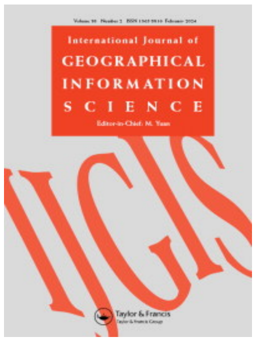


International Journal of Geographical Information Science

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tgis20

A reinforcement learning-based routing algorithm for large street networks

Diya Li, Zhe Zhang, Bahareh Alizadeh, Ziyi Zhang, Nick Duffield, Michelle A. Meyer, Courtney M. Thompson, Huilin Gao & Amir H. Behzadan

To cite this article: Diya Li, Zhe Zhang, Bahareh Alizadeh, Ziyi Zhang, Nick Duffield, Michelle A. Meyer, Courtney M. Thompson, Huilin Gao & Amir H. Behzadan (2024) A reinforcement learning-based routing algorithm for large street networks, International Journal of Geographical Information Science, 38:2, 183-215, DOI: 10.1080/13658816.2023.2279975

To link to this article: https://doi.org/10.1080/13658816.2023.2279975

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
2024, VOL. 38, NO. 2, 183–215
https://doi.org/10.1080/13658816.2023.2279975

RESEARCH ARTICLE

OPEN ACCESS

A reinforcement learning-based routing algorithm for large street networks

Diya Li[a], Zhe Zhang[a,b], Bahareh Alizadeh[c], Ziyi Zhang[b], Nick Duffield[b,d], Michelle A. Meyer[e], Courtney M. Thompson[a], Huilin Gao[f] and Amir H. Behzadan[c]

[a]Department of Geography, Texas A&M University, College Station, TX, USA; [b]Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA; [c]Department of Construction Science, Texas A&M University, College Station, TX, USA; [d]Texas A&M Institute of Data Science, College Station, TX, USA; [e]Department of Landscape Architecture & Urban Planning, Texas A&M University, College Station, TX, USA; [f]Department of Civil & Environmental Engineering, Texas A&M University, College Station, TX, USA

ABSTRACT

Evacuation planning and emergency routing systems are crucial in saving lives during disasters. Traditional emergency routing systems, despite their best efforts, often struggle to accurately capture the dynamic nature of flood conditions, road closures, and other real-time changes inherent in urban disaster logistics. This paper introduces the ReinforceRouting model, a novel approach to optimizing evacuation routes using reinforcement learning (RL). The model incorporates a unique RL environment that considers multiple criteria, such as traffic conditions, hazardous situations, and the availability of safe routes. The RL agent in this model learns optimal actions through interaction with the environment, receiving feedback in the form of rewards or penalties. The ReinforceRouting model excels in executing prompt and accurate route planning on large road networks, outperforming traditional RL algorithms and shortest-path-based algorithms. A higher safety score and episode reward of the model are demonstrated when compared to these classical methods. This innovative approach to disaster evacuation planning offers a promising avenue for enhancing the efficiency, safety, and reliability of emergency responses in dynamic urban environments.

ARTICLE HISTORY
Received 2 February 2023
Accepted 31 October 2023

KEYWORDS
Disaster evacuation; reinforcement learning; Geographic Information Science and Systems (GIS); artificial intelligence; routing algorithm

1. Introduction

Changes in the global climate amplify the risk of water-related disasters such as flooding in urban areas. Since 1990, water-related disasters have accounted for 90% of the 1000 most severe disasters (Hendricks *et al.* 2022). They are the most frequent and

CONTACT Zhe Zhang zhezhang@tamu.edu

expensive natural disasters nationwide, impacting millions of people's lives and thousands of communities every year. Despite the cascading impacts of these extreme events, existing climate adaptation platforms do not fully understand the dynamics of urban community-level response due to the lack of detailed disaster management and efficient evacuation plans at the street level (Gharaibeh *et al.* 2021). Evacuation and emergency routing systems play a critical role in saving lives and minimizing damages during disaster events. Traditional emergency routing systems usually use remote-sensing images and hydrology methods to simulate the movement of the flood (Henonin *et al.* 2013, Feng *et al.* 2015) to obtain the flooding information. However, data obtained this way may result in imprecise flood prediction results due to its failure to consider reshaped surface topography and micro-topographic variations commonly seen in urban environments (Jha *et al.* 2012, Alizadeh *et al.* 2021, Kharazi and Behzadan 2021). Accurate and reliable data are crucial for the successful dispatching of rescue teams and navigating individuals from flooded areas to safe places during emergencies. Recent research (Cavdur *et al.* 2016, Yan *et al.* 2020) showed that the number of deaths is highly related to the efficiency of evaluation plans and routing during emergency events. Therefore, establishing a well-thought-out and timely evacuation strategy is critical in dynamic flooding situations where lives can be at risk (Meyer *et al.* 2018).

Traditional routing algorithms such as capacitated scheduling algorithm (Osman and Ram 2013), genetic algorithm (Gomes and Straub 2017), and cellular automata-based evacuation (Li *et al.* 2021a), have been used in the past for approximate solutions in evacuation routing during flood emergencies. However, the following limitations have been identified for the existing routing algorithms (Ding *et al.* 2021). Firstly, traditional routing algorithms are often based on pre-defined rules or heuristics and may not be able to adapt to real-time changes in flood conditions, road closures, or other dynamic factors. For example, in fast-changing flood situations, the traditional routing algorithms cannot update the routes in real-time based on the emergency situation (Duraipandian 2019, Ding *et al.* 2021). Second, traditional routing algorithms struggle with scalability issues when dealing with large-scale flood evacuations involving a large number of affected individuals, multiple evacuation routes, and varying capacities of transportation resources. The computational complexity of these algorithms may increase significantly with the size and complexity of the evacuation scenario, leading to longer computation times and potential delays evacuation process. Finally, traditional routing algorithms lack the ability to deal with diverse data sources, such as gauge data, information from first responders (eg volunteered geographic information such as New York City 311 data), or inputs from affected individuals. The traditional methods rely solely on a few predefined parameters or heuristics, which makes it hard to capture the complexity and dynamics of flood emergencies. To overcome these limitations, innovative approaches, such as reinforcement learning techniques can be used.

Recent contributions in deep reinforcement learning (RL) (Sutton and Barto 2018) have shown unique advantages in solving conditional routing problems (Nazari *et al.* 2018, Levy *et al.* 2020). In RL frameworks, agents represent individual people or a vehicle in a navigation environment, which are simulated to iteratively learn the policies to maximize the reward feedback through interacting with an environment. This unique trait of RL makes it a natural choice for many data mining problems requiring incremental decisions. For example, RL can learn to solve complex route optimization problems in a dynamic environment by collecting experience, while traditional algorithms focus more on static environments (Qiu *et al.* 2019). Traditional methods also usually need to recalculate the route when the road network changes, but the RL approach can incrementally adapt to an unknown environment without retraining the entire data to reduce the computational time (Su *et al.* 2004). Furthermore, when a route optimization problem becomes a complex sequential decision-making process due to unpredictable natural or artificial causes, solving those problems by traditional algorithms also becomes extremely complex or even NP-hard (Abe *et al.* 2019).

Although RL has been widely adopted in previous work for pathfinding and route optimization (Godfrey and Powell 2002, Xiong *et al.* 2017, Wei et al. 2018, Kim and Kim 2021), there are still many limitations to the current RL routing optimization approach. For instance, using RL in large graph networks can be challenging due to inefficient exploration strategies. Here, the RL exploration refers to the process of discovering and learning from new states and actions to improve the agent's decision-making. Traditional RL methods may suffer from exploration issues in large graphs, as the search space can be large, leading to long computation times and sub-optimal solutions (Arora *et al.* 2017, Manchanda *et al.* 2019). Moreover, action settings in previous graph routing environments lack consideration of practical usage in the real world. For example, Levy *et al.* (2020) used compass direction (North, Northeast, East, Southeast, South, Southwest, West, Northwest) as an action space, which ignores the actual road network structure and causes invalid actions. For example, an agent will keep the same direction and slightly turn left or right when the agent wants to exit from a controlled-access highway to a freeway. Sharma *et al.* (2021) directly used nodes as an action space for fire evacuation route planning, but it is not applicable in a large network system with tens of thousands of nodes.

To address the challenges mentioned above, we developed a geospatial cyberinfrastructure-enabled reinforcement learning algorithm to improve the routing efficiency in a large real-world road network. The algorithm was trained using the National Science Foundation-funded FASTER (Acquisition of FASTER – Fostering Accelerated Sciences Transformation Education and Research) supercomputer to handle large road networks in a spatial database and support training data-loading tasks (Li and Zhang 2021).

Our RL routing algorithm uniquely considers multiple factors, including safety, reliability, and efficiency, in routing calculations under dynamic and variable weather and flooding conditions. This work makes several significant contributions to the fields of routing and emergency management research:

- We developed a novel graph-based RL environment, complete with efficient action and reward policies, which facilitates sophisticated routing optimization.

- We integrated state-of-the-art reward optimization methods to train our graph-based RL algorithm, even under the complexities of large graph-based environments.
- We successfully scaled up the RL agent to accommodate large graph-based maps (eg over one thousand nodes) through the use of behavior cloning optimization methods.
- We incorporated near-real-time flood data in our experimental tests to assess the performance of the graph-based RL algorithm under realistic conditions.

The remainder of this article is organized as follows. Section 2 introduces the literature background of our research. Section 3 presents the procedures of data preparation. Section 4 describes our methodology of flooding environment development and the details of various optimization techniques. Section 5 implements experiments using the proposed method and analyzes the results. Section 6 discusses this method for different applications. The last section discusses conclusions drawn from the study.

2. Background

Floods are one of the most common hazards in the United States (Perry 2000, Zhang *et al.* 2014, 2019, Xu *et al.* 2020, Li *et al.* 2021b), which cause widespread devastation, resulting in the loss of life and damages to personal property and critical public health infrastructure. During a disaster, rapid response and effective evacuation activities are important in minimizing the loss of life or harm to the public during natural disasters (Murray-Tuite and Wolshon 2013, Huang *et al.* 2016, Yang and Shekhar 2017). Recent studies (Hai-bo and Yu-bo 2017, Qiu *et al.* 2019, Yin *et al.* 2019, Li *et al.* 2021a) found that traffic delays and unknown road conditions are the biggest challenges for efficient rescue. Moreover, in Lim et al. (2013)'s comprehensive review, flood evacuation models with different optimized variables, including travel times, travel costs, unknown traffic, travel distance, and identification of evacuation routes with an emphasis on flooding situations, are the main difficulties in flood disaster management. Many researchers focused on modeling flood evacuation as a decision-support system that combines all the information to build various spatial analysis models to help decision-makers (Liu *et al.* 2006, Zhang *et al.* 2016, Lee *et al.* 2020). For example, Liu *et al.* (2006) developed an adaptive evacuation route model based on the traditional Dijkstra shortest path algorithm (Dijkstra 1959) to pursue the goal of minimizing the total evacuation time. Later, Zhang *et al.* (2016) developed a GIS-based decision support system that can acquire situational information on flood evolution, feasible routes, and high-risk areas for the flood detention basin. To evaluate the performance of the flood evacuation model, Li *et al.* (2019b) simulated the flood evacuation with a multi-agent system in a virtual reality environment. In recent studies, Lee *et al.* (2020) modeled the spatial and temporal inundation information with a non-linear auto-regressive model to plan the evacuation route, and (Li *et al.* 2021a) developed an algorithm that couples high-resolution hydrodynamic and cellular automata-based evacuation route planning for flooding situations.

However, these existing routing algorithms suffer from several limitations (Delling *et al.* 2012, Chen *et al.* 2014, Zhang, Yang, and Zhao 2016). They are often inefficient in navigating complex and dynamic road networks. This is because they rely on predefined road information, which may not accurately reflect the actual road conditions during flooding events (Zhang, Yang, and Zhao 2016). Moreover, Staroverov *et al.* (2020) found that traditional routing algorithms are not well-suited for real-time updates and are limited in their ability to quickly adapt to changing road conditions. For example, the capacitated scheduling algorithm and genetic algorithm can only find approximate solutions (Kumari and Geethanjali 2010), and the cellular automata-based evacuation model is limited by the accuracy of the given information (Trindade *et al.* 2016). Only a few research are focused on developing routing algorithms that can be applied to real-time changing road networks with less external traffic data. For example, Delling *et al.* (2012) developed a robust mobile route planning model with limited connectivity information in the road network. Similarly, Mirahadi and McCabe (2021) proposed an evacuation management model that uses Dijkstra's algorithms to dynamically calculate and foresee consequences, and thus create an evacuation decision-support strategy. However, classical routing algorithms often assumed that there is only one objective and that the problem's preset environment is completely deterministic. For example, Chen *et al.* (2014)'s path optimization model for vehicle evacuation uses the greedy methodology that tries out all possible routes in a network, but the model cannot work in a changing environment with unpredictable weather conditions and other social factors. Machine Learning (ML) researchers take advantage of the increasing computing power of GPUs to adapt a supervised neural network to solve the pathfinding problem and demonstrate the potential ability to solve many other location-based problems (Wang *et al.* 2009, Kumari and Geethanjali 2010, Yin *et al.* 2023). More recently, alternative methods have been proposed utilizing clustering-based methods with evaluation algorithms for Vehicular ad hoc networks (VANETs). For instance, based on Bagherlou and Ghaffari (2018)'s proposed routing protocol with simulated annealing and radial basis function (RBF) neural networks, Mohammadnezhad and Ghaffari (2019) developed a reliable routing algorithm using simulated annealing for clustering and radial basis function neural networks for cluster head selection, showing efficiency in terms of route discovery rate and packet delivery rate. Later Kheradmand *et al.* (2022) improved the previous method's performance using Harris Hawks Optimization (HHO). These methods, although efficient, rely on the specific conditions of the network and the number of generated clusters.

A particular challenge in ML-based path planning models concerns the generalization issue and the selection of training and test data. This challenge raises the questions of (1) whether the trained ML model will still work in a different environment (eg a different city) and (2) whether it will work if we lack training and test data. Deep reinforcement learning (DRL), a type of machine learning technique (Sutton and Barto 2018), has recently been employed in such complex sequential decision-making processes to minimize loss and maximize the long-term gain of an intelligent agent. The choice of DRL in our project was natural because many real-world path planning problems require an incremental decision-making process, and the RL method has no dependencies on batch path training datasets (Zhang *et al.* 2021). Moreover, compared with classical path-finding methods (such as the Dijkstra algorithm and A* algorithm (Hart *et al.* 1968)) and other supervised machine learning methods, RL-based route planning can gain experience by interacting through a memorized reward function with the training environment and evaluating the feedback from the training environments, eventually performing as a self-adjusted intelligent decision-support agent for real-world users (Bi *et al.* 2019). In Chamola *et al.* (2021)'s survey about machine learning in disaster management, the DRL method is regarded as a self-sustainable system – this unique trait makes it very promising to support disaster management research.

Many researchers have been working on integrating RL techniques to navigate vehicles in various scenarios Walker *et al.* (2019), Levy *et al.* (2020), Sharma *et al.* (2021), Wang *et al.* (2021), He *et al.* (2022). Levy *et al.* (2020) used DRL to develop SafeRoute algorithms to help pedestrians safely navigate the city by avoiding street harassment and crime. However, their limited consideration of real-time changing street conditions resulted in no significant improvement over simple avoidance routing. Machine learning or reinforcement learning-based routing algorithms cannot navigate in large road networks due to scalability issues. They are limited by the size of the data they can process, which affects the accuracy and speed of the routing result (Geng *et al.* 2021). For example, both Tian and Jiang (2018) and Sharma *et al.* (2021) have considered using DRL to design evacuation routes for fire disasters in building environments, but only indoors due to scalability issues. DRL methods have also been applied to complex multi-task path-finding scenarios Wang *et al.* (2019, 2021). (Wang *et al.* 2019) and Wang *et al.* (2021) have proposed using multi-agent DRL methods to help a group of vehicles design efficient routes in complex environments. Other research (Shi *et al.* 2023) takes advantage of the traffic light to optimize the travel time using DRL. However, no existing studies have provided an efficient solution to handle large-scale, real-world flood evacuation events.

Sample efficiency is another challenge that many researchers have faced when designing a routing algorithm (Kakade 2003, Sohn *et al.* 2021). Sohn *et al.* (2021) showed that achieving good learning in Markov Decision Processes (MDPs) path planning problems requires a large number of samples in RL algorithms. To address this, Pathak *et al.* (2017) and Seo *et al.* (2021) proposed using external neural network encoders to embed the environment features as intrinsic rewards to optimize the training procedures of an RL agent. Recent researchers (Christiano *et al.* 2017, Kumar *et al.* 2021) have found that integrating the behavioral cloning method (Schaal 1999) in RL training can improve the policy's performance. In a recent study on human-centered RL, Li *et al.* (2019a) demonstrated the importance of using human feedback in DRL, which can improve applicability to real-life problems.

In aggregate, *Table 1* below summarizes the key characteristics of several routing algorithms that are used in real-world cases. According to the comprehensive survey about the routing algorithm by Tyagi *et al.* (2022), we included the environment settings, algorithm type, completeness, and limitations as comparison attributes. The environment setting can be either VANETs, or static, where the road network is pre-defined and unchanging, or dynamic, where the road network can change in real-time. The algorithm setting can include traditional methods like Dijkstra's algorithm, GIS-based methods, robust mobile routing, greedy methods, and ML methods such as supervised neural networks (NN), simulated annealing (SA), and deep reinforcement learning (DRL). The completeness of the algorithm refers to whether it provides an exact solution or a heuristic (approximate) solution.

Table 1. Comparison of real-world routing algorithms.

Reference	E.	Algorithms	C.	Comment
Liu *et al.* (2006)	S*	Dijkstra	E	Inflexible to changes
Zhang *et al.* (2016)	S*	GIS-based	H	Limited in real-time route updating
Delling *et al.* (2012)	D*	Robust Mobile	H	Limited connectivity
Mohammadnezhad and Ghaffari (2019)	V*	SA & NN	H	Rely on existing condition
Mirahadi and McCabe (2021)	D*	Dijkstra	E	Single objective, deterministic
(Chen *et al.* 2014)	S*	Greedy	H	Inflexible to changes
Lee *et al.* (2020)	S*	Supervised NN	H	Generalization issue
Li *et al.* (2021a)	D*	Cellular Automata-based	H	Limited accuracy
Levy *et al.* (2020)	D*	DRL (SafeRoute)	H	Limited real-time updates
Sharma *et al.* (2021)	S*	DRL on q-matrix	H	Limited to indoor environments
Wang *et al.* (2021)	D*	Multi-agent DRL	H	Limited to small-scale scenarios
He *et al.* (2022)	D*	DRL	H	Navigation Comparison
Shi *et al.* (2023)	S*	Adaptive DRL	H	Limited to small-scale network

E.: Environment; S*: Static; D*: Dynamic; V*: VANETs; C.: Completeness; E: Exact; H: Heuristic.

3. Data collection and processing

This study can be applied to any transportation network in urban or rural areas. Point data with longitude and latitude can be used to represent the Points of Interest (POI) that a user wants to avoid. In our experiment, we used transportation network data collected from New York City and Houston. For the New York City case study, we used New York City 311 data as the POI data, and for the Houston case study, we used gauge data as the POI data.

For the transportation data, the original map information was collected from OpenStreetMap, a free collaborative world map (Bennett 2010). In our experiments (Section 5), we chose to export map information for the downtown areas of Houston and New York City. We used the Houston road network to demonstrate the algorithm's performance, and then used the New York City road network to show that our algorithm can also be applied to other cities in the US with different types of POI data to demonstrate the scalability and flexibility of our algorithm. We first converted the routing map (street network data) into graph-based data using the methods introduced in Section 4.1. For each node, the route map contains:

1. Node id: unique ID of each node.
2. Longitude and Latitude of the node location.
3. Elevation (meter): the elevation information is collected using OpenTopography Krishnan *et al.* (2011).
4. Lowland (Boolean): the lowland attributed of node n is preprocessed using elevation data:

$$\mathbf{L}_n = \begin{cases} \text{True} & \forall \mathcal{E}_N > \mathcal{E}_n \\ \text{False} & \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{E}_N$ denotes as elevation set of neighbors of node n, $\mathcal{E}_n$ denotes as elevation of node n.

Each edge contains the following information:

1. Edge id: unique ID of the edge.
2. Node ids: The ID pair of nodes that connect the edge.
3. Preset attributes: show the preset open street map attributes, including the number of lanes, highway type (eg tertiary, secondary, etc,.), one-way (Boolean), bridge (Boolean), road length (meter), speed limits (kmph), average speed (kmph).
4. Grade ($-90 \sim 90$): show the slope of the road, calculated by the elevation of nodes; usually less than 30.
5. Bearing ($0 \sim 360$): show the bearing direction of the road, calculated by the Longitude and Latitude of nodes (e.g. 0 represents north and 90 represents east).

When examining graph theory, a graph can be depicted by an adjacency matrix. This matrix showcases the segments of the graph, with indicators such as $0, 1$ that demonstrate whether a particular edge connects to the node or not. However, mapping a real-world map into an adjacency matrix poses significant difficulties due to the intricate attribute information that the map holds. This information includes street type, street speed limit, elevation, and grade. To manage and update this data, we utilize NetworkX (Hagberg *et al.* 2008, Rossi and Ahmed 2015). We also incorporate additional data sources, such as the BluPix application[1] and the New York City 311 data[2], which is a public dataset recording non-emergency service requests in New York City. For the purpose of street flood analysis, we used a processed version of the New York City 311 data.[3]

4. Methodology

4.1. Preliminaries and formulation

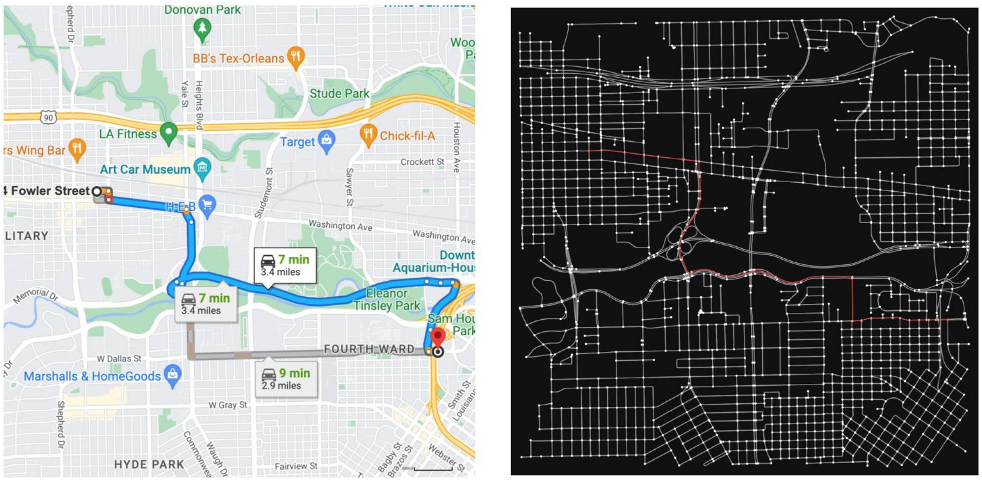
4.1.1. Graph representation in pathfinding problem

A road network can be modeled as a directed Graph $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ denotes a set of n vertices in graph and $\mathbf{E} = \{e_1, e_2, \ldots, e_m\}$ denotes the set of roads as m edges in graph. Figure 1 illustrates the comparison between the real-world representation of the routing system and the graph-based routing system. For example, Figure 1(a) shows an example of navigating from point A to point B using Google map, while Figure 1(b) represents the graph-based view Herman *et al.* (2000).

4.1.2. Markov decision process (MDP)

In the RL, the routing system can model as an MDP tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \rho)$, where $\mathcal{S}$ denotes as state set, $\mathcal{A}$ denotes s an action set, ρ denotes as a transition probability, $\mathcal{R}$ is a reward function and ρ is initial state distribution (Sutton and Barto 2018). Each state s contains all the information (described in Section 4.2) that the agent observed. The value of a policy π is denoted by

$$V_\pi(s) = \mathbb{E}^\pi\left[\sum_t \gamma^t r_t | s_0 = s\right] \tag{2}$$



(a) Example of using Google Map to navigate from (29.7712254,-95.4072904) to (29.7577685,-95.3742963) in Houston

(b) Graphical representation of using open street map NetworkX (OSMnx) to plot street-level road network in similar center Houston area. The red vertices and edges represent route navigation generated by classical Dijkstra algorithms

Figure 1. A demonstration of a real-world road map and graph road network.

where $r \in \mathcal{R}$ denotes a reward, and $\gamma \in [0, 1]$ denotes a discount factor, which cares for the rewards the agent achieved in the past, present, or future. Equation 2 aims to achieve the optimal policy π^* which maximizes the expected return:

$$\pi^* = \arg\ \max_\pi \mathbb{E}_{s\sim\rho}[V_\pi(s)] \tag{3}$$

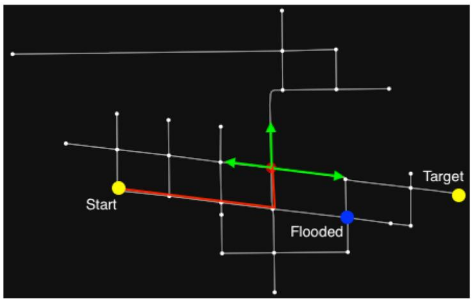
The optimal policy π^* will take the best action from the action space (described in Section 4.2) to lead the agent to reach the goal.

4.2. Environment settings

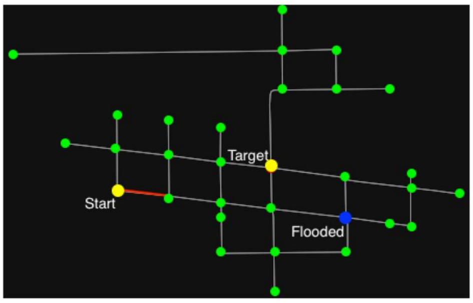
4.2.1. States

The state $s \in \mathcal{S}$, also known as an observation, represents the agent's current status on the map. In RL applications, the settings of the observation space and action space play an important role in the downstream task of RL. The state of an agent contains the current node and target node, which allows the agent to recognize that state and take appropriate action to reach the goal. If the agent observes the current node at a later training time with a target node in the opposite direction from before, the agent may take the opposite action to correct the direction. Furthermore, in the early training stage, the agent tends to be more curious, meaning the agent will try to explore more unseen states by taking as many unknown actions as possible, especially in a large environment (eg the Houston metro area). In our model, the states include two parts: graph observation and flooding observation.

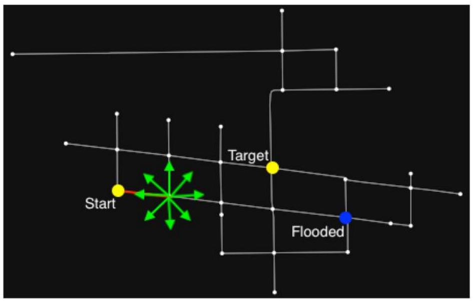
In a flood scenario, the agent can get an overview of flood depth information from our graph environment. Additionally, the raw flooding information only contains



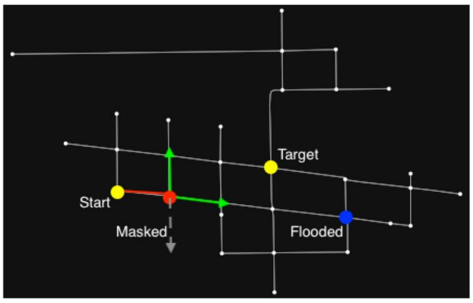
(a) An agent takes action during the training. The green arrow is three actions (Turn left, Proceed, and Turn right) in action space.



(b) A simple demonstration of action space in Demircan *et al.* (2011), Sharma *et al.* (2020) that all nodes in the observation are available actions that let the agent choose a valid one.



(c) A simple demonstration of action spaces in Levy *et al.* (2020) that contains eight compass directions. The action space in Geng *et al.* (2021) has a similar design but only focuses on four directions.



(d) Action mask in RL navigation environment. The grey dotted line represents a masked action.

Figure 2. A comparison of different action designs in RL navigation research. The yellow nodes are the start and target points, and the blue node is a flooded point. This sub-graph is sampled near (29.7685519,-95.3772329) in Houston downtown.

longitude, latitude, and flood depth (inches). Therefore, we need to convert the flooding information to distance with uniform measures (meters). Given two nodes (the current node λ and the flooded node ϕ), distance is calculated using Equation (20) (Appendix A). Figure 2(a) shows where flooded points are located in a road network. With fixed latitude and longitude information, the agent can obtain an overview of flooding formation directly from the graph. However, using latitude and longitude coordinates to represent the flooded point does not give the agent any direct information (Levy *et al.* 2020). It is challenging to store all reported floods in our observation. Thus, to use the node attribute to represent flooding information in graph observation, we used the graph embedding method (generated by node2vec (Grover and Leskovec 2016)) to embed the flooded nodes. Given a k-nearest reported stop sign, the flooding observation can be expressed as a $2 \times d'$ matrix $s_{f_n} = (e_n, e_f - e_n)$, where e_n denotes the node embedding of the current node, and e_f denotes the sum of the embedding representation of all reported flooded points. In the real-world experiment, our environment setting becomes more complicated to optimize the simulation of real-world dynamic flooding events. So we involve a new mechanism called 'evolving' to simulate dynamic flooding (described in Section 5).

4.2.2. Actions

Actions in the environment represent moving from one street node to another. An agent performs an action to receive an updated state to observe the environment and prepare for the next action. An evacuation agent's action depends on its state and then determines its behavior. Figure 2(a) demonstrates how an agent takes action during training. The agent has three discrete actions (shown as green arrows) in one state. Through a policy network, the policy will eventually converge to an optimal policy that will take the best action to lead the agent to reach the target node. In our environment, the action space is discrete, which follows the human decision-making process in wayfinding, such as 'Turn left,' 'Proceed,' and 'Turn right.' However, not all nodes are located at the intersection with four sides. Some nodes are fork roads, and some nodes are highway exits with two sides. Therefore, the action needs to be masked (described in Section 4.3.2). In our study, the action was formatted as a cardinal number in the training step.

4.2.3. Reward function

The reward function evaluates the action of the agent in each state. In a policy network, the task of the agent is to maximize the reward function. The reward measures whether the agent reaches the target node or whether the agent falls into a flooded situation. In our environment, the agent optimizes various preferences so the reward function must be designed with several factors. Since the main goal of our evacuation routing problem is to reach the destination safely, we embed the safety factor into the reward as a function of distance from the current node to the closest flooded points. A list of the reported flooded depth is traversed and updated for each step during the training step. In the urban area, flows greater than 9 cm depth and 1.5–2 m/s velocity can generate a loss of stability for subjects weighing 50–60 kg (Russo *et al.* 2013) and the effect evacuation time is 10 min (Vicario *et al.* 2020). Even though our evacuation plan is designed for vehicle routing, the safety of passengers is equally important and needs to take into consideration. We assumed the flooded depth of over 9 cm is a dangerous flooding node and the velocity of flooding is constant at 1 m/s. Then, the reward for the flooding factor can be expressed as follows:

$$r_{flood} = -\min(0, \max\left(C, log_2 \frac{d_f}{T}\right)), r_{flood} \in [C, 0] \tag{4}$$

where $C \in [-\infty, 0]$ is an adjustable parameter in the environment setting to limit the negative reward of the flooding factor, d_f is the closest distance between the current node to a dangerous flooding node and T denotes an effect evacuation distance. In Equation (4), we use a logarithm function to control drops in the reward curve during training. Through this process, the agent can learn to leverage the penalty and take a faster road when the distance between the current position and the closest flooded point is acceptable. In addition to the flooding factor, the final goal of reaching the destination also needs to be added to the reward function. Considering the AI safety issue that agents should not try to introduce or exploit the reward function to get more reward (Leike *et al.* 2017), the goal reward should be simple and effective (Chevalier-Boisvert *et al.* 2018, Sutton and Barto 2018, Levy *et al.* 2020). Note that the reward function of the flooding factor will also not generate any positive reward to the agent to avoid reward abuse. Thus, our reward is simply set as:

$$R = \begin{cases} 1 + r_{flood} & \text{if current node} == \text{target node} \\ r_{flood} & \text{otherwise} \end{cases} \tag{5}$$

Using a simple reward function in RL routing methods for evacuation scenarios can result in sparse reward environments, where most of the reward signals are equal to or less than 0.0. Sparse reward environments can pose challenges for RL algorithms as they may lead to slow learning or difficulty in finding optimal policies due to the lack of informative feedback. In such an environment, agents have to navigate (and change the underlying state of the environment) over long periods of time, without receiving much (or any) feedback (Pathak *et al.* 2017, Moritz *et al.* 2018). Section 4.3.3 shows the methods for dealing with sparse rewards in our RL flooding evacuation model.

4.3. Flood evacuation model using reinforcement learning

This section introduces detailed information about the optimization process and adaptation methods. We applied our graph-based RL algorithm to a large-scale road network to support risk-informed decision-making. In our study, agents explored the study area with a flooding event in four stages.

4.3.1. Policy network

MDPs are the bases for an RL framework that can underlay the unknown state probability distribution and transition probability to get an optimal policy π^* to decide which road should go. Several methods have been discovered to find the optimized π^* (Watkins and Dayan 1992, Schulman *et al.* 2017). Recently, the policy-gradient method has shown state-of-the-art improvements toward graph RL research and navigation tasks (Schulman *et al.* 2017, Bøhn *et al.* 2019, Silva *et al.* 2020). In policy gradient methods, build upon an estimator of policy gradient and plug it into a stochastic gradient ascent algorithm that adjusts the weights of the policy towards the maximum rewards. The most common gradient estimator $\hat{g}$ is given by:

$$\hat{A}_t(s_t, a_t) = Q(s_t, a_t) - \hat{V}(s_t) \tag{6a}$$

$$\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log_{\pi_\theta}(a_t|s_t)\hat{A}_t\right] \tag{6b}$$

where π_θ denotes stochastic policy, $\hat{A}_t$ is the estimator of the advantage function at time step t, a is action follows $a_t \sim \pi(a_t|s_t)$ and s is state follows $s_t + 1 \sim P(t+1|s_t, a_t)$ for $t \leq 0$. In this case, the expectation $\hat{\mathbb{E}}_t$ is the empirical average over a finite trajectory τ of our environment samples. Here we define the trajectory τ as a sequence of state, action, and rewards:

$$\tau = \{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots\} \tag{7}$$

$Q(s_t, a_t)$ and $\hat{V}(s_t)$ in Equation (6a) are state-action value function derived from equation (2) via MDP in the trajectory. This can be intuitively taken as the difference of Q

value (future discounted rewards) and the average of action which it would have taken at state. It shows the extra reward that an agent could obtain by taking a turn in an intersection. Since $\hat{g}$ is a gradient estimator like other estimators in deep learning models, it is obtained by an objective function (or loss function) that differentiates the objective:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log_{\pi_\theta}(a_t|s_t)\hat{A}_t\right] \tag{8}$$

where θ is the policy parameter. This policy gradient optimization makes RL frameworks more like general optimization problems where the deep learning training method is applicable. Then training method involves an objective to minimize the loss function, where the parameterized approximator can be established. Conversely, reinforcement learning consists of an unknown and non-differentiable dynamic model, which causes the increased variance of the gradient estimator. Considering these limitations, we use Proximal Policy Optimization (PPO) (Schulman *et al.* 2017) to optimize the policy. In PPO methods, the objective function is expressed as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \tag{9}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}$ is the ratio of the probability under the new and old policies and ϵ is a tunable hyper-parameter (generally 0.1 or 0.2). The clip term $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ in equation 9 modifies a surrogate objective function (Schulman *et al.* 2015) by clipping the probability ratio, where the minimum of the clipped and unclipped objective is taken and the final objective is a lower bound on the unclipped objective (Schulman *et al.* 2017). In the experiment Section 5, the agent adapted a well-established learnable estimator, so a tuned neural network will be introduced in the next step to train the evacuation model.

4.3.2. Action mask

As stated in the previous Section 4.2, the existing flood evacuation algorithms lack the capability of dealing with complex environments. For example, Sharma *et al.* (2021) uses nodes in observation as action to train an agent to plan an evacuation route in a fire situation. It requires the agent to first learn to memorize the neighborhood nodes and choose a valid action via Q-learning (shown in Figure 2(b)). Figure 2(c) also shows an example of action space without an active mask. Once the observation extends to a large number of nodes (eg over a thousand nodes), the Q-learning method will be extremely hard to converge (Low *et al.* 2019). In contrast, Figure 2(d) demonstrates how the action mask works in our flooding evacuation model, and only valid action is presented to the policy. To remove the invalid actions and adapt the policy network to our environment, we designed an action mask method to avoid invalid actions affecting our policy network, which masks all invalid actions in each state. To incorporate the change in action, the policy network in Section 4.3.1. needs the following modifications:

1. The trajectory τ in Equation (7) only use valid actions
2. Only valid actions are calculated in Equation (6b) during the gradient descent.

The policy network first outputs logits (also known as scores without normalization in the neural network) and then converts them into an action probability with an activation function. In this case, the MDP is an action set with three directions $A = \{a_0, a_1, a_2\}$ denoted as turn left, proceed, and turn right respectively. The decision to limit the action space to these three options was made based on a comprehensive statistic of real-world road networks Appendix D. Our studies showed that intersections requiring more than these three actions, such as 5-way intersections or situations necessitating a U-turn, constitute a minor percentage of the total intersections in major cities. Moreover, expanding the action space to include these additional actions would significantly increase the complexity of the model. Further, consider a policy π_θ in Equation (6b) parameterized by $\theta = [l_0, l_1, l_2] = [1.0, 1.0, 1.0]$. Assume we directly use θ as the output logits for easier representation. Then in an initial state s_0 that an agent in a start node, we have:

$$\begin{aligned} \pi_\theta(\cdot|s_0) &= [\pi_\theta(a_0|s_0), \pi_\theta(a_1|s_1), \pi_\theta(a_2|s_2)] = \text{activation}([l_0, l_1, l_2]) \\ \pi_\theta(a_i|s_0) &= \frac{\exp(l_i)}{\sum_j \exp(l_j)} \end{aligned} \tag{10}$$

Assume a_0 is invalid for state s_0, the re-normalized probability distribution $\pi'_\theta(\cdot|s_0)$ will be calculated by an activation function (eg softmax) as $[0.5, 0.00, 0.5]$. Thus when an agent traverses a road network, every time an agent makes a decision at an intersection, only a valid turn will be chosen by the policy network.

4.3.3. Exploration

In reinforcement learning, 'agent exploration' plays an important role in the training process. The existing literature indicated that agents in reinforcement learning algorithms suffer from sparse reward issues (Savinov *et al.* 2018). For example, the agent cannot receive a reward until it arrives at a destination in a disaster event. One solution to this problem is to design an intrinsic reward by the agent itself, following steps such as observing, memorizing, and recalling (Savinov *et al.* 2018). Let the intrinsic reward generated by the agent at time t be r_t^i, the original reward be r_t^e and the trade-off parameter between the exploration and exploitation be β. The optimized reward can be:

$$r_t = r_t^i + \beta \cdot r_t^e \tag{11}$$

In this way, a new objective of an agent can be developed by maximizing the sum of these two rewards. Pathak *et al.* (2017) used self-supervised representation learning to encode the observation feature for exploration. Conversely, Seo *et al.* (2021) encourages exploration without introducing representation learning but utilizes a k-nearest neighbor state entropy estimator in a randomly initialized encoder. In our policy network, we adapt these two intrinsic reward methods to our agents.

1. **Reward by auto-encoder representation learning:** The auto-encoder representation learning follows the idea of Pathak *et al.* (2017). Given a raw observation s_t, an auto-encoder neural network (Lange and Riedmiller 2010) is used to encode it to a feature vector ϕ, where the feature vector stores the information of the road

situation (eg the distance between the goal node and current node). The auto-encoder consists of two modules: encoder and decoder (also called inverse model and forward model). The predicted action $\hat{a}_t$ from action a_t taken by the agent from the state s_t can defined as:

$$\hat{a}_t = g_e(\phi(s_t, \theta_e), \phi(s_{t+1}, \theta_e); \theta_i) \tag{12}$$

where θ_i and θ_e are the learnable parameters in the encoder neural network g_e and are trained to optimize the minimal discrepancy between the original action and predicted action. In the decoder g_d, the a_t and $\phi(s_t, \theta_e)$ can be used to predict feature encoding of the s_{t+1} :

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t, \theta_e), a_t; \theta_f) \tag{13}$$

where f is the function that is trained to optimize the regression loss between the predicted estimate of $\hat{\phi}(s_{t+1})$ and the actual $\phi(s_{t+1})$ and finally, the intrinsic reward is computed as:

$$r_t^i = ||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})||_2^2, \eta > 0 \tag{14}$$

In the section of experiment 5, we introduce our tuned auto-encoder network for intrinsic reward agents.

1. **Reward by random-encoder:** The random-encoder (Seo *et al.* 2021) is based on a k-nearest neighbor entropy estimator (Singh *et al.* 2003). The random encoder is based on a state entropy, which is calculated based on its distance from k-nearest neighbor states present in the replay buffer in the representation space. Let $X \subset \mathbb{R}^q$ be a random variable with a probability function p. The random-encoder starts by using the differential entropy (Michalowicz *et al.* 2013) to create a high-dimensional feature space (or representation space) to monitor the observation of a moving agent. The high-dimensional space can be represented as $\mathcal{H}(X) = -\mathbb{E}_{x \sim p(x)}[\log p(x)]$. Since our observation of road network in low dimensional, we employed particle-based k-nearest neighbors (k-NN) as training samplers for X as follows:

$$\hat{\mathcal{H}}_N^k = \frac{1}{N} \sum_{i=1}^{N} \log ||x_i - x_i^{k-\mathrm{NN}}||_2 \tag{15}$$

where $x_i^{k-\mathrm{NN}}$ denotes the k-NN of x_i within a set $\{x_i\}_{i=1}^N$ (Seo *et al.* 2021). The difference between the states in the low-dimensional feature space of a randomly initialized encoder can be calculated using the distance measure. By utilizing Equation (15), we can treats each transition as a particle (Liu and Abbeel 2021), the intrinsic reward of the random-encoder can be expressed as:

$$r_t^e = \log(||y_i - y_i^{k-\mathrm{NN}}||_2 + 1) \tag{16}$$

where y_i is a fixed representation outputs from a random-encoder and $y_i^{k-\mathrm{NN}}$ is the k-

nearest neighbors of y_i. Measuring the difference between states in the feature space can enable a more stable intrinsic reward as the pair of states does not change during training (Seo *et al.* 2021). The comparison of those two explorations is present in Section 5.

4.3.4. Scaling optimization

Navigating large road networks often leads to sparse rewards, making it challenging for agents to reach their goals. The design of scaling optimization assists the agent in demonstrating robustness in real-world applications (Li *et al.* 2019a). Researchers, inspired by human imitation behavior, employ imitation learning (IL) (Schaal 1999) to learn from historical trajectories to construct an agent. This approach circumvents the need to learn from sparse rewards or manually specify a reward function. The resulting agent can be designed for higher horizons or multi-task goals in industry applications (Magzhan and Jani 2013).

In this section, we present a scaling optimization design that imitates expert decisions to support agent navigation in a road network comprising over 80k nodes. The behavior cloning optimization trains an agent using a sequence of decisions from human experts $\hat{\tau}_i \in \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_m\}$, wherein each decision comprises expert trajectories as follows:

$$\hat{\tau}_i =< (s_1^i, a_1^i), (s_2^i, a_2^i), \ldots > \tag{17}$$

For each trajectory τ in the dataset $\hat{\tau}$, the scaling optimization estimates the advantage function $\hat{A}^{\pi}(s_t, a_t) = (R_t - V_\theta(s_t))/c$ for time $t = 1, \ldots, t$. This function measures the relative quality of an action at a given state compared to the average action at that state under the policy. Here, $V_\theta(s_t)$ represents the value function discussed in Section 4.1, and R_t represents the modified reward function $R_t = \alpha \cdot r_t^i + \beta \cdot r_t^e + \chi \log(||z_t - z_t^e||_2 + 1)$, where z is a path representation containing a sequence of node ids at timestamp t. The term z_i^e denotes an expert path decision. In the experiment settings (Section 5), we utilized a random sampling method to summarize the path feature and the path dimension was adjusted according to statistical results. We used the implementation of the monotonic advantage reweighted imitation learning (Wang *et al.* 2018) method in Ray Rllib (Liang *et al.* 2017) for behavior cloning optimization by maximizing the following estimation with hyperparameter β:

$$\mathbb{E}_{(s_t, a_t) \in \hat{\tau}} \exp(\beta \hat{A}^{\pi}(s_t, a_t)) \log \pi_\theta(a_t|s_t) \tag{18}$$

In traditional pathfinding problems, the shortest path can be approximated using greedy or heuristic algorithms. For our experiment, we utilized a trajectory generated by a multicriteria decision-making routing algorithm under flood emergency conditions (Alizadeh *et al.* 2022) that we had previously collected. When the environment becomes static—ie the flood ceases to evolve and avoidance areas are no longer a concern—the routing algorithm simplifies to a general shortest path finding the problem. In such circumstances, the decisions made by these algorithms can be considered expert decisions.

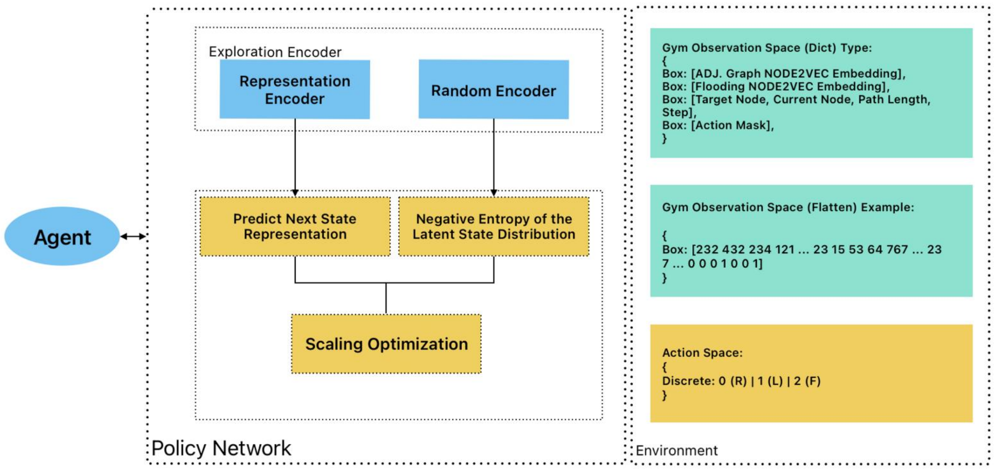
5. Experiments and results

5.1. Flooding environments

We can obtain flooding information in urban areas using various information sources. Using the lowland attribute we added in Section 3, if the weather of the flooding environment is rainfall or the statistic precipitation level is high, we claimed that the lowland intersections have a higher chance of being potential flooded points according to the existing flooding information. In a real-world environment, flooding conditions in urban areas are more uncertain due to weather factors. Thus, we added the 'evolving' mechanism to simulate this situation. The 'evolving' setting can be configured as a parameter of the flooding environment. If the 'evolving' setting is enabled, we will randomly choose many intersections labeled with lowland (excluding the origin point and goal point) as new flooding points with average flooded depth. For example, if the 'evolving' is configured as 10, a new flooded point will generate every 10 steps in an episode. In the next episode, the environment will also be reset to clean up all generated flooded points and keep only stop signs and gauge flooding information.

5.2. Experiments and training

Our research was demonstrated in two distinct study areas: Houston and New York City. In the case of Houston, flood data was sourced from the period of Hurricane Harvey (August 17, 2017 to September 3, 2017). The New York City environment was selected to demonstrate the scalability of our algorithm. Comprehensive information regarding our experiments is provided in Appendix B. Figure 3 illustrates the workflow of how our policy neural network interacts with the flooding environment. Within the policy neural network, we utilize an exploration encoder, comprising a representation encoder and a random encoder, to aid agents in identifying the destination point



```
Gym Observation Space (Dict) Type:
{
Box: [ADJ. Graph NODE2VEC Embedding],
Box: [Flooding NODE2VEC Embedding],
Box: [Target Node, Current Node, Path Length,
Step],
Box: [Action Mask],
}
```

```
Gym Observation Space (Flatten) Example:

{
Box: [232 432 234 121 ... 23 15 53 64 767 ... 23
7 ... 0 0 0 1 0 0 1]
}
```

```
Action Space:
{
Discrete: 0 (R) | 1 (L) | 2 (F)
}
```

Figure 3. A demonstration of our policy neural network interacts with a flooding environment. The expected reward is integrated with intrinsic reward, extrinsic reward, and a scaling optimization.

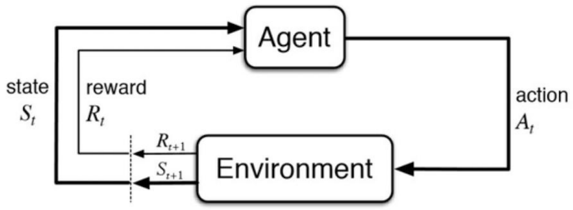
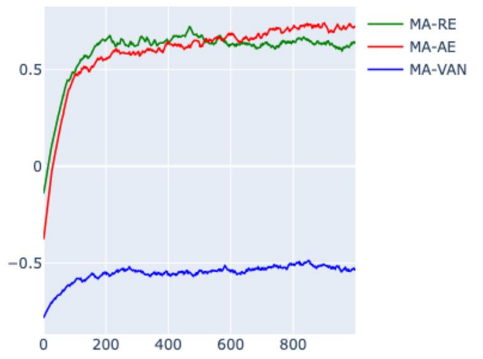
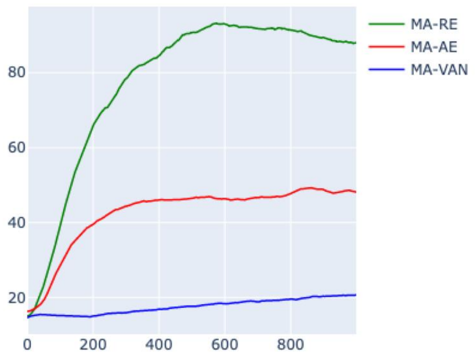


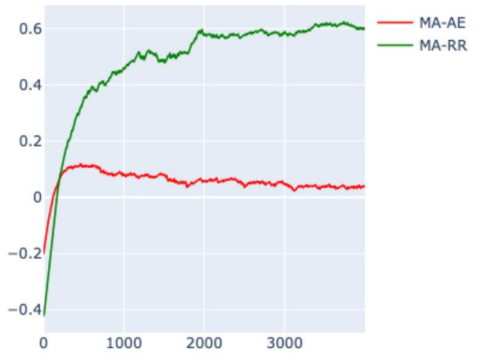
Figure 4. A demonstration of reinforcement learning simulation process.



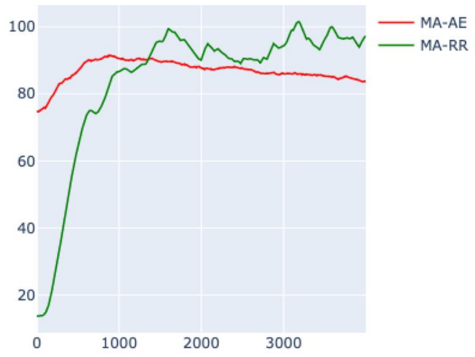
(a) The moving-average returned episode reward of Houston small size map.



(b) The moving-average returned episode length of Houston small size map.



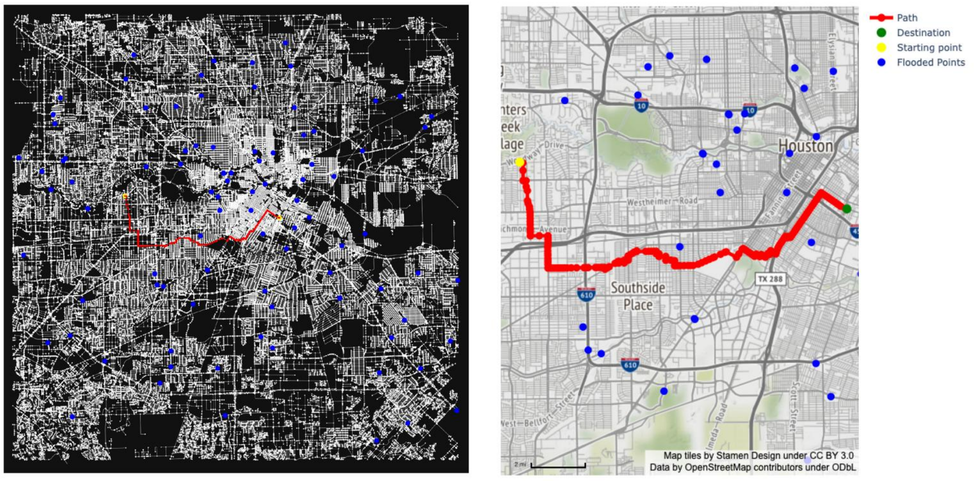
(c) The moving-average returned episode reward of Houston large size map.



(d) The moving-average returned episode length of Houston large size map.

Figure 5. A comparison of results of different optimization methods in our experiments. The y-axis is returned episode reward and returned episode length and the x-axis is the training steps. Here we use an exponentially weighted moving average with 500 window size to smooth the line chart for better visualization. The smoothed line chart is marked with the prefix 'MA-'. The 'RE' is the experiment of the random-encoder method. The 'AE' is the experiment of the auto-encoder representation learning method. The 'VAN' is the vanilla version of proximal policy optimization. The 'SC' is the scaling optimization method.

within a road network. The observation space, depicted on the left side, consists of various types of information, including neighborhood graph embedding, flooding graph embedding, target node, current node, path information, and action mask. The



(a) Route demonstration in training environment. (b) Evacuation route demonstration in open street map.

Figure 6. A demonstration of our policy neural network interacting with flooding environment.

action design, detailed in Section 4.3.2, is characterized by three directional movements: right (R), left (L), and forward (F). The reinforcement learning simulation process is depicted in Figure 4. In our configuration, the graph functions as the environment, with the simulation process running continuously until convergence of the policy network is achieved. The graph environment provides a reward as feedback to the policy network for parameter optimization, based on the actions taken by the agent. The simulation episode concludes when the agent reaches the destination point or exceeds the maximum horizon (episode length), at which point the environment returns a terminal signal. As demonstrated in Figure 5, the overall simulation comprises approximately 2 million episodes, with each episode containing around 100 steps. This robust simulation process ensures a comprehensive exploration of the environment and the effectiveness of our policy network.

In the training step, we followed the definition of the path finding algorithm (Hart *et al.* 1968), so an agent starts from an arbitrary node and moves along the edge between nodes as a path until it reaches the destination node. Since the hyper-parameter tuning for different models varies from experiment to experiment, we used ‘Tune’ (Liaw *et al.* 2018) to adjust the neural network structure and all other configurations. For example, in the Houston experiment, the tuned neural network structure for a random-encoder method is $256 \times 256 \times 256$ fully-connected hidden layers with ‘RELU’ activation functions (Agarap 2018). In the experiment of the auto-encoder method, the inverse model and forward model were both set to 256×256 and the hidden layers were tuned to $256 \times 256 \times 256$ with ‘RELU’ activation functions. The PPO policy network was tuned with 128 full-connected hidden layer for action masking with fully-connected hidden layers with a dimension of $256 \times 512 \times 256$. The learning rate for small and large environments was set to 0.0003 and 0.0005, respectively. In general, the tunable reward factors α and β can be set to 0.5. As mentioned in the literature (Pathak *et al.* 2017), the reward factor can also be optimized by linear or exponential

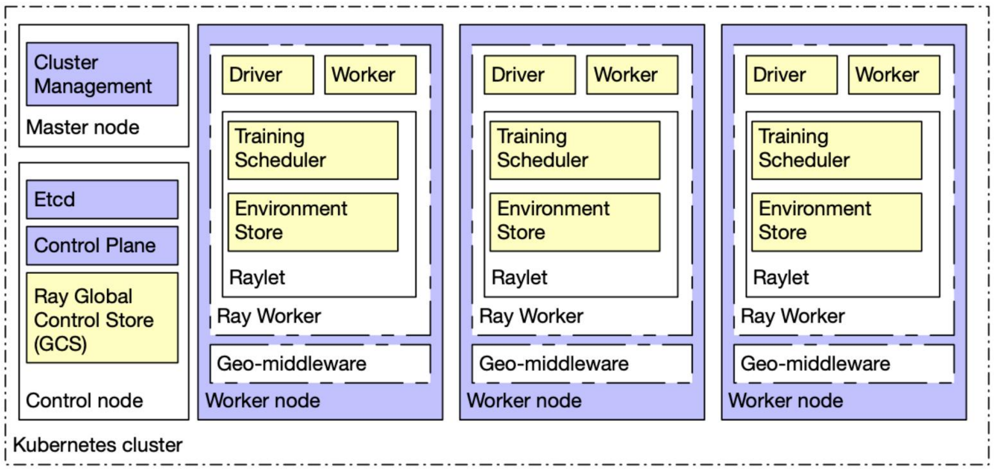


Figure 7. A illustration of using high-performance cyberinfrastructure architecture to implement the reinforcement learning routing algorithms. We used four nodes to balance the workloads for training and each node has a 64-core CPU with an NVIDIA Ampere A30 Tensor Core GPU to accelerate the computing.

decaying function. During the large experiment training, we increased the weights of χ up to 0.5 following a linear space to help the agent follow the scaling optimization (Figure 6 shows the result of evacuation routing in a large flooding environment with 0.5 weighted scaling optimizations). We reset the environment in every training epoch to randomly sample a different node pair as the start node and destination node for generalization purposes. At the beginning of the training (for the first 100 epochs), we manually control the distance between the start node and destination node to let the agent learn the reference route faster (Goecks *et al.* 2019). Besides, we developed a data loader to process multi-dimensional data collected from each observation. In the data loader, the observations are flattened, and the different states in observation are flattened as $1 \times N$ dimensions. We developed a high-performance geospatial cyberinfrastructure to optimize computational efficiency (Li and Zhang (2021)). For example, we used a Ray cluster (Liang *et al.* 2017) with three nodes to balance the workload of RL training. Figure 7 illustrates the general framework of the geospatial cyberinfrastructure used in this project. We used four nodes to balance the workloads for training, and each node has a 64-core CPU with an NVIDIA Ampere A30 Tensor Core GPU to accelerate computing. The master node and control node control all computing resources in our cluster and maintain the weights of the policy network. Each worker node is a replica set of training units, and it clones the flooding environment to perform distributed training.

5.3. Evaluation and results

We evaluated our model in several experiments. Because our task is focused on safety evacuation paths during the flooding event, the quality of the paths is measured using a safety metric that can be expressed as below:

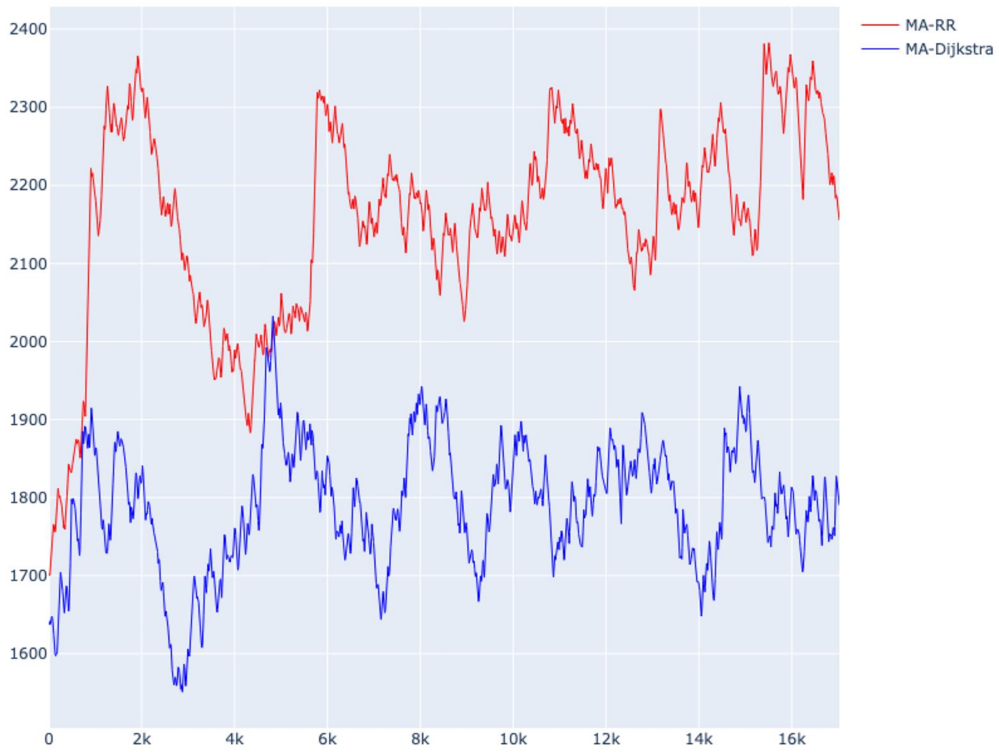


Figure 8. A demonstration of routing evaluation on random sampled Houston environment in 16,000 times. The red line is the smoothed moving-average using our method and the blue line is the smoothed moving-average using traditional method. The y-axis is the distance (meters) to the closest flooded point and the x-axis is the number of sampled routes.

$$\text{Safety} = \frac{\sum_{i=0}^{n} \min(dist(i, \mathcal{F}))}{n} \tag{19}$$

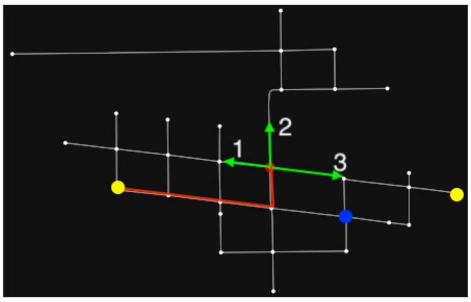
where n is the number of nodes along the path, i is the index of the nodes, and function *dist* calculates the closest distance between the current node i and the flooded node-set $\mathcal{F}$. Figure 8 illustrates our recorded results from 16,000 route samplings. The blue line depicts the safety test results obtained using a traditional Dijkstra routing algorithm, while the red line represents the results from the scaling optimization. Both 'MA-' lines have been smoothed using an exponentially weighted moving average with a window of 1000 for clearer visualization. In a real flood event, a distance dropping to zero signifies a vehicle being inundated. The experimental results indicate that the traditional evacuation routing algorithm more frequently traverses flooded points, thereby potentially compromising passenger safety. Although our model occasionally approaches flooded points (eg around the 14,000th sample), it predominantly maintains an evacuation route that is safely distanced from flooding. This proximity to flood points is a result of the model balancing the effects of the reward discount (γ) and the safety factor. Importantly, our method does not directly traverse flooded points, suggesting that it surpasses traditional algorithms in terms of safety evaluation.

In reinforcement learning, the model's performance is evaluated using various metrics, including the episode reward and the episode length. The episode reward, comprising a sequence of states, actions, and rewards ending with a terminal state, reflects the model's ability to accumulate rewards within a specific environment and RL setting. Meanwhile, the episode length, indicating the number of actions an agent takes in an episode, can shed light on the agent's exploration capabilities and strategy utilization.

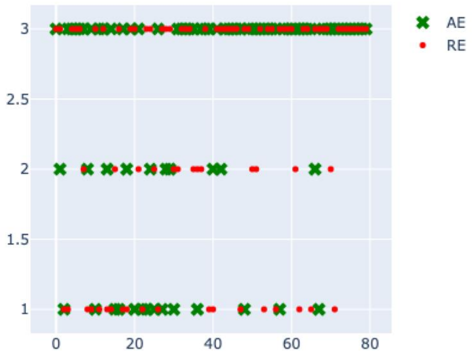
In our study, the episode length also represents the number of interactions the agent has with the environment. We employed distributed training techniques, scaling up to 4000 replications per iteration across three working nodes. We recorded the episode rewards and episode length after 400,000 trained environments for a small set of environments and after 1,600,000 trained environments for a larger set, using these as evaluation metrics.

Figure 5 presents the evaluation results across different training environments, with the y-axis indicating returned episode reward and length, and the x-axis representing training steps. An exponentially weighted moving average was used to smooth the line chart for enhanced visualization. Figure 5(a) demonstrates the training evaluation on Houston's small size map (a rendered demonstration can be found at 1). The detailed evaluation score is also listed in Appendix C. The results indicate that reinforcement learning models with extrinsic reward augments outperform the vanilla version of proximal policy optimization in terms of episode rewards. The auto-encoder optimization had a slower convergence than the random-encoder. However, the auto-encoder representation method had a better final performance at the conclusion of the training because the random-encoder takes a different exploration strategy during the training. Figure 5(b) shows the episode length evaluation of the random-encoder is much higher than the auto-encoder method. This means the random-encoder recommends an evacuation route with more detours to avoid flooded areas, thereby providing a relatively bad consideration of the 'shortest' factor.

Therefore, we used the auto-encoder proximal policy optimization (PPO) model as our base model to generate optimal evacuation routes. To better analyze the actions taken by different optimization methods, we sampled those actions in a small test environment (Figure 9(a)). When an agent takes an action during training (green arrows are the three available actions that the agent can choose), we record each action and plot its distribution. Figure 9(b) shows the comparison of the auto-encoder representation method and the random-encoder method. We found out that the red dot (random-encoder) did not show good convergence for action 3 (Figure 9(b)) in the later training process, which means that the random-encoder had a much longer exploration period. This is also in line with the previously mentioned longer episode length in the evaluation of the random encoder. Based on the auto-encoder, we added scaling optimization for large environment training. Figure 5(c) shows the scaling optimization (green line) with the behavior cloning method to plan the evacuation route in a large city road network. The results show that PPO methods with auto-encoder optimization can only produce good results when the agent is far away from the submerged point and barely able to reach the destination point. Figure 5(d) also shows that the episode length of the scaling optimization method remains within a reasonable range.



(a) An agent takes action during the training. The green arrows are three actions (Turn left, Proceed, and Turn right) in action space.

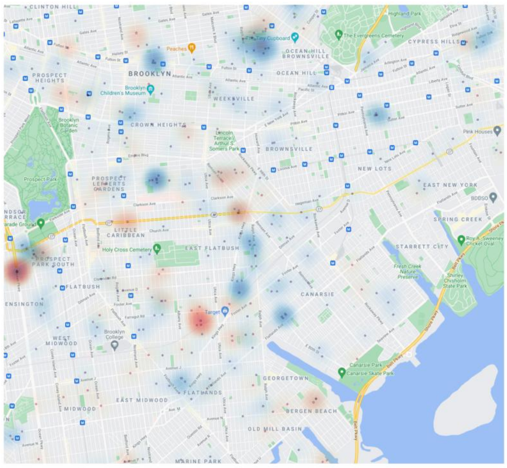


(b) The y-axis is the action that the agent takes. The x-axis is the training steps in one episode.

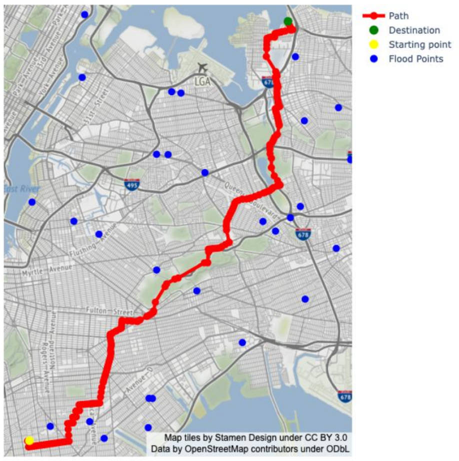
Figure 9. The exploration and exploitation comparison of auto-encoder representation learning method (AE) and random-encoder method (RE). In this small environment (shown in the left figure), the best action is 'turn right', which is numbered action '3'.

In Figure 5, the episode curves for the large and small Houston maps differ significantly. This divergence is primarily attributed to the exploration-exploitation trade-off and environmental non-stationarity. Initially, the agent's exploration strategy identifies rewarding action sequences, leading to an episode reward and length surge. However, as the agent refines its policy amidst continued exploration, temporary setbacks may occur, explaining the subsequent reward and length drop. This phenomenon is particularly pronounced in large, complex environments where simplistic models may struggle to capture the full environmental complexity, resulting in suboptimal performance. The non-stationarity of the environment, as detailed in Section 5.1, also plays a crucial role. Our model's evolving mechanism, simulating urban flooding's uncertain nature, introduces new flooding points during episodes in large road networks, increasing environmental complexity and unpredictability. This contrasts with small road networks, where shorter episode horizons limit the evolving mechanism's impact. On the small road network, the agent rapidly learns an optimal policy due to the environment's relative simplicity and stability. Conversely, on the large road network, the agent requires more time to adapt to the complexity and variability introduced by the evolving mechanism, explaining the initial reward and length increase, subsequent drop, and ultimate convergence.

We conducted simulations to investigate how our policy neural network interacted with the flooding environment using data collected during the Hurricane Harvey period from August 17, 2017, to September 3, 2017. The results of this simulation are presented in Figure 6. In the presence of changing flood conditions, the map information in traditional navigation software becomes distorted. However, our agent can improve the reference path even without this information by combining street information within the near-visible range and external flood information to develop an optimized evacuation route. Our approach generated an evacuation route for both the training environment (Figure 6(a)) and OpenStreetMap (Figure 6(b)). The outcomes



(a) 311 calls made during the week following Hurricane Ida (blue) and the week following Hurricane Henri (red) from Wu (2021)


Path
Destination
Starting point
Flood Points
LGA
Map tiles by Stamen Design under CC BY 3.0
Data by OpenStreetMap contributors under ODbL

(b) The results of using New York City 311 data to test the adaptability of our proposed framework. The yellow dot and green dot represent starting point and destination point, respectively.

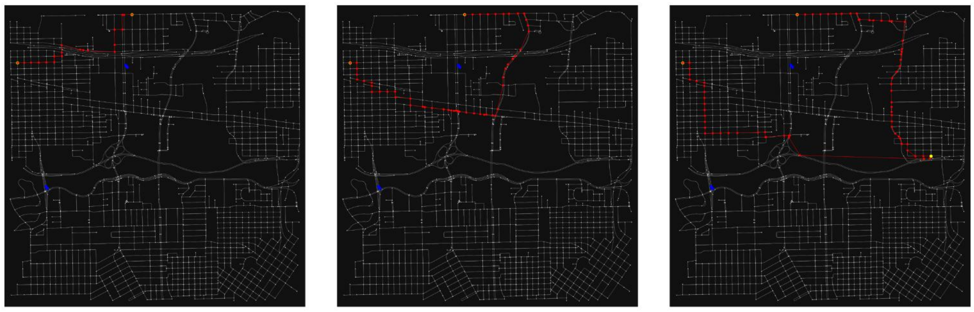
Figure 10. An illustration of using New York City 311 data to build a flooding environment for testing our reinforcement learning routing algorithm. (a) 311 calls made during the week following Hurricane Ida (blue) and the week following Hurricane Henri (red) from Wu (2021). (b) The results of using New York City 311 data to test the adaptability of our proposed framework. The yellow dot and green dot represent starting point and destination point, respectively.

indicate that our approach can efficiently plan a secure route during a flood event with limited information on the actual road network.

Our proposed framework worked well using various datasets. For example, Wu (2021) used New York City 311 data (NYCOpenData 2022) to create a near-real-time flooding map to help people understand the impact of Hurricane Ida. Figure 10(a) shows the flooding map in New York City created by New York City 311 data. Simply replace the flooding dataset with the previously mentioned (Section 3) data format, and we train and run our model into a different environment. The results of using New York City 311 data instead of the BluPix dataset to test the adaptability of our proposed framework are thus plotted in Figure 10(b). Our model still works well in a new environment using different data sources.

6. Discussion

The reinforcement learning method estimates each state-action value to optimize disaster evacuation strategies. More route strategies and spatial patterns can be found and investigated by choosing different reward factors instead of the proposed optimization techniques. For example, we can use road type attributes as a reward factor to change the road preference for the evacuation agent. Figure 11(a) shows the regular evacuation plan in a small environment without road type preferences. By rewarding the route with a road type $r^{[\mathrm{road}]} = \frac{L_P}{L}$ (eg L_P denotes 'primary' road type and L



(a) Using road type "all" to set road preference. (b) Using road type "primary" to set road preference. (c) Multi-stops route planning in Houston small road network.

Figure 11. Reinforcement learning evacuation planning model adaptations. The yellow dots represent two endpoints between a route. The blue dots represent flooding points in the environment. (a) Using road type 'all' to set road preference. (b) Using road type 'primary' to set road preference. (c) Multi-stops route planning in Houston small road network.

denotes the total length from start point to destination), the agent will choose a different route to complete the task (shown in Figure 11(b)).

Compared with other heuristic algorithms, reinforcement learning shows good adaptability so developers can easily add multiple tasks without redesigning the policy optimization method (Cai *et al.* 2019). For example, emergency managers often need to add stops during evacuation planning (eg picking up their child from school). By adding a bonus point in the road network that returns an extra reward to the agent, we can fine-tune the model to plan a multi-stop route in a dynamic environment (shown in Figure 11(c)).

As shown in Figure 8, our method is not significantly affected when attempting to create a 'safe' route versus a 'short' route. The reinforcement learning model seeks to make an informative decision that leverages the different conditions to create a better route. However, a 'do not get flooded' bottom line still restricts the agent's behavior. This restriction is beneficial for developers and users to understand its strict characteristics for future development and usage.

7. Future works

Although our analytical experiments demonstrate good results in route planning, testing on real flooding events is needed to validate our algorithms. Future research will focus on a field experiment with actual weather and road conditions to put the theoretical work into practice. This will also provide an opportunity to test the potential inclusion of more complex actions in the action space, thus further enhancing the practicality and versatility of our model. Extending this approach to broader impacts, our RL routing algorithm can also be adapted for future self-driving systems and even exoplanet rovers. Given the similarities of disaster events, exoplanet rovers also lack external guidance. With limited online information (GPS or other satellite imagery), the reinforcement learning agents can be trained to navigate the rover to its destination.

Moreover, the inherent flexibility of reinforcement learning provides the possibility of incorporating more complex actions into the action space, including 5-way intersections and U-turns. At the same time, these actions can enrich the algorithm's capacity and enable the agent to navigate through more complicated road networks. On the other hand, these new features will increase the algorithm's computational complexity in terms of convergence and exploration which should be considered in the implementation stage. As part of our future endeavors, we plan to explore additional training and optimization techniques, such as the use of a pre-trained routing policy (Wu *et al.* 2023). We anticipate that such improvements could considerably broaden the applicability of our model to a diverse array of road situations.

8. Conclusion

In this study, we developed an RL-based routing algorithm to help people navigate in urban areas during flooding events under complex, information-limited, and dynamic road network conditions. Our graph-based RL model, equipped with real-world action design and reward settings, has demonstrated practicality and feasibility in navigating real-world scenarios with multiple flood information sources. The key findings and implications of our study are:

- Our model effectively learns spatial information around flooding areas, producing high-quality evacuation routes even in the face of changing road conditions (eg road closures caused by flooding) and limited neighborhood information.
- The integration of reinforcement learning into the routing algorithm represents a significant intellectual contribution to GIScience, enabling navigation in complex and dynamic large road networks.
- The practical implications of our research extend to the field of disaster management, providing safe evacuation routes during flooding events by embedding routing information during the training processes.
- The versatility of our method allows for adaptation to other disaster events such as earthquakes or volcanic eruptions, making it a valuable tool for urban navigation.

Despite these promising results, our research has limitations, particularly regarding the scalability of the model to larger road networks (eg nationwide) and the need for training resources. Future research directions should focus on improving the model's scalability and incorporating effective training methods, such as pre-training techniques. We believe that these improvements will further enhance the utility of our model in disaster management and safety planning.

Notes

1. https://blupix.geos.tamu.edu/MapPage
2. https://data.cityofnewyork.us/Social-Services/NYC-311-Data/jrb2-thup
3. https://github.com/mebauer/nyc-311-street-flooding.

Acknowledgement(s)

We thank the anonymous reviewers for their constructive comments and suggestions.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This study is funded by award #NA18OAR4170088 from the National Oceanic and Atmospheric Administration (NOAA), U.S. Department of Commerce, and Texas A&M TAMIDS Data Resource Development Program.

Data and codes availability

The data and codes are available at https://doi.org/10.6084/m9.figshare.24422143.

Notes on contributors

Diya Li is a Ph.D. Student in Texas A&M University. His research interests include spatially explicit Artificial Intelligence, Generative Large Language Models, Spatial Decision-making, and Cyberinfrastructure.

Zhe Zhang holds a Ph.D. in Geoinformatics from Aalto University, Finland. Her primary research area is Geographic Information Science, and within it, a focus on Cyberinfrastructure, knowledge-driven spatial decision-making, Geospatial Artificial Intelligence, and spatial uncertainty modeling. Her interdisciplinary research investigates intersections between decision science, and advanced cyberinfrastructure to address societal challenges related to climate change, coastal sustainability, natural hazards, and disaster management.

Bahareh Alizadeh holds a Ph.D. in Construction Science from Texas A&M University. Her research interests include Artificial Intelligence (AI), Deep Learning, Computer Vision, and Statistical Analysis.

Ziyi Zhang is a Ph.D. Student in Computer Engineering in Texas A&M University. His research interests include Time Series Analysis, Machine Learning, Causality, and Optimization.

Nick Duffield holds a PhD in Physics from the University of London, UK. He is Director of the Texas A&M Institute of Data Science and Royce E. Wisenbaker I Professor in the Department of Electrical & Computer Engineering. His research interests include Data Science and Computer Networking.

Michelle A. Meyer holds a Ph.D. in Sociology from Colorado State University. Her research interests include disaster mitigation and recovery, social vulnerability, and volunteer organizations in disaster management.

Courtney M. Thompson holds a Ph.D. in Geography from the University of Idaho. Her research interests include spatial statistical analyses, risk perception, and social vulnerability.

Huilin Gao holds a Ph.D. in Water Resources Engineering from Princeton University. Her research interests include hydrological modeling, satellite remote sensing, water resources management, and climate change.

Amir H. Behzadan holds a Ph.D. in Civil Enginering from the University of Michigan, Ann Arbor. His research interests include urban computing, disaster resilience, applied AI/ML, and human-environment interaction.

ORCID

Zhe Zhang http://orcid.org/0000-0001-7108-182X

References

Abe, K., *et al.*, 2019. Solving np-hard problems on graphs by reinforcement learning without domain knowledge. *ArXiv, abs/1905.11623*.

Agarap, A.F., 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv: 1803.08375*.

Alizadeh, B., *et al.*, 2022. Human-centered flood mapping and intelligent routing through augmenting flood gauge data with crowdsourced street photos. *Advanced Engineering Informatics*, 54, 101730.

Alizadeh, B., *et al.*, 2021. Feasibility study of urban flood mapping using traffic signs for route optimization. *arXiv preprint arXiv:2109.11712*.

Arora, A., Galhotra, S., and Ranu, S., 2017. Debunking the myths of influence maximization: An in-depth benchmarking study. In: *Proceedings of the 2017 ACM international conference on management of data*, 651–666.

Bagherlou, H., and Ghaffari, A., 2018. A routing protocol for vehicular ad hoc networks using simulated annealing algorithm and neural networks. *The Journal of Supercomputing*, 74 (6), 2528–2552.

Bennett, J., 2010. *Openstreetmap*. Birmingham: Packt Publishing Ltd.

Bi, C., *et al.*, 2019. Evacuation route recommendation using auto-encoder and Markov decision process. *Applied Soft Computing*., 84, 105741.

Bøhn, E., *et al.*, 2019. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 523–533.

Cai, Q., *et al.*, 2019. Reinforcement learning driven heuristic optimization. *ArXiv*, abs/1906.06639.

Cavdur, F., Kose-Kucuk, M., and Sebatli, A., 2016. Allocation of temporary disaster response facilities under demand uncertainty: An earthquake case study. *International Journal of Disaster Risk Reduction*, 19, 159–166.

Chamola, V., *et al.*, 2021. Disaster and pandemic management using machine learning: a survey. *IEEE Internet of Things Journal*, 8 (21), 16047–16071.

Chen, Yz., *et al.*, 2014. Path optimization study for vehicles evacuation based on Dijkstra algorithm. *Procedia Engineering*, 71, 159–165.

Chevalier-Boisvert, M., Willems, L., and Pal, S., 2018. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

Christiano, P.F., *et al.*, 2017. Deep reinforcement learning from human preferences. *ArXiv*, abs/1706.03741.

Delling, D., *et al.*, 2012. Robust mobile route planning with limited connectivity. In *Symposium on Algorithm Engineering and Experiments*(ALENEX).

Demircan, S., Aydin, M., and Durduran, S.S., 2011. Finding optimum route of electrical energy transmission line using multi-criteria with q-learning. *Expert Systems with Applications*, 38 (4), 3477–3482.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 (1), 269–271.

Ding, Q., *et al.*, 2021. An overview of machine learning-based energy-efficient routing algorithms in wireless sensor networks. *Electronics*, 10 (13), 1539.

Duraipandian, M., 2019. Performance evaluation of routing algorithm for manet based on the machine learning techniques. *Journal of Trends in Computer Science and Smart Technology*, 01 (01), 25–38.

Feng, Q., Liu, J., and Gong, J., 2015. Urban flood mapping based on unmanned aerial vehicle remote sensing and random forest classifier—a case of Yuyao, China. *Water*, 7 (12), 1437–1455.

Geng, Y., *et al.*, 2021. Deep reinforcement learning based dynamic route planning for minimizing travel time. *In*: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 1–6.

Gharaibeh, N., *et al.*, 2021. Potential of citizen science for enhancing infrastructure monitoring data and decision-support models for local communities. *Risk Analysis*, 41 (7), 1104–1110.

Godfrey, G.A., and Powell, W.B., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, II: Multiperiod travel times. *Transportation Science*, 36 (1), 40–54.

Goecks, V.G., *et al.*, 2019. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. *arXiv preprint arXiv: 1910.04281*.

Gomes, R., and Straub, J., 2017. Genetic algorithm for flood detection and evacuation route planning. In: *Proceedings of Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIII*, 10198.

Grover, A., and Leskovec, J., 2016. node2vec: Scalable feature learning for networks. *In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Hagberg, A., Swart, P., and S Chult, D., 2008. *Exploring network structure, dynamics, and function using networkx*. Los Alamos, NM: Los Alamos National Lab (LANL).

Hai-bo, M., and Yu-bo, S., 2017. Study on evacuation routes under unexpected event based on petri net. In *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, 773–778.

Hart, P.E., Nilsson, N.J., and Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4 (2), 100–107.

He, Q., *et al.*, 2022. A comparison of reinforcement learning models of human spatial navigation. *Scientific Reports*, 12 (1), 13923.

Hendricks, M.D., Meyer, M.A., and Wilson, S.M., 2022. Moving up the ladder in rising waters: Community science in infrastructure and hazard mitigation planning as a pathway to community control and flood disaster resilience. *Citizen Science: Theory and Practice*, 7 (1), 18.

Henonin, J., *et al.*, 2013. Real-time urban flood forecasting and modelling–a state of the art. *Journal of Hydroinformatics*, 15 (3), 717–736.

Herman, I., Melancon, G., and Marshall, M.S., 2000. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6 (1), 24–43.

Huang, S.K., Lindell, M.K., and Prater, C.S., 2016. Who leaves and who stays? A review and statistical meta-analysis of hurricane evacuation studies. *Environment and Behavior*, 48 (8), 991–1029.

Jha, A.K., Bloch, R., and Lamond, J., 2012. *Cities and flooding: a guide to integrated urban flood risk management for the 21st century*. Washington: World Bank Publications.

Kakade, S.M., 2003. *On the sample complexity of reinforcement learning*. London: University of London.

Kharazi, B.A., and Behzadan, A.H., 2021. Flood depth mapping in street photos with image processing and deep neural networks. *Computers, Environment and Urban Systems*, 88, 101628.

Kheradmand, B., *et al.*, 2022. Cluster-based routing schema using Harris Hawks optimization in the vehicular ad hoc networks. *Wireless Communications and Mobile Computing*, 2022, 1–15.,

Kim, J., and Kim, K., 2021. Optimizing large-scale fleet management on a road network using multi-agent deep reinforcement learning with graph neural network. *In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 990–995.

Krishnan, S., *et al.*, 2011. Opentopography: a services oriented architecture for community access to lidar topography. *In: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, 1–8.

Kumar, A., *et al.*, 2021. Should i run offline reinforcement learning or behavioral cloning? *In: International Conference on Learning Representations*.

Kumari, S.M., and Geethanjali, N., 2010. A survey on shortest path routing algorithms for public transport travel. *Global Journal of Computer Science and Technology*, 9 (5), 73–76.

Lange, S., and Riedmiller, M., 2010. Deep auto-encoder neural networks in reinforcement learning. In: *The 2010 international joint conference on neural networks (IJCNN)*. IEEE, 1–8.

Lee, Y.H., *et al.*, 2020. Flood evacuation routes based on spatiotemporal inundation risk assessment. *Water*, 12 (8), 2271.

Leike, J., *et al.*, 2017. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*.

Levy, S., *et al.*, 2020. Saferoute: Learning to navigate streets safely in an urban environment. *ACM Transactions on Intelligent Systems and Technology*, 11 (6), 1–17.

Li, D., and Zhang, Z., 2021. Urban computing cyberinfrastructure: visualizing human sentiment using social media and augmented reality. *In: Proceedings of the 4th ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities*, 27–31.

Li, B., *et al.*, 2021a. A coupled high-resolution hydrodynamic and cellular automata-based evacuation route planning model for pedestrians in flooding scenarios. *Natural Hazards*, 110 (1), 607–628.

Li, Y., *et al.*, 2021b. Constructing reservoir area–volume–elevation curve from TanDEM-X DEM data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 2249–2257.

Li, G., *et al.*, 2019a. Human-centered reinforcement learning: A survey. *IEEE Transactions on Human-Machine Systems*, 49 (4), 337–349.

Li, Y., *et al.*, 2019b. Flood evacuation simulations using cellular automata and multiagent systems-a human-environment relationship perspective. *International Journal of Geographical Information Science*, 33 (11), 2241–2258.

Liang, E., *et al.*, 2017. Ray rllib: A composable and scalable reinforcement learning library. *arXiv preprint arXiv:1712.09381*, 85.

Liaw, R., *et al.*, 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Lim, H., Lim, M.B., and Piantanakulchai, M., 2013. A review of recent studies on flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, 10, 147–162.

Liu, H., and Abbeel, P., 2021. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34, 18459–18473.

Liu, Y., Hatayama, M., and Okada, N., 2006. Development of an adaptive evacuation route algorithm under flood disaster. *Annuals of Disaster Prevention Research Institute*, 49, 189–195.

Low, E.S., Ong, P., and Cheah, K.C., 2019. Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous Systems*, 115, 143–161.

Magzhan, K., and Jani, H.M., 2013. A review and evaluations of shortest path algorithms. *International Journal of Scientific & Technology Research*, 2 (6), 99–104.

Manchanda, S., *et al.*, 2019. Learning heuristics over large graphs via deep reinforcement learning. *arXiv preprint arXiv:1903.03332*.

Meyer, M.A., *et al.*, 2018. Previous hurricane evacuation decisions and future evacuation intentions among residents of southeast Louisiana. *International Journal of Disaster Risk Reduction*, 31, 1231–1244.

Michalowicz, J.V., Nichols, J.M., and Bucholtz, F., 2013. *Handbook of differential entropy*. Boca Raton: Crc Press.

Mirahadi, F., and McCabe, B., 2021. Evacusafe: A real-time model for building evacuation based on Dijkstra's algorithm. *Journal of Building Engineering*, 34, 101687.

Mohammadnezhad, M., and Ghaffari, A., 2019. Hybrid routing scheme using imperialist competitive algorithm and RBF neural networks for VANETs. *Wireless Networks*, 25 (5), 2831–2849.

Moritz, P., *et al.*, 2018. Ray: A distributed framework for emerging {AI} applications. *In*: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561–577.

Murray-Tuite, P., and Wolshon, B., 2013. Evacuation transportation modeling: An overview of research, development, and practice. *Transportation Research Part C*, 27, 25–45.

Nazari, M., *et al.*, 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in Neural Information Processing Systems*, 31,

NYCOpenData. 2022. 311 service requests from 2010 to present: Nyc open data. Available from: https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9.

Osman, M.F.S., and Ram, B., 2013. Two-phase evacuation route planning approach using combined path networks for buildings and roads. *Computers & Industrial Engineering.*, 65 (2), 233–245.

Pathak, D., *et al.*, 2017. Curiosity-driven exploration by self-supervised prediction. *In: International Conference on Machine Learning*. PMLR, 2778–2787.

Perry, C.A., 2000. *Significant floods in the United States during the 20th century – USGS measures a century of floods*. Lawrence, Kansas: U.S. Geological Survey.

Qiu, X., Wan, K., and Li, F., 2019, Autonomous robot navigation in dynamic environment using deep reinforcement learning. *2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 338–342.

Rossi, R.A., and Ahmed, N.K., 2015. The network data repository with interactive graph analytics and visualization. *In*: *AAAI'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. Available from: https://networkrepository.com.

Russo, B., Gómez, M., and Macchione, F., 2013. Pedestrian hazard criteria for flooded urban areas. *Natural Hazards*, 69 (1), 251–265.

Savinov, N., *et al.*, 2018. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*.

Schaal, S., 1999. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3 (6), 233–242.

Schulman, J., *et al.*, 2015. Trust region policy optimization. In: *International conference on machine learning*. PMLR, 1889–1897.

Schulman, J., *et al.*, 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv: 1707.06347*.

Seo, Y., *et al.*, 2021. State entropy maximization with random encoders for efficient exploration. In: *International Conference on Machine Learning*. PMLR, 9443–9454.

Sharma, J., *et al.*, 2021. Deep q-learning with q-matrix transfer learning for novel fire evacuation environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51 (12), 7363–7381.

Shi, Y., *et al.*, 2023. An adaptive route guidance model considering the effect of traffic signals based on deep reinforcement learning. In *IEEE Intelligent Transportation Systems Magazine*.

Silva, S.H., Alaeddini, A., and Najafirad, P., 2020. Temporal graph traversals using reinforcement learning with proximal policy optimization. *IEEE Access.*, 8, 63910–63922.

Singh, H., *et al.*, 2003. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23 (3-4), 301–321.

Sohn, S., *et al.*, 2021. Shortest-path constrained reinforcement learning for sparse reward tasks. *arXiv preprint arXiv:2107.06405*.

Staroverov, A., *et al.*, 2020. Real-time object navigation with deep neural networks and hierarchical reinforcement learning. *IEEE Access.*, 8, 195608–195621.

Su, M.C., *et al.*, 2004. A reinforcement-learning approach to robot navigation. IEEE International Conference on Networking, Sensing and Control, 1, 665–669 .

Sutton, R.S., and Barto, A.G., 2018. *Reinforcement learning: An introduction*. London: MIT press.

Tian, K., and Jiang, S., 2018. Reinforcement learning for safe evacuation time of fire in Hong Kong-Zhuhai-Macau immersed tube tunnel. *Systems Science & Control Engineering*, 6 (2), 45–56.

Trindade, A., *et al.*, 2016. A placement and routing algorithm for quantum-dot cellular automata. In: *2016 29th symposium on integrated circuits and systems design (SBCCI)*. IEEE, 1–6.

Tyagi, N., Singh, J., and Singh, S., 2022. A review of routing algorithms for intelligent route planning and path optimization in road navigation. *Recent Trends in Product Design and Intelligent Manufacturing Systems: Select Proceedings of IPDIMS 2021*, 851–860.

Vicario, S.A., *et al.*, 2020. Unravelling the influence of human behaviour on reducing casualties during flood evacuation. *Hydrological Sciences Journal*, 65, 2359–2375.

Walker, O., *et al.*, 2019. A deep reinforcement learning framework for UAV navigation in indoor environments. *In*: *2019 IEEE Aerospace Conference*. IEEE, 1–14.

Wang, H., *et al.*, 2021. Energy-efficient 3d vehicular crowdsourcing for disaster response by distributed deep reinforcement learning. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3679–3687.

Wang, Q., *et al.*, 2018. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31.

Wang, Q., *et al.*, 2019. Improved multi-agent reinforcement learning for path planning-based crowd simulation. *IEEE Access.*, 7, 73841–73855.

Wang, X., Qu, H., and Yi, Z., 2009. A modified pulse coupled neural network for shortest-path problem. *Neurocomputing*, 72 (13–15), 3028–3033.

Watkins, C.J., and Dayan, P., 1992. Q-learning. *Machine Learning*, 8 (3–4), 279–292.

Wei, C., *et al.*, 2018. Look-ahead insertion policy for a shared-taxi system based on reinforcement learning. *IEEE Access.*, 6, 5716–5726.

Wu, J., 2021. Mapping hurricane Ida's impact. Available from: https://sustainabilityandthecity.commons.gc.cuny.edu/2021/09/20/mapping-flooding-from-hurricane-ida/.

Wu, J., *et al.*, 2023. Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving. *Engineering*, 21, 75–91.

Xiong, W., Hoang, T., and Wang, W.Y., 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Xu, B., *et al.*, 2020. The transborder flux of phosphorus in the Lancang-Mekong river basin: Magnitude, patterns and impacts from the cascade hydropower dams in China. *Journal of Hydrology*, 590, 125201.

Yan, L., *et al.*, 2020. Mobirescue: Reinforcement learning based rescue team dispatching in a flooding disaster. In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 111–121.

Yang, K., and Shekhar, S., 2017. Evacuation route planning. In *ACM SIGSPATIAL 2017*.

Yin, Z., Li, D., and Goldberg, D.W., 2023. Is ChatGPT a game changer for geocoding – a benchmark for geocoding address parsing techniques. arXiv preprint arXiv:2310.14360.

Yin, Z., *et al.*, 2019. An nlp-based question answering framework for spatio-temporal analysis and visualization. *In*: *Proceedings of the 2019 2nd international conference on geoinformatics and data analysis*, 61–65.

Zhang, F., *et al.*, 2021. Reinforcement learning path planning method with error estimation. *Energies*, 15 (1), 247.

Zhang, J., Yang, T., and Zhao, C., 2016. Energy-efficient and self-adaptive routing algorithm based on event-driven in wireless sensor network. *International Journal of Grid and Utility Computing*, 7 (1), 41–49.

Zhang, W., *et al.*, 2016. Emergency evacuation planning against dike-break flood: a GIS-based DSS for flood detention basin of Jingjiang in central China. *Natural Hazards*, 81 (2), 1283–1301.

Zhang, Z., *et al.*, 2014. A fuzzy multiple-attribute decision-making modelling for vulnerability analysis on the basis of population information for disaster management. *International Journal of Geographical Information Science*, 28 (9), 1922–1939.

Zhang, Z., *et al.*, 2019. A cybergis-enabled multi-criteria spatial decision support system: A case study on flood emergency management. *International Journal of Digital Earth*, 12 (11), 1364–1381.

Appendix A. Distance calculation

Distance calculation between two points in our project use great circle distance formula:

$$d = r\Delta\sigma = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda)r \tag{20}$$

where λ and ϕ denote the geographical longitude and latitude of two points 1 and 2; $\Delta\sigma$ denote as the central angle between two points and r denote the radius of the earth.

Appendix B. Road network summary

Table B1. Road networks summary of Houston and New York in our experiments.

City	Avg. Grade	Med. Grade	Interactions	Edges
Houston (Small)	2.0%	1.3%	1521	4088
New York City (Small)	3.6%	2.5%	1281	2572
Houston (Large)	1.1%	0.7%	72486	182525
New York City (Large)	2.0%	1.3%	81099	205896

Appendix C. Evaluation summary

Table C1. Summary of evaluation scores.

Method	Environment	Metric.	Score
Dijkstra	New York City (Large)	Safety	1450.23
ReinforceRouting	New York City (Large)	Safety	1663.01
Dijkstra	Houston (Large)	Safety	1798.06
ReinforceRouting	Houston (Large)	Safety	1963.90
Vanilla-RL	Houston (Small)	Reward	−0.54
RE-RL	Houston (Small)	Reward	0.71
Vanilla-RL	Houston (Small)	Episode Length	20.5
RE-RL	Houston (Small)	Episode Length	87.7
RE-RL	Houston (Large)	Reward	0.12
ReinforceRouting	Houston (Large)	Reward	0.62
Vanilla-RL	Houston (Large)	Episode Length	20.5
ReinforceRouting	Houston (Large)	Episode Length	87.7

Appendix D. Road network statistic

Table D1. Road networks statistic of top four largest cities in U.S.

City	Avg. Degree	Med. Degree	Std.	N0*	N1*	N2*	N3*	N4*	N5*
New York City	2.51	3.00	0.905	0.12	15.16	29.65	42.35	12.60	0.11
Los Angeles	2.77	3.00	0.977	0.08	15.81	13.45	47.90	22.62	0.11
Chicago	2.74	3.00	0.911	0.12	11.28	23.06	45.77	19.58	0.13
Houston	2.48	3.00	0.934	0.14	17.35	30.48	38.66	13.34	0.02

*N*x* means the percentage of the nodes in the road networks have *x* neighbors.