PlanRanker: Towards Personalized Ranking of Train Transfer Plans

Jia Xu
Guangxi University
Nanning, Guangxi, China
xujia@gxu.edu.cn

Wanjie Tao*
Alibaba Group
Hangzhou, Zhejiang, China
wanjie.twj@alibaba-inc.com

Zulong Chen*
Alibaba Group
Hangzhou, Zhejiang, China
zulong.czl@alibaba-inc.com

Jin Huang
Alibaba Group
Hangzhou, Zhejiang, China
fengchu.hj@alibaba-inc.com

Huihui Liu
Alibaba Group
Hangzhou, Zhejiang, China
jinyu.lhh@alibaba-inc.com

Hong Wen
Alibaba Group
Hangzhou, Zhejiang, China
qinggan.wh@alibaba-inc.com

Shenghua Ni
Alibaba Group
Hangzhou, Zhejiang, China
shenghua.nish@alibaba-inc.com

Qun Dai
Nanjing University of Aeronautics and Astronautics
Nanjing, Jiangsu, China
daiqun@nuaa.edu.cn

Yu Gu
Northeastern University
Shenyang, Liaoning, China
guyu@mail.neu.edu.cn

ABSTRACT

Train transfer plan ranking has become the core business of online travel platforms (OTPs), due to the flourish development of high-speed rail technology and convenience of booking trains online. Currently, mainstream OTPs adopt rule-based or simple preference-based strategies to rank train transfer plans. However, the insufficient emphasis on the costs of plans and the negligence of considering reference transfer plans make these existing strategies less effective in solving the personalized ranking problem of train transfer plans. To this end, a novel personalized deep network (*PlanRanker*) is presented in this paper to better address the problem. In PlanRanker, a *personalized learning* component is first proposed to capture both of the query semantics and the target transfer plan-relevant personalized interests of a user over the user's behavior log data. Then, we present a *cost learning* component, where both of the price cost and the time cost of a target transfer plan are emphasized and learned. Finally, a *reference transfer plan learning* component is designed to enable the whole framework of PlanRanker to learn from reference transfer plans which are pieced together by platform users and thus reflect the wisdom of crowd. PlanRanker is now successfully deployed at Alibaba Fliggy, one of the largest OTPs in China, serving millions of users every day for train ticket reservation. Offline experiments on two production datasets and a country-scale online A/B test at Fliggy both demonstrate the superiority of the proposed PlanRanker over baselines.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599887

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Instance-based learning*.

KEYWORDS

Personalized ranking, Train transfer plans, Reference transfer plans, Deep network, Contrastive learning, Attention mechanism

ACM Reference Format:
Jia Xu, Wanjie Tao*, Zulong Chen*, Jin Huang, Huihui Liu, Hong Wen, Shenghua Ni, Qun Dai, and Yu Gu. 2023. *PlanRanker*: Towards Personalized Ranking of Train Transfer Plans. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599887

1 INTRODUCTION

With the rapid development of railway and the construction of large-scale passenger line all over the world, traveling by trains plays an increasingly more critical role in people's daily life. According to the statistics of National Railway Administration of the People's Republic of China, the national railway transport volume is as large as 1.61 billion in past year 2022 [21]. Train transfer plans, which response to an Origin-Destination (OD) query like those direct plans, are quite important to serve such massive passengers. Taking Figure 1 as an example, given an origin railway station (o) and a destination railway station (d) in China, train transfer plans are indispensable, since: 1) there is not a direct plan from $o = Dali$ to $d = Chengdu$; 2) the direct plan has a higher price cost from $o = Guilin$ to $d = Xi'an$; and 3) the direct plan has a higher time cost from $o = Guangzhou$ to $d = Suzhou$. Hence, personalized train transfer plan rankings that ranks plans according to each passenger's preferences so as to improve user experience and increase the post-view click-through and conversion rate (CTCVR) has become the core business of many online travel platforms (OTPs) in China, such as 12306

China Railway[1], Ctrip[2], and Alibaba Fliggy[3]. According to recent statistics of Fliggy, the proportion of orders w.r.t. train transfer plans has reached 37% within orders created for train reservations.

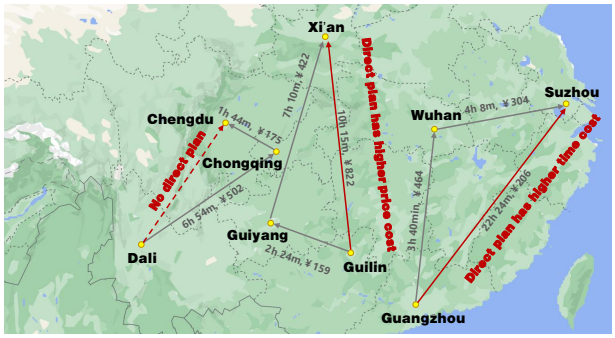


Figure 1: Importance of train transfer plans (h-hour; min-minute; a direct plan: a red solid line with arrow).

The most commonly used strategy by recent OTPs is time cost based or users' preferences based ranking methods. For example, 12306 China Railway ranks plans by time cost as default, while Ctrip tries to rank plans for each user based on the user's preferences and gives consideration to the diversity of departure time and transfer railway stations. Although recently researchers from academia and industry have proposed several route recommendation strategies [1, 2, 4, 10, 16–18, 28], there are still two limitations faced by train transfer plan ranking systems. 1) *Insufficient emphasis on the costs of a plan.* According to the experience of Alibaba Fliggy, the costs (i.e., price cost and time cost) of a train transfer plan have an important impact on the popularity of the plan among platform users. Hence, a plan having lower costs is supposed to be assigned with a higher rank. 2) *Failing to consider reference transfer plans.* A reference transfer plan for an *OD* query generally a certain number of successive train trips initiated by platform users that enables the passenger to travel from the origin station to the destination station by trains. Apparently, reference transfer plans reflect crowd wisdom. They thus are usually cost-effective or with some attractive characteristics (e.g., the transit cities are more friendly to passengers), which are very helpful to optimize the predicted ranks of target transfer plans.

To address the above two limitations, we propose ***PlanRanker***, a novel deep neural networks for personalized ranking problem of train transfer plans. In PlanRanker, a *Personalized Learning Component* is at first proposed to capture the semantics of an OD query and the target transfer plan-relevant personalized preferences of a user over the user's behavior logs. Then, to overcome the first limitation of recent works, we propose a *Cost Learning Component* where the idea of monotonic learning is applied to emphasize the importance of price cost and time cost of a target transfer plan in computing its rank. Meanwhile, to fully utilize the information of reference plans given by platform users for the same query and handle the second limitation of existing works, we design a *Reference Transfer Plan Learning Component* in PlanRanker, which integrates both instance-level and cluster-level contrastive learning to ensure the whole learning framework of PlanRanker absorbing the crowd wisdom. Our major contributions are summarized as follows:

- To the best of our knowledge, we make the first attempt to formally define the personalized ranking problem of train transfer plans identified from real-life scenarios.
- We design PlanRanker, a novel deep network that effectively solves the personalized ranking problem of train transfer plans. In specific, a series of optimization strategies, i.e., learning from plan costs and learning from reference transfer plans, are carefully designed and applied in PlanRanker that well address two limitations of related works. PlanRanker is now successfully deployed at Alibaba Fliggy platform and has achieved surprising results in serving millions of users each day for train ticket reservations.
- Offline experiments on two Fliggy production datasets and an online A/B test at Fliggy mobile APP both show that the proposed PlanRanker consistently and significantly outperforms all baselines w.r.t. several evaluation metrics. In particular, PlanRanker gains averagely 4.68% increase at CTCVR compared with the best baseline, which is a remarkable improvement for industrial application scenarios.

2 RELATED WORKS

The closest line of works to our train transfer plan ranking problem is the route or transportation recommendation. Route recommendation aims to provide a route between the origin location and the destination location in a road network based on a given cost function, which has become a core component in map service, such as Google Maps and Baidu Maps. The route recommendation task is achieved based on different types of trajectory data, e.g., GPS data [31] or POI check-in data [3]. A common strategy to the task is to apply the shortest path queries [8] with a predefined cost function [23], which is usually solved by classic route-finding algorithms, such as Dijkstra's algorithm and A^* algorithm. Some related studies focus on the cost functions that maximize the profit of taxi drivers or experience of customers [22, 32], while the popularity indicator learned from users' historical traveling behaviors can also be used to define the cost function and recommend popular routes [5, 29, 30]. However, all these studies are non-personalized. Personalized route recommendation takes users' preferences in the calculation of the cost function, which are more friendly to individual's needs. T-Drive [31] recommends the practically fastest route to a destination at a given departure time based on taxi drivers' intelligence. [13] provides personalized routes for individual driver according to their driving preferences (e.g., time-efficient or fuel-efficient). In [16], spatio-temporal autocorrelations within road networks and the semantic coherence of route sequences are utilized to improve the recommended multi-modal routes. Other personalized route planning studies include [19], [28], and [4]. Recently, many works are proposed to address the route planning problem from shared mobility domain, which focus on improving the efficiency of urban transportation [27] or on raising the profit [24–26]. Nevertheless, all these methods cannot be directly generalized to address the multi-model recommendations, which are

[1]https://www.12306.cn/
[2]https://www.ctrip.com/
[3]https://www.fliggy.com

properly solved by a few machine learning based techniques. To name a few, FAVOUR [2] provides personalized, situation-aware multi-modal route proposals based on user-provided profiles and survey data. [1] discusses the recommendation of faster and convenient multi-model routes that satisfy travelers' preferences based on real-time transit data. Trans2vec [10] combines the historical travel behaviors and user/OD relevance for learning transport mode preference and recommending routes. Most recently, Hydra [17, 18], is proposed which adaptively recommends uni-modal and multi-modal routes according to the user preferences and the situational context. Another work that is highly relevant to ours is PFRN [11], a personalized flight itinerary ranking network which is designed based on a group of attentive mechanisms. Specifically, PFRN makes use of the learned travel intentions of users to improve the ranking results, but it is designed to rank only the direct itineraries of flights.

Although some of the aforementioned studies, e.g., Hydra and PFRN, share some similar concepts and solutions that can be transferred to the context of train transfer plan ranking, they fail to pay sufficient attention to the costs of a transfer plan which are generally cared by passengers and neglect the importance of reference transfer plans in enhancing the predicted ranks of target transfer plans, which lead to unsatisfactory ranking results.

3 PRELIMINARIES

In this section, we first introduce important definitions and then formalize the problem investigated in this paper.

Let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ represent a set of $|\mathcal{U}|$ users, $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ be a set of $|\mathcal{A}|$ train numbers, and $\mathcal{L}$ denote a set of $|\mathcal{L}|$ railway stations. We define several essential concepts as follows.

Definition 3.1. ***Origin-Destination (OD) Query.*** An OD query $q(o, d) \in \mathcal{Q}$, is a query initiated by a user $u \in \mathcal{U}$ which corresponds to a cross-regional train trip of the user. Here, $o, d \in \mathcal{L}$ represent the origin railway station and the destination railway station of the trip, respectively. Notice that the departure date from o is not considered by the OD query. This is because we assume the trains operated in each day are commonly the same, which is consistent with the situation in China.

Definition 3.2. ***Train Trip.*** A train trip satisfying an OD query $q(o, d)$ in the form of a six-tuple $p^q =< a, o, d, t_o, t_d, c >$ is a trip by the train No. a which leaves from the railway station $o \in \mathcal{L}$ at the time t_o and arrives at the railway station $d \in \mathcal{L}$ at the time t_d, having a time cost $(t_d - t_o) \in \mathbb{R}$ and a price cost $c \in \mathbb{R}$.

Definition 3.3. ***Train Transfer Plan.*** Given an OD query $q(o, d) \in \mathcal{Q}$, a train transfer plan for the query is denoted as $P^q \in \mathcal{P}^q$, where $P^q = \{p_1^q, \ldots, p_{|P^q|}^q\}$ is an ordered sequence of train trips satisfying the following constrains:

- $p_1^q.o = q.o$ and $p_{|P^q|}^q.d = q.d$;
- For any two successive train trips $p_l^q, p_k^q \in P^q$ with $l < k$, $p_l^q.t_d < p_k^q.t_o$ holds.

Given a train transfer plan P^q and any two successive train trips in the plan (i.e., $p_l^q, p_k^q \in P^q$ with $l = k - 1$), we have:

- $p_l^q.d = p_k^q.o$ indicates a transfer at the same station, while $p_l^q.d \neq p_k^q.o$ represents a transfer at different stations;
- The transfer time between p_l^q and p_k^q is $p_k^q.t_o - p_l^q.t_d$;
- The time cost of the plan is $p_{|P^q|}^q.t_d - p_1^q.t_o$;
- The price cost of the plan is $\sum_{l=1}^{|P^q|} p_l^q.c$.

Notice that a transfer plan may contain arbitrary amount of successive train trips.

Definition 3.4. ***Reference Transfer Plan.*** Given an OD query $q(o, d) \in \mathcal{Q}$, its reference transfer plan represented by $R^q \in \mathcal{R}^q$ is a transfer plan for the query that consists of a list of successive train trips manually pieced together by users at an OTP.

With the above notations and concepts, the problem investigated in this paper is formalized as follows:

Definition 3.5. ***Personalized Ranking of Train Transfer Plans.*** Given an OD query $q(o, d) \in \mathcal{Q}$ initiated by a user $u \in \mathcal{U}$, let $\mathcal{B}^u = \{b_1^u, \ldots, b_{|\mathcal{B}|}^u\}$ denote a sequence of historical behaviors (e.g., clicking or booking plans) of user u at an OTP, and let $\mathcal{R}^q = \{R_1^q, \ldots, R_{|\mathcal{R}^q|}^q\}$ be a list of reference transfer plans for the query. Given a list of target transfer plans for the query q, denoted by $\mathcal{T}^q = \{T_1^q, \ldots, T_{|\mathcal{T}^q|}^q\}$, the objective of the personalized train transfer plan ranking problem is to predict the probability $Pr(T_z|q, \mathcal{T}^q, \mathcal{B}^u, \mathcal{R}^q)$ of the z^{th} target train transfer plan chosen by user u.

4 PLANRANKER

4.1 Overview

Figure 2 shows an overview of PlanRanker. It mainly contains four components, namely ***embedding layer*** (EL), ***personalized learning*** (PL), ***cost learning*** (CL), and ***reference transfer plan learning*** (RTPL). EL generates embeddings for all inputs. PL mainly learns the target transfer plan-relevance personalized interests of users via some attentive mechanisms. CL leverages the idea of monotonic learning to emphasize the importance of price cost and time cost of a target transfer plan in the ranking procedure. Then, RTPL makes the whole network of PlanRanker learn from reference transfer plans by conducting both of the instance-level and cluster-level contrastive learning.

4.2 Embedding Computation

As illustrated by Fig. 2, mainly four categories of inputs, i.e., *OD query*, *user behavior sequence*, *target transfer plan*, and *reference transfer plan*, are used to train the PlanRanker model, and each input category is represented by a group of features. In specific, an OD query $q(o, d)$ contains the IDs of both of the origin railway station $o \in \mathcal{L}$ and the destination railway station $d \in \mathcal{L}$ corresponding to the query. The behavior sequence of a user $u \in \mathcal{U}$, denoted by $\mathcal{B}^u = \{b_1^u, \ldots, b_i^u, \ldots\}$ contains time-based ordered transfer plan IDs that have been historically clicked, booked, or put in favorites by user u at an OTP. A target transfer plan (or a reference transfer plan) for the query q, represented as T^q (or R^q), contains the information of plan ID, transfer railway station, price cost, and time cost of the plan. Each feature of an input category (e.g., o='Beijing' in an OD query) is firstly transformed into a high-dimensional sparse one-hot vector (e.g., [0,…,1,…,0] denotes the city Beijing in China). Then, following the widely-applied embedding technique proposed in [20], the high-dimensional sparse one-hot vector of each feature is

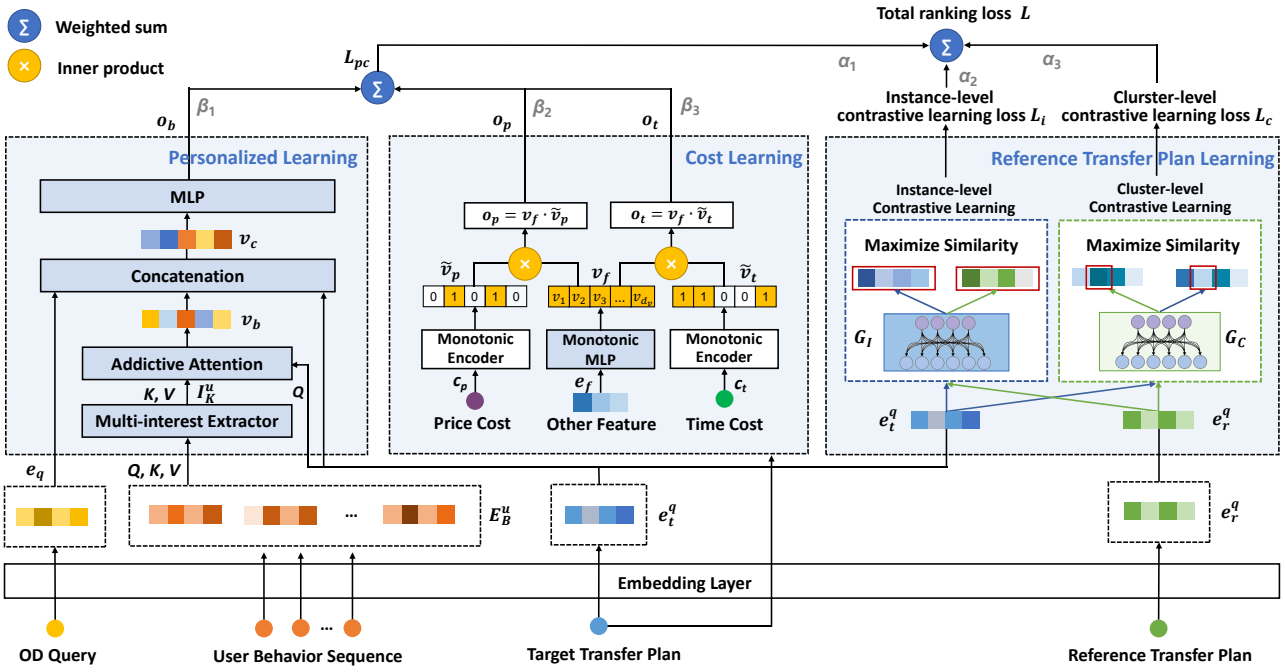


Figure 2: Framework of the PlanRanker model.

converted to a low-dimensional dense vector. After the computation of the embedding layer, the embedding vectors of the OD query, target transfer plan, and reference transfer plan are denoted by $\boldsymbol{e_q} \in \mathbb{R}^d$, $\boldsymbol{e_t^q} \in \mathbb{R}^d$, and $\boldsymbol{e_r^q} \in \mathbb{R}^d$, respectively, where d is the dimensionality of each vector. The behavior sequence of user u, which is composed by a sequence of transfer plan IDs, is denoted as a matrix $\boldsymbol{E_B^u} = [\ldots; \boldsymbol{e}(b_i^u); \ldots]^T \in \mathbb{R}^{N \times d}$, where N is the number of user behaviors and every $\boldsymbol{e}(b_i^u) \in \mathbb{R}^d$ in $\boldsymbol{E_B^u}$ is the embedding vector of the i^{th} transfer plan historically interacted by user u.

4.3 Personalized Learning

In the personalized learning component (PL), at the beginning, multiple interests of a user are extracted via a multi-interest extractor, which is an implementation of the multi-interest extractor layer in the MIND (Multi-Interest Network with Dynamic Routing) model [15]. In specific, the extractor utilizes multiple representation vectors to express distinct interests of users separately, rather than representing user interests by only one representation vector which has been proven less precise than the multi-vector solution. To learn multiple representation vectors, a clustering procedure is applied to group a user's historical behaviors into K clusters. Transfer plans interacted by users from one cluster are expected to be closely related and collectively represent one particular aspect of user interests. The representation vector for every resulted cluster is inferred in the multi-interest extractor subsequently, which finally outputs a matrix $\boldsymbol{E_I^u} = [\boldsymbol{I_1^u}; \ldots; \boldsymbol{I_K^u}]^T \in \mathbb{R}^{K \times d}$, where d is the dimensionality of the representation vector for every user interest and $\boldsymbol{I_i^u}$ is the representation vector of the i^{th} user interest.

Since not all extracted representation vectors of user interests are relevant to the target transfer plan, different weights should be assigned to each representation vector according to its relevance to the plan. To achieve this, we apply an addition attention mechanism with the embedding of the target transfer plan, i.e., $\boldsymbol{e_t^q}$, as the query to adaptively learn the weight for the representation vector of each user interest in $\boldsymbol{E_I^u} = [\boldsymbol{I_1^u}; \ldots; \boldsymbol{I_K^u}]^T$. Specifically, the weight of the i^{th} representation vector $\boldsymbol{I_i^u} \in \boldsymbol{E_I^u}$, i.e., α_i, is computed as follows:

$$\begin{aligned} \alpha_i &= \frac{exp(a_i)}{\sum_{j=1}^{|E_I^u|} exp(a_j)}, \\ a_i &= z^T \tanh\left(\boldsymbol{W^t e_t^q} + \boldsymbol{W^i I_i^u}\right), \end{aligned} \tag{1}$$

where $\boldsymbol{e_t^q} \in \mathbb{R}^d$ is the embedding vector of the target transfer plan and $\boldsymbol{I_i^u} \in \mathbb{R}^d$ is the embedding vector of the i^{th} user behavior which corresponds to the i^{th} column in matrix $\boldsymbol{E_I^u}$. $\boldsymbol{W^t} \in \mathbb{R}^{d_h \times d}$, $\boldsymbol{W^i} \in \mathbb{R}^{d_h \times d}$, $\boldsymbol{z} \in \mathbb{R}^{d_h}$ are learnable parameters.

Then, by using a weighted sum pooling operation as shown in Equation 2, the matrix $\boldsymbol{E_I^u}$ is converted to a feature vector $\boldsymbol{v_b}$ to generate a new representation of the user behavior sequence which is target transfer plan-relevant.

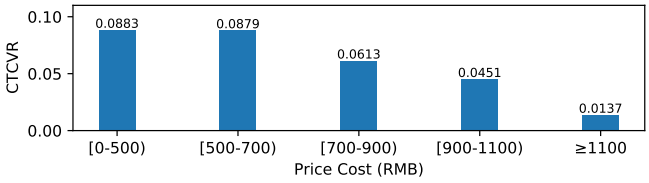
$$\boldsymbol{v_b} = \sum_{i=1}^{|E_I^u|} (\alpha_i \times \boldsymbol{I_i^u}). \tag{2}$$

With the additive attention mechanism, behaviors more relevant to the target transfer plan are weighted higher and dominate the target transfer plan-relevant representation of user's behaviors, i.e., $\boldsymbol{v_b}$. $\boldsymbol{v_b}$ is then concatenated with the embeddings of the OD query ($\boldsymbol{e_q}$) and the target transfer plan ($\boldsymbol{e_t^q}$) in a concatenation layer and derive a vector $\boldsymbol{v_c} = \boldsymbol{u_b} \oplus \boldsymbol{e_q} \oplus \boldsymbol{e_t^q}$ with $\boldsymbol{v_c} \in \mathbb{R}^{3d}$. Finally, the

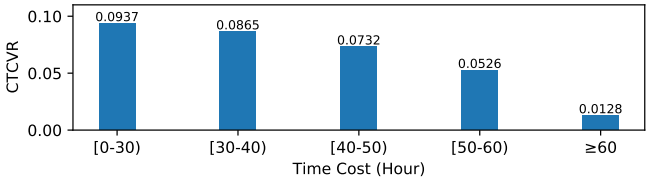
vector $\boldsymbol{v_c}$ is processed by an MLP layer which outputs the value o_b. o_b reflects the compliance degree of the target transfer plan w.r.t. semantic of the OD query q and the interests of user u.

4.4 Cost Learning

Figure 3 illustrates the statistics of users' CTCVR (post-view click through and conversion rate) towards returned train transfer plans for one popular OD query *'Beijing→Sanya'* w.r.t. different price (or time) cost intervals, which are derived at the Alibaba Fliggy platform from the date 01/05/2022 to the date 31/07/2022. The figure shows that users are quite sensitive to the price (or time) cost of train transfer plans, preferring to click or book those transfer plans with lower costs. Such important observation from productive practice inspires us to design a cost learning component (CL) in PlanRanker to ensure the ranking results of target transfer plans satisfying two intuitions: '*cheaper is better*' and '*less time is better*'.



(a) Statistics of CTCVR w.r.t. different price cost intervels



(b) Statistics of CTCVR w.r.t. different time cost intervels

Figure 3: Statistics of CTCVR w.r.t. different costs.

As illustrated in Figure 2, the inputs of CL are the original data of the target transfer plan, which are then divided into three parts, namely the price cost (c_p), the time cost (c_t), and other feature ($\boldsymbol{e_f}$). Notice that c_p and c_t are both real numbers, while $\boldsymbol{e_f}$ is computed following the embedding idea having been explained in Section 4.2. Then, c_p (or c_t) is encoded by a monotonic encoder (ME), which transfers c_p (or c_t) into a binary representation vector $\boldsymbol{v_p}$ (or $\boldsymbol{v_t}$) having a dimensionality of d_v. Take the price cost c_p for example and let min_c^q and max_c^q be the minimum price cost and maximum price cost in terms of all transfer plans satisfying the OD query q, separately. ME firstly computes $\Delta = (c_p - min_c^q)$ to derive the gap between the price cost of the current target transfer plan and the smallest price cost for q. Then, the domain $[min_c^q, max_c^q]$ is evenly divided into L bins, where the index of the left-most bin is 1 and the index of the right-most bin is L. Assuming that c_p is within the bin l, then $\boldsymbol{v_p}$, i.e., the binary representation vector of c_p, is derived by setting all bins whose indexes are not smaller than l to 1, while the remaining bins are set to 0. Obviously, the number of 1s in the binary representation vector increases monotonously as the price cost drops. Similarly, ME can calculate the vector representation $\boldsymbol{v_t}$ for the time cost c_t of the current target transfer plan. As for the inputted embedding of other features, i.e., $\boldsymbol{e_f}$, it is processed by a monotonic MLP layer whose weights are all nonzero positive values, to get a new representation of other features that has the same dimensionality as $\boldsymbol{v_p}$ (or $\boldsymbol{v_t}$), denoted as $\boldsymbol{v_f} =< v_1, \ldots, v_{d_v} >$. After that, the derived $\boldsymbol{v_p}$ and $\boldsymbol{v_t}$ are separately perturbed to get their perturbation vectors $\tilde{\boldsymbol{v}}_{\boldsymbol{p}}$ and $\tilde{\boldsymbol{v}}_{\boldsymbol{t}}$, by randomly changing the locations of 1s in them. Then, CL executes the inner product between $\boldsymbol{v_f}$ and $\tilde{\boldsymbol{v}}_{\boldsymbol{p}}$ (or $\tilde{\boldsymbol{v}}_{\boldsymbol{t}}$), and derives the output real number o_p (or o_t). It is easy to understand that o_p (or o_t) rises monotonously as the price cost (or the time cost) declines. This is because all values in $\boldsymbol{e_f}$ are nonzero and positive. Under such circumstances, a lower price cost (or time cost) indicates a larger amount of 1s in $\tilde{\boldsymbol{v}}_{\boldsymbol{p}}$ (or $\tilde{\boldsymbol{v}}_{\boldsymbol{t}}$), which makes the result of inner production, i.e., o_p (or o_t), become larger. Hence, the o_p and o_t outputted by CL respond to the intuitions '*cheaper is better*' and '*less time is better*', respectively.

4.5 Reference Transfer Plan Learning

Reference transfer plans each of which contains a list of successive train trips manually pieced together by some user at the OTP, reflect crowd wisdom of platform users and thus offer critical information to evaluate the quality of target transfer plans. The reference transfer plan learning component (RTPL) in PlanRanker is designed to improve the predicted ranks of target transfer plans by the information of reference transfer plans. To achieve this goal, RTPL applies the contrastive learning, i.e., a promising paradigm of unsupervised learning. In specific, contrastive learning maps the data of both target transfer plans and reference transfer plans into a feature space wherein the similarities of positive pairs are maximized while those of negative pairs are minimized [9]. The implementation details of the two modules in RTPL, namely instance-level contrastive learning (I-CL) and cluster-level contrastive learning (C-CL), are provided in the following subsections.

4.5.1 Instance-level contrastive learning. In the I-CL module, a pair is set to be composed by two transfer plans, to enable the contrastive learning between them. Two plans in a positive pair satisfy the same OD query, whereas two plans in a negative pair satisfy different OD queries. Formally, given a size of mini-batch M, RTPL executes two types of data augmentations on every sample of target transfer plan (i.e., $\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}} \in \mathbb{R}^d$) and reference transfer plan (i.e, $\boldsymbol{e}_{\boldsymbol{r_j}}^{\boldsymbol{q_j}} \in \mathbb{R}^d$), and derives $2M$ data samples that are represented by a feature matrix $\mathcal{F} = [\boldsymbol{e}_{\boldsymbol{t_1}}^{\boldsymbol{q_1}}; ..., \boldsymbol{e}_{\boldsymbol{t_M}}^{\boldsymbol{q_M}}; \boldsymbol{e}_{\boldsymbol{r_1}}^{\boldsymbol{q_1}}; ...; \boldsymbol{e}_{\boldsymbol{r_M}}^{\boldsymbol{q_M}}]^T \in \mathbb{R}^{2M \times d}$. Note that an $\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}} (1 \le i \le M)$ and an $\boldsymbol{e}_{\boldsymbol{r_j}}^{\boldsymbol{q_j}} (1 \le j \le M)$ are computed by the same embedding strategy and thus they are in the same feature space. For a certain $\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}}$, there are totally $2M - 1$ pairs, amongst which we select a sample of reference transfer plan $\boldsymbol{e}_{\boldsymbol{r_i}}^{\boldsymbol{q_i}}$ that satisfies the same OD query q_i as $\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}}$ does to generate a positive pair $\{\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}}, \boldsymbol{e}_{\boldsymbol{r_i}}^{\boldsymbol{q_i}}\}$, and leave other $2M - 2$ pairs to be negative.

To lighten the information loss induced by the contrastive loss, the contrastive learning is not directly performed on the feature matrix $\mathcal{F}$. Instead, a two-layer non-linear MLP is employed, denoted by $G_I(\cdot)$, to map every sample to a subspace via $\boldsymbol{z}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}} = G_I(\boldsymbol{e}_{\boldsymbol{t_i}}^{\boldsymbol{q_i}})$ (or $\boldsymbol{z}_{\boldsymbol{r_j}}^{\boldsymbol{q_j}} = G_I(\boldsymbol{e}_{\boldsymbol{r_j}}^{\boldsymbol{q_j}})$), where the instance-level contrastive loss is applied. Then, the similarity between two instances in a pair is computed by cosine distance:

$$s(z_{a_i}^{q_i}, z_{b_j}^{q_j}) = \frac{z_{a_i}^{q_i} \cdot [z_{b_j}^{q_j}]^T}{\|z_{a_i}^{q_i}\| \|z_{b_j}^{q_j}\|}, \tag{3}$$

where $a, b \in \{t, r\}$ and $i, j \in [1, M]$. To optimize the pair-wise similarities, without loss of generality, the loss for a given instance of target transfer plan (i.e., $\boldsymbol{e}_{t_i}^{q_i}$) is calculated by:

$$l_{t_i}^{q_i} = -log \frac{exp\left(s(z_{t_i}^{q_i}, z_{r_i}^{q_i})/\tau_I\right)}{\sum_{j=1}^{M}\left[exp\left(s(z_{t_i}^{q_i}, z_{t_j}^{q_j})/\tau_I\right) + exp\left(s(z_{t_i}^{q_i}, z_{r_j}^{q_j})/\tau_I\right)\right]}, \tag{4}$$

where τ_I is the instance-level temperature parameter. For all positive pairs being identified in the dataset, the instance-level contrastive loss L_i is computed over each augmented sample, namely,

$$L_i = \frac{1}{2M}\sum_{i=1}^{M}\left(l_{t_i}^{q_i} + l_{r_i}^{q_i}\right). \tag{5}$$

4.5.2 Cluster-level contrastive learning. This C-CL module proposes to compute the rank of a target transfer plan by considering the label of cluster that it belongs to.

Formally, let the matrix $\boldsymbol{Y^t} \in \mathbb{R}^{M \times K^*}$ be the output result of C-CL for a mini-batch $\mathcal{F} = [\boldsymbol{e}_{t_1}^{q_1}; ..., \boldsymbol{e}_{t_M}^{q_M}; \boldsymbol{e}_{r_1}^{q_1}; ...; \boldsymbol{e}_{r_M}^{q_M}]^T \in \mathbb{R}^{2M \times d}$ w.r.t. the samples of target transfer plans in it (and the matrix $\boldsymbol{Y^r} \in \mathbb{R}^{M \times K^*}$ for the samples of reference transfer plans in it), where M is the bath size and K^* is the amount of clusters. Then, $Y_{i,j}^t$ is the probability of sample i being allocated to the cluster j. Apparently, each sample belongs to only one cluster, and thus all rows in $\boldsymbol{Y^t}$ are one-hot ideally. Under such circumstances, the i^{th} column in $\boldsymbol{Y^t}$ is deemed to be a representation of the i^{th} cluster and all columns should be different from each other.

Similar to the presented $G_I(\cdot)$ in I-CL (see Section 4.5.1), we employ another two-layer MLP named $G_C(\cdot)$ to project the feature matrix $\mathcal{F}$ into a K^*-dimensional space by $\tilde{\boldsymbol{y}}_{t_i}^{q_i} = G_C(\boldsymbol{e}_{t_i}^{q_i})$ (or $\tilde{\boldsymbol{y}}_{r_j}^{q_j} = G_C(\boldsymbol{e}_{r_j}^{q_j})$), where $\tilde{\boldsymbol{y}}_{t_i}^{q_i}$, i.e., the i^{th} row of $\boldsymbol{Y^t}$ (or $\tilde{\boldsymbol{y}}_{r_j}^{q_j}$, i.e., the j^{th} row of $\boldsymbol{Y^r}$) are the soft labels of samples $\boldsymbol{e}_{t_i}^{q_i}$ (or $\boldsymbol{e}_{r_j}^{q_j}$). Here, letting $\boldsymbol{y}_{t_i}^{q_i}$ be the i^{th} column in $\boldsymbol{Y^t}$, it is the representation of cluster i in $\boldsymbol{Y^t}$. Then, we combine $\boldsymbol{y}_{t_i}^{q_i}$ with $\boldsymbol{y}_{r_i}^{q_i}$ (i.e., the representation of cluster i in $\boldsymbol{Y^r}$) to generate a positive cluster pair $\{\boldsymbol{y}_{t_i}^{q_i}, \boldsymbol{y}_{r_i}^{q_i}\}$, while leaving the remaining $2K^* - 2$ pairs to be negative. Again, the cosine distance is utilized to quantify the similarity between cluster pairs, i.e.,

$$s(\boldsymbol{y}_{a_i}^{q_i}, \boldsymbol{y}_{b_j}^{q_j}) = \frac{[\boldsymbol{y}_{a_i}^{q_i}]^T \cdot \boldsymbol{y}_{b_j}^{q_j}}{\|\boldsymbol{y}_{a_i}^{q_i}\| \|\boldsymbol{y}_{b_j}^{q_j}\|}, \tag{6}$$

where $a, b \in \{t, r\}$ and $i, j \in [1, M]$. Without loss of generality, the loss function below is applied to distinguish cluster $\boldsymbol{y}_{t_i}^{q_i}$ from all other clusters except $\boldsymbol{y}_{r_i}^{q_i}$.

$$\hat{l}_{t_i}^{q_i} = -log \frac{exp\left(s(\boldsymbol{y}_{t_i}^{q_i}, \boldsymbol{y}_{r_i}^{q_i})/\tau_C\right)}{\sum_{j=1}^{K^*}\left[exp\left(s(\boldsymbol{y}_{t_i}^{q_i}, \boldsymbol{y}_{t_j}^{q_j})/\tau_C\right) + exp\left(s(\boldsymbol{y}_{t_i}^{q_i}, \boldsymbol{y}_{r_j}^{q_j})/\tau_C\right)\right]}. \tag{7}$$

τ_C is the cluster-level temperature parameter. By traversing all clusters, the cluster-level contrastive learning loss is computed as:

$$L_c = \frac{1}{2K^*}\sum_{i=1}^{K^*}\left(\hat{l}_{t_i}^{q_i} + \hat{l}_{r_i}^{q_i}\right) - H(Y), \tag{8}$$

where $H(Y) = \sum_{i=1}^{K^*}\left[Pr\left(\boldsymbol{y}_{t_i}^{q_i}\right)\log Pr\left(\boldsymbol{y}_{t_i}^{q_i}\right) + Pr\left(\boldsymbol{y}_{r_i}^{q_i}\right)\log Pr\left(\boldsymbol{y}_{r_i}^{q_i}\right)\right]$ is the entropy of cluster allocation probabilities $Pr(\boldsymbol{y}_{t_i}^{q_i}) = \sum_{j=1}^{M}(Y_{j,i}^k / \|Y\|_1), k \in \{t, r\}$ within a mini-batch. Notice that $H(Y)$ is used to avoid the trivial solution that most samples are allocated to the same cluster.

4.6 Training and Serving

All parts of the modules in PlanRanker are trained jointly by back-propagation. In the training phase, we use the train transfer plans purchased by the users as the labels and minimize the following loss function:

$$L = \alpha_1 L_{pc} + \alpha_2 L_i + \alpha_3 L_c, \tag{9}$$

where $\alpha_1, \alpha_2, \alpha_3$ are learnable parameters satisfying $\alpha_1 + \alpha_2 + \alpha_3 = 1$. In Equation 9, L_i and L_c are the loss values of the instance-level contrastive learning module and the cluster-level contrastive learning module in the RTPL component of PlanRanker respectively, and L_{pc} is the computed loss w.r.t. the PL component and CL component in PlanRanker. L_{pc} is computed as:

$$L_{pc} = -\frac{1}{|S|}\sum_{T^* \in S}\left(I(T^*)\log Pr(T^*) + (1 - I(T^*))\log(1 - Pr(T^*))\right), \tag{10}$$

where $|S|$ is the cardinality of the training dataset S, T^* is a target transfer plan in S, and $Pr(T^*)$ represents the probability (or score) of T^* being purchased by the user which is jointly predicted by the PL and CL components in PlanRanker. An indicator variable I is used to label each target transfer plan. For example, $I(T^*) = 1$ represents that T^* is a transfer plan booked by the user, and 0 otherwise. To compute $Pr(T^*)$, Equation 11 below is applied.

$$Pr(T^*) = \mathbf{sigmoid}(\boldsymbol{v_{pr}}), \tag{11}$$

where $\boldsymbol{v_{pr}} = \beta_1 \boldsymbol{v_b} + \beta_2 \boldsymbol{v_p} + \beta_3 \boldsymbol{v_c}$ is the weighted sum of the outputs of the PL and CL components in PlanRanker, and β_1, β_2, and β_3 are automatically learned parameters.

In the serving phase, PlanRanker computes the probability of a given target transfer plan purchased by the user also by Equation 11. Train transfer plans corresponding with the top-k highest probabilities will be returned and recommended to the user.

5 EXPERIMENT

5.1 Experimental Setup

5.1.1 Datasets. There are no public datasets that contain the information of stations, vehicles, and users' log data for browsing and purchasing tickets, which are all indispensable for evaluating PlanRanker. Hence, two Fliggy productivity datasets, i.e., the train dataset and the flight dataset, are used in the experiment, all of which are released online[4]. In specific, besides the information of stations and vehicles, the train dataset (or flight dataset) contains user logs collected at the Fliggy platform from the date 16/08/2022

[4] https://github.com/PlanRanker/TEST

(or 01/10/2019) to the date 17/09/2022 (or 01/12/2019), while the logs of the date 18/09/2022 (or 02/12/2019) are used as the testing data and the remaining logs are used as the training data. Notice that only the transfer plans that are clicked and purchased by users are deemed as positive (+) samples, while others are treated as negative (-) samples. Statistics of the two datasets are listed in Table 1.

Table 1: Statistics of two datasets. (M-million)

Features	Train dataset		Flight dataset	
	Training	Testing	Training	Testing
# of samples	2,782M	89.1M	12.6M	0.41M
# of + samples	272M	8.0M	1.21M	0.04M
# of - samples	2,51M	74.5M	11.35M	0.37M
# of users	47M	1.8M	0.16M	0.005M
# of transfer plans	52M	17M	0.12M	0.06M
# of cities	348	343	113	109

5.1.2 Baselines. We compare PlanRanker with two categories of baselines for personalized recommendation: 1) the rule-based methods that are usually adopted by OTPs; 2) model-based methods.

- **Cheapest**: It ranks transfer plans by their price costs.
- **Shortest**: It ranks transfer plans by their time costs.
- **LR**[6]: It is a well-known logistic regression model which can be used to predict the ranks of different items.
- **GBDT**[7]: It is a scalable tree-based model for recommending or ranking tasks, which is usually used in industry.
- **Hydra**[18]: It is the state-of-the-art multi-modal route recommendation method using Trans2vec [10] and GBDT techniques.
- **MIND**[15]: It is a deep neural network that can effectively extract users' diverse interests for personalized recommendation.
- **PFRN**[11]: It is a personalized flight itinerary ranking method that never considers the condition of transfer itineraries.

Notice that Hydra, MIND, and PFRN are deep neural network based methods.

5.1.3 Other settings. We use standard metrics of Area Under Curve (AUC), Normalized Discounted Cumulative Gain (NDCG) [12], and recall to evaluate different methods. Note that NDCG is a ranking measure widely used in practice. The calculation of NDCG follows:

$$NDCG@k = \frac{DCG}{IDCG}, \qquad DCG = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{12}$$

where rel_i represents the relevance of the recommended transfer plan for the ranking location i, while k stands for the size of the recommendation list. IDCG denotes a list of the best recommended transfer plans returned for a user by PlanRanker (IDCG stands for the ideal DCG in order of ranking). NDCG is the normalized DCG, having a value between 0 and 1. The closer NDCG to 1, the better the recommendation effect. Besides NDCG, the larger the AUC or recall is, the better the recommended results outputted by a method.

Adam [14] is used as the optimizer with the learning rate of 0.001 for all methods and the batch size is set to 512. In addition, the truncation length of a user's historical behaviors considered in the experiment is 50. For the setting of the proposed PlanRanker, the number of interests set for the multi-interest extractor is 5, the number of bins used to derive the binary vector representation for price cost (or time cost) is set to 5, and the amount of clusters used by the cluster-level contrastive learning is 8, to ensure the PlanRanker can achieve its best performance.

5.2 Offline Evaluation

5.2.1 Comparison of all methods. By observing experimental results in Table 2, we achieve the following conclusions.

- The two rule-based methods are apparently beaten by all model-based methods on two datasets, which indicates ranking transfer plans purely based on their price costs or time costs is not an advantageous strategy.
- All deep learning solutions (i.e., Hydra, PFRN, MIND, and PlanRanker) consistently outperform other model-based methods (i.e, lR and GBDT) on all datasets, revealing the power of deep learning for improving the ranking results.
- The state-of-the-art route recommendation method Hydra gets satisfactory results compared with LR and GBDT, since Hydra learns embeddings of transfer plans via a heterogeneous transportation graph to enhance its recommendation performance.
- MIND shows next-best performance in the experiment. In particular, compared with Hydra, MIND gains averagely 2.70% increase in AUC, while on average 4.32% and 3.94% improvements in NDCG and recall respectively on two datasets, which indicates the importance of expressing distinct interests of users by multiple representation vectors rather than by a single representation vector in personalised recommendation.
- The proposed PlanRanker significantly outperforms all baselines. In particular, compared with the next-best solution MIND, PlanRanker averagely improves the AUC by 3.07% and raises the NDCG and recall by 6.84% and 3.51% respectively on two datasets, which is a big progress made by an industrial recommendation system. This is because PlanRanker emphasizes the importance of price cost (or time cost) of a transfer plan in the learning process and utilizes the reference transfer plans to enhance its prediction performance for plan ranks.
- As a personalized ranking solution, PFRN is sub-optimal in ranking transfer plans, since: 1) it is designed for ranking direct itineraries rather than transfer itineraries; 2) no optimization designed focusing on the two costs of an itinerary; 3) reference itineraries are ignored in its learning framework.
- Finally, Table 2 shows that PlanRanker performs better on Flight dataset. Particularly, PlanRanker outperforms PFRN, which is originally designed for personalized rankings of direct flight transfer plans, by 3.74% in AUC, while 8.88% and 4.61% in NDCG and recall respectively. This may be because: 1) users who book flights are more sensitive to price (or time) time cost of transfer plans than users who book trains, which increases the benefit of cost learning (CL) deployed in PlanRanker; 2) the flight transfer plans are generally less than the train transfer plans for a certain OD query, which makes users' choices more focus and thus improves the predicted rankings of transfer plans by PlanRanker.

A t-test is also conducted and its results show that the p-values under all evaluation metrics are all less than 0.05, which means PlanRanker performs statistically better than the best baseline MIND.

5.2.2 Ablation study. The ablation studies are carried out to further understand the importance of every optimization strategies proposed in PlanRanker. In specific, PlanRanker is compared with its five variants listed below:

- **PR-MIND**: The multi-interest extractor w.r.t. MIND [15] in the personalized learning (PL) component is eliminated from PlanRanker.
- **PR-PL**: The PL component is deleted from PlanRanker.
- **PR-CL**: The cost learning (CL) component is removed from PlanRanker.
- **PR-RTPL**: The preference target plan learning (RTPL) component is removed from PlanRanker.
- **PR-CL-RTPL**[6]: Both of the CL and RTPL components are deleted from PlanRanker.

The results of ablation study for the proposed PlanRanker on train dataset are listed in Table 3. As indicated by the table, each component (i.e., PL, CL, and RTPL) makes a considerable contribution to ensure the quality of the predicted ranks of transfer plans by PlanRanker. In particular, the performance gain brought by the PL component is as large as 5.68% in AUC, 9.77% in NDCG, and 6.75% in recall. This shows that learning based on users' historical preferences plays the most vital role in guaranteeing the quality of personalized rankings of train transfer plans. Meanwhile, we observe that without using the MIND strategy in the PL component, PlanRanker suffers from an averagely 5.1% decline w.r.t. all evaluation metrics, since MIND has been proven to be very effective in capturing varying nature of user's interests [15]. Table 3 also displays that the involvement of the CL component brings 4.1% increase in AUC, while 8.62% and 4.5% improvements in NDCG and recall respectively, since the price and time costs of every transfer plan are properly emphasized in the ranking process. Another observation from Table 3 is that the introduction of the RTPL component is also very helpful in ensuring the performance of PlanRanker, although the performance degradation of PlanRanker caused by deleting RTPL is less than that caused by removing PL or CL. Finally, we find that when both CL and RTPL are disable, PlanRanker suffers from a 5.15% decrease in AUC, and 9.34% and 6.04% declines in NDCG and recall separately, which again verifies the indispensability of emphasizing the costs of transfer plans and the learning based on reference transfer plans.

5.3 Online A/B Test

With the success in offline experiments, we deploy PlanRanker in daily environment at Fliggy. The best rule-based method, i.e., Shortest, the best model-based baseline, i.e., MIND, and PlanRanker are involved in the online experiment to cope with real traffic generated by 2, 000, 000 users at Fliggy for one week in May 2022. To ensure the fairness, the scheduling engine of Fliggy is revised to approximately assign $\frac{1}{3}$ daily traffics to each method. The following CTCVR (post-view click through and conversion rate) is used to evaluate different methods for serving online traffic.

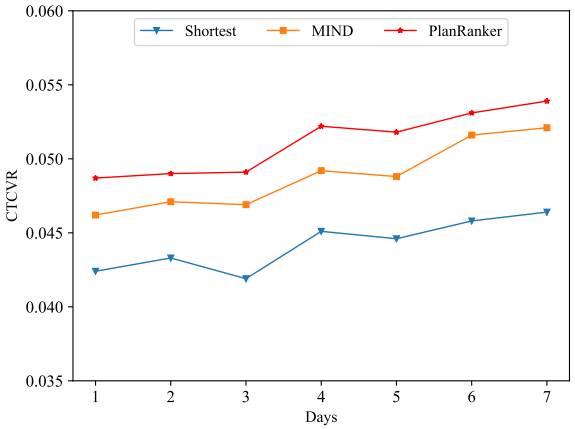


Figure 4: Online CTCVR of different methods at Fliggy from May 18, 2022 to May 24, 2022.

$$CTCVR = \frac{\text{The number of purchased transfer plans}}{\text{The number of impressed transfer plans}} \tag{13}$$

Experimental results are illustrated in Figure 4. Figure 4 shows that the proposed PlanRanker has a significant improvement at CTCVR. In particular, the CTCVR is on average improved by 15.60% compared with the shortest solution that only considers the time cost of each transfer plan. Meanwhile, we are delighted to see that PlanRanker gains on average 4.68% improvement at CTCVR compared with the best baseline MIND, due to the sufficient emphasizing on the costs of transfer plans and the effective learning based on reference transfer plans. This is a significant improvement made under the industrial application scenarios.

5.4 Deployment and Efficiency

5.4.1 Deployment of PlanRanker at Fliggy. PlanRanker has been deployed at Fliggy, one of the most popular online travel platforms (OTPs) in China. It has successfully optimized the train search ranking function at Fliggy and serves millions of users each day. Deployment of PlanRanker at Fliggy is illustrated in Figure 5. As shown by the figure, in the offline training stage, logs of user feedback are gathered which are then stored at the MaxCompute platform of Alibaba Group and the training dataset is built by processing the stored logs. After that, the PlanRanker model is trained in a distributed way using Tensorflow with 10 parameter servers and 50 workers deployed at Alibaba PAI (Platform of Artificial Intelligence) [5]. Notice that each parameter server owns 6 CPU cores with 32 GB RAM, being responsible for fetching part of training samples, computing gradients of parameters, and sending the gradients to parameter servers. Then, the trained PlanRanker model is uploaded to the real-time prediction (RTP) center. When a user initializes an OD query to search for train plans at Fliggy, the OD query is firstly parsed, after which the candidate train transfer plans, the user's profile together with historical behaviors are fetched from

[5]https://help.aliyun.com/product/30347.html

Table 2: Comparison of different methods on two datasets. (Best values are in bold; next-best values are underlined).

Categories	Methods	Train dataset			Flight dataset		
		AUC	NDCG@15	Recall@15	AUC	NDCG@15	Recall@15
Rule-based	Cheapest	0.612	0.553	0.679	0.632	0.579	0.696
	Shortest	0.643	0.565	0.690	0.671	0.586	0.715
Model-based	LR	0.664	0.570	0.721	0.683	0.593	0.741
	GBDT	0.701	0.613	0.772	0.723	0.634	0.794
	Hydra	0.710	0.621	0.783	0.736	0.642	0.803
	PFRN	0.726	0.636	0.806	0.747	0.657	0.827
	MIND	0.734	0.649	0.815	0.752	0.671	0.836
	PlanRanker	**0.757**	**0.696**	**0.844**	**0.776**	**0.721**	**0.867**
Improvement (w.r.t. next-best)		3.04%	6.75%	3.44%	3.09%	6.93%	3.58%
p-value (w.r.t. next-best)		0.0127	0.0134	0.0129	0.0125	0.0131	0.0125

Table 3: Ablation study on Train dataset (Best values are in bold; next-best values are underlined).

Methods	AUC	NDCG@15	Recall@15
PR-MIND	0.731	0.640	0.812
PR-PL	0.714	0.628	0.787
PR-CL	0.726	0.636	0.806
PR-RTPL	0.734	0.649	0.815
PR-CL-RTPL	0.718	0.631	0.793
PlanRanker	**0.757**	**0.696**	**0.844**

the feature center. According to the parsed OD query and all fetched data, the RTP center calculates the ranking score for each candidate train transfer plan based on the trained PlanRanker model. Finally, k train transfer plans with the highest ranking scores are ranked and exposed to the user in the Fliggy Mobile APP.

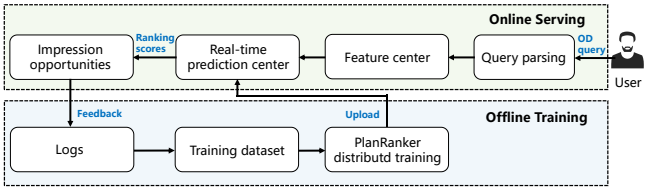


Figure 5: Illustration of deployment of PlanRanker at Fliggy.

5.4.2 Efficiency evaluation. Following the deployment strategy of PlanRanker, we deploy each comparison method at Fliggy. The training time and inference time of different methods are shown in Table 4. In Table 4 we can observe that there is no significant difference in training time of different model-based methods when using the large-scale cluster deployed at Fliggy. The deep neural networks based methods (i.e., Hydra, MIND, PFRN, and PlanRanker) get a little bit more training time compared with GBDT, due to their complexity. PlanRanker consumes 9 minutes more in training time than MIND, since MIND is deployed in the CL component of PlanRanker. It is also observed that Hydra uses less time in the training phase, since it focuses on route recommendation and only the user preferences and the situational context are considered in its learning architecture. Cheapest (or Shortest) is a rule-based method and thus does not need to be trained. Noting that the online inference time of all methods is within 0.03 second, which makes them meet the needs of industrial applications.

Table 4: Evaluation of efficiency of different methods on Train dataset (h-hour; m-minute; s-second; ms-millisecond).

Methods	Training time	Inference time
Cheapest	-	2ms
Shortest	-	2ms
GBDT	35m 18s	12ms
Hydra	52m 23s	21ms
MIND	1h 07m	28ms
PFRN	1h 23m	30ms
PlanRanker	1h 16m	29ms

6 CONCLUSION

Summary. In this paper, we present PlanRanker, a novel personalized deep network for train transfer plan ranking. In addition to learn users' personalized interests over their historical behaviors, PlanRanker sufficiently emphasizes the importance of price cost and time cost of a transfer plan for its rank and utilizes the idea of contrastive learning to capture vital information contained in reference transfer plans during the learning process. Experimental results on two production datasets shows the promising performance of PlanRanker in ranking train transfer plans.

Generalization. Though PlanRanker is proposed under the context of train transfer plan ranking, it can be easily generalized to address the personalized ranking problem of flights or regular routes, since both of these problems need to pay attention to the price and consumed time of a plan and the consideration of reference plans are equally important to improve their ranking results.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 62067001), the Innovation Project of Guangxi Graduate Education (No. JGY2021003). This work is partially supported by the Guangxi Natural Science Foundation (No. 2019JJA170045).

REFERENCES

[1] N. Borole, D. Rout, N. Goel, P. Vedagiri, and T. V. Mathew. 2013. Multimodal Public Transit Trip Planner with Real-time Transit Data. *Procedia - Social and Behavioral Sciences* (2013), 775–784.

[2] Paolo Campigotto, Christian Rudloff, Maximilian Leodolter, and Dietmar Bauer. 2016. Personalized and situation-aware multimodal route recommendations: the FAVOUR algorithm. *IEEE Transactions on Intelligent Transportation Systems* (2016), 92–102.

[3] Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016. Learning points and routes to recommend trajectories. In *Proceedings of CIKM'16*. 2227–2232.

[4] Lisi Chen, Shuo Shang, Christian S. Jensen, Bin Yao, Zhiwei Zhang, and Ling Shao. 2019. Effective and Efficient Reuse of Past Travel Behavior for Route Recommendation. In *Proceedings of KDD'19*. 488–498.

[5] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *Proceedings of ICDE'11*. 900–911.

[6] Jan Salomon Cramer. 2002. The origins of logistic regression. (2002).

[7] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[8] L. Fu, D. Sun, and L. R. Rilett. 2006. Heuristic shortest path algorithms for transportation applications: State of the art. *Computers & Operations Research* (2006), 3324–3343.

[9] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *In Proceedings of CVPR'06*, Vol. 2. IEEE, 1735–1742.

[10] L. Hao, T. Li, R. Hu, Y. Fu, and Hui Xiong. 2019. Joint representation learning for multi-modal transportation recommendation. In *Proceedings of AAAI'19*. 1036–1043.

[11] Jinhong Huang, Yang Li, Shan Sun, Bufeng Zhang, and Jin Huang. 2020. Personalized Flight Itinerary Ranking at Fliggy. In *Proceedings of CIKM'20*. 2615–2623.

[12] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *In Proceedings of SIGIR'00*.

[13] D. Jian, B. Yang, C. Guo, and Z. Ding. 2015. Personalized route recommendation using big trajectory data. In *Proceedings of ICDE'15*. 543–554.

[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[15] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of CIKM'19*. 2615–2623.

[16] Hao Liu, Jindong Han, Yanjie Fu, Jingbo Zhou, Xinjiang Lu, and Hui Xiong. 2020. Multi-modal transportation recommendation with unified route representation learning. *Proceedings of the VLDB Endowment* (2020), 342–350.

[17] H. Liu, Y. Tong, J. Han, P. Zhang, and H. Xiong. 2020. Incorporating Multi-Source Urban Data for Personalized and Context-Aware Multi-Modal Transportation Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[18] H. Liu, Y. Tong, P. Zhang, X. Lu, and H. Xiong. 2019. Hydra: A Personalized and Context-Aware Multi-Modal Transportation Recommendation System. In *Proceedings of KDD'19*. 2314–2324.

[19] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M Ni. 2013. Finding time period-based most frequent path in big trajectory data. In *Proceedings of SIGMOD'13*. 713–724.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS'13*. 3111–3119.

[21] people.cn. [n.d.]. The national railway completes 1.61 billion passenger trips in 2022. http://finance.people.com.cn/n1/2023/0103/c1004-32598895.html.

[22] Yuhua Qiu and Xiaolong Xu. 2018. RPSBPT: A route planning scheme with best profit for taxi. In *Proceedings of MSN'18*. 121–126.

[23] R. J. Szczerba and P. Galkowski. 2000. Robust algorithm for real-time route planning. *IEEE Transactions on Aerospace & Electronic Systems* (2000), 869–878.

[24] Y. Tong, J. She, B. Ding, L. Wang, and C. Lei. 2016. Online mobile micro-task allocation in spatial crowdsourcing. In *Proceedings of ICDE'16*. 49–60.

[25] Yongxin Tong, Libin Wang, Zimu Zhou, Lei Chen, Bowen Du, and Jieping Ye. 2018. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In *Proceedings of SIGMOD'18*. 773–788.

[26] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu. 2017. Flexible online task assignment in real-time spatial data. *Proceedings of the VLDB Endowment* (2017), 1334–1345.

[27] Y. Tong, Y. Zeng, Z. Zhou, C. Lei, and X. Ke. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* (2018), 1633–1646.

[28] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. 2019. Empowering A* Search Algorithms with Neural Networks for Personalized Route Recommendation. In *Proceedings of KDD'19*. 539–547.

[29] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of KDD'12*. 195–203.

[30] Z. Yu. 2015. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology* (2015), 1–41.

[31] Nicholas Jing Yuan, Z. Yu, C. Zhang, W. Xie, X. Xing, G. Sun, and H. Yan. 2010. T-drive: driving directions based on taxi trajectories. In *Proceedings of SIGSPATIAL'10*. 99–108.

[32] Chak Fai Yuen, Abhishek Pratap Singh, Sagar Goyal, Sayan Ranu, and Amitabha Bagchi. 2019. Beyond shortest paths: Route recommendations for ride-sharing. In *Proceedings of WWW'19*. 2258–2269.

SUPPLEMENTARY MATERIALS

A IMPACT OF PARAMETERS

Appendix A discusses the impact of different parameters on the performance of the proposed PlanRanker.

A.1 Impact of the Number of Interests

Figure 6 displays the impact of the number of interests K set for the multi-interest extractor deployed in the personalized learning component (PC) of PlanRanker. As shown in the figure, when K equals to 5, PlanRanker achieves its best performance in terms of both NDCG@25 and Recall@25, while introducing more interests beyond 5 reduces its performance. This possibly because 5 is an appropriate amount of interests for platform users that can be learned and extracted from their historical behavior logs. Therefore, we set K used in PlanRanker and its variants to 5 in the experiment.

A.2 Impact of the Number of Bins

Figure 7 displays the influence of the number of bins L on the effectiveness of PlanRanker. L is an important parameter used to partition the domain of price cost (or time cost), so that the binary representation vector of a transfer plan's price cost (or time cost) can be computed in the cost learning (CL) component of PlanRanker. As shown in Figure 7, when setting L to 5 PlanRanker achieves its best performance, since a smaller L is not enough to distinguish different levels of costs and a greater L cannot well summarize the costs by their common characteristics. Hence, $L = 5$ is used in all experiments for PlanRanker and its related variants.

A.3 Impact of the Number of Clusters

Figure 8 illustrates the impact of the number of clusters K^* on the performance of PlanRanker. K^* is used to set the amount of clusters concerned by the cluster-level contrastive learning module which is within the reference transfer plan learning component (RTPL) of PlanRanker. As demonstrated by the figure, increasing K^* from 2 to 8 improves the performance of PlanRanker, while increasing K^* beyond 8 bringing no remarkable benefit. This indicates that 8 is a reasonable number of clusters for distinguishing different patterns of transfer plans. Hence, K^* is set to 8 in PlanRanker and its relevant variants in all experiments.

B EVALUATION OF SERVING NEW USERS

A online A/B test is also conducted to evaluate the performance of the proposed PlanRanker model and the best baseline MIND in serving new users who do not own any historical behaviors in the system. In the test, the new users account for 15% of daily access workloads in our Fliggy mobile APP. According to the new online A/B test results, when handling new users, the CTCVR of PlanRanker is averagely 6.53% higher than that of the best baseline MIND for one week. This is because PlanRanker still can benefit from emphasizing the time cost (or price cost) of train transfer plans and considering the reference transfer plans (i.e., crowd wisdom) when ranking the train transfer plans for the new users. On the other hand, without historical behaviors of users, MIND generates very poor rankings of train transfer plans, since it highly relies on historical behaviors of users.

C FREQUENTLY-USED OF NOTATIONS

Table 5 summarizes the frequently-used notations and their descriptions.

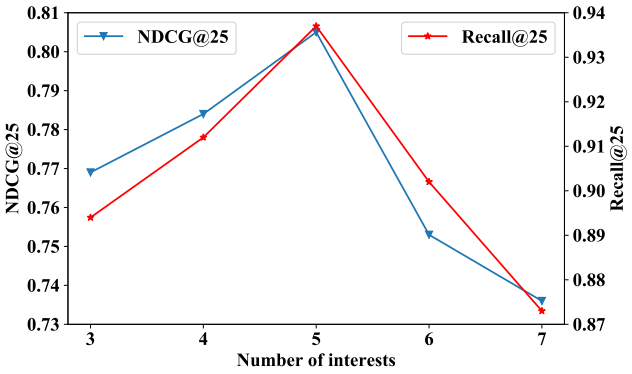


Figure 6: Impact of the number of interests.

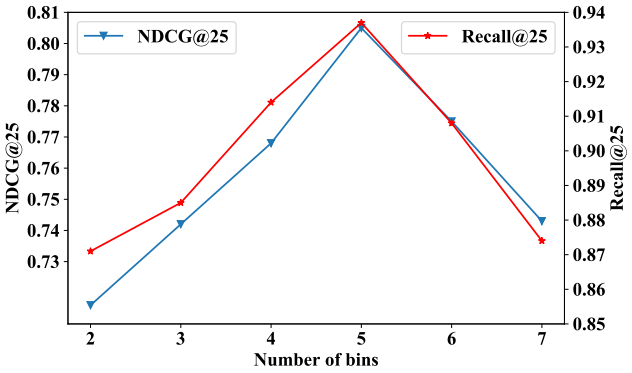


Figure 7: Impact of the number of bins.

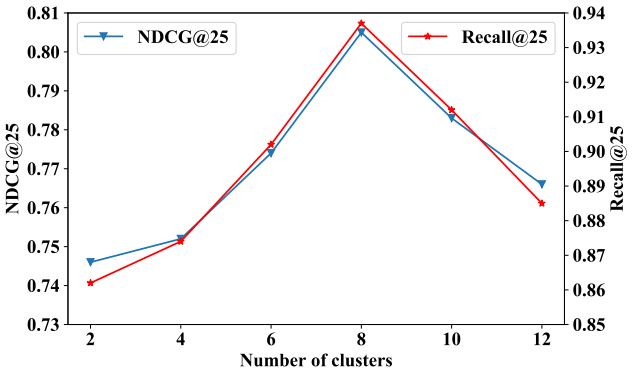


Figure 8: Impact of the number of clusters.

Table 5: Description of Frequently-used Notations

Notations	Descriptions
$\mathcal{U} = \{u_1, \ldots, u_{\lvert\mathcal{U}\rvert}\}$	A set of $\lvert\mathcal{U}\rvert$ users. u_i is the i^{th} user in the set.
$\mathcal{A} = \{a_1, \ldots, a_{\lvert\mathcal{A}\rvert}\}$	A set of $\lvert\mathcal{A}\rvert$ train numbers. a_j is the j^{th} train number in the set.
$\mathcal{L} = \{l_1, \ldots, l_{\lvert\mathcal{L}\rvert}\}$	A set of $\lvert\mathcal{L}\rvert$ railway stations. l_k is the k^{th} railway station in the set.
$q(o, d)$, $o, d \in \mathcal{L}$	An OD query. o is the original railway station and d is the destination railway station.
$p^q =< a, o, d, t_o, t_d, c >$	A train trip w.r.t. OD query $q(o, d)$. t_o is the departure time from o and t_d is the arrival time to d. The time cost and price cost of p^q are $(t_d - t_o)$ and c, respectively.
$P^q = \{p_1^q, \ldots, p_{\lvert P^q\rvert}^q\}$	A train transfer plan w.r.t. OD query $q(o, d)$. It is composed by a sequence of $\lvert P^q\rvert$ successive train trips.
$\mathcal{R}^q = \{R_1^q, \ldots, R_{\lvert\mathcal{R}^q\rvert}^q\}$	A reference transfer plan w.r.t. OD query $q(o, d)$. It is consists of a list of successive train trips manually pieced together by users at an OTP.
$\mathcal{T}^q = \{T_1^q, \ldots, T_{\lvert\mathcal{T}^q\rvert}^q\}$	A list of target transfer plans w.r.t. OD query $q(o, d)$.
$\mathcal{B}^u = \{b_1^u, \ldots, b_{\lvert\mathcal{B}\rvert}^u\}$	A sequence of historical behaviors (e.g., clicking or booking plans) of user u at an OTP.