International Journal of Parallel Programming (2020) 48:137–156
https://doi.org/10.1007/s10766-019-00649-8

Planning Above the API Clouds Before Flying Above the Clouds: A Real-Time Personalized Air Travel Planning Approach

Zelin Liu[1] · Jian Cao[1] · Yudong Tan[2] · Quanwu Xiao[2] · Mukesh Prasad[3]

Received: 9 August 2019 / Accepted: 4 November 2019 / Published online: 22 November 2019

Abstract
The rapid growth of the airline industry has resulted in the availability of a large number of flights, however this can also create a paralyzing problem. Flight information on all airlines across the world can be obtained via the Internet. Today, passengers trend to be interested in user personalized service. How to effectively find a passenger's most preferred air travel plan, which might include multiple transfers from millions of possible choices with certain constraints, such as time and price, is a critical challenge. This paper presents an efficient air travel planning approach, which can find a number of air travel plans by invoking the APIs offered by airline companies. At the same time, these plans also best match the customer's preference based on an analysis of historical orders. An algorithm to extract user preference features is introduced and heuristic rules to speed up the K path search process under constraints are presented. The experiment results show that the proposed model finds optimal air travel plans efficiently on a real-world dataset.

Keywords Air travel planning · Personalized recommendation · Network heuristic search

✉ Jian Cao
cao-jian@cs.sjtu.edu.cn

Zelin Liu
lzllzl2013051@sjtu.edu.cn

Yudong Tan
ydtan@Ctrip.com

Quanwu Xiao
qwxiao@ctrip.com

Mukesh Prasad
Mukesh.Prasad@uts.edu.au

1 Shanghai Jiao Tong University, Shanghai, China

2 Ctrip.com International, Ltd., Shanghai, China

3 University of Technology, Sydney, Australia

1 Introduction

The airline industry has developed very quickly in recent years. Passengers have to search for flights from a multiple of choices which can be difficult when there is no direct flight. Therefore, passengers need air travel planning services. Many online travel agencies (OTAs) and travel booking websites provide these services. However, developing an efficient trip planning service poses huge challenges:

1. *The search space is extremely large* There are thousands of flights between a pair of cities. If every possible route is taken into consideration, there will be many stop choices and the number of possible travel plans will increase exponentially.
2. *Information changes dynamically* The availability of airline tickets constantly changes and at the same time, airlines adjust their ticket prices dynamically relying on their yield management systems.
3. *Passengers have different preferences* Air travel plan preferences vary from person to person. For example, passengers with a higher price sensitivity prefer cheaper tickets and are willing to transfer several times for the sake of saving money. Although OTAs provide different filters to help passengers search for flights which meet their personal requirements, even if these filters are set correctly, often too many results are returned which makes the choice difficult.

Addressing the challenges associated with generating personalized air travel plans needs costly computations and also many invocations to access real-time ticket information. Currently, air travel planning services offered by OTAs or travel booking websites rely on several techniques to reduce computation and communication costs, such as filtering some flights in advance based on rules and using static information rather than real-time information. Moreover, most of them do not provide personalized air travel plans automatically.

The goal of this paper is to develop an efficient approach to find personalized air travel plans in real time. As shown in Fig. 1, airline companies provide up-to-date air ticket information through web APIs. These APIs from a large number of airline companies form an API cloud. In order to speed up the search process, a flight network is maintained between cities in advance. In this network, the nodes are cities and the edges are air tickets.The properties of a ticket include airline, flight class and price. We search for paths that satisfy the given constraints, such as ticket price and flight time, and which also match the users' preferences. Due to changes in the availability of tickets and prices from time to time during the search process, the edges need to be updated by invoking APIs.Since there are so many flights, the size of this network is very large and the search time together with the invocation time is huge if we use the original network. Therefore, we need to find heuristics to reduce the search space. The network search problem with multiple constraints has been studied for many years [1–3], however, there is a need to investigate how to find personalized air travel plans. Also, recommender systems have been widely studied and applied [4–6] but unfortunately, it is not possible to generate all plans in advance, which is a necessary condition for traditional recommender systems.

Therefore, this paper proposes an approach to address the personal air travel planning problem on a large-sized flight network. The contributions include:

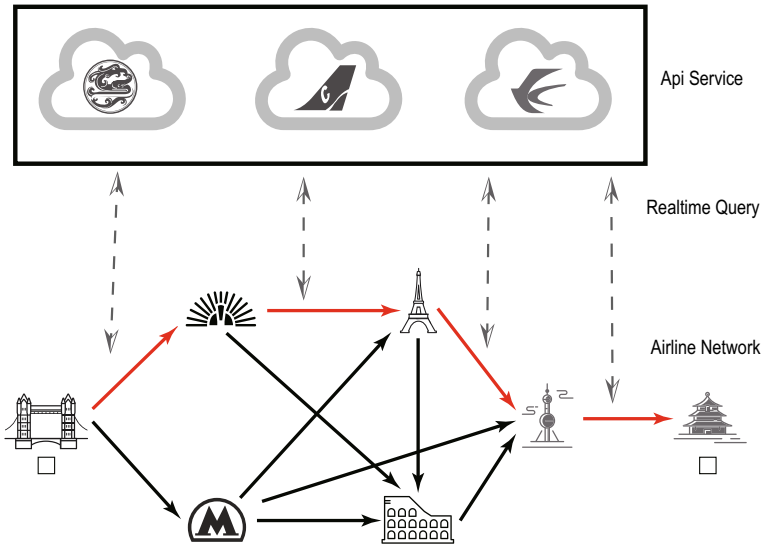


Fig. 1 The air travel planning problem

1. To learn a user's inherent preferences from their historical orders, a metric called UPS (User Preference Score) is proposed to quantify how much an air travel plan matches a user's preferences. Each path in the network has a UPS value, and the paths with the best UPS values are those that best meet the customer's needs.
2. A ticket price lowest bound estimation approach based on historical data is designed to reduce the search space.
3. A heuristic path searching algorithm called KBFP (k Best Flight travel Plan searching algorithm) is proposed, which can find the k paths with best UPS values under the given constraints, such as time and price, in flight networks.

The rest of the paper is structured as follows. Section 2 introduces the related work. In Sect. 3, the method to learn a user's preferences is introduced and a metric to measure how well a path satisfies the user's preference is proposed. Section 4 presents a heuristic algorithm to find the best k paths in a multiple constrained network and verify its rationality and efficiency by comparing it with other general algorithms. Section 5 provides a method to estimate the lowest bound of the ticket price for pruning the paths in the network. The experiment results to evaluate the performance of KBFP and its counterparts are introduced in Sect. 6 and finally, conclusions are drawn in Sect. 7.

2 Related Work

The network search problem with multiple constraints has been studied for many years [1–3]. As previously mentioned, the path searching and planning problem on flight networks can be regarded as such a problem. The Tabu Search, a heuristic search strategy employing local search methods for mathematical optimization, has been

applied to solve vehicle routing and scheduling problems with time window constraints [7–9]. Constraints are often introduced into flight travel planning problems so that they can be modeled as QoS-aware network problems. Heuristic algorithms have been designed to find paths with the minimum cost in a QoS-aware network with constraints [2,3,10]. Many researchers have also studied bio-inspired algorithms to solve QoS-aware network problems [11,12]. Other researchers focused on the constrained shortest path with uncertainties. The work in [13] describes an algorithm for the stochastic shortest path problem where path costs are a weighted sum of the expected cost and cost standard deviation. The work in [14] studies the robust optimization methodology and stochastic optimization techniques to deal with the constrained shortest path problem, in which the transit times are subject to uncertainty.

Research on personalized travel services has attracted much attention in recent years [4–6]. For example, a tourist-area-season topic model was introduced to find users' preferences [15], in which a model to extract the contents of the travel packages and the interests of the tourists was proposed. Context information and the footprints of users are used to model the interests of tourists [16,17]. Researchers have analyzed Flickr data to find the relations between users' preferences and locations [18]. Complex models, such as Bayesian networks, are used to extract user preferences. A Bayesian personalized ranking algorithm is used to find the probability distribution of each user's interest in each item according to the rankings [6,19]. A tensor factorization method is used to recommend flights according to the attributes learned from historical data in several domains [20]. The work in [21] also uses tensor factorization to solve point-of-interest recommendation problems. Several metrics have been introduced to evaluate the performance of recommendation systems [22,23]. Furthermore, some researches has been applied to the airline industry. The work in [24] established a heterogeneous information network (HIN) to analyze the travel behaviors and patterns of air passengers. The work in [25] proposes a way in which techniques such as collaborative filtering, content based filtering, etc can be hybridized to generate recommendations for the airline industry. The Average Rank Score (ARS) represents the average rankings of the items on the recommendation list which is given by the recommendation system. A smaller ARS value indicates a better system. The coverage rate shows how often the recommended items are actually chosen.

Although path searching algorithms and personalized travel service models have been studied, they have not yet been integrated together. The method proposed in this paper efficiently searches personalized paths using the UPS value to prune the search space. The proposed method meets personal requirements while simultaneously ensuring search efficiency.

3 User Preference Modeling

This section introduces the features selected to represent a user's preferences for flight travel plans. Then, an algorithm to calculate the users' preferences based on collaborative filtering is introduced since there is often not enough data on many users. Finally, a metric called UPS is proposed to measure how well an air travel plan fits a given user's preference model.

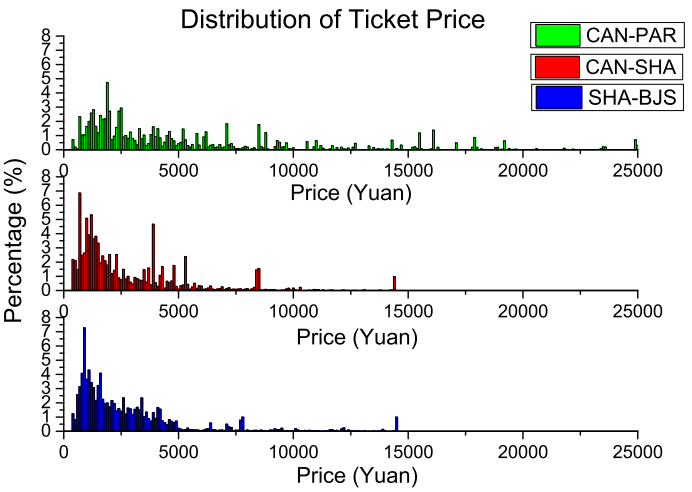


Fig. 2 Distribution of ticket price

3.1 Feature Selection

According to the historical orders and other studies [26], three features are most important in measuring a person's preference for a flight:

1. *Airlines* represents the user's favorite airlines and can be denoted by a vector in which each dimension represents one airline and the value represents the degree to which the user prefers this airline.
2. *Flight class* represents a user's preferred flight class. In our experiments, class is divided into economy, business and commercial.

where p_{high} and p_{low} represent the highest and the lowest price for the same flight, respectively. p denotes the price of the ticket the user selects.

3. *Price* represents a user's price sensitivity. Prices vary on different flights which means this cannot indicate a person's preference. Here we introduce a metric called price-sensitivity to show to what extent a person prefers cheaper tickets. For a given pair of departure and arrival cities, price-sensitivity is calculated as follows:

$$sen = \frac{p_{high} - p}{p_{high} - p_{low}} \tag{1}$$

Figure 2 shows the distributions of ticket price while Fig. 3 shows the distributions of price-sensitivities of users on selected air lines. It can be observed from Figs. 2 and 3 that the distribution of price sensitivity on different airlines is similar, which implies price sensitivity is a better model of users' price preferences compared with the original ticket price.

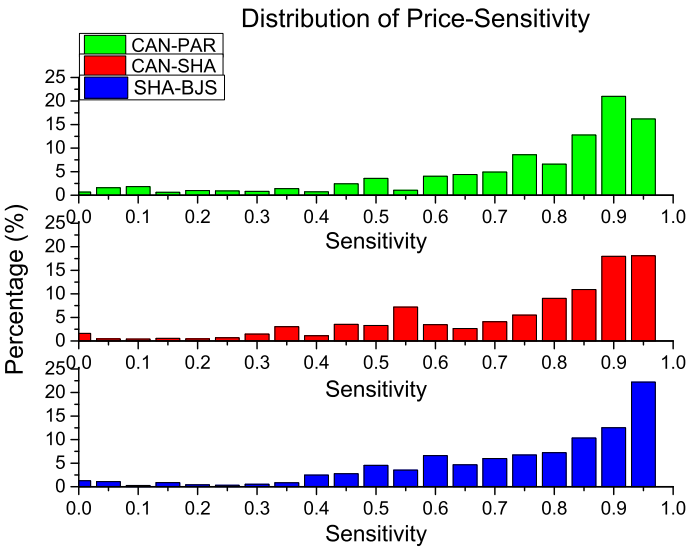


Fig. 3 Distribution of ticket price sensitivity

3.2 Preference Representation

In this section we discuss how to model a user's preference from their historical orders.

We define $Prf_u =< A_u, C_u, p_u >$ as the preference representation of user u. The airline preference is represented as:

$$A_u = [a_1, a_2, a_3, \ldots, a_n] \quad s.t. \quad 0 \leq a_i, \sum_{i=1}^{n} a_i = 1 \tag{2}$$

where n equals the number of different airlines appearing in the historical orders. Class preference is defined as:

$$C_u = [c_1, c_2, c_3, \ldots, c_m] \quad s.t. \quad 0 \leq v_i, \sum_{i=1}^{m} c_i = 1 \tag{3}$$

where m equals the number of different classes appearing in the historical orders. Price preference p_u is a real number between 0 and 1.

We denote each order as $Ord_o =< A_o, C_o, p_o >$. A_o is an n dimensional vector indicating the airline information in this order and when the ith airline is responsible for this order, then the ith element is 1 otherwise it is 0. C_o is an m dimensional vector indicating class information in this order. When the jth class ticket is selected in this order, then the jth element is 1 otherwise it is 0. p_o is sensitivity to price which is calculated using Eq. 1. We denote O_u as the order set which was previously placed by user u. To model user u's preferences from the historical orders, we calculate their preferences as follows:

$$
\begin{aligned}
Prf_u &=< A_u, C_u, p_u > \\
s.t. \quad A_u &= \frac{\sum_{o \in O_u} A_o}{|O_u|} \\
C_u &= \frac{\sum_{o \in O_u} C_o}{|O_u|} \\
p_u &= \frac{\sum_{o \in O_u} p_o}{|O_u|}
\end{aligned} \tag{4}
$$

Two problems persist during preference modeling:

1. *Cold start problem* This is a common problem in recommender systems. If a user has very few or even no historical orders, this user's preferences cannot be modeled in a reliable way.
2. *Sparsity problem* This occurs when the users' historical orders do not cover all the options, or even most options. For example, the orders show a user prefers certain airlines, but this doesn't mean that he has no interest in those airlines which he hasn't used.

We take advantage of the idea of collaborative filtering to solve these two problems. For each user, we find other users with a high similarity to him, and we migrate these users' preference information to the current user. For user u_x and u_y, we define their similarity as:

$$
Sim(u_x, u_y) = \frac{(A_{u_x} \cdot A_{u_y} + C_{u_x} \cdot C_{u_y} + (1 - |p_{u_x} - p_{u_x}|))}{3} \tag{5}
$$

For a given user u_x, we first find a set of users with the highest similarity. Then we denote these users as set U_{u_x} and the feature vector of user u_x can be calculated as:

$$
\begin{aligned}
A'_{u_x} &= \frac{\sum_{y \in U_{u_x}} Sim(u_x, u_y) \cdot A_{u_y}}{\sum_{y \in U_{u_x}} Sim(u_x, u_y)} \\
C'_{u_x} &= \frac{\sum_{y \in U_{u_x}} Sim(u_x, u_y) \cdot C_{u_y}}{\sum_{y \in U_{u_x}} Sim(u_x, u_y)} \\
p'_{u_x} &= \frac{\sum_{y \in U_{u_x}} Sim(u_x, u_y) \cdot p_{u_y}}{\sum_{y \in U_{u_x}} Sim(u_x, u_y)}
\end{aligned} \tag{6}
$$

In order to verify the effectiveness of the adjustments of the preference information, the passengers are clustered into five groups according to their similarity. From Fig. 4, it can be seen that by using Eqs. (5) and (6), the similarities between users in the same group increase, and the variance decreases which means the sparsity problem is lightened to some degree.

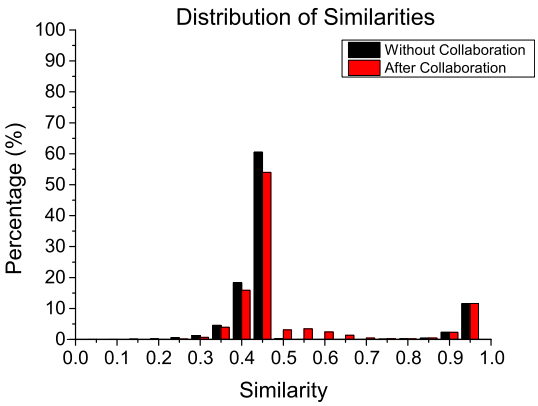


Fig. 4 The distribution of similarities of users in the same group

3.3 User Preference Score

In order to find an air travel plan which fits a user's preferences most, we propose a metric called User Preference Score (UPS) to measure the degree to which a user's preferences are satisfied.

An air travel plan is denoted as p and each transfer is an edge in this path. Therefore, p consists of a set of sequentially connected edges. We denote $e(w, v)$ as the edge between city w and v and it represents a direct flight ticket between these two cities. $A_{e(w,v)}$, $C_{e(w,v)}$, $p_{e(w,v)}$ are the airline feature, class feature and price feature of this ticket. The representations are the same as those of the orders. For user u_x, with his preference defined in the previous section, we define UPS as:

$$UPS(u_x, p) = \sum_{e(w,v)\in p} (\Delta(A_{u_x}, A_{e(w,v)}) + \Delta(C_{u_x}, C_{e(w,v)}) + |p_{u_x} - p_{e(w,v)}|) \quad (7)$$
$$where \quad \Delta(X, Y) = 1 - X \cdot Y$$

The UPS has two characteristics:

1. *Additive* This characteristic enables us to design the KBFP algorithm (detailed the next section).
2. *Monotonic* The smaller the UPS, the better the path. This characteristic enables us to find k paths with a minimum UPS score using the KBFP algorithm.

4 KBFP Algorithm

A flight network is dynamic as the attributes of the edges change from time to time. When a search starts, in a case where the current flight information is not known, an API invocation is needed to retrieve up-to-date information. For brevity, we omit the description of the API invocation to obtain flight information.

4.1 Problem Definition

Finding an air travel plan in a flight network is similar to the QoS (Quality of Service) routing problem. Finding the k shortest constrained paths in the network is a commonly discussed problem, defined as follows:

Suppose the entire network is represented by a graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, \ldots, v_n\}$ represents the set of vertexes, and $E = \{e(v_i, v_j)|v_i, v_j \in V\}$ represents all the edges in graph G.

The weight of each edge is a vector of R dimensions, representing multiple attributes. The rth attribute of the weight is denoted as:

$$W_r(e(v_i, v_j)), \quad r = 1, 2, \ldots, R \tag{8}$$

We denote the cost of an edge as:

$$W_0(e(v_i, v_j)) = f(W_1(e(v_i, v_j)), W_2(e(v_i, v_j)), \ldots, W_R(e(v_i, v_j)), \theta) \geq 0 \tag{9}$$

and f is a cost function where θ represents the possible parameters in f.

Starting from point v_s, a path to v_t consists of a set of sequentially connected edges and is denoted by $p(v_s, v_{s+1}, \ldots, v_t)$. Given two paths $p(v_a, v_{a+1}, \ldots, v_b)$ and $p(v_c, v_{c+1}, \ldots, v_d)$, suppose $v_a = v_c$ and $v_b = v_d$, if any two m-th edges from two paths are equal, then$p(v_a, v_{a+1}, \ldots, v_b) = p(v_c, v_{c+1}, \ldots, v_d)$. Two paths $p(v_a, v_{a+1}, \ldots, v_b)$ and $p(v_c, v_{c+1}, \ldots, v_d)$ can be connected if and only if $v_b = v_c$, and the result is also a path from v_a through v_b to v_d, which is denoted as $p(v_a, v_{a+1}, \ldots, v_c, v_{c+1}, \ldots, v_d) = p(v_a, v_{a+1}, \ldots, v_b) + p(v_c, v_{c+1}, \ldots, v_d)$.

The total cost and the attribute value of the total weight of a path can be calculated as follows:

$$W_r(p(v_x, v_{x+1}, \ldots, v_y)) = \sum_{e(v_i, v_j) \in p(v_x, v_{x+1}, \ldots, v_y)} W_r(e(v_i, v_j)) \quad r = 0, 1, 2, \ldots, R \tag{10}$$

For points v_i, v_j, v_s in G, suppose there exists three paths, i.e., $p(v_i, \ldots, v_s, \ldots v_j)$, $p(v_i, \ldots, v_s)$ and $p(v_s, \ldots, v_j)$. If $p(v_i, \ldots, v_s, \ldots, v_j) = p(v_i, \ldots, v_s) + p(v_s, \ldots, v_j)$, we have:

$$W_r(p(v_i, \ldots, v_s, \ldots, v_j)) = W_r(p(v_i, \ldots, v_s)) + W_r(p(v_s, \ldots, v_j)) \tag{11}$$

Now we can describe the problem in a more mathematical way.

Given a graph $G = (V, E)$, a starting point v_s and a termination point v_t, and the upper bound constraint C_r on the rth attribute of the weight ($r = 1, 2, \ldots, R$), we need to find k paths which satisfy:

$$W_r(p(v_s, \ldots, v_t)) \leq C_r, \quad r = 1, 2, \ldots, R \tag{12}$$

with minimum cost $W_0(p(v_s, \ldots, v_t))$.

4.2 Necessary Concepts

Before introducing KBFP, we first introduce the following concepts which are the key parts of the algorithm:

Feasible Path Set

For graph G, given a starting point v_s, an end point v_t, and a set of upper bound constraints on each attribute of the weight of all paths starting from v_s and ending at v_t, i.e., $C(v_s, v_t) = \{C_r(v_s, v_t)\}, r = 1, 2, \ldots, R$, the feasible path set is:

$$\begin{aligned}\mathbf{F}(v_s, v_t, C(v_s, v_t)) = \{p(v_s, \ldots, v_t) | W_r(p(v_s, \ldots, v_t)) \\ \leq C_r(v_s, v_t), r = 1, 2, 3, \ldots, R\}\end{aligned} \tag{13}$$

The Lowest Distance Bound

The lowest bound $L_r(v_s, v_t, C(v_s, v_t))$ is the minimum value of the r th attribute of all paths in the feasible path set and is defined as follows:

$$\begin{aligned}L_r(v_s, v_t, C(v_s, v_t)) = min W_r(p(v_s, \ldots, v_t)) \\ for\ \forall p(v_s, \ldots, v_t) \in \mathbf{F}(v_s, v_t, C(v_s, v_t)), r = 1, 2, \ldots, R\end{aligned} \tag{14}$$

Minimum Heuristic Distance

Given graph G, a path starts from v_s, goes through v_x and ends at v_t, and the constraint set $C(v_s, v_t)$, the minimum heuristic distance $H_r(p(v_s, \ldots, v_t))$ is defined as follows:

$$H_r(p(v_s, \ldots, v_t)) = W_r(p(v_s, \ldots, v_x)) + E_r(v_x, v_t, C'_r) \tag{15}$$

such that

$$\begin{aligned}C'_r(v_x, v_t) = C_r(v_s, v_t) - W_r(p(v_s, \ldots, v_x)) \\ E_r(v_x, v_t, C'_r(v_x, v_t)) \leq L_r(v_x, v_t, C'_r(v_x, v_t)) \\ r = 1, 2, 3, \ldots, R\end{aligned} \tag{16}$$

where $E_r(v_x, v_t, C'_r(v_x, v_t))$ is the estimated lowest bound of W_r of different paths from v_x to v_t.

In the search process, we start from the starting point, and add every possible edge into our search space. In fact, each time we add an edge, we can judge whether we can prune this branch by using the minimum heuristic distance. Since the definitions confirm that if the minimum heuristic distance is larger than constraint C_r, the paths cannot meet the constraints anymore. On the other hand, calculating the lowest distance bound is rather complex since we have to traverse all possible paths on all attributes of the weight to get the results. However, estimating $E_r(v_x, v_t, C_r(v_x, v_t))$ will be much easier as we only focus on one attribute of the weight at a time and fortunately, quite a few algorithms can be applied for this task.

Lemma 1 *Each path $p \in \boldsymbol{F}(v_s, v_t, C(v_s, v_t))$ will not be pruned by the minimum heuristic distance rule.*

Proof If there exists a path $p \in \mathbf{F}(v_s, v_t, C(v_s, v_t))$, and it is pruned by the minimum heuristic distance rule at point v_x, then we have $p(v_s, \ldots, v_t) = p(v_s, \ldots, v_x) + p(v_x, \ldots, v_t)$ and for at least one r, $H_r(p(v_s, \ldots, v_x)) > C_r(v_s, v_t)$.

This means $W_r(p(v_s, \ldots, v_x)) + E_r(v_x, v_t, C_r(v_s, v_t) - W_r(p(v_s, \ldots, v_x)) > C_r(v_s, v_t)$ so that $W_r(p(v_s, \ldots, v_x)) + W_r(p(v_x, \ldots, v_t)) > C_r(v_s, v_t)$.

This means path p cannot satisfy the constraints, which leads to a contradiction. □

4.3 Pseudo Code of KBFP

The pseudo-code of the KBFP algorithm is shown in Algorithm1 as follows:

Algorithm 1 $KBFP(G, v_s, v_t, C(v_s, v_t), K)$

```
n ⇐ 0, W_r(p(v_s, v_s)) = 0, Paths = p(v_s, v_s) is a heap, Result ⇐ ∅
Calculate the dijkstra for each point regarded as metrics D
while Paths ≠ ∅ and k ≤ K do
    p(v_s, ..., v_x) ⇐ heappop(Paths)
    if x = t then
        Result ⇐ Result.Add(p(v_s, ..., v_x))    k = k + 1
    else
        edges ⇐ {e(v_x, v_z)|v_z ∈ V and e(v_x, v_z) ∈ E}
        while edges ≠ ∅ do
            e(v_x, v_y) ⇐ edges.dequeue
            if v_y ∉ p(v_s, ..., v_x) then
                p(s, y) ⇐ p(v_s, ..., v_x) + e(v_x, v_y)
                for all r, W_r(p(v_s, ..., v_y)) = W_r(p(v_s, ..., v_x)) + W_r(p(v_x, ..., v_y))
                if for all r, H_r(p(v_s, ..., v_y)) ≤ C_r(v_s, v_t) then
                    Paths.heappush(p(v_s, ..., v_y))
                end if
            end if
        end while
    end if
end while
return Result
Paths is a minimum heap sorted by W_0(p(v_s, ..., v_x))
```

4.4 KBFP Algorithm Characteristics

Next, we analyze some characteristics of the KBFP algorithm:

1. *Completeness* This can be proven by *lemma* 1.
2. *Optimality* The K paths returned by the algorithm have a smaller cost W_o^p than any other path. Let's assume that path u and v are two feasible paths. If KBFP finds path u is better than path v, this means there are two paths in the heap, path u from v_s to v_t and path v from v_s to v_x. KBFP chooses path u because $W_0(u) < W_0(v)$. Meanwhile, path v may be expanded later which means the final cost of path v will be even larger. Thus, its optimality is proven.
3. *Algorithm complexity analysis* Suppose the KBFP algorithm generates k feasible paths, R is the number of constraints in the path search, M is the total number of

paths in the heap paths, H is the maximum number of edges in the k paths, and d is the maximum number of degrees in graph G. The space complexity of the algorithm is $O(M)$ and the time complexity is $O(dM(R + H + logM))$.

Now we discuss how to use the KBFP searching algorithm to solve the personalized flight recommendation problem by combining the UPS function. The flight network is also represented as $G = (V, E)$. Each edge $e(v_d, v_a)$ represents a direct flight between city v_d (departure city) and v_a (arrival city). The weight attributes for edge $e(v_d, v_a)$ include:

1. $W_1(e(v_d, v_a))$: flight-time
2. $W_2(e(v_d, v_a))$: ticket-price
3. $W_3(e(v_d, v_a))$: airline feature
4. $W_4(e(v_d, v_a))$: class feature
5. $W_5(e(v_d, v_a))$: price sensibility

The cost function is defined as:

$$\begin{aligned} W_0(e(v_d, v_a)) = \Delta(A_{u_x}, W_3(e(v_d, v_a)) \\ +\Delta(C_{u_x}, W_4(e(v_d, v_a)) + |p_{u_x}, W_5(e(v_d, v_a))| \end{aligned} \tag{17}$$

W_0 satisfies Eq. (9), and the KBFP algorithm retrieves k paths that satisfy Eq. (11). This means these k transfer plans satisfy the time and money constraints. Furthermore, these k paths have minimal $W_0(p(v_s, \ldots, v_t))$ such that:

$$\begin{aligned} W_0(p(v_s, \ldots, v_t)) = & \sum_{e(v_w, v_v) \in p} (\Delta(A_{u_x}, A_{e(v_w, v_v)}) \\ & + \Delta(C_{u_x}, C_{e(v_w, v_v)}) + |p_{u_x} - p_{e(v_w, v_v)}|) \\ = & UPS(u_x, e(v_w, v_v)) \end{aligned} \tag{18}$$

which implies that the k paths found have the minimal UPS values.

5 Estimation of the Lowest Bounds of the Ticket Prices

Using heuristic distance (Dijkstra distance) for pruning paths is the key component of the KBFP algorithm and can improve the search efficiency greatly. However, determining the ticket price heuristic distance is difficult since information on every ticket needs to be collected. Querying APIs to obtain all ticket prices is necessary if we want to calculate the Dijkstra distance. As mentioned in the introduction, querying APIs takes a huge amount of time so it is impossible to query all ticket information for every user's query. We need a more efficient method to calculate the heuristic distance in terms of the price constraints.

It is obvious that a given airline's air ticket price for travel between a pair of cities will not be lower than a value. The air travel planning service will be called repeatedly, which means we can estimate the lowest bound based on the latest ticket

price information. We propose a ticket price lowest bound estimation algorithm using the Poisson distribution with a confidence interval.

Suppose, for a given pair of cities and corresponding airlines, the ticket prices we have collected are $P = \{p_i | i = 1, 2, 3 \ldots n\}$. It is assumed the ticket prices follow the Poisson distribution $Pt(X) = \frac{\lambda^X e^{-\lambda}}{X!}$, as shown in Fig. 2.

We sample the data using a sampling method with a confidence interval. Suppose, N is the size of the data, σ is the standard deviation, d represents the absolute error bound and $1 - \alpha$ for the confidence interval. d and α are decided according to the requirements. The sampling data size n satisfies:

$$\frac{1}{n} = \frac{1}{N} + \frac{d^2}{Z_{\frac{\alpha}{2}2}\sigma^2} \tag{19}$$

After we sample enough data, we can use a maximum-likelihood estimation to estimate λ. The log likelihood probability is:

$$\begin{aligned} lnPt(P;\lambda) &= ln\prod_{i=1}^{n}\frac{\lambda^{p_i} * e^{-\lambda}}{p_i!} \\ &= \sum_{i=1}^{n} -\lambda + p_i ln\lambda - ln(p_i!) \\ \frac{\partial lnPt(P;\lambda)}{\partial\lambda} &= -n + \frac{\sum_{i=1}^{n} p_i}{\lambda} \end{aligned} \tag{20}$$

which implies:

$$\lambda = \frac{\sum_{i=1}^{n} p_i}{n} \tag{21}$$

After we calculate the distribution, we can obtain the lowest price bound estimation based on the percentage of this distribution.

6 Performance Evaluation

6.1 Dataset and Experiment Settings

We estimate the performance including search time, API query times, coverage and average ranking which are useful metrics, as previously mentioned. We use the flight data crawled on the Internet from 2017.01.01 to 2017.06.31 for the 50 most popular cities which includes information on a total of 21,076,167 air tickets. We also use another data set from an OTA that contains 8000 real orders over the same period. The reason why we selected the 50 most popular cities is given in the next section.

The constraints are defined as follows: flight time is no longer than 10 hours and total ticket price is not higher than 10,000 Yuan. We use the UPS function (KBFP-

UPS), total flight time (KBFP-Time) and total ticket price (KBFP-Cost) as the cost function respectively. This means that the algorithms will find the travel plans with the minimum UPS value, ticket price or total flight time. Heuristic distance in KBFP is calculated based on the estimated ticket lowest bound with a percentile of 10%. Each algorithm recommends 50 air travel plans at most for each query. In the experiment, each time the travel path is broadened, an API invocation is needed to retrieve the ticket information for the given pair of cities. To make the experiment more realistic, we invoke the real APIs from ctrip.com and the average query time is 0.33s.

6.2 Flight Network Optimization

The original flight network is incredibly large as there are thousands of flights and billions of paths. To represent and process this will be expensive in terms of time and memory space. However, many air travel plans are never accepted in practice. For example, if we fly from Shanghai to New York, we may transit in Peking, but we would never fly to Brazil and then to New York .

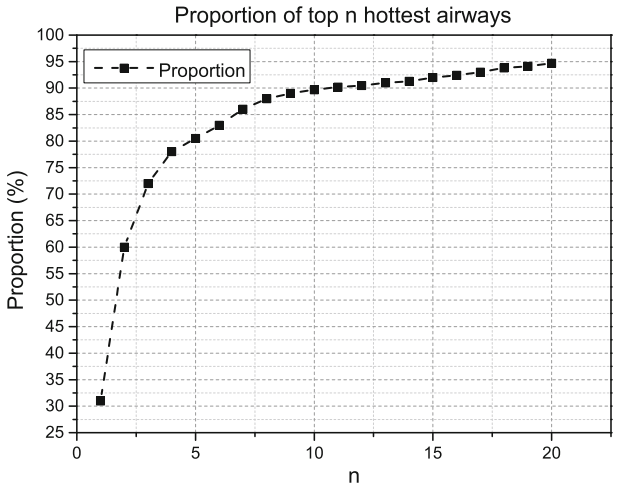
Therefore, under such an assumption, we can reduce the flight network as follows:

1. *Reduce the number of transit cities*
   As shown in Fig. 5a, in all the historical orders, there are 940 different departure and arrival city pairs and the top ten pairs with the highest frequency account for 89% of all flight plans, and the top 20 city pairs account for nearly 95%. Regarding these cities, the distribution of the occurrence of cities in orders is very similar to the distribution of the city pairs. The top ten most visited cities account for 93% of the total number of visits to all 175 cities while the top 20 account for nearly 97% as shown in Fig. 5b. As a result, in our experiment, we only select the most popular 50 cities and the flights between them.

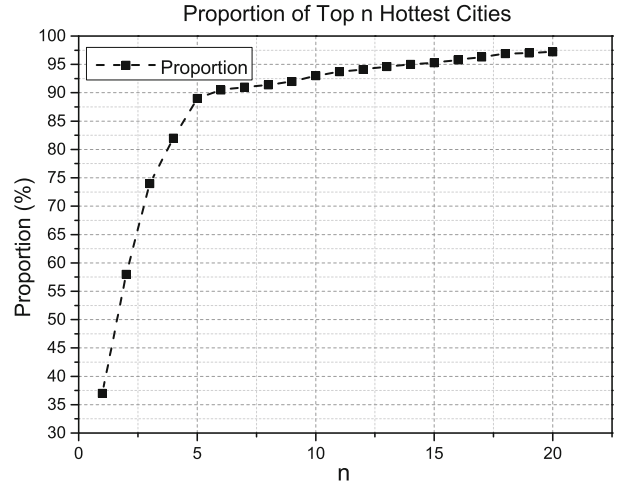
   To include unpopular cities in the network, one method is to attach them to nearby popular cities. This strategy is a common practice when we manually search the plan.
2. *Limit the times of transfer*
   On most occasions, travellers will not organize an air flight plan which has too many different flights.In our experiment, we limit the number of transits to less than 5. This conclusion is also supported by the historical orders.

6.3 Personalized Flight Travel Plan Search Performance

100 users' historical orders were selected randomly from the testing order dataset. For each user, a search query with a given departure and arrival time and the other settings is generated. Furthermore, the size of the network is changed indirectly by changing the time span between the departure and arrival time.



(a) Proportion of the top n city pairs in the users' historical orders



(b) Proportion of the top n cities in the users' historical orders

Fig. 5 Proportion of the top n cities and pairs in the users' historical orders

Figure 6 shows the search time needed for all algorithms which increases linearly, although the KBFP algorithms increase much more slowly than DFS. Figure 7 shows that KBFP with the UPS algorithm needs the least time to query the APIs.

Figures 8 and 9 show that KBFP-UPS has much better performance than the others in personalized travel plan recommendation in terms of coverage and ARS. The performance degradation speed of KBFP-UPS is almost the same as the others.

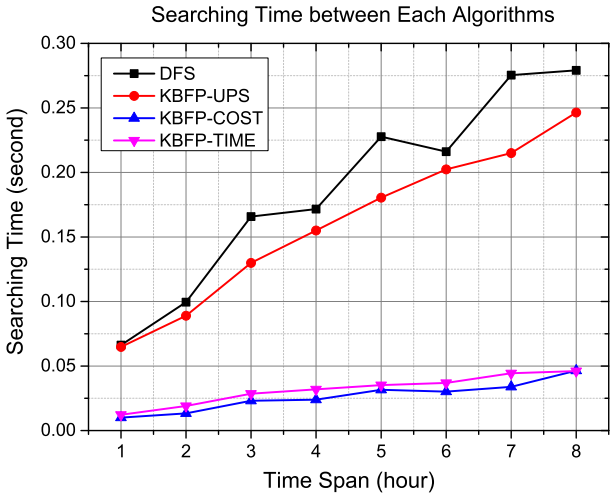


Fig. 6 Efficiency of the different algorithms

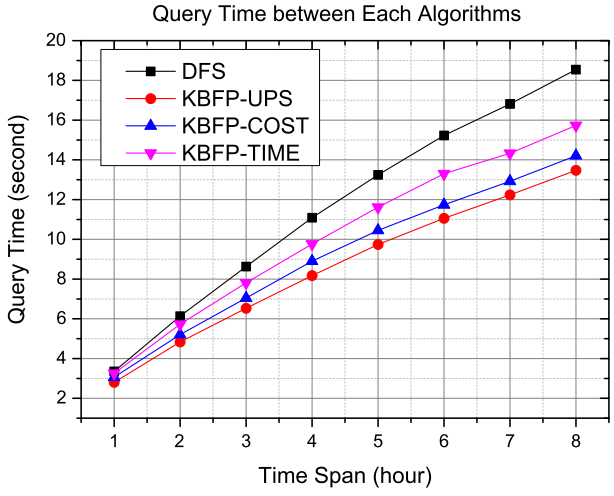


Fig. 7 API queries needed for the different algorithms

Table 1 shows the total computation cost of each algorithm during the different time spans in the personalized airline planning service. It can be observed that the KBFP-UPS is faster than the other algorithms.

6.4 The Performance of KBFP with the Lowest Bound Estimations

Experiments were conducted to verify the performance of KBFP with the lowest bound estimations for ticket prices. We choose $d = 100$ (error bound of the estimated price is 100 yuan in RMB) and $1 - \alpha = 0.95$ which implies $Z_{\frac{\alpha}{2}} = 1.96$. Table 2 shows the

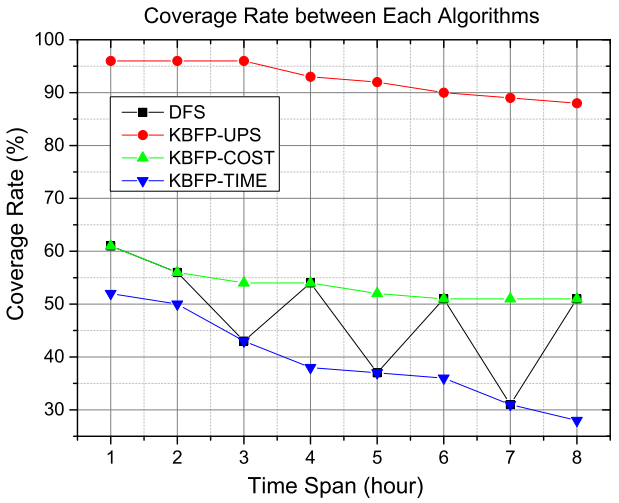


Fig. 8 Coverage for different algorithms

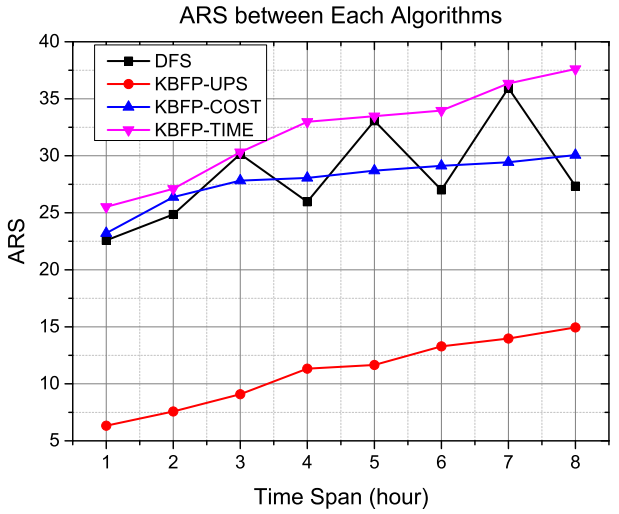


Fig. 9 Recommended path rankings for different algorithms

ARS and average API query time when using the UPS-KBFP algorithm whereas we use the actual lowest ticket price obtained from the dataset and the estimated lowest ticket price as the heuristic distance, respectively. It should be noted that in reality, we do not have the actual lowest price as it is impossible to maintain real-time price information on all tickets.

It can be observed from Table 2 that the estimation-based method has almost the same performance compared with the actual price-based one. This indicates the lowest bound estimated is very close to the actual one. Furthermore, ARS becomes worse and

Table 1 Total time cost of the planning service

Time Span(h)	Total time cost(s)			
	1	2	3	4
DFS(s)	3.42	6.23	8.80	11.26
KBFP-UPS(s)	2.87	4.92	6.66	8.33
KBFP-Time(s)	3.08	5.22	7.07	8.93
KBFP-Cost(s)	3.27	5.75	7.84	9.80
Time Span(h)	5	6	7	8
DFS(s)	13.47	15.44	17.09	18.82
KBFP-UPS(s)	9.92	11.26	12.45	13.72
KBFP-Time(s)	10.48	11.77	12.96	14.26
KBFP-Cost(s)	11.65	13.34	14.38	15.78

Table 2 KBFP Performance using the lowest bounds based on actual and estimated ticket prices

Span(h)	ARS			Query Time(s)		
	Actual	10%	30%	Actual	10%	30%
1	6.33	6.48	7.66	2.80	2.80	2.79
2	7.58	7.58	8.99	4.84	4.84	4.79
3	9.09	9.10	10.63	6.53	6.67	6.52
4	11.33	11.33	11.6	8.17	8.25	8.18
5	11.65	11.65	11.92	9.74	9.78	9.70
6	13.29	13.29	13.65	11.06	11.06	10.98
7	13.97	13.97	14.37	12.24	12.35	12.11
8	14.95	14.95	15.35	13.48	13.48	13.38

the query time becomes shorter when the percentile rises, which indicates the higher the lowest bound, the more paths that are pruned.

7 Conclusions

This paper proposes an approach which efficiently finds the optimal flight travel plans satisfying a user's preferences. The approach is composed of a user preference modeling phase and a heuristic search process. The heuristic algorithm named KBFP finds the paths with the minimum cost in a multiple constrained network and its efficiency is due to its powerful pruning process. The user preference modeling process provides a way to calculate user preferences from historical data. The UPS metric proposed in this paper is used in the KBFP algorithm and finds the best flight travel plans. The combination of the preference model and KBFP finds flight travel plans efficiently by pruning unnecessary branches. Furthermore, this paper proposes a method to predict the ticket price lower bound and significantly decreases the time spent on querying web services while maintaining performance. The proposed model outperforms the others on a real-world dataset.)

Acknowledgements This work is partially supported by National Key Research and Development Plan (No. 2018YFB1003800).

References

1. Al Nasr, K., Ranjan, D., Zubair, M., Chen, L., He, J.: Solving the secondary structure matching problem in cryo-EM de novo modeling using a constrained K-shortest path graph algorithm. IEEE/ACM Trans. Comput. Biol. Bioinform. **11**(2), 419–430 (2014)
2. Chu, C.H., Gu, J., Hou, X.D., Gu, Q.: A heuristic ant algorithm for solving QoS multicast routing problem. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), vol. 2, pp. 1630–1635. IEEE (2002)
3. Wang, H., Lu, X., Zhang, X., Wang, Q., Deng, Y.: A bio-inspired method for the constrained shortest path problem. Sci. World J. (2014). https://doi.org/10.1155/2014/271280
4. Cheng, A.J., Chen, Y.Y., Huang, Y.T., Hsu, W.H., Liao, H.Y.M.: Personalized travel recommendation by mining people attributes from community-contributed photos. In: Proceedings of the 19th ACM International Conference on Multimedia, pp. 83–92. ACM (2011)
5. Yang, P., Zhang, T., Wang, L.: TSRS: trip service recommended system based on summarized co-location patterns. In: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, pp. 451–455. Springer, Cham (2018)
6. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
7. Archetti, C., Speranza, M.G., Hertz, A.: A tabu search algorithm for the split delivery vehicle routing problem. Transp. Sci. **40**(1), 64–73 (2006)
8. Escobar, J.W., Linfati, R., Toth, P., Baldoquin, M.G.: A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. J. Heuristics **20**(5), 483–509 (2014)
9. Wassan, N.A., Simeonova, L., Salhi, S., Nagy, G.: A Reactive Tabu Search for the Fleet Size and Mix Vehicle Routing Problem with Backhauls (2015)
10. Liu, G., Ramakrishnan, K.G.: A* Prune: an algorithm for finding K shortest paths subject to multiple constraints. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), vol. 2, pp. 743–749. IEEE (2001)
11. Lee, C.J., Jung, J.Y., Lee, J.R.: Bio-inspired distributed transmission power control considering QoS fairness in wireless body area sensor networks. Sensors **17**(10), 2344 (2017)
12. Dorigo, M., St*ü*tzle, T.: Ant colony optimization: overview and recent advances. In: Handbook of Metaheuristics, pp. 311–351. Springer, Cham (2019)
13. Shahabi, M., Unnikrishnan, A., Boyles, S.D.: An outer approximation algorithm for the robust shortest path problem. Transp. Res Part E Logist. Transp. Rev. **58**, 52–66 (2013)
14. Mokarami, S., Hashemi, S.M.: Constrained shortest path with uncertain transit times. J. Global Optim. **63**(1), 149–163 (2015)
15. Liu, Q., Ge, Y., Li, Z., Chen, E., Xiong, H.: Personalized travel package recommendation. In: 2011 IEEE 11th International Conference on Data Mining, pp. 407–416. IEEE (2011)
16. Majid, A., Chen, L., Chen, G., Mirza, H.T., Hussain, I., Woodward, J.: A context-aware personalized travel recommendation system based on geotagged social media data mining. Int. J. Geogr. Inf. Sci. **27**(4), 662–684 (2013)
17. Liu, Q., Chen, E., Xiong, H., Ge, Y., Li, Z., Wu, X.: A cocktail approach for travel package recommendation. IEEE Trans. Knowl. Data Eng. **26**(2), 278–293 (2012)
18. Jiang, S., Qian, X., Mei, T., Fu, Y.: Personalized travel sequence recommendation on multi-source big social media. IEEE Trans. Big Data **2**(1), 43–56 (2016)
19. Huang, Y., Bian, L.: A Bayesian network and analytic hierarchy process based personalized recommendations for tourist attractions over the Internet. Expert Syst. Appl. **36**(1), 933–943 (2009)
20. Cao, J., Xu, Y., Ou, H., Tan, Y., Xiao, Q.: PFS: a personalized flight recommendation service via cross-domain triadic factorization. In: 2018 IEEE International Conference on Web Services (ICWS), pp. 249–256. IEEE (2018)

21. Yao, L., Sheng, Q.Z., Qin, Y., Wang, X., Shemshadi, A., He, Q.: Context-aware point-of-interest recommendation using tensor factorization with social regularization. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1007–1010. ACM (2015)
22. Zhou, T., Ren, J., Medo, M., Zhang, Y.C.: Bipartite network projection and personal recommendation. Phys. Rev. E **76**(4), 046115 (2007)
23. Lü, L., Liu, W.: Information filtering via preferential diffusion. Phys. Rev. E **83**(6), 066119 (2011)
24. He, Z., Liu, J., Xu, G., Huang, Y.: Heterogeneous item recommendation for the air travel industry. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 407–419. Springer, Cham (2019)
25. Bahulikar, S., Upadhye, V., Patil, T., Kulkarni, B., Patil, D.: Airline recommendations using a hybrid and location based approach. In: 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 972–977. IEEE (2017)
26. Cao, J., Yang, F., Xu, Y., Tan, Y., Xiao, Q.: Personalized flight recommendations via paired choice modeling. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1265–1270. IEEE (2017)