This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

Personalized Route Recommendation with Neural Network Enhanced A^* Search Algorithm

Jingyuan Wang, Ning Wu, and Wayne Xin Zhao

Abstract—In this work, we study an important task in location-based services, namely *Personalized Route Recommendation (PRR)*. Given a road network, the PRR task aims to generate user-specific route suggestions for replying to users' route queries. A classic approach is to adapt search algorithms to construct pathfinding-like solutions. These methods typically focus on reducing search space with suitable heuristic strategies. For these search algorithms, heuristic strategies are often handcrafted, which are not flexible to work in complicated task settings. In addition, it is difficult to utilize useful context information in the search procedure. To develop a more principled solution to the PRR task, we propose to improve search algorithms with neural networks for solving the PRR task based on the widely used A^* algorithm. The main idea of our solution is to automatically learn the cost functions in A^* algorithms, which is the key of heuristic search algorithms. Our model consists of two main components. First, we employ attention-based Recurrent Neural Networks (RNN) to model the cost from the source to the candidate location by incorporating useful context information. Instead of learning a single cost value, the RNN component is able to learn a time-varying vectorized representation for the moving state of a user. Second, we propose to use an estimation network for predicting the cost from a candidate location to the destination. For capturing structural characteristics, the estimation network is built on top of position-aware graph attention networks. The two components are integrated in a principled way for deriving a more accurate cost of a candidate location for the A^* algorithm. Extensive experiment results on three real-world datasets have shown the effectiveness and robustness of the proposed model.

Index Terms—Route Recommendation, A^* Search, Graph Neural Networks, Attention, Deep Learning.

1 Introduction

NOWADAYS, GPS-enabled mobile devices have been widely used by a large number of users, and their trajectory data has been accumulated in a dramatic rate [1], [2], [3], [4]. In the literature, various studies have been proposed to utilize large-volume trajectory data for improving real-world applications. As one of the important applications, *Personalized Route Recommendation (PRR)* plays a key role in many online location-based services (*e.g.,* online map), which aims to generate user-specific route suggestions on instant queries about the path planing from a source to a destination on a road network [5], [6].

Given a large and complex road network, PRR can be considered as a challenging pathfinding task. Early studies mainly focus on how to extend existing heuristic search algorithms (*e.g.,* Dijkstra shortest path algorithms and A^* search algorithm) [7], [8] for the PRR task. Relying on suitable heuristics, they aim to obtain high-quality responses by effectively reducing the search space. For heuristic search algorithms, it is key to develop an effective cost function. A common way adopted by previous studies is to heuristically set the cost function according to empirical experiences and human knowledge. However, it is difficult to utilize various kinds of context information in the search process. For accurate route recommendation, an effective approach should be able to model various influencing factors and rich context information, including personalized preference, spatial-temporal influence and road network constraint. To develop more flexible solutions, machine learning methods have been applied to solve the PRR task [9], [10]. They are able to characterize the location dependencies or spatial-temporal information in a principled way. With the revival of deep learning, neural networks have provided more powerful technical solutions to the PRR task. For example, sequential neural models, *i.e.,* Recurrent Neural Networks (RNN), have been widely used for modeling sequential trajectory data [11], [12], [13], [14].

For the PRR task, the aforementioned two kinds of approaches have their own merits. On one hand, heuristic search algorithms are specially suitable for the PRR task since it essentially solves a pathfinding problem on graphs given the source and destination. With elaborate heuristics, it can generate high-quality approximate solutions in an efficient way. On the other hand, deep learning methods are powerful to capture the complex data characteristics using learnable neural networks [15], [16]. They are able to learn effective mapping mechanisms from input to output or expressive feature representations from raw data in an automatic way. Based on these discussions, this work aims to combine the merits of both kinds of approaches in a principled manner. Our solution is inspired by recent progress of deep learning in strategy-based games (*e.g.,* Go and Atari) [17], which incorporate learnable components in the heuristic search algorithms.

To this end, we propose to improve search algorithms with neural networks for solving the PRR task based on the widely used A^* algorithm [18]. The main idea of our

- *J. Wang and N. Wu are with the School of Computer Science and Engineering, Beihang University, the State Key Laboratory of Software Development Environment, and Beijing Advanced Innovation Center for Big Data and Brain Computing, China.*
  E-mail: {jywang,wuning}@buaa.edu.cn
- *W.X. Zhao (corresponding author) is with Gaoling School of Artificial Intelligence, Renmin University of China, and Beijing Key Laboratory of Big Data Management and Analysis Methods.*
  E-mail: batmanfly@gmail.com.

Manuscript revised xxx.

solution is to automatically learn the cost functions in A^* algorithms, which is the key of heuristic search algorithms. For this purpose, we mainly consider addressing three important issues. First, we need to define a suitable form for the cost in the PRR task. Different from traditional graph search problems, a simple heuristic cost cannot directly optimize the goal of our task [7]. For example, the identified route based on the shortest distance may not meet the personalized needs of a specific user. Second, we need to design effective models for implementing the cost function which instructs the search process, which is a key step in our work. Third, we need to utilize rich context or constraint information for improving the task performance, including spatial-temporal influence and user preference.

To define a suitable form for the search cost, we first formulate the PRR task as a conditional probability ranking problem by computing the sum for the negative log of conditional probabilities for each point in a candidate trajectory. We use this form of cost to instruct the learning of the two cost functions in A^* algorithm, namely $G(\cdot)$ and $H(\cdot)$, which measure the current and future cost respectively.

For $G(\cdot)$ function, an observable trajectory is given as input, *i.e.,* a subsequence of locations from the source location to the candidate location. We aim to compute the likelihood for this observable trajectory as the current cost. For this purpose, we propose to use attention-based RNNs to model the observable trajectory. Compared with heuristic search algorithm [18], our $G(\cdot)$ function has two major advantages. First, we can incorporate useful context information to better capture sequential trajectory behaviors, including spatial-temporal information, personalized preference and road network constraint. Second, instead of simply computing a single cost, our model also learns a time-varying vectorized representation for the moving state of a user, which can be subsequently used in $H(\cdot)$ function.

For $H(\cdot)$ function, the current location, the destination location and the road network are given as input. We aim to estimate the future cost from the current location to the destination location. This task is more challenging than computing the observable cost, since the future trajectory is unobserved. Our solution is based on graph neural networks (GNN) [19], [20], [21]. By effectively learning node representations, we can well capture road network characteristics and make accurate prediction. However, a major problem is that vanilla GNN (*e.g.,* GCN [21] and GAT [20]) is not suitable to directly model road networks. A major reason is that road networks typically have a large graph diameter, where some nodes are distant from each other. In order to increase the receptive field, vanilla GNN needs stack multiple layers, which tends to cause the over-smoothing issue [22]. Besides, existing GNN architectures have limited power in capturing the position information of a given node with respect to all other nodes of the graph. Considering these difficulties, we propose to adopt position-aware GNN [19] for better learning node representations over road networks, where important locations are selected as anchor points and message passing is conducted between anchor points and common points. In this way, our approach can better capture the overall characteristics of road networks by learning more effective node embeddings. We further design two kinds of network architectures based on either geographical distance or user preference. Finally, a reinforcement learning based approach is adopted to estimate the future cost based on the learned node representations.

Combining the above two parts, our approach is able to automatically learn the cost functions without handcrafting heuristics, which is the major contribution of this work. It is able to effectively utilize context information and characterize complex trajectory characteristics, which elegantly combines the merits of A^* search algorithms and deep learning. Extensive results on the three datasets have shown the effectiveness and robustness of the proposed model.

2 RELATED WORK

Our work is related to the following research directions.

Route Recommendation Algorithms. With the availability of user-generated trajectory information, route recommendation has received much attention from the research community [3], [5], [6], [23], [24], which aims to generate reachable paths between the source and destination locations. The task can be defined as either *personalized* [5], [6], [25] or *non-personalized* [8], [10], [23], [26], and constructed based on different types of trajectory data, *e.g.,* GPS data [26], [27] or POI check-in data [28], [29]. In the literature, various algorithms have been developed for route recommendation. Wei et al. [7] utilized graph search algorithms for identifying the path over the road network; Chen et al. [29] proposed probabilistic POI transition/ranking models are employed to recommend probable routes; Yuan et al. [26] proposed to mine diver-direction from the historical GPS trajectories of a large number of taxis. Overall, these studies focus on search algorithms or probabilistic models by considering additional constraints, *e.g.,* road networks or time. Our work is built on top of search based solutions, and the novelty lies in the automatic learning of the cost functions using neural networks. Our model is flexible to incorporate rich context or constraint information.

Trajectory Data Mining with Deep Learning. Recent years have witnessed the success of deep learning in modeling complex data relations or characteristics. As early studies, location/trajectory embedding method are applied to to solve trajectory-related tasks [30]. More recently, Recurrent Neural Network (RNN) together with its variant Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been widely used for modeling sequential trajectory data. Zhou et al. [31] proposed topic-enhanced memory networks for POI recommendation problem; Zhen et al. [32] utilized hierarchical RNN to capture high-level information in trajectories; Gao et al. [33] proposed variational RNN to model the latent variable of trajectories data; Wu et al. [34] took road network constraints into consideration when designing recurrent neural network; Feng et al. [35] proposed a multi-modal embedding RNN with attention mechanism to predict human mobility; Liu et al. [36] proposed spatial-temporal RNN to model spatial-temporal context information; Ai et al. [11] proposed a Space time feature-based RNN to model spatial-temporal information. These studies mainly focus on short-term trajectory behaviors, *e.g.,* one-step location recommendation [36], which are not suitable for solving the current task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

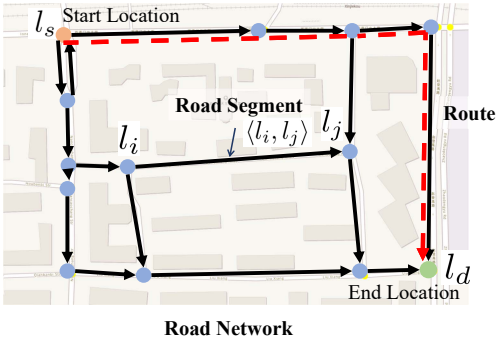


Fig. 1. Illustration of the notations in this work.

Learning-enhanced Heuristic Search. These studies in this direction aim to automatically improve or optimize the search algorithms with machine learning methods. Early works include the use of machine learning in creating effective, likely-admissible or improved heuristics [37], [38], [39]. Lelis et al. [37] proposed to predicts the optimal solution cost of a problem instance without finding the actual solution; Ernandes et al. [38] introduced the related concept of likely-admissible heuristics where admissibility requirement is relaxed in a probabilistic sense; Samadi et al. [39] proposed an artificial neural network to combine several features into a single heuristic value. More recently, deep learning has significantly pushed forward the research of this line. The main idea is to leverage the powerful modeling capacity of neural networks for improving the tasks that require complicated solving strategies, including the Go game [17] and route recommendation [25]. Our work is highly inspired by these pioneering works, but have a quite different focus on the studied task, *i.e.*, personalized route recommendation.

Our current study significantly extends previous work [25] in multiple aspects. First, we enhance the $H(\cdot)$ function (the key of the A^* search algorithm) for estimating the future cost with a totally new position-aware GNN. We design two variants of position-aware GNN to incorporate distance and preference information into the estimation network. Second, we add various kinds of content details, including search algorithms, training details, and parameter settings, and more discussions and explanations throughout the paper.

3 Preliminaries

In our task, we assume road network information is available for the pathfinding task, which is the foundation of urban transportation for users.

Definition 1. **Road Network**. A road network is a directed graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, where $\mathcal{L}$ is a vertex set of locations and $\mathcal{E} \subset \mathcal{L} \times \mathcal{L}$ is an edge set of road segments. A vertex $l_i \in \mathcal{L}$ (*i.e.*, a location) represents a road junction or a road end. An edge $e_{l_i,l_j} = \langle l_i, l_j \rangle \in \mathcal{E}$ represents a directed road segment from vertex l_i to vertex l_j.

Definition 2. **Route**. A route (*a.k.a.*, a path) p is an ordered sequence of locations connecting the source location l_s with the destination location l_d with m intermediate locations, *i.e.*, $p : l_s \rightarrow l_1 \rightarrow ... \rightarrow l_m \rightarrow l_d$, where each pair of consecutive locations $\langle l_i, l_{i+1} \rangle$ corresponds to a road segment $e_{l_i,l_{i+1}}$ in the road network.

The moving trajectory of a user on the road network can be recorded using GPS-enabled devices. Due to instrumental inaccuracies, the sampled trajectory points may not be well aligned with the locations in $\mathcal{L}$. Following [40], we can preform the procedure of *map matching* for aligning trajectory points with locations in $\mathcal{L}$.

Definition 3. **Trajectory**. A trajectory t is a time-ordered sequence of m locations (after map matching) generated by a user, *i.e.*, $t : \langle l_1, b_1 \rangle \rightarrow \langle l_2, b_2 \rangle \rightarrow ... \rightarrow \langle l_m, b_m \rangle$, where b_i is the visit timestamp for location l_i.

A trajectory is a user-generated location sequence with timestamps, while a route is pre-determined by the road network. For a route, the start and end points are important to consider, which correspond to the source and destination of a travel. We present an illustrative example in Fig. 1.

In the task of PRR, a user can issue *travel queries*.

Definition 4. **Query**. A query q is a triple $\langle l_s, l_d, b \rangle$ consisting of source location l_s, destination location l_d and departure time b.

With the above definitions, we now define the studied task.

Definition 5. **Personalized Route Recommendation (PRR)**. Given a dataset $\mathcal{D}$ consisting of historical trajectories, for a query $q : \langle l_s, l_d, b \rangle$ from user $u \in \mathcal{U}$, we would like to infer the most possible route p^* from l_s to l_d made by user u, formally defined as solving the optimal path with the highest conditional probability:

$$p^* = \arg\max_p \Pr(p|q, u, \mathcal{D}). \tag{1}$$

The PRR task is formulated as a conditional ranking problem. For solving this task, we first present a traditional A^*-based algorithm in Section 4, and then present our proposed approach in Section 5.

4 A Heuristic A^* Solution for PRR

The task of PRR can be framed as a graph-based search problem. In this setting, we view the road network as a graph, and study how to find possible route(s) that start from the source node and end at the destination node.

4.1 Review of A^* Algorithm

In the literature [18], A^* search algorithm is widely used in pathfinding and graph traversal due to its performance and accuracy. Starting from a source node of a graph, it aims to find a path to the given destination node resulting in the smallest *cost*. It maintains a tree of paths originating at the source node and extending those paths one edge at a time until its termination criterion is satisfied. At each extension, A^* evaluates a candidate node n based on a *cost function* $F(n)$:

$$F(n) = G(n) + H(n), \tag{2}$$

where $G(n)$ is the cost of the path from the source to n (we call it *observable cost* since the path is observable), and $H(n)$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

is an estimate of the cost required to extend the future path to the goal (we call it *unobserved cost* since the actual optimal path is unknown). The key part of A^* is the setting of the heuristic function $F(\cdot)$, which has an important impact on the final performance.

4.2 A Simple A^*-based Approach for PRR

Considering our task, the goal is to maximize the conditional probability of $\Pr(p|q,u,\mathcal{D})$. We can equally minimize its negative log: $-\log \Pr(p|q,u,\mathcal{D})$. Given a possible path $p: l_s \rightarrow l_1 \rightarrow l_2 \cdots \rightarrow l_m \rightarrow l_d$, consisting of m intermediate locations, we can factorize the path to compute its cost according to the chain rule in probability as

$$-\log \Pr\left(p|q,u,\mathcal{D}\right) = -\sum_{i=0}^{m} \log \Pr\left(l_{i+1}|l_s \rightarrow l_i, q, u\right), \quad (3)$$

where $l_0 = l_s$ and $l_{m+1} = l_d$, and $\mathcal{D}$ is dropped for simplifying notations. Here, we develop the probability conditioned on user u. As will be explained later, we will consider user preference and historical trajectories as context. This formula motivates us to set the cost functions of A^* algorithm in a similar form. Assume a partial route has been generated, *i.e.*, $p: l_s \rightarrow l_1 \cdots \rightarrow l_{i-1}$, we can compute the observable cost of a candidate l_i for extension as

$$G\left(l_s \rightarrow l_i\right) = -\sum_{k=1}^{i-1} \log \Pr\left(l_{k+1}|l_s \rightarrow l_k, q, u\right). \quad (4)$$

To compute the conditional transition probabilities, the first-order Markov assumption is usually adopted, so we have $\Pr(l_{k+1}|l_s \rightarrow l_k, q, u) = \Pr(l_{k+1}|l_k, q, u)$. Following [7], [10], we can further use user-specific or overall transition statistics to estimate the probabilities (with smoothing). While, to compute the cost of $H(l_i \rightarrow l_d)$ is more difficult, since the optimal sub-route from l_i to l_d is unknown. We cannot directly apply the similar method in Eq. (4) for $H(\cdot)$. In practice, we can use different heuristics to set $H(\cdot)$, including the shortest spatial distance [18], [41] and the historical likelihood [7].

4.3 Analysis

For our task, the A^*-based approach is more appealing than a greedy best-first search algorithm. By decomposing the entire cost into two parts, it leaves room on the elaborated setting of $G(\cdot)$ and $H(\cdot)$ for different tasks. Although it has been shown that A^*-like algorithms perform well in the task of route recommendation [7], [41], [42], we see three weak points for improvement.

First, A^* algorithm is a general framework in which cost functions have to be heuristically set. It is difficult to incorporate varying context information, *e.g.*, personalized preference and spatial-temporal influence. Second, the cost function usually relies on the heuristic computation or estimation, which is easy to suffer from data sparsity. For example, the estimation of transition probabilities in Eq. (4) may not be accurate when the historical transitions between two locations are sparse. In this case, even the computation of observable cost $G(\cdot)$ is likely to be problematic. Third, the PRR task is challenging, and a simple heuristic search strategy may not be capable of performing effective pathfinding in practice, *e.g.*, the route that has the shortest spatial distance may not be the final choice of a user [8], [10].

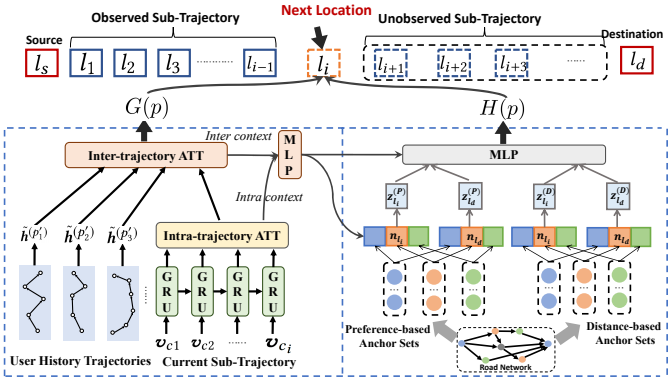


Fig. 2. The overall architecture of the NASR+ model. $G(\cdot)$ learns the cost from the source to a candidate location, called *observable cost*; $H(\cdot)$ predicts the *unobserved cost* from a candidate location to the destination.

With these considerations, we next present our solution for addressing the above difficulties of A^* in PRR.

5 The NASR+ Model

In the section, we present the proposed *Neuralized A-Star based personalized route Recommendation (NASR+)* model[1].

5.1 Model Overview

Our model is developed based on the general A^* algorithm framework. For node evaluation, we decompose the entire cost function $F(\cdot)$ into two parts, namely *observable cost* and *unobserved cost*, which correspond to the cost functions $G(\cdot)$ and $H(\cdot)$. Traditionally, both $G(\cdot)$ and $H(\cdot)$ are heuristically computed or set. While, our idea is to automatically learn the two functions with neural networks instead of using heuristics. Specially, we use Recurrent Neural Networks (RNN) to implement $G(\cdot)$ and another estimation network to implement $H(\cdot)$. In our neural network for function $G(\cdot)$, we not only compute a single cost value, but also learn a time-varying *moving state* for a specific user. The moving state encodes necessary trajectory information of a user till the evaluation time, which will be fed into the computation of $H(\cdot)$. Once the two networks are learned, we can compute the cost of a candidate location for path extension. We present the overall architecture for the proposed model in Fig. 2. We also present the used notations and their explanations in Table 1.

5.2 Modeling the Observable Cost with RNN

This part studies the learning of function $G(\cdot)$ for observable cost. Given an observed sub-route $l_s \rightarrow l_1 \rightarrow l_2 \cdots \rightarrow l_i$, as shown in Eq. (4), the problem becomes how to effectively learn the conditional transition probabilities $\Pr(l_{k+1}|l_s \rightarrow l_k, q, u)$. Simple frequency-based estimation method will suffer from data sparsity in large search space even with first-order Markov assumption. In this case, the computed observable cost will not be reliable to be used. In

1. We use the suffix of the plus symbol in naming our model for discriminating this work from our KDD paper [25].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Notations, explanations, and configurations in this work.

Group	Notation	Explanation	Configuration
Input ($\Theta^{(I)}$)	$\boldsymbol{v}_u \in \mathbb{R}^{K_U}$	The embedding vector for user u.	$K_U = 128$
	$\boldsymbol{v}_l \in \mathbb{R}^{K_L}$	The embedding vector for location l.	$K_L = 512$
	$\boldsymbol{v}_{di(b)} \in \mathbb{R}^{K_D}$	The embedding vector of the weekday index for a timestamp b.	$K_D = 24$
	$\boldsymbol{v}_{hi(b)} \in \mathbb{R}^{K_H}$	The embedding vector of the hour index for a timestamp b.	$K_H = 48$
	$\boldsymbol{v}_{x_i} \in \mathbb{R}^{K_X}$	The context vector concatenated by $\boldsymbol{v}_u$, $\boldsymbol{v}_{l_i}$, $\boldsymbol{v}_{di(b_i)}$, and $\boldsymbol{v}_{hi(b_i)}$.	$K_X = 712$
$G(\cdot)$ ($\Theta^{(G)}$)	$\boldsymbol{h}_i^{(p)} \in \mathbb{R}^{K_R}$	The hidden vector of l_i in trajectory p learned by the GRU network.	$K_R = 512$
	$\tilde{\boldsymbol{h}}^{(p)} \in \mathbb{R}^{K_R}$	The hidden vector of p produced by the intra-trajectory attention.	$K_R = 512$
	$\boldsymbol{h}^{(p)} \in \mathbb{R}^{K_R}$	The hidden vector of p produced by the inter-trajectory attention.	$K_R = 512$
	$\boldsymbol{w}_1$ $\boldsymbol{W}_1$,$\boldsymbol{W}_2$	The attention parameters in Eq. (8).	$\mathbb{R}^{256}$ $\mathbb{R}^{256\times 512}$
$H(\cdot)$ ($\Theta^{(H)}$)	$\boldsymbol{n}_{l_k} \in \mathbb{R}^{K_G}$	The representation of location l_k learned by graph neural network.	$K_G = 512$
	$\boldsymbol{M}_{l_i} \in \mathbb{R}^{K_G \times A}$	The matrix of messages $m_{l_i}^{(a)}$.	$A = 64$
	$\boldsymbol{z}_{l_k}^{(D)} \in \mathbb{R}^{K_P}$	The representation of location l_k learned by Distance-based PA-GNN.	$K_P = 64$
	$\boldsymbol{z}_{l_k}^{(P)} \in \mathbb{R}^{K_P}$	The representation of location l_k learned by Preference-based PA-GNN.	$K_P = 64$
	$\boldsymbol{w}_{out}$ $\boldsymbol{W}_3$ $\boldsymbol{W}_4$	The learnable parameters in our PA-GNNs.	$\mathbb{R}^{512}$ $\mathbb{R}^{1024\times 512}$ $\mathbb{R}^{1536\times 512}$
	A	The number of anchor-sets.	64

addition, PRR is a user-centric task, and a single cost value may not be enough to describe what has been observed. Instead, the *moving state* of a user and useful context information should be considered. To address these difficulties, we propose to use Recurrent Neural Networks to implement the cost function $G(\cdot)$.

5.2.1 Embedding Rich Context Information

As the prerequisite module, we embed rich context information into dense vectors, which will be subsequently used by other components. First, we set up an embedding vector $\boldsymbol{v}_u \in \mathbb{R}^{K_U}$ for user u, encoding necessary personalized user information. Then, for each location $l \in \mathcal{L}$, we set up a corresponding embedding vector $\boldsymbol{v}_l \in \mathbb{R}^{K_L}$. For trajectory behaviors, temporal information is also important to consider. Following [35], for each visit timestamp b, we use two embedding vectors $\boldsymbol{v}_{di(b)} \in \mathbb{R}^{K_D}$ and $\boldsymbol{v}_{hi(b)} \in \mathbb{R}^{K_H}$, where $di(b)$ and $hi(b)$ are functions transforming b into corresponding weekday index (1 to 7) and hour index (1 to 24) respectively. At i-th time step, we concatenate the above embedding vectors of user u's location l_i into a single embedding vector to form an enhanced representation of contextual information as

$$\boldsymbol{v}_{x_i} = \boldsymbol{v}_u \| \boldsymbol{v}_{l_i} \| \boldsymbol{v}_{di(b_i)} \| \boldsymbol{v}_{hi(b_i)}, \tag{5}$$

where "$\|$" is the concatenation operation. Here, the representation $\boldsymbol{v}_{x_i}$ contains contextual information of user ID (user preference), location ID (location characteristics) and temporal information (periodical patterns). It will be flexible to incorporate more kinds of context features. We can also enhance the context representations with more complicated networks, *e.g.*, multi-layer perceptron.

5.2.2 Encoding the Observed Sub-Trajectory with RNN

For the PRR task, it is important to model the trajectory characteristics of users' moving behaviors, which can be considered as a sequential process. We utilize RNNs to model such sequential behaviors. Given an observed sub-trajectory $p : l_s \to l_1 \cdots \to l_i$ generated by u, we employ the widely used GRU network [43] to encode it into a vector

$$\boldsymbol{h}_i^{(p)} = \text{GRU}(\boldsymbol{v}_{x_i}, \boldsymbol{h}_{i-1}^{(p)}), \tag{6}$$

where $\boldsymbol{h}_i^{(p)} \in \mathbb{R}^{K_R}$ is the hidden vector produced by the GRU network and $\boldsymbol{v}_{x_i}$ is the context vector defined in Eq. (5), where $\boldsymbol{v}_{x_i}$ is taken as input to incorporate context information at each step. The vector $\boldsymbol{h}_i^{(p)}$ encodes the *moving state* of a user at the i-th time step. Note that we use the superscript to index the trajectory and the subscript to index locations. Unlike traditional A^* search algorithm, we learn an informative state representation of a user at each step, providing more information than a single cost value. The learned moving state will be subsequently used by the $H(\cdot)$ function.

5.2.3 Enhanced Moving States with Attention Mechanism

An observed sub-trajectory can be short and noisy. We further propose to use two types of attention to improve the learning of moving state by leveraging data dependence.

Intra-Trajectory Attention. We first apply the method [44] to compute the attention between locations in the same trajectory as

$$\tilde{\boldsymbol{h}}_i^{(p)} = \sum_{k=1}^{i} \text{att}\left(\boldsymbol{h}_i^{(p)}, \boldsymbol{h}_k^{(p)}\right) \cdot \boldsymbol{h}_k^{(p)}, \tag{7}$$

where $\tilde{\boldsymbol{h}}_i^{(p)}$ denotes the improved state representation with intra-trajectory attention and $\text{att}(\cdot,\cdot)$ is an attention function in the form of

$$\begin{aligned} \text{att}\left(\boldsymbol{h}_i^{(p)}, \boldsymbol{h}_k^{(p)}\right) &= \frac{\exp(\alpha_{i,k})}{\sum_{k'=1}^{i} \exp(\alpha_{i,k'})}, \\ \alpha_{i,k} &= \boldsymbol{w}_1^\top \cdot \tanh\left(\boldsymbol{W}_1 \cdot \boldsymbol{h}_k^{(p)} + \boldsymbol{W}_2 \cdot \boldsymbol{h}_i^{(p)}\right), \end{aligned} \tag{8}$$

where $\boldsymbol{w}_1$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the parameter vector or matrices to learn. With intra-trajectory attention, we can discover more important characteristics by considering the entire trajectory. After intra-trajectory attention, we use the state representation of the last location for encoding the entire sub-trajectory, *i.e.*, $\tilde{\boldsymbol{h}}^{(p)} = \tilde{\boldsymbol{h}}_i^{(p)}$.

Inter-Trajectory Attention. The information from a single trajectory is usually limited. In order to capture overall moving patterns for a specific user, we further consider incorporating historical trajectories generated by the user. Given the current trajectory p, we attend it to each of the other historical trajectories, denoted as p', as

$$\boldsymbol{h}_i^{(p)} = \sum_{p' \in \mathcal{P}^u} \text{att}\left(\tilde{\boldsymbol{h}}_i^{(p)}, \tilde{\boldsymbol{h}}^{(p')}\right) \cdot \tilde{\boldsymbol{h}}^{(p')}, \tag{9}$$

where $\mathcal{P}^u$ denotes the set of historical trajectories generated by u. The vector $\tilde{\boldsymbol{h}}^{(p')}$ is an representation vector of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

historical trajectory p', which is generated by an attention in the form of

$$\tilde{\boldsymbol{h}}^{(p')} = \sum_{k=1}^{m} \text{att}\left(\boldsymbol{h}_m^{(p')}, \boldsymbol{h}_k^{(p')}\right) \cdot \boldsymbol{h}_k^{(p')}. \tag{10}$$

$\text{att}(\cdot,\cdot)$ is the attention function which is similar to Eq. (8) but with different learnable parameters.

5.2.4 Observable Cost Computation

Once we have learned the hidden state for the current timestamp, we are able to compute the probability of the next location using a softmax function with road network constraint as

$$\Pr(l_i|l_s \to l_{i-1}, q, u) = \frac{\exp\left(\text{LT}(p^{l_s \to l_i})\right)}{\sum_{l' \in \mathcal{L}_{l_{i-1}}} \exp\left(\text{LT}(p^{l_s \to l'})\right)}, \tag{11}$$

where $\text{LT}(p) = \boldsymbol{w}_2^\top \cdot \boldsymbol{h}^{(p)}$ is a linear transformation function taking as input the hidden state learned for a trajectory p in Eq. (9), and $\boldsymbol{w}_2$ is a parameter vector to learn. Here, we compute the probability of a candidate location l_i by normalizing over all the neighboring locations of l_{i-1} in the road network. After defining $\Pr(l_i|l_s \to l_{i-1}, q, u)$ in Eq. (11), we sum the negative log probability of each location in a trajectory $l_s \to l_i$ as the value of $G(\cdot)$

$$G(l_s \to l_i) = -\sum_{j=2}^{i} \log \Pr\left(l_j|l_s \to l_{j-1}, q, u\right). \tag{12}$$

Note that we do not set $G(l_s \to l_i)$ using simple distance functions, since we would like to learn more useful information from the observed trajectories. Typically, a user would select a route based on many factors. Our computation form for $G(\cdot)$ naturally fits the defined goal of our task in Eq. (1). To learn the neural network component, given $G(\cdot)$ in Eq. (12), we set a loss for all the observed trajectories over all users as

$$Loss_1 = \sum_{u \in \mathcal{U}} \sum_{p \in \mathcal{P}^u} G(p). \tag{13}$$

5.3 Modeling the Unobserved Cost with Estimation Networks

Besides the observable cost, we need to estimate the cost from a candidate location to the destination. Specially, we introduce an estimation network to implement $H(\cdot)$. This part is more difficult to model since no explicit trajectory information is observed. In order to better utilize the road network information and user preference for estimation, we build the estimation network on top of a position-aware graph neural network.

5.3.1 PG-GNN for Modeling Road Networks

We consider using graph neural networks for learning effective node representations for capturing graph characteristics. Generally, the update of graph neural networks [20] can be given as

$$\boldsymbol{N}^{(z+1)} = \text{GNN}\left(\boldsymbol{N}^{(z)}\right) \tag{14}$$

where $\boldsymbol{N}^{(z)} \in \mathbb{R}^{K_G \times |\mathcal{L}|}$ denotes the matrix consisting of node representations at the z-th iteration, and the l_k-th column $\boldsymbol{n}_{l_k} \in \mathbb{R}^{K_G}$ corresponds to the representation of node l_k, *i.e.*, location $l_k \in \mathcal{L}$. For initialization, we set $\boldsymbol{n}_{l_k}^{(0)} = \boldsymbol{v}_{l_k}$ with the learned location embeddings in Sec. 5.2.1.

However, vanilla GNN (*e.g.*, GCN [21] and GAT [20]) are not directly suitable to model road networks. With the same scale of node number, road networks usually have a larger graph diameter than "*small-world* networks" [45], which is likely to cause the over-smoothing issue for multi-layered GNNs. Inspired by recent work [19], our idea is to select a small number of anchor points from the road network, and the message passing in GNNs is conducted between anchor points and locations. Such a kind of GNNs is called position-aware graph neural networks (PA-GNNs). Overall, the general PA-GNN contains two major parts:

Construction of Anchor Set. First, we introduce the concept of anchor set. In order to generate position-aware node embeddings, the perquisite step is to select several representative nodes as *anchors*. By calculating the relations between anchor sets and target nodes, we could map target locations into an embedding space by considering relative position information *w.r.t.* anchor points. Following [19], we generate $A = \lambda \times \lfloor \log |\mathcal{L}| \rfloor^2$ anchor-sets from using some selection algorithm, denoted as $\mathcal{S}_a \subset \mathcal{L}$, where $a = 1, ..., \lambda \times \lfloor \log |\mathcal{L}| \rfloor^2$ and λ is a hyper-parameter. Following [19], we select the centric point from an anchor set as the representative anchor point in this set.

Learning Position-Aware Node Embeddings. PA-GNNs consist of three learning steps to generate position-aware embedding for the node l_i. In the first step, it constructs a message passing function $\mathfrak{F}(\cdot,\cdot)$ to describe the position relationship between l_i and a given location l':

$$\mathfrak{F}\left(\boldsymbol{n}_{l_i}, \boldsymbol{n}_{l'}\right) = \text{Dist}\left(l_i, l'\right) \cdot \left(\boldsymbol{n}_{l_i} \| \boldsymbol{n}_{l'}\right) \cdot \boldsymbol{W}_3, \tag{15}$$

where $\boldsymbol{W}_3 \in \mathbb{R}^{2K_G \times K_G}$ is a learnable parameter matrix and $\text{Dist}(l_i, l')$ is the distance function that can be implemented in different ways. In the second step, the PA-GNN uses the message passing function to generate representations of l_i for each anchor-set. The representation vector for the anchor-set $\mathcal{S}_a$, the representation vector, denoted as $\boldsymbol{m}_{l_i}^{(a)}$, is calculated as

$$\boldsymbol{m}_{l_i}^{(a)} \leftarrow \text{AP}\left(\{\mathfrak{F}(\boldsymbol{n}_{l_i}, \boldsymbol{n}_{l'}) \mid \forall l' \in \mathcal{S}_a\}\right), \tag{16}$$

where $\text{AP}(\cdot)$ is an average pooling function to combine output vectors of $\mathfrak{F}(\boldsymbol{n}_{l_i}, \boldsymbol{n}_{l'})$ for all $l' \in \mathcal{S}_a$ as an single representation vector. In the third step, PA-GNN further combines $\boldsymbol{m}_{l_i}^{(a)}$ of all anchor-sets using average pooling as

$$\boldsymbol{n}_{l_i} \leftarrow \text{AP}(\{\boldsymbol{m}_{l_i}^{(a)} \mid a = 1, \ldots, A\}). \tag{17}$$

Plugging $\boldsymbol{n}_{l_i}$ generated by Eq. (17) into Eq. (14), position information expressed between l_i and the anchor points can be involved in the update process of GNN.

Finally, the position-aware node embedding of l_i is calculated as

$$\boldsymbol{z}_{l_i} = \sigma\left(\boldsymbol{M}_{l_i}^\top \cdot \boldsymbol{w}_{out}\right), \tag{18}$$

where $\boldsymbol{M}_{l_i} \in \mathbb{R}^{K_G \times A}$ is a matrix that consists of $\boldsymbol{m}_{l_i}^{(a)}$ (Eq. 16) as columns, $\boldsymbol{w}_{out} \in \mathbb{R}^{K_G}$ is a weight vector, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

$\sigma(\cdot)$ is a non-linear mapping function. In this way, the k-th entry in $\boldsymbol{z}_{l_i}$ measures the importance of the k-th anchor-set for l_i.

5.3.2 Instantiation of PA-GNN for PRR

In our framework, we have two major parts to set, namely the anchor-sets and the $\mathrm{Dist}()$ function. Here, we present two variants for instantiating the above framework, namely *distance-based PA-GNN* and *preference-based PA-GNN*. The distance-based PA-GNN sets the two parts with distance information, and the preference-based PA-GNN utilizes user preference information to enhance the performance.

Distance-based PA-GNN. In road networks, there are many locations there are no trajectory passing through them. This problem may cause these locations are very hard be selected in route search. In order to alleviate this problem, we designed as pure distance-based PA-GNN variant. In this variant, we apply the K-means algorithm to derive anchor-sets. Specifically, the variant takes the latitude and longitude as features of locations on a road network, and uses a standard K-means algorithm to cluster the locations as a number of sets, which are used as anchor-sets of the PA-GNN. It has been shown that K-means algorithm is suitable to deal with distance-based metrics [46]. In this way, the spatial distance information is incorporated into the anchor-sets. Then, we run the Dijkstra algorithm to derive the shortest distance between an anchor location l' and the candidate location l_i. The shortest distance on the road network is used to set $\mathrm{Dist}(l_i, l')$. In this way, the distance and spatial structure information of the road network is incorporated into the PA-GNN. The $\mathrm{Dist}()$ function of the distance-based PA-GNN does not reply on user preference information (historical trajectories), so it improves the robustness of our model over the locations that are seldom visited.

Preference-based PA-GNN. In the second variant, we incorporate user preference information into anchor-sets and the distance function. For constructing the anchor-sets, instead of directly clustering locations, we cluster trajectory points of users in the training set. The locations that correspond to points in the same cluster constitute an anchor-set. Note that different from the above distance-based method, a location can appear multiple times in the clustering algorithm here. In this way, a frequently visited location can receive more attention in the clustering algorithm. We adopt DBSCAN [47] as the clustering algorithm since it is able to consider the density of data points to reflect the collective preference of the crowd, and therefore is suitable for generating preference-based anchor-sets. Next, we study how to incorporate personalized preference into the distance function. Recall that we can derive a moving state $\boldsymbol{h}^{(p)}$ defined in Eq. (9) summarizing the behavioural characteristics of a user for an observed sub-trajectory p. In order to incorporate $\boldsymbol{h}^{(p)}$ into the distance, we implement the $\mathrm{Dist}()$ as a function of $\boldsymbol{n}_{l_i}, \boldsymbol{n}_{l'}, \boldsymbol{h}^{(p)}$ as

$$\mathrm{Dist}(l_i, l') = \sigma\left(\boldsymbol{W}_4^\top \cdot (\boldsymbol{n}_{l_i} \| \boldsymbol{n}_{l'} \| \boldsymbol{h}^{(p)})\right), \tag{19}$$

where $\boldsymbol{W}_4$ is a learnable parameter matrix and $\sigma()$ is a non-linear mapping function. The moving state $\boldsymbol{h}^{(p)}$ used in Eq. (19) denotes the embedding of *observed trajectory* for the start location to current search location, *i.e.*, $l_s \rightarrow l_{i-1}$. As shown in Eq. (5), the moving state $\boldsymbol{h}^{(p)}$ also encodes the user and context information, so that it can reflect the user preference to some extent. Given the moving state, the personal preference information has been modeled to compute the distance between two locations. Moreover, the embedding $\boldsymbol{h}^{(p)}$ also plays an important role in providing useful information through G network to H network, while the two functions are usually isolated in previous studies.

Note that Eq. (19) can be also used in the distance-based variant. However, for the distance-based approach, we would like to develop a simple solution with a pure distance measure (*i.e.*, the shortest path on the graph) to fully exploit the spatial information of the road network. It helps alleviate the data sparsity problem for seldom visited locations. If we introduce the user personalized information into distance-based variant, the utility of spatial information might be diluted.

According to the two variants, we can obtain two different node representations for a location by Eq. (18). We use $\boldsymbol{z}_{l_i}^{(D)}$ and $\boldsymbol{z}_{l_i}^{(P)}$ to denote the representations learned according to the distance-based and preference-based PA-GNNs, respectively.

5.3.3 Unobservable Cost Estimation

After obtaining the node representations, we use a Multi-Layer Perceptron component to infer the cost from the candidate location l_i to the destination l_d. Formally, we have

$$H\left(l_i \rightarrow l_d\right) = \mathrm{MLP}\left(\boldsymbol{h}_i^{(p)}, \boldsymbol{z}_{l_i}^{(D)}, \boldsymbol{z}_{l_d}^{(D)}, \boldsymbol{z}_{l_i}^{(P)}, \boldsymbol{z}_{l_d}^{(P)}\right), \tag{20}$$

where the MLP component takes as input the moving state $\boldsymbol{h}^{(p)}$ and the two kinds of representations of the current and destination locations l_i and l_d. In Eq. (20), the vectors $\boldsymbol{z}_{l_d}^{(D)}$, $\boldsymbol{z}_{l_i}^{(D)}$ respectively are representations of l_i and l_d learned by the distance-based PA-GNN, and $\boldsymbol{z}_{l_d}^{(P)}$, $\boldsymbol{z}_{l_i}^{(P)}$ are learned by the preference-based PA-GNN. We expect the two kinds of representations can capture road network characteristics in different views, so that we can obtain more comprehensive information for making the estimation.

Next, we study how to define the loss of the entire estimation network. The learning of $H(l_i \rightarrow l_d)$ relies on the optimal sub-route from l_i to l_d. Our task aims to minimize the future cost. When a location l_i is selected, an immediate cost c_i will be yielded according to

$$c_i = -\log \Pr\left(l_i | l_s \rightarrow l_{i-1}, q, u\right), \tag{21}$$

where $\Pr(l_i | l_1 \rightarrow l_{i-1}, q, u)$ is the probability computed in Eq. (11). In our model, the groundtruth cost is equal to the step cost from the current location to the destination: $H(l_i \rightarrow l_d) = \sum_{j=i+1}^{T} c_j$, where T is the timestamp arriving at l_d. If we have the real route from the current location l_i to destination l_d, the prediction error of the estimation network is calculated as

$$\delta_{l_i \rightarrow l_d} \quad = \quad \left\| H(l_i \rightarrow l_d) - \sum_{j=i+1}^{T} c_j \right\|^2, \tag{22}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1 The training algorithm for the NASR+ model.

Input: A trajectory dataset $\mathcal{D}$.
Output: Model parameters $\Theta^{(G)}$,$\Theta^{(H)}$, and $\Theta^{(I)}$.
Randomly initialize $\Theta^{(G)}$, $\Theta^{(H)}$ and $\Theta^{(I)}$.
Pre-learn the $\Theta^{(I)}$ and $\Theta^{(G)}$ by minimizing the $Loss_1$ in Eq. (13).
for *episode* $= 1$ to $|\mathcal{D}|$ **do**
  Acquire a random trajectory t from $\mathcal{D}$.
  Perform SGD on Eq. (13) *w.r.t.* $\Theta^{(G)}$ and $\Theta^{(I)}$.
  for $i = length(t)$ to 0 **do**
    Acquire location $l_{i+1}, ..., l_d$ and corresponding time from trajectory t.
    Acquire observed cost $c_{i+1}, ..., c_d$ using Eq. (21).
    Compute $y_{l_i \to l_d} = \sum_{k=i+1}^{d} c_k$.
    Perform SGD on $\left\| H(l_i \to l_d; \Theta^{(H)}) - y_{l_i \to l_d} \right\|^2$ (Eq. 22) *w.r.t.* $\Theta^{(H)}$ and $\Theta^{(I)}$.
  end for
end for
return $\Theta^{(G)}$,$\Theta^{(H)}$, $\Theta^{(I)}$.

To learn the estimation network component, we set a loss for all the observed trajectories over all users as

$$Loss_2 = \sum_{u \in \mathcal{U}} \sum_{p \in \mathcal{P}^u} \sum_{l_i \in p} \delta_{l_i \to l_d}. \tag{23}$$

5.4 Learning and Analysis

In this part, we present the learning algorithm for optimizing our approach, and given detailed analysis on model merits and complexities.

5.4.1 Model Learning

It is not easy to directly optimize the entire approach, which contains two different components for G and H networks. Here, we follow the training practice in deep learning [48] to pre-learn the two networks separately. For G network, we directly learn the RNN component according to Eq. (13). To pre-learn H network, we first fix the parameters in G network, and utilize it to learn the moving state $\boldsymbol{h}^{(p)}$. Furthermore, in training set, we can utilize the real optimal path (*i.e.,* the path adopted by a user) to derive the ground-truth future cost. In this way, the loss $Loss_2$ in Eq. (23) can be directly optimized with ground-truth costs.

Next, we alternatively optimize the parameters in $\Theta^{(G)}$ and $\Theta^{(H)}$. At each iteration, we first update $\Theta^{(G)}$ and $\Theta^{(I)}$. Then, we take a supervised learning way. We construct labels $y_{l_i \to l_d}$ by acquiring observed cost $c_{i+1}, ..., c_d$ using Eq. (21). When computing the observed cost, we use the RNN component with current parameters of $\Theta^{(G)}$, and $\Theta^{(I)}$. After obtaining the loss $\|H(l_i \to l_d; \Theta^{(H)}) - \sum_{j=i+1}^{d} c_j\|^2$, we perform stochastic gradient descent (SGD) to update the parameters $\Theta^{(H)}$ and $\Theta^{(I)}$. The above iteration process is repeated over all the trajectories in training dataset.

5.4.2 Model Analysis

Compared with traditional heuristic search algorithms, NASR+ has the following merits. First, it does not require to manually set functions with heuristics, but automatically learns the functions from data. Second, it can utilize various kinds of context information and capture more complicated personalized trajectory characteristics. Third, it is able to coordinate and integrate the two components by sharing useful information or parameters in a principled way. Note that traditional search algorithms neglect the importance of $G(\cdot)$, which computes the cost of observed sub-trajectories. In our model, the implementation of $G(\cdot)$ not only learns the cost but also a vectorized user state representation, *i.e.,* the moving state of a user. This state vector is subsequently used for the learning of preference-based PA-GNN by providing useful preference information for current user. Besides, as we discussed in Sec. 4, not all the observable cost can be directly computed, usually requiring estimation or approximation. Neural networks are helpful to improve the computation of $G(\cdot)$ by producing more robust results.

In some cases, there exist some unreachable route requests from users. Hence, it needs to determine whether a request can be satisfied as early as possible. Since our algorithm is extended by A^* search algorithm, it is able to judge that the request is unreachable when the open set is empty when searching a path towards the destination. The reachability verification can be further accelerated with other efficient graph search algorithms [49], [50].

Now we compare the time complexities between our GNN and vanilla GNN. For GNNs, an important, common measure in time complexity is the number of message passing. In our approach, at each iteration, each node communicates with A anchor points, so that the total number of message passing is $A \times |\mathcal{L}|$, where $|\mathcal{L}|$ is the number of locations on a road network. As a comparison, vanilla GNN (*e.g.,* GCN [21]) requires a total number of $|\mathcal{E}|$ message passings at each iteration, which can be rewritten as $\bar{D} \times |\mathcal{L}|$, where $\bar{D}$ is the average degree of nodes in the road network. In our work, we follow [19] to set $A = \log^2 |\mathcal{L}|$, which is around several hundreds for $|\mathcal{L}| \approx 10000$. To reduce the number of message passing, we also remove weak links between anchor points and locations, so that the overall complexity can be in a reasonable range in practice.

6 EXPERIMENTS

In this section, we first set up the experiments, and then present the performance comparison and result analysis.

6.1 Experimental Setup

6.1.1 Construction of the Datasets

To evaluate the performance of our proposed model, we use three real-world trajectory datasets. The *Beijing taxi* dataset is collected by more than 18,000 taxis in Beijing, China from Nov. 1, 2011 to Nov. 30, 2011. A trajectory record takes the form of $\langle tid, te, longitude, latitude, state \rangle$, where tid is the unique ID of a taxi, and $state$ informs whether the taxi is carrying any passengers at time te. The state information can be used to identify boundary marks for trajectories, *i.e.,* "*No passengers*" indicates the stop of a travel. The *Porto taxi* dataset is a public trajectory dataset, released by a Kaggle trajectory prediction competition [2]. The dataset contains a complete year (from July 1, 2013 to June 30, 2014) of the trajectories for all the 442 taxis running in the city of Porto, in Portugal. The dataset organizes one ride of a taxi as a trajectory. In both the Beijing taxi and Porto taxi

2. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
Statistics of the three datasets after preprocessing.

Statistics	Beijing taxi	Porto taxi	Beijing bicycle
#users	18,298	442	196,591
#trajectories	302,654	284,100	484,421
#records	16,040,662	8,523,000	6,442,890
#locations	15,208	8,224	15,500
#road segments	20,198	9,457	22,010

datasets, we use taxi IDs to group trajectories, and treat the corresponding driver as the studied user. The *Beijing bicycle* dataset is collected by the *ofo* bicycle sharing company [3] from July 1, 2017 to July 31, 2017. The dataset organizes one ride of a user as a trajectory, which contains 484,421 trajectories from 196,591 users. We can identify a user with a unique user ID in trajectory records. The *Beijing taxi* trajectory data is sampled every minute, while the *Beijing bicycle* dataset is sampled every 10 seconds. The *Porto taxi* dataset is with a sampling period of 15 seconds. For the three datasets, we collect corresponding road network information from *Open Street Map* [4].

After obtaining the datasets, we adopt the open source tool *fmm* [5] to match sampled trajectory points with locations in the road network. In this way, the sequence of sampled trajectory points are converted into a time-ordered location sequence. Since the original sampling frequency is high, we remove consecutive points on the same road segment and replace the subsequence with the start and end points of a road segment. In this way, we transform the trajectory data into timestamped location sequences. We present the statistics of the three datasets after preprocessing in Table 2.

6.1.2 Evaluation Metrics

For the PRR task, we adopt a variety of evaluation metrics widely used in previous works [5], [51]. Given an actual route p, we predict a possible route p' with the same source and destination. Following [5], [51], we use *Precision*, *Recall* and *F1-score* as evaluation metrics: $Precision = \frac{|p \cap p'|}{|p'|}$, $Recall = \frac{|p \cap p'|}{|p|}$ and $F1 = \frac{2*P*R}{P+R}$. *Precision* and *Recall* compute the ratios of overlapping locations *w.r.t.* the actual and predicted routes respectively. Besides, we use the *Edit distance* as a fourth measure, which is the minimum number of edit operations required to transform the predicted route into the actual route. Note the source and destination locations are excluded in computing evaluation metrics.

6.1.3 Task and Experiment Setting

For each user, we divide her/his trajectories into three parts with a ratio of 7 : 1 : 2, namely training set, validation set and test set. We learn the model with training set, and optimize the model with validation set. Instead of reporting the overall performance on all test trajectories, we generate three types of queries *w.r.t.* the number of locations in the trajectories, namely *short* (10 to 20 locations), *medium* (20 to 30 locations) and *long* (more than 30 locations). In test set, given a trajectory, the first and last locations are treated as the source and destination respectively, and the rest locations are hidden. Each method is required to recover the missing route between the source and destination.

3. https://www.ofo.com/
4. https://www.openstreetmap.org/
5. https://www.github.com/cyang-kth/fmm

6.1.4 Methods to Compare

We consider the following methods for comparison:

• *RICK* [7]: It builds a routable graph from uncertain trajectories, and then answers a user's online query (a sequence of point locations) by searching top-k routes on the graph.

• *MPR* [10]: It discovers the most popular route from a transfer network based on the popularity indicators in a breadth-first manner.

• *CTRR* [5]: It proposes a collaborative travel route recommendation algorithm by considering a user's personal travel preference. A road network graph is weighted based on the log-inversed travel behaviour probability that is transformed from the user historical trajectories. Finally, the least weighted route is discovered with Dijkstra's algorithm.

• *STRNN* [36]: Based on RNNs, it models local temporal and spatial contexts in each layer with transition matrices for different time intervals and geographical distances. The original STRNN is used for next-location prediction. We adapt it to our task in an auto-regressive way, *i.e.*, a predicted location at the current step will be fed into the network to produce the location at the next step. If a predicted location is the destination, the algorithm will be ended. Otherwise we will end it if the predicted route is 50% longer than the real route.

• *DeepMove (DM)* [35]: It is a multi-modal embedding RNN that can capture the complicated sequential transitions by jointly embedding the multiple factors that govern the human mobility. To adapt it to route recommendation, we follow the similar method of STRNN to produce auto-regressive prediction. A major advantage of DeepMove is that it can leverage various kinds of context features in the prediction model. For a fair comparison, we adopt the same kinds of context features as in our approach (Eq. (5)).

• *NASR* [25]: it is a base version of our model, which shares the same component for the $G(\cdot)$ function but adopts a simpler implementation of the $H(\cdot)$ function without considering position-aware node embeddings.

• *NASR+*: it is the current model.

Among these baselines, RICK and MPR are heuristic search based methods, CTRR is a machine learning method, and STRNN and DeepMove are deep learning methods. The parameters in all the models have been optimized using the validation set.

To reproduce our results, Table 1 lists the parameters and their configurations in our model. We organize the notations in three groups, namely the input, $G(\cdot)$ and $H(\cdot)$. The last column reports the parameter configurations that are able to reproduce the experimental results in our paper. We also set up a GitHub project to share all the code for reproducibility at the link: https://github.com/bigscity/NASR.

In the experiment, the parameters in GRU, GNN and MLP in our model are initialized by a truncated normal distribution with zero mean and 0.01 variance, and the biases are initialized as zeros. We use the Adaptive Moment Estimation (Adam) optimizer to train with a learning rate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

of 0.0001. The batch size is set as 100 for the pre-training of $G(\cdot)$, and is set as 50 for the joint training of $G(\cdot)$ and $H(\cdot)$. The epoch number for the the pre-training is set as 50 and for the joint training is set as 20 until convergence. To avoid overfitting, we apply dropout in both GRU and fully connected layers, and the dropout rate is 0.5.

6.2 Result and Analysis

We present the results of all the comparison methods in Table 3. From the table we can observe followings.

First, heuristic search methods, *i.e.*, RICK and MPR, perform well, especially the RICK method. RICK fully characterizes the road network information and adopts the informed A^* algorithm. This result verified the effectiveness of the A^* algorithm in the PRR task. As a comparison, MPR mainly considers the modeling of transfer network and uses a relatively simple BFS search procedure.

Second, the matrix factorization based method CTRR does not perform better than RICK and MPR. A possible reason is that CTRR can not well utilize the road network information. Besides, it has limited capacities in learning complicated trajectory characteristics. In our experiments, CTRR tends to generate short route recommendations, giving very bad recall results for medium and long queries. Third, the deep learning method DeepMove performs very well among all the baselines, while STRNN gives a worse performance. Compared with STRNN, DeepMove considers more kinds of context information and designs more advanced sequential neural networks.

Finally, the model NASR and its improved version NASR+ are consistently better than all the baselines in all cases, yielding very good performance even on long queries. Comparing the two versions of our model, we can see NASR+ further produces substantial improvement over the base model NASR. The major difference between the two versions of our model lies in the fact that NASR+ is able to learn distance and preference information using the position-aware node embeddings. For the PRR task, the positional and personalized information of locations are particularly important to consider. The proposed PA-GNN can learn such characteristics by setting up anchor sets and measuring the relations between anchor locations and target locations from the positional and personalized perspective.

By summarizing these results, we can see heuristic search methods are competitive to solve the PRR task, especially when suitable heuristics are used and context information is utilized. Besides, deep learning is also able to improve the performance by leveraging the powerful modeling capacity. Our proposed models are able to combine both the benefits of heuristic search and neural networks, and hence it performs best among the comparison methods.

6.3 Detailed Model Analysis

In this section, we perform a series of detailed analysis on NASR+ for further verifying its effectiveness. Due to space limit, we only report the results of F1 scores on the *Beijing taxi* dataset. The rest results show the similar findings, and are omitted here.

6.3.1 The Effect of the RNN Component

We first examine the effect of the RNN component with different variants. We have incorporated two kinds of attentions, namely inter- and intra-trajectory attention in Sec. 5.2. Here, we consider three variants of the attention mechanism for implementing $g(\cdot)$: *without attention (NA)*, *using only intra-trajectory attention (IA)* and *using both intra- and inter-trajectory attention (BA)*. Recall our RNN component is also able to learn a vectorized representation for the moving state of users. We further prepare a variant for verifying the effect of the learned moving state in the estimation network, namely the model that does not provide the moving state to the $H(\cdot)$ function, denoted by ($BA_{\neg S}$). In Fig. 3(a), it can be seen that the performance rank is as follows: $NA < IA < BA$ and $BA_{\neg S} < BA$. It shows that both inter- and intra-trajectory attention are important to improve the performance of the PRR task. Especially, the learned moving state from the RNN component is useful for the estimation network. When the moving state is incorporated, the performance of the joint model has been substantially improved.

6.3.2 The Effect of the Estimation Network

Predicting the future cost (*i.e.*, $H(\cdot)$) of a candidate location is especially important for our task. We use an estimation network for implementing $H(\cdot)$, which replaces the traditional heuristics. We now examine the performance of different variants for the estimation network. In this part, we fix the RNN component as its optimal setting. Then we prepare four variants for the estimation network as comparisons, including (1) *ED* using Euclid distance as heuristics, (2) *SP* using the scalar product between the embeddings of the candidate and destination locations, (3) *NASR* using the base version of our model which does not consider positional information nodes, and (4) *NASR+* using our complete model. In Fig. 3(b), it can be observed that the performance rank is as follows: $ED < SP < NASR < NASR+$. We can see that the simplest spatial distance baseline *ED* gives the worst performance, which indicates simple heuristics may not work well in our task. Position-aware graph neural networks are more effective to capture geographical and personalized characteristics from graphs. When incorporating these information, our estimation network is able to outperform the base version.

6.3.3 The Effect of the Estimation Network on Search Space

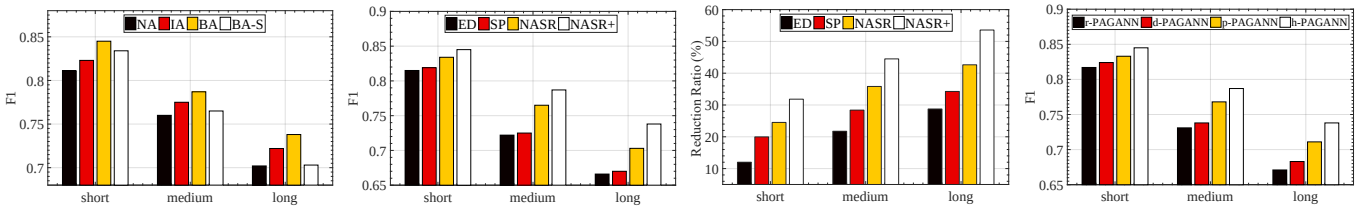
Search space is an importance metric to evaluate the effect of heuristics. We now examine the performance of different variants for the estimation network on the reduction of search space. We measure the reduced ratio yielded by a search algorithm over the original search space. In this part, we fix the RNN component as its optimal setting. Then we prepare four variants for the estimation network as comparisons, including (1) *ED* using Euclid distance as heuristics, (2) *SP* using the scalar product between the embeddings of the candidate and destination locations, (3) *NASR* using the base version of our model which does not consider positional information nodes, and (4) *NASR+* using our complete model. Overall, Fig. 3(c) shows similar

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
Performance comparison using four metrics on three datasets. All the results are better with larger values except the EDT measure. With paired t-test, the improvement of the NASR over all the baselines is significant at the level of 0.01.

Datasets	Metric	Precision							Recall						
	Length	RICK	MPR	CTRR	STRNN	DM	NASR	NASR+	RICK	MPR	CTRR	STRNN	DM	NASR	NASR+
Beijing Taxi	Short	0.712	0.347	0.558	0.491	0.742	0.821	**0.834**	0.723	0.372	0.164	0.384	0.756	0.848	**0.856**
	Medium	0.638	0.253	0.276	0.446	0.642	0.757	**0.782**	0.651	0.261	0.067	0.350	0.654	0.773	**0.793**
	Long	0.586	0.169	0.194	0.359	0.562	0.684	**0.735**	0.589	0.173	0.045	0.214	0.575	0.709	**0.741**
Porto Taxi	Short	0.697	0.359	0.701	0.442	0.721	0.804	**0.813**	0.705	0.381	0.358	0.372	0.726	0.832	**0.840**
	Medium	0.622	0.271	0.416	0.403	0.619	0.729	**0.742**	0.634	0.293	0.106	0.326	0.628	0.754	**0.765**
	Long	0.565	0.184	0.305	0.340	0.547	0.657	**0.679**	0.578	0.198	0.036	0.218	0.568	0.671	**0.690**
Beijing Bicycle	Short	0.652	0.303	0.587	0.559	0.673	0.788	**0.799**	0.670	0.313	0.272	0.330	0.685	0.802	**0.814**
	Medium	0.568	0.217	0.603	0.461	0.582	0.715	**0.737**	0.574	0.226	0.142	0.304	0.589	0.724	**0.747**
	Long	0.503	0.129	0.613	0.297	0.487	0.641	**0.675**	0.519	0.139	0.045	0.206	0.492	0.663	**0.694**

Datasets	Metric	F1-score							EDT						
	Length	RICK	MPR	CTRR	STRNN	DM	NASR	NASR+	RICK	MPR	CTRR	STRNN	DM	NASR	NASR+
Beijing Taxi	Short	0.717	0.359	0.253	0.431	0.749	0.834	**0.845**	4.594	8.287	9.082	7.551	4.362	3.376	**2.941**
	Medium	0.644	0.257	0.108	0.392	0.648	0.765	**0.787**	8.273	16.321	23.110	14.725	8.730	5.728	**5.192**
	Long	0.587	0.171	0.073	0.268	0.568	0.703	**0.738**	11.283	25.873	27.493	22.705	12.059	8.314	**7.102**
Porto Taxi	Short	0.701	0.370	0.474	0.404	0.723	0.818	**0.826**	4.801	8.104	6.935	8.790	4.496	3.563	**3.194**
	Medium	0.628	0.282	0.169	0.360	0.623	0.741	**0.753**	8.619	15.032	18.294	13.368	8.930	5.949	**5.280**
	Long	0.571	0.191	0.065	0.266	0.557	0.663	**0.684**	11.379	21.349	31.745	19.603	12.297	8.572	**7.339**
Beijing Bicycle	Short	0.661	0.308	0.372	0.414	0.679	0.795	**0.806**	5.183	8.924	7.784	7.092	4.629	3.719	**3.183**
	Medium	0.571	0.221	0.229	0.367	0.585	0.720	**0.742**	8.972	17.497	20.966	14.503	9.039	6.253	**5.319**
	Long	0.511	0.134	0.084	0.243	0.489	0.671	**0.684**	11.891	22.028	57.997	21.324	12.692	8.794	**7.395**



(a) Examining the RNN component. (b) Examining the estimation network. (c) Examining the search space. (d) Examining the generation of anchor sets.

Fig. 3. Detailed analysis of our model on the dataset of Beijing taxi using F1 measure.

observations for different variants: $ED < SP < NASR < NASR+$. Our full model NASR+ leads to the maximum reduction over the search space, which further indicates the effectiveness of the learned position-aware node embeddings.

6.3.4 The Effect of Different PA-GNN Variants

A major novelty of our model is it can learn more effective node embeddings by integrating the distance- and preference-based PA-GNN variants. Here, we would like to examine the effect of different ways to learn the position-aware node embeddings on the model performance. We consider four variants for the estimation network as comparisons, including (1) *r-PAGNN* using the original implementation in [19], (2) *d-PAGANN* using our distance-based variant, (3) *p-PAGANN* using our preference-based variant, and (4) *h-PAGANN* using a hybrid of our proposed two variants. In Fig. 3(d), it can be observed that the performance rank is as follows: *r-PAGNN* < *d-PAGNN* < *p-PAGNN* < *h-PAGNN*. The original PA-GNN performs worst, which indicates a simple application of PA-GNN may not work well on our task. The two variants are effective to improve the model performance, and the preference-based variant seems to yield a better performance. By integrating the two variants, our model achieves the best performance.

TABLE 4
Effect of different context features on Beijing taxi dataset using F1 measure.

Model	Short	Medium	Long
NASR+	0.845	0.787	0.738
¬week	0.813	0.751	0.715
¬hour	0.801	0.732	0.703
¬user	0.795	0.725	0.697

6.3.5 Effect of Different Context Features

In Eq. (5), besides location ID, we have considered different features to enrich the context information, namely user ID, weekly index and hourly index. Here, we study the effect of context features on the performance of our task. We keep our approach with all the context features as a reference. Then, we remove one kind of context feature at one time, and examine how it affects the performance: (1) *¬week* removes the week context, (2) *¬hour* removes the hour context, and (3) *¬user* removes the user context. The comparison results are reported in Table 4. From this table, we can observe that the three kinds of features contribute to the final performance. In particular, user ID seems to be more important the other two kinds of temporal features.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

TABLE 5
Examining the influence of data sparsity on the model performance with Beijing dataset using F1 measure.

Type	≤ 5	≤ 10	≤ 15
NASR+	0.755	0.814	0.837
RICK	0.569	0.621	0.673
DeepMove	0.575	0.639	0.701

This finding is reasonable since the task itself has a personalized setting.

6.3.6 The Influence of Data Sparsity

Our current task setting is highly based historical trajectory data, *i.e.,* learning the route recommendation model using these historical data. A potential issue is that data sparsity (*e.g.,* some locations are seldom been visited) will influence the algorithm performance. Here, we examine the influence of data sparsity on the route recommendation algorithm. In specific, we only consider the recommendation performance on the infrequently visited locations at different sparsity levels (≤ 5 times, ≤ 10 times and ≤ 15 times). We select the baselines of RICK and DeepMove for comparison. Table 5 reports the performance of the three methods at different sparsity levels. As we can see, our model performs consistently better than the other two baselines, especially for the most sparse case. By designing a more effective model architecture, our approach is more resistible to the data sparsity.

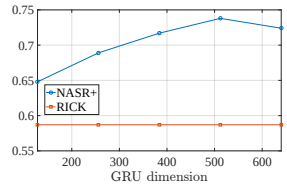
6.4 Parameter Tuning

In our model, there are several parameters to tune. We present the tuning results of F1 scores of four important parameters in Fig. 4.

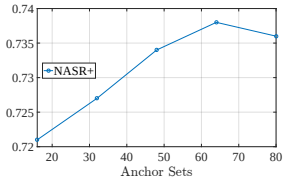
Since the $G(\cdot)$ is developed based on GRU, an important parameter to consider is the dimensionality K_R of the hidden state in GRU. We vary the embedding size from 128 to 640, with a gap of 128. As shown in Fig. 4(a), the optimal embedding size is around 500. Overall, the range from 400 to 600 gives good performance.

To implement the $H(\cdot)$ function, we need to set the number of anchor sets for use. Overall, using more anchor sets will increase the capacity of learning structural characteristics, and meanwhile incur a high model complexity. Intuitively, it should be set to a suitable value neither too large nor too small. To see the influence of the number of anchor sets, we vary it from 16 to 80 with a gap of 16 for the Beijing dataset. Fig. 4(b) presents the varying results for different numbers of anchor sets. It can be observed that using 64 anchor sets gives the best performance for the Beijing dataset.

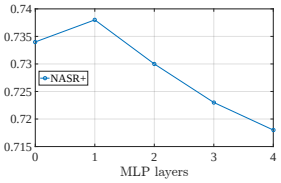
In the $H(\cdot)$ network, another parameter to tune is the number of hidden layers in MLP defined in Eq. (20). We vary the number of layers from 0 to 4. From Fig. 4(c), it can be observed that using relatively fewer hidden layers lead to a better performance. The optimal performance is achieved with one hidden layer. Finally, we also tune the number of hidden layers in GNN defined in Eq. (14). We vary the number of GNN layers from 1 to 4. Overall, using two hidden layers achieves the optimal performance as shown in Fig. 4(d).



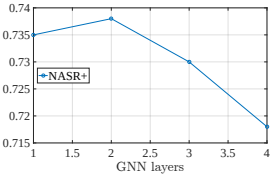
(a) The hidden dimensions in GRU.



(b) The # of acnchor sets in the estimation network.



(c) The # of hidden layers in MLP.



(d) The # of graph neural network layer.

Fig. 4. Parameter sensitivity on long queries with Beijing taxi dataset using F1 measure.

6.5 Qualitative Analysis on Model Interpretability

Previously, we have shown the effectiveness of our model in the PRR task. In this part, we qualitatively analyze why NASR+ is able to yield a good performance.

In NASR+, the improved graph neural network is the core component for modeling road network information. It can generate informative node representations for encoding structural characteristics. To see this, we present an illustrative example in Fig. 5. A user is currently located at l_i and moving towards the destination l_d. Recall that, according to Eq. (18), the elements in $\boldsymbol{z}_{l_k}^{(P)}$ represent the importance levels of anchor-sets on the representation of l_k, called *anchor-set importance* for short. For each location l_k on the road network, we plot the anchor-set importance with varying darkness degrees on a location in an associated sub-figure. A darker triangle means the anchor-set is more important for location l_k. For comparison, we plot anchor-set importance expressed by $\boldsymbol{z}^{(P)}$ for locations on both the actual and shortest routes. As we can see, the anchor-set importance subfigure for l_i is more similar to those for locations from the actual route (upper right) than the shortest route (left lower). By inspecting into the dataset, we find the shortest route contains several side road segments that are possibly in traffic congestion at the visit time. Another interesting observation is that the user indeed visits the locations in the actual route more times in historical trajectories. These observations indicate that our model is able to learn effective node representations for identifying more important locations to explore for the PRR task.

Next, we continue to study how the learned cost function helps the search procedure in NASR+. Figure 6 presents a sample trajectory from a specific user. Given the source and destination, we need to predict the actual route. By comparing Fig. 6(a) (the original search space) and Fig. 6(b) (the reduced search space by NASR+), it can be seen that our model is able to effectively reduce the search space, *i.e.,* 65% search space has been reduced for this case. When zooming into a subsequence of this route, we further compare the estimated cost values for two candidate locations (green points) in Fig 6(c). Although the upper location has

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3068479, IEEE Transactions on Knowledge and Data Engineering

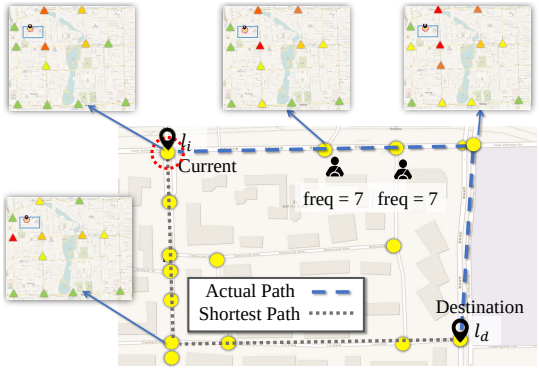


Fig. 5. Visualization of the learned representation using improved position-aware graph neural networks. The yellow circles denote locations in the road network. The colored triangles denote the centric points of anchor-sets in the road network. A darker triangle in a subfigure indicates the corresponding author-set is more important to the associated location. "*freq*" denotes the visit frequency by the user in historical trajectories.

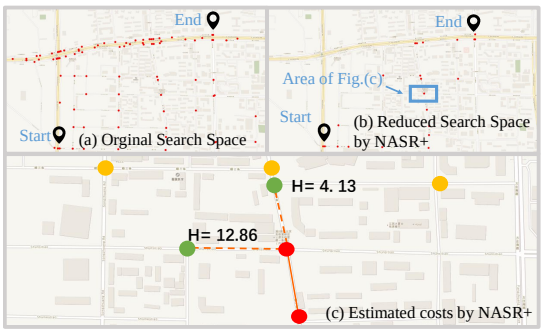


Fig. 6. Visualization of the search procedure with the estimated costs by the NASR+ model. In (c), red points have been already explored and green points are candidate locations to extend in A^* search algorithm.

a longer distance with the explored locations, it is located on the main road that is likely to lead to a better traffic condition. Our model is able to predict a lower cost for the upper location by effectively learning such trajectory characteristics from road network and historical data.

7 CONCLUSIONS

In this paper, we took the initiative to use neural networks to automatically learn the cost functions in A^* for the PRR task. We first presented a simple A^* solution for solving the PRR task, and formally defined the suitable form for the search cost. Then, we set up two components to learn the two costs respectively, *i.e.*, the RNN component for $G(\cdot)$ and the estimation network for $H(\cdot)$. The two components were integrated in a principled way for deriving a more accurate cost of a candidate location for search. A major novelty of this model lies in the estimation network, which is developed based on position-aware graph attention networks. By selecting suitable anchor sets, the estimation network is more capable of learning distance and preference structure characteristics of road networks. We constructed extensive experiments for verifying the effectiveness and robustness of the proposed model. Interestingly, besides the system performance, we have found that the proposed model is also able to effectively reduce the search space.

A possible extension of our work is the incorporation of prior information. Currently, we mainly learn the model and data representations through the historical trajectories, which can be either sparse or noisy. In order to improve our model, it will be also useful to inject prior information about traffic conditions, *e.g.*, the congestion time of a crossroad. Besides, our approach relies on the cost function for the decision of each candidate location. However, the cost function is difficult to understand, *e.g.*, why a candidate location has a large estimated cost. As future work, we will consider designing more interpretable estimation network for deriving the cost. Besides, our elaborate model structure also introduce some additional computational complexity. Therefore, improving the computational efficiency is also an extension direction of our model.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2019YFB2102103), the National Natural Science Foundation of China (Grant No.92046010, 61872369, 61572059). Dr. Jingyuan Wang's work was partially supported by BRICS STI Framework Programme: Response to COVID-19 global pandemic (MFQuantiC), the CCF-DiDi Gaia Collaborative Research Funds for Young Scholars, and the Fundamental Research Funds for the Central Universities (Grant No. YWF-20-BJ-J-839).

REFERENCES

[1] J. Wang, C. Chen, J. Wu, and Z. Xiong, "No longer sleeping with a bomb: a duet system for protecting urban safety from dangerous goods," in *KDD'17*. ACM, 2017, pp. 1673–1681.
[2] J. Wang, X. He, Z. Wang, J. Wu, N. J. Yuan, X. Xie, and Z. Xiong, "CD-CNN: a partially supervised cross-domain deep learning model for urban resident recognition," in *AAAI'18*, 2018.
[3] J. Wang, J. Wu, Z. Wang, F. Gao, and Z. Xiong, "Understanding urban dynamics via context-aware tensor factorization with neighboring regularization," *IEEE TKDE*, 2019.
[4] L. Tang, Z. Duan, Y. Zhu, J. Ma, and Z. Liu, "Recommendation for ridesharing groups through destination prediction on trajectory data," *IEEE TITS'19*, pp. 1–14, 2019.
[5] G. Cui, J. Luo, and X. Wang, "Personalized travel route recommendation using collaborative filtering based on gps trajectories," *IJED*, vol. 11, no. 3, pp. 284–307, 2018.
[6] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *ICDE*, April 2015, pp. 543–554.
[7] L. Y. Wei, Y. Zheng, and W. C. Peng, "Constructing popular routes from uncertain trajectories," in *KDD'12*, 2012, pp. 195–203.
[8] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *SIGMOD'13*, 2013, pp. 713–724.
[9] H. Wu, J. Mao, W. Sun, B. Zheng, H. Zhang, Z. Chen, and W. Wang, "Probabilistic robust route recovery with spatio-temporal dynamics," in *KDD'16*, 2016, pp. 1915–1924.
[10] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *ICDE'11*, 2011, pp. 900–911.
[11] A. Al-Molegi, M. Jabreel, and B. Ghaleb, "STF-RNN: space time features-based recurrent neural network for predicting people next location," in *SSCI'16*, 2016, pp. 1–7.
[12] C. Yang, M. Sun, W. X. Zhao, Z. Liu, and E. Y. Chang, "A neural network approach to jointly modeling social networks and mobile trajectories," *ACM T-IS*, vol. 35, no. 4, pp. 36:1–36:28, 2017.
[13] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, "Colight: Learning network-level cooperation for traffic signal control," in *CIKM'19*, 2019, pp. 1913–1922.
[14] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *ICLR'18*, 2018.

[15] J. Wang, X. Wang, C. Li, J. Wu *et al.*, "Deep fuzzy cognitive maps for interpretable multivariate time series prediction," *IEEE TFS'20*, 2020.

[16] S. Guo, C. Chen, J. Wang, Y. Liu, X. Ke, Z. Yu, D. Zhang, and D.-M. Chiu, "Rod-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data," *IEEE TMC'19*, 2019.

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE TSSC'68*, vol. 4, no. 2, pp. 100–107, 1968.

[19] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *ICML'19*, 2019, pp. 7134–7143.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR'18*, 2018.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR'16*, 2016.

[22] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[23] N. Wu, J. Wang, W. X. Zhao, and Y. Jin, "Learning to effectively estimate the travel time for fastest route recommendation," in *CIKM'19*. ACM, 2019, pp. 1923–1932.

[24] J. Wang, N. Wu, X. Lu, X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using kalman filter," *IEEE TKDE'19*, 2019.

[25] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering A* search algorithms with neural networks for personalized route recommendation," in *SIGKDD'19*. ACM, 2019, pp. 539–547.

[26] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *SIGSPATIAL*. ACM, 2010, pp. 99–108.

[27] H. Liu, L. Y. Wei, Y. Zheng, and M. Schneider, "Route discovery from mining uncertain trajectories," pp. 1239–1242, 2011.

[28] S. Shafique and M. E. Ali, "Recommending most popular travel path within a region of interest from historical trajectory data," in *SIGSPATIAL*. ACM, 2016, pp. 2–11.

[29] D. Chen, C. S. Ong, and L. Xie, "Learning points and routes to recommend trajectories," in *CIKM*. ACM, 2016, pp. 2227–2232.

[30] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang, "Content-aware hierarchical point-of-interest embedding model for successive POI recommendation." in *IJCAI'18*, 2018, pp. 3301–3307.

[31] X. Zhou, C. Mascolo, and Z. Zhao, "Topic-enhanced memory networks for personalised point-of-interest recommendation," *SIGKDD'19*, 2019.

[32] S. Zheng, Y. Yue, and P. Lucey, "Generating long-term trajectories using deep hierarchical networks," in *NIPS'17*, 2017, pp. 1543–1551.

[33] Q. Gao, F. Zhou, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Predicting human mobility via variational attention," in *WWW'19*. ACM, 2019, pp. 2750–2756.

[34] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *ICJAI'17*, 2017, pp. 3083–3090.

[35] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *WWW'18*, 2018, pp. 1459–1468.

[36] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: a recurrent model with spatial and temporal contexts," in *AAAI'16*, 2016, pp. 194–200.

[37] L. Lelis, R. Stern, and S. J. Arfaee, "Predicting solution cost with conditional probabilities," in *ASSC'11*, 2011.

[38] M. Ernandes and M. Gori, "Likely-admissible and sub-symbolic heuristics," in *ECAI'04*. Citeseer, 2004, pp. 613–617.

[39] M. Samadi, A. Felner, and J. Schaeffer, "Learning from multiple heuristics." in *AAAI'08*, 2008, pp. 357–362.

[40] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *IJGIS'18*, vol. 32, no. 3, pp. 547–570, 2018.

[41] K. Nachtigall, "Time depending shortest-path problems with applications to railway networks," *EJOR'95*, vol. 83, no. 1, pp. 154–166, 1995.

[42] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in *ICDE'06*. IEEE, 2006, pp. 10–10.

[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *ICML'15*, 2015, pp. 2067–2075.

[44] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR'15*, 2015.

[45] S. Milgram, "The small world problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[46] J. Han, M. Kamber, and A. K. H. Tung, "Spatial clustering methods in data mining: A survey," in *Geographic data mining and knowledge discovery*, 2001.

[47] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD'96*, vol. 96, no. 34, 1996, pp. 226–231.

[48] Y. Wu, W. Wu, D. Yang, C. Xu, and Z. Li, "Neural response generation with dynamic vocabularies," in *AAAI'18*, vol. 32, no. 1, 2018.

[49] M. Thorup, "Compact oracles for reachability and approximate distances in planar digraphs," *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 993–1024, 2004.

[50] T. Kameda, "On the vector representation of the reachability in planar directed graphs," *Information Processing Letters*, vol. 3, no. 3, pp. 75–77, 1975.

[51] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations." in *IJCAI'15*, vol. 15, 2015, pp. 1778–1784.

Jingyuan Wang received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China. He is currently an Associate Professor of School of Computer Science and Engineering, Beihang University, China. His is also the head of the BIGSCity lab, and Vice Director of the Beijing City Lab (BCL). He published more than 30 papers on top journals and conferences, such as SIGKDD, AAAI, ICDM, IEEE TKDE, IEEE TMC, ACM TOIS, *etc.*. His general area of research is data mining and machine learning, with special interests in smart cities.

Ning Wu received the MEng degree in computer science from the Beihang University, China, in 2021. He is currently a data scientist in Microsoft, working on Bing search engine. He has published several papers in international conferences and journals such as SIGKDD, CIKM, IEEE TKDE. His research interests are urban data analysis and natural language processing.

Wayne Xin Zhao received the PhD degree from Peking University in 2014. He is currently a tenured associated professor in Gaoling School of Artificial Intelligence, Renmin University of China. His research interests are web text mining and natural language processing. He has published a number of papers in international conferences and journals such as ACL, SIGIR, SIGKDD, WWW, ACM TOIS, and IEEE TKDE. He is a member of the IEEE.