Personalized Trip Recommendation with Multiple Constraints by Mining User Check-in Behaviors

Eric Hsueh-Chan Lu
Dept. of Computer Science and Information Engineering
National Taitung University
No.684, Sec. 1, Zhonghua Rd., Taitung City, Taitung County 950, Taiwan (R.O.C.)
luhc@nttu.edu.tw

Ching-Yu Chen and Vincent S. Tseng*
Dept. of Computer Science and Information Engineering
National Cheng Kung University
No.1, University Rd., Tainan City 701, Taiwan (R.O.C.)
*Correspondence: tsengsm@mail.ncku.edu.tw

ABSTRACT
In recent years, researches on travel recommendation have attracted extensive attentions due to the wide applications and one of the active topics is constraint-based trip recommendation for meeting user’s personal requirements. Although a number of studies on this topic have been proposed in literatures, most of them only regard the user-specific constraints as some filtering conditions for planning the trip. In fact, immersing the constraints into travel recommendation systems to provide a personalized trip is desired for users. Furthermore, the time complexity of trip planning from a set of attractions is sensitive to the scalability of travel regions. Hence, how to reduce the computational cost by parallel cloud computing techniques is also a critical issue. In this paper, we propose a novel framework named Personalized Trip Recommendation (PTR) to efficiently recommend the personalized trips meeting multiple constraints of users by mining user’s check-in behaviors. In PTR, an automatic module is first proposed to estimate the scores of attractions by considering both of user-based preferences and temporal-based properties. Then, a trip planning algorithm named Parallel Trip-Mine+ is proposed to efficiently plan the trip which satisfies multiple user-specific constraints. To our best knowledge, this is the first work on travel recommendation that considers the issues of multiple constraints, social relationship, temporal property and parallel computing simultaneously. Through comprehensive experimental evaluations on a real check-in dataset obtained from Gowalla, the proposed PTR is shown to deliver excellent performance.

Categories and Subject Descriptors
I.2.6 [Artificial Intelligence]: Learning; J.4 [Computer Applications]: Social and Behavior Sciences

General Terms
Algorithms, Performance, Design, Experimentation, Verification.

Keywords
Trip Planning, Recommendation, Multiple Constraints, Location-Based Social Network, Data Mining.

1. INTRODUCTION
Traveling is one of the most important entertainments in a modern society. Traditionally, before traveling to an unfamiliar city, one of possible ways of trip planning is that tourists may ask travel agencies to schedule a trip or directly buy a tour package. For example, the trip planned by travel agencies usually includes some famous attractions such as Metropolitan Museum of Art and Times Square when the targeted city is New York. However, such popular trip may not be satisfied by everyone. With the advances of intelligent mobile devices and Web 2.0 techniques, many kinds of applications on web services and Location-Based Services (LBSs) [11] such as Gowalla [4], FourSquare and Facebook have been developed. Based on these approaches, users can easily record and share their daily lives and travel experiences via their mobile devices. Hence, another possible way of trip planning is to search the travel information from websites and plan the trip. This information benefits tourists to add or remove some attractions provided by travel agencies for planning a personalized trip. Take a scenario as an example. Suppose that Tom is a tourist and he wants to travel to New York. He may not know which attractions are worth to visit because this is the first time he comes to New York. He may search for the attractions in New York by the travel guide websites such as Lonely Planet and Yahoo Travel and schedule a travel trip by the trip planning websites such as HomeckAbroad. However, the whole procedure needs to spend much time to plan a personalized trip since the amount of travel information is very huge. Besides, it is hard to value whether the attraction is worth to visit or not. Although Tom can additionally search some travel blogs and check the comments about the attractions, it takes more time to search the information and to put collected information together for trip planning.

As mentioned above, although the travel information can be conveniently obtained from various sources, it is hard to distinguish that which attractions are suitable to visit while the travel constraints specified by tourists are different. Following the same scenario described above, suppose that Tom has only 8 hours and 100 dollars to travel New York. The intuitive idea is to choose the interesting attractions and arrange them to the most interesting trip in a proper order that satisfies the constraints of time and budget. However, such idea is inefficient since there are thousands of attractions in large cities such as New York. Besides, arranging the attractions as the most interesting trip meeting multiple constraints needs to consider many of real aspects such as attraction relations and temporal properties. Take New York as an example. The opening time of Metropolitan Museum of Art is only in the morning and afternoon. Thus it is incorrect to arrange this attraction in the evening. On the contrary, Times Square is more suitable to be arranged in the evening. Therefore, it is essential to develop an efficient and personalized travel recommendation system which can automatically recommend the suitable attractions at suitable time for tourists with the most interesting trip plan that satisfies the multiple constraints of tourists.

In this work, we aim to address two main issues: 1) How to evaluate the interesting degrees of each attraction; and 2) How to efficiently plan attractions as a trip based on the multiple user-specific constraints? For the first issue, a number of studies [1][2][5][6][8][14] are proposed to discuss about attraction recommendation. However, most of them only gather information provided by travel experts and utilize an inference model to make recommendation. In the Web 2.0 generation, the intelligence from social media such as Location-Based Social Networks (LBSNs) should be further considered for recommendation. For the second issue, there are also many studies [9][10][15][16][17] focus on
trip planning. Nevertheless, few of them consider the topics of multi-constraints trip planning and the computational efficiency, simultaneously. With the advances of the cloud computing architectures and parallel computing techniques, MapReduce benefits to divide a huge task into many small jobs and assign these jobs to large number of computers to conquer simultaneously. Therefore, the efficiency of trip planning can be significantly improved by many kinds of multi-core programming APIs such as OpenMP, Java Thread and CUDA.

To provide an efficient and personalized travel recommendation system with multi-constraints, we have to deal with the following four challenges: 1) How to discover attractions in an unfamiliar region? A tourist must be unfamiliar with the nearby attractions if the tourist has never been here. Therefore, we need to design an approach to automatically gather data and distinguish which locations are actually attractions; 2) How to evaluate the interesting score of an attraction based on user preference? The score of an attraction can be evaluated by using check-in logs and attraction-related information from Gowalla. The personalized score of an attraction can be evaluated by using check-in logs and attraction-related information; and 3) How to efficiently plan a trip in real time? Tourists do not endure that the response time over a few seconds when they query the travel recommendation system. However, there are thousands of attractions in a large city such as New York. The computational complexity of trip planning may be very high. Therefore, how to improve the efficiency of trip planning is a critical issue in such a real-time query system.

In this paper, we propose a novel framework, named Personalized Trip Recommendation (PTR), to efficiently plan a personalized trip with multiple constraints for users. PTR consists of two major modules: 1) Attraction Scoring Module for evaluating the personalized score of each attraction by considering both of user-based preferences and temporal-based properties. We collect the related data including user check-in behaviors and attraction-related information from Gowalla. The personalized score of an attraction can be evaluated by using check-in logs and attraction information; and 2) Parallel Multi-Constrains Trip Planning Module for efficient planning of the trip based on multiple user-specific constraints such as travel time constraint, budget constraint, etc. In this module, an efficient algorithm named Trip-Mine$^*$ is proposed to plan the optimal trip meeting these constraints from the top-k attractions evaluated by the first module. To improve the computational efficiency of trip planning, Trip-Mine$^*$ is extended as Parallel Trip-Mine$^*$ by integrating parallel computing techniques. Furthermore, we also consider the diversity of a trip to recommend the more varied trip for users. In this work, we extract valuable attraction information from LBSNs as knowledge bases to support an efficient and personalized travel recommendation system that considers multi-constraints at the same time. The key contributions of this paper are four-fold:

- We propose the Personalized Trip Recommendation (PTR) framework, a new approach for trip planning which considers multi-constraints, user preferences and temporal properties, simultaneously. The problems and ideas have not been well explored in the research community.
- We propose an attraction scoring module for automatically estimating the interesting score of attraction by considering user-based preferences and temporal-based properties.
- We propose a novel algorithm named Parallel Trip-Mine$^*$ for efficiently planning the trip with multi-constraints. Besides, we also consider the diversity of a trip to recommend the more interesting trip for users.
- We conduct a series of experiments based on a real data crawled from Gowalla to evaluate the performance of PTR. The results show superior performance of PTR in terms of planned effectiveness and efficiency.

The remainder of this paper is organized as follows. We briefly review the related work in Section 2. In Section 3, we formulate the problem. In Section 4, our proposed framework will be presented in details. In Section 5, we perform an empirical performance evaluation. Finally, in Section 6, we summarize our conclusions and future work.

2. Related Work

In this section, we briefly classify the relevant previous studies into three categories: 1) Location-based social network (LBSN); 2) Travel attraction recommendation; and 3) Travel trip planning.

Location-Based Social Network (LBSN). In recent years, more and more famous LBSN approaches have been developed, such as Gowalla [4], FourSquare and Facebook, that allow users to share their location-related information and real-life experiences in anywhere at any time. One of the main functions on LBSN services is “check-in”. Users can check-in their current locations, which are also called spots, by handheld mobile devices. The check-in records contain the name of spot, the time of check-in and the category of spot. For example, John checks-in the spot of Central Park in New York which belongs to “City Park”. The check-in records are also propagated and showed on the pages of their friends. Therefore, the real-life experiences can be easily shared and discussed in the virtual world. We target Gowalla as our research LBSN source.

Travel Attraction Recommendation. Before planning a trip, we should know which attractions are popular or interesting and what time is suitable for visiting. In general, this task can be viewed as the problem of attraction recommendation. There are a number of studies that focus on location recommendation. Albowd et al. presented the Cyberguide [1] project which is a mobile context-aware tour guide. This work is designed by the contextual information, such as the current location of user and history of past locations. In [6], Huang et al. proposed an intelligent system to provide personalized recommendations of attractions by using Bayesian network techniques and the Analytic Hierarchy Process (AHP) method in an unfamiliar city. This work considered the travel behaviors from not only of the individual user but also other related users. In [8], Kim et al. designed a system, called TripTip, to recommend users the next place by considering the similarity between previous visiting places and the next place. In TripTip, the similarity is measured by the features of user-given place tags and the reasons for visiting the place. In [5], Horozov et al. proposed an enhanced Collaborative Filtering (CF) method to provide more reasonable and useful recommended results based on the current locations of users. However, most of them rely on information defined by domain experts while this kind of
information is rare on the world. Besides, information defined by domain expert may be easily out-of-date. Some of studies discussed the issue of location recommendation based on the user-generated data from LBSNS and geo-tag photos. In [14], Ye et al. collected the data from FourSquare and further recommend locations for users by the user-based CF method and geographical factors. In [2], Clements et al. estimated the similarity among landmarks by geo-tag photos. This work built a location similarity model to make personalized recommendation. However, the above studies do not consider the temporal property which is an important factor in real applications.

**Travel Trip Planning.** An automatic travel recommendation system should recommend not only the interesting attractions but also the travel trips. Lee et al. proposed an ontological and multi-agent based travel recommendation system [9] in Tainan city. This system selects the top three of attractions and the top five of local restaurants by fuzzy logics and plans the optimal trip from these attractions by the ant colony optimization algorithm. However, it is restricted that the number of attractions in the recommended trip is eight. Zheng et al. proposed a HITS-based inference model [16] to evaluate the degrees of user experiences and location interests based on GPS trajectory data. This study considered that interesting places might be accessed by many greater travel experts and experienced tourists would visit many interesting places. The HITS-based inference model is compared with two baseline methods, namely Rank-By-Count (RBC) and Rank-By-Frequency (RBF). Both of these two methods ranked locations according to the popularities of locations. They further discovered the classical travel sequences among locations. However, GPS trajectory data is comparatively difficult to obtain and still not readily available. In [15], Yoon et al. evaluated the quality of a trip by four additional metrics, i.e., Elapse Time Ratio (ETR), Stay Time Ratio (STR), Interest Density Ratio (IDR) and Classical Travel Sequence Ratio (CTSR). All of the candidate trips are ranked by Euclidean distance and CTSR to recommend for users. In [10], Lu et al. planned the optimal trip by a dynamic programming method from the Panoramio geo-tag photos. However, this work did not consider the efficiency of trip planning. In [17], Zhou et al. also used the geo-tag photos to extract the scene features and scene topics of attractions. After computing the matching scores between attractions features and user profile, three approximate methods d-LOA, v-LOA and GOA are proposed to plan the trip. However, the planning method could not ensure to find the global optimal trip. Lu et al. proposed the Trip-Mine [12] algorithm for efficient planning of the optimal trip with highest score under a user-specific travel time constraint. They also designed three optimization mechanisms including Attraction Sorting, Lower Bound Checking, and Score Checking to further improve the mining efficiency. However, the computation time still exponentially increases by the number of attractions increases since this is still an NP complete problem.

**3. Problem Statement**

In this section, we first define some terms used in this work and then specify our research goal. To clear explain the following definitions, Figure 1 is taken as an example.

**Definition 1. Attraction.** \( A = \{a_1, a_2, \ldots, a_{|A|}\} \) and \( C = \{c_1, c_2, \ldots, c_{|C|}\} \) denote the collection of attractions and categories, respectively. For each attraction \( a \in A, a \) has an attraction Category \( CAT(a) \), an Attraction Score \( AS(a, t, a) \), an Attraction Cost \( AC(a) \) and a Stay Time \( STG(a) \), which represent the category of attraction \( a \), how interesting this attraction is for user \( u \) at time \( t \), how much money users often spend in attraction \( a \) and how long users often stay in attraction \( a \), respectively.

**Figure 1. An example of a trip map network.**

Definition 2. Check-In. Let \( ci = (u, t, a) \) be a check-in record which means user \( ci.u \) checks-in attraction \( ci.a \) at time \( ci.t \). \( D = \{ci_1, ci_2, \ldots, ci_{|D|}\} \) denotes a check-in database that contains \( |D| \) check-in records.

**Definition 3. Route.** \( R = \{r_1, r_2, \ldots, r_{|R|}\} \) denotes the collection of routes. For each \( r \in R, r = (a_s, a_e) \), where \( a_s, a_e \in A \) represent two attractions and \( i \neq j \). In this paper, the measurement of each route \( r \) is defined as the Travel Time \( TT(a_s, a_e) \). We assume that \( TT(a_s, a_e) = TT(a_s, a_e) \) and the travel time is approximately estimated by using the average travel speed in routine profile.

**Definition 4. Travel Map.** \( M = (A, R) \) denotes the map of travel region. Take Figure 1 as an example. There is one current location \( a_s \), four attractions, i.e., \( a_1 \) to \( a_4 \), and 10 routes in the trip map network. The departure time of user at \( a_s \) is 8:00. For the attraction \( a_1 \), the stay time is 30, i.e., \( ST(a_1) = 30 \). The travel time of route between \( a_1 \) and \( a_2 \) is 30, i.e., \( TT(a_1, a_2) = 30 \).

**Definition 5. Trip.** A trip \( tp = <a_1, a_2, \ldots, a_{|k|}> \), also denoted as n-trip, which is orderly composed of one or several attractions, where \( n \) indicates the number of attractions (i.e., \( |tp| = n \)). Let \( AT(a_{k_s}) \) be the Arrived Time of \( a_{k_s} \), which is defined as (1), where \( x \geq 1, \forall a_{k_s} \in A, <a_1, a_{|k|} > \) is one of trips in Figure 1.

\[
AT(a_{k_s}) = AT(a_{k_s}) + ST(a_{k_s}) + TT(a_{k_s}, a_{k_s})
\]

**Definition 6. Trip Score.** Given a user \( u \), a start time \( t \) and a trip \( tp = <a_1, a_2, \ldots, a_{|k|}> \), the Trip Score \( TPS(u, t, tp) \) is defined as (2), which represents how interesting this trip is. In Figure 1, suppose that the departure time for user \( u \) from "a_s" is 8:00. Hence, \( AT(a_{|k|}) \) is 8:30 and \( ST(a_{|k|}) \) is 9:30. Assume that \( AS(u, 8:30, a_1) \) is 0.4 and \( AS(u, 9:30, a_2) \) is 0.3. Hence, the trip score is \( TPS(<a_1, a_2>) = 0.4 + 0.3 = 0.7 \).

\[
TPS(u, t, tp) = \sum_{i=1}^{n} AS(u, AT(a_i), a_i)
\]

**Definition 7. Trip Time.** Let \( a_s \) be the start location of the user. For a trip \( tp = <a_1, a_2, \ldots, a_{|k|}> \), the Trip Time \( TPT(a_s, tp) \) is formulated as (3), which represents how long the users travel around this trip. For example, \( TPT(a_s, <a_1, a_2>) = TT(a_s, a_1) + TT(a_1, a_2) + ST(a_2) = 140 \).

\[
TPT(a_s, tp) = TT(a_s, a_1) + TT(a_1, a_2) + \sum_{j=1}^{k-1} TT(a_j, a_{j+1}) + \sum_{i=1}^{n} ST(a_i)
\]

**Definition 8. Trip Cost.** Let \( tp = <a_1, a_2, \ldots, a_{|k|}> \) be a trip, the Trip Cost \( TPC(tp) \) is formulated as (4), which represents how
much money the users need to spend around this trip. For example, \( TPC(a_1, a_2) = A(C_u) + A(C_t) = 20 \).

\[
TPC(tp) = \sum_{a_k} AC(a_k) \tag{4}
\]

**Definition 9. Travel Time and Budget Constraint.** The constraints of travel time and budget are defined as how much time and money the user has in this trip, respectively.

**Definition 10. Valid Trip.** Let \( a_u \), \( c_1 \), and \( c_2 \) be the current location of the user, the user-specific travel time constraint and the user-specific budget constraint, respectively. A trip \( tp = \langle a_1, a_2, \ldots \rangle \) is called a Valid Trip if \( TPT(a_u, tp) \) is less than or equal to \( c_1 \) and \( TPC(tp) \) is less than or equal to \( c_2 \); otherwise, it is an Invalid Trip. In Figure 1, the current location is \( a_u \), the travel time constraint \( c_1 \) is 210 and the travel budget \( c_2 \) is 50. A trip \( tp = \langle a_1, a_2 \rangle \) is a valid trip since \( TPT(a_u, tp) = 140 \leq c_1 \) and \( TPC(tp) = 20 \leq c_2 \). However, a trip \( tp' = \langle a_5, a_2 \rangle \) is not a valid trip since \( TPT(a_u, tp') = 215 \) which is greater than \( c_1 \).

**Definition 11. Optimal Trip.** Let \( a_u \) and \( t \) be the current location of the user \( u \) and the departure time of \( u \), respectively. A valid trip \( tp = \langle a_1, a_2, \ldots, a_n \rangle \) is called the Optimal Trip if there is no valid trip \( tp' \) such that \( TPS(u, t, tp) \) is greater than \( TPS(u, t, tp) \). Notes that there may be not only one optimal trip in some constraint settings. In Figure 1, there are several valid trips in the trip map network, e.g., \( \langle a_1, a_2, a_1, a_2, a_1, a_2, a_2 \rangle \), etc. In this case, the optimal trip is \( \langle a_1, a_2 \rangle \) since the travel score of \( \langle a_1, a_2 \rangle \) is greater than other valid trips.

**Problem Formulation.** With the above definitions, the main problem we address in this paper is formulated as follows. Given a current location of the user \( u \) and several user-specific constraints such as travel time and travel budget, our goal is to develop an efficient trip recommendation framework which provides the optimal trip meeting multiple user-specific constraints. We expect the planner can efficiently and accurately return the trip answer.

### 4. Proposed Methods

In this section, we first give the proposed framework of **Personalized Trip Recommendation (PTR)**. Next, three main techniques of PTR, including attraction finding, attraction scoring and multi-constraints trip planning, are presented in detail.

#### 4.1 System Framework

In this paper, we aim to design a travel recommendation system which can suggest the user a personalized trip meeting multiple constraints in real time. Figure 2 shows, PTR consists of two stages. The first stage is an offline data mining mechanism which can collect travel related data from Gowalla. First, we can obtain the POI information of each attraction directly from the collected data. Then, we evaluate the user-based and temporal-based attraction scores of each attraction by check-in logs. Finally, we pick up the top-k attractions with highest user-based attraction scores as the input data of our second stage. The second stage is an online query mechanism. In this stage, users need to input a start location, a start time, a budget constraint and a time constraint via their handheld mobile devices when they need to plan a trip. After receiving user’s request, we first fusion the temporal-based attraction score and user-based attraction score by a user-specific weight. Next, the **Parallel Trip-Mine** algorithm is proposed to plan trip. The algorithm considers both of the time constraint and budget limitation at the same time. For improving the computational efficiency, we further apply a parallel technique in the process of path planning. Finally, we re-rank the trip ranking list by considering the diversity of trip categories and recommend the trip with the highest score to users.

#### 4.2 Attraction Finding

The first task of PTR is to find attractions. We choose the famous LBSN site, Gowalla, as our research dataset. On Gowalla, users must to choose a location, which is also called spot, before they want to check-in. There are four main elements in each spot, which are "spot name", "the number of users who have checked-in the spot", "the number of check-ins which the spot has been made" and "spot category". Users can either choose an existing spot or create a new spot by themself. Intuitively, we can directly take these spots as attractions. However, it is not work due to the dataset have two characteristics. 1) **Duplicated Spots**. There usually exist a lot of spots in a large and famous attraction. Take Figure 3(a) as an example. There are a lot of spots in Central Park. These spots are not only some well-known statues and land marks in Central Park but also duplicate spots which all represent Central Park but created by different users. 2) **Functional Street**. A number of spots with the same categories are usually located in a certain small region. For example, as Figure 3(b) shows, these spots belong to the category of Asian Food while most of them are aggregated on the W 32nd Street because many of Asian restaurants are located on this street. The planned trip may contain many attractions with the same category or flock around in a small region if these spots are considered as the distinct attractions. To solve the above problems, the near spots with the same category are clustered by clustering algorithms such as DBSCAN [3] as an attraction.

#### 4.3 Attraction Scoring

For each attraction, PTR needs to understand that how interesting the attraction is for a specific user at a specific time before
planning the trip. This task is called **Attraction Scoring** which consists of two aspects: 1) **User-based attraction Score** (US), which is to measure how interesting the attraction is for a specific user. Four strategies are proposed to estimate USs; 2) **Temporal-based attraction Score** (TS), which is to measure how suitable users visit the attraction at a specific time. Different attractions may have different suitable time periods. We propose a strategy to estimate TS. Finally, US and TS are fused by a user-specific weight parameter as the **Attraction Score (AS)**. Because the database of check-in will be used in the following sections, we recall Definition 2. \( D = \{ c_i, c_2, \ldots, c_{|D|} \} \) denotes a check-in database, where \( c_i = (u, t, a) \in D \) indicates a check-in record which means user \( c_i \) checks-in attraction \( c_i.a \) at time \( c_i.t \).

### 4.3.1 User-based Attraction Score Estimation

To estimate US, the first proposed strategy named **Ranking-By-Preference (RBP)** which is based on the personal preferences of users. Then, three kinds of Collaborative Filtering (CF)-based strategies including Friend-based CF (FCF), Similarity-based CF (SCF) and Similarity-Friend-based CF (SFCF) are proposed to estimate attraction scores based on the LBSN from other users. Let \( u \) and \( a \) be a user and an attraction, respectively, the goal of user-based attraction score estimation is to calculate US(\( u, a \)).

#### 4.3.1.1 Ranking-By-Preference

As mentioned in Section 4.2, each attraction \( a \) in Gowalla belongs to a category, i.e., CAT(\( a \)). We think that there is a strong relationship between user preferences and attraction categories. For instance, a user frequently checks-in attractions which belong to “City Park” may imply that the user loves to engage in outdoor activities. Hence, it is appropriate to recommend the user the attractions such as “Central Park” and “Liberty State Park”. For a user \( u \), the idea of **Ranking-By-Preference (RBP)** is to count the number of personal check-in attractions which belongs to the category \( c \). Then, all of the values, i.e., \( |D(u, c)| \), are normalized as the preference score from 0 to 1, denoted as \( ps(u, c) \), which is defined as (5), where \( C \) indicates the set of all categories.

\[
ps(u, c) = \frac{|D(u, c)|}{\arg \max_{c \in C} \text{Max}(|D'(u, c')|)} \tag{5}
\]

For a user \( u \), the preference vector, denoted as \( P^*(u) \), is obtained by \( u \)’s preference score of each category, which is defined as (6).

\[
P^*(u) = ps(u,c_1), ps(u,c_2), \ldots, ps(u,c_{|C|}) > \tag{6}
\]

For a user \( u \) and an attraction \( a \), the **User-based attraction Score** by RBP, denoted as \( US_{RBP}(u, a) \), is defined as (7).

\[
US_{RBP}(u, a) = ps(u,a,c_1) \tag{7}
\]

#### 4.3.1.2 Friend-based CF

User-based attraction score may be estimated by the concept of Collaborative Filtering (CF). Hence, the first proposed CF-based strategy is **Friend-based CF (FCF)** that considers the friendship information as social links to estimate the user-based attraction score. The basic idea of FCF is to estimate the score of an attraction for a user by considering the check-in behaviors of the users’ friends. Note that the friendship between users can be collected directly from Gowalla. The **User-based attraction Score by FCF**, denoted as \( US_{SCF}(u, a) \), is defined as (8), where \( FU \) indicates the set of all the friends of \( u \). The functions \( \text{hasCheckedIn}(u, a) \) and \( \text{isFriend}(u, v) \) equal to 1 if user \( u \) has checked-in attraction \( a \) and user \( u \) and \( v \) exist a friend relationship, respectively. Otherwise, the two functions equal to 0.

\[
US_{SCF}(u,a) = \sum_{v \in FU} \frac{\text{hasCheckedIn}(v,a) \times \text{isFriend}(u,v)}{\sum_{v \in FU} \text{isFriend}(u,v)} \tag{8}
\]

#### 4.3.1.3 Similarity-based CF

As mentioned in Section 4.3.1.1, the preference vector of user \( u \) \( P^*(u) \) can be obtained by considering the personal check-in history of \( u \). We can further use \( P^*(u) \) to calculate the similarity between \( u \) and other users. Let \( u \) and \( v \) be two users, the similarity between \( u \) and \( v \) can be measured by the cosine similarity measurement as shown in (9). Based on the similarity, the second proposed strategy is **Similarity-based CF (SCF)** that considers the user similarities as social links to estimate the user-based attraction score. The **User-based attraction Score by SCF**, denoted as \( US_{SCF}(u, a) \), is defined as (10), where \( SU \) indicates the set of top-\( m \) similar users of \( u \).

\[
sim(u, v) = \frac{\sum_{c \in C} ps(u,c) \times ps(v,c)}{\sqrt{\sum_{c \in C} ps(u,c)^2 \times \sum_{c \in C} ps(v,c)^2}} \tag{9}
\]

\[
US_{SCF}(u,a) = \sum_{v \in SU} \frac{\text{hasCheckedIn}(v,a) \times \text{sim}(u,v)}{\sum_{v \in SU} \text{sim}(u,v)} \tag{10}
\]

#### 4.3.1.4 Similarity-Friend-based CF

The last proposed strategy is **Similarity-Friend-based CF (SFCF)** which is the fusion of FCF and SCF. In SCF, user-based attraction score is estimated based on not only similar users but also friends. The **User-based attraction Score by SFCF**, denoted as \( US_{SFCF}(u, a) \), is defined as (11), where \( \lambda \) (limited from 0 to 1) is used to control the weight between similarity and friendship.

\[
sim(u, v) = \sum_{v \in SU} \frac{\text{hasCheckedIn}(v,a) \times \text{sim}(u,v)}{\sum_{v \in SU} \text{sim}(u,v)} \tag{11}
\]

#### 4.3.2 Temporal-based Attraction Score Estimation

In real travel experiences, different attractions should have different suitable visiting time periods. For example, **Times Square** in New York is more suitable to be visited in the evening. Hence, for each attraction \( a \), \( PTR \) defines 24 **Temporal-based attraction Scores** (TS), denoted as \( TS(a, t) \), in a day, where \( t \) indicates a specific time period. TS is considered from two aspects: 1) **What time is more suitable for visiting this attraction?** For an attraction, we sum up the number of check-ins for every time period and normalize all of the values. This strategy is called **Normalized-By-Time (NBT)**. The **Temporal-based attraction Score by NBT**, denoted as \( TS_{NBT}(a, t) \), is defined as (12), where \( |D'(a, t)| \) represents the number of check-ins of attraction \( a \) during time period \( t \), and \( T \) indicates the set of all time periods, i.e, 24 time periods. 2) **Which attractions are popular now?** For a time period, we sum up the number of check-ins for every attraction and normalize all of the values. This strategy is called **Normalized-By-Attraction (NBA)**. The **Temporal-based attraction Score by NBA**, denoted as \( TS_{NBA}(a, t) \), is defined as (13), where \( A \) indicates the set of all attractions.

\[
TS_{NBT}(a,t) = \frac{|D'(a,t)|}{\arg \max_{a, t} \text{Max}(|D'(a, t)|)} \tag{12}
\]

\[
TS_{NBA}(a,t) = \frac{|D'(a,t)|}{\arg \max_{a,t} \text{Max}(|D'(a,t)|)} \tag{13}
\]

\( TS_{NBT} \) and \( TS_{NBA} \) are fused by the harmonic mean since we want both of the two values are relative high. Therefore, we can ensure
that the time period \( t \) is really suitable for visiting the attraction \( a \) and the attraction \( a \) is suitable for visiting during the time period \( t \). The final \( TS(a, t) \) can be defined as (14).

\[
TS(a, t) = \frac{2 \times TS_{\text{us}}(a, t) \times TS_{\text{ts}}(a, t)}{TS_{\text{us}}(a, t) + TS_{\text{ts}}(a, t)} \tag{14}
\]

As mentioned above, we obtain both the user-based attraction score and temporal-based attraction score for each attraction. In the next section, we will describe how to fuse them when a user wants to plan a trip.

4.4 Multi-Constraints Trip Planning

The online query stage of PTR consists of three main processes: 1) Attraction Score Fusion. The user-based and temporal-based scores are fused by a user-specific temporal factor; 2) Trip Planning. We propose the Parallel Trip-Mine\(^+\) algorithm to plan trips under the user-specific budget constraint and time constraint; and 3) Diversity-based Trip Re-ranking. We pick up top-\( k \) trips with highest scores for re-ranking based on the diversity of trip.

4.4.1 Attraction Score Fusion

After obtaining the user-based attraction score and temporal-based attraction score for each attraction, the main purpose of PTR is to plan a trip which not only satisfies user preference but also matches the suitable visiting time. We first select the top-\( k \) attractions with highest user-based attraction scores. It can ensure that these \( k \) attractions are more satisfied with the user’s preference. Then, for these \( k \) attractions, their temporal-based attraction scores and user-based attraction scores are fused as the final Attraction Scores, denoted as \( AS(u, a, t) \) and defined as (15), which represents the score of attraction \( a \) for user \( u \) during time period \( t \). The parameter \( \alpha \) is set by users to control the weight between user-based and temporal-based attraction score.

\[
AS(u, a, t) = \alpha \times TS(a, t) + (1 - \alpha) \times US(u, a) \tag{15}
\]

4.4.2 Trip Planning

To clear explain our design, we use Figure 1 as an example. Suppose that Tom is a tourist and he has 210 minutes and 50 USD to travel. He starts from \( a_0 \) at 8:00 and \( a_1 \) to \( a_4 \) represent four different attractions. Each attraction has the stay time and attraction cost. For two attractions, the approximate travel time is different. Each attraction has the stay time and attraction cost. For two attractions, the approximate travel time is different when the orders of attraction permutations are different. However, in our problem, the trip scores are different when the orders of attraction permutations are different. Hence, Trip-Mine is extended to Trip-Mine\(^+\) solving the new problem with dynamic attraction scores. Following the idea of Trip-Mine, Trip-Mine\(^+\) is also designed as an Apriori-based algorithm. To explain the policy of candidate attraction set generation, we use Figure 4 as an example. Suppose that \( VT \) is the set of all valid trips. Initially, \( VT = \emptyset \). We first generate 4 candidate 1-sets and check whether both the trip time and trip cost satisfy the two constraints. In this example, all of the candidate 1-sets are valid. Hence, we put all candidate 1-sets into \( VT \). Next, two valid 1-sets are joined as a candidate 2-set. There are 6 candidate 2-sets. By checking their trip time and trip cost, we observe that \( \{a_2, a_5\} \) is not valid because its trip time exceeds the travel time constraint. Hence, \( \{a_2, a_5\} \) can be pruned, and it does not join the candidate 3-set generation. For these candidate 2-sets, we need to generate their permutations and calculate the current Attraction Score of each attraction for getting the trip scores. We put all valid trips generated from candidate 2-sets into \( VT \). There are 3 candidate 3-sets. By checking their trip cost, we observe that \( \{a_1, a_2, a_3\} \) is not valid. Hence \( \{a_1, a_2, a_3\} \) can be pruned. Again, for these candidate 3-sets, we need to generate their permutations to get trip scores and check whether any permutation satisfies the travel time constraint. For the candidate 3-sets \( \{a_1, a_2, a_3\} \), all permutations are invalid. Hence, \( \{a_1, a_2, a_3\} \) can be pruned. For the candidate 3-sets \( \{a_1, a_2, a_3\} \), all permutations are valid. Hence, we put all permutations into \( VT \). The algorithm stops since no candidate attraction set is generated. Finally, we can obtain a list of trips ranked by the trip score. The optimal trip until now is \( \{a_1, a_2, a_3\} \) since the trip score is the highest from the valid trips set, while all the top-\( k \) trips will be considered to be re-ranked in the next step.

<table>
<thead>
<tr>
<th>Location</th>
<th>Duration</th>
<th>Attraction Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_0 )</td>
<td>8:00</td>
<td>0.0</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>8:30-9:00</td>
<td>0.3*(0.5)+0.5*(1-0.5) = 0.4</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>9:40-10:10</td>
<td>0.7*(0.5)+0.8*(1-0.5) = 0.75</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>10:30-10:50</td>
<td>0.2*(0.5)+0.3*(0.5) = 0.25</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>11:20</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 1. Temporal-based and user-based attraction scores.

<table>
<thead>
<tr>
<th>Location</th>
<th>Duration</th>
<th>Attraction Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>8:00</td>
<td>0.0</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>8:30-9:00</td>
<td>0.3*(0.5)+0.5*(1-0.5) = 0.4</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>9:40-10:10</td>
<td>0.7*(0.5)+0.8*(1-0.5) = 0.75</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>10:30-10:50</td>
<td>0.2*(0.5)+0.3*(1-0.5) = 0.25</td>
</tr>
<tr>
<td>( a_5 )</td>
<td>11:20</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 2. A simple trip.
Trip-Mine\(^+\) planning, we integrate the parallel computing techniques into still a NP-hard problem. To improve the efficiency of trip stores all valid trips generated in Thread 1. Initially, although the computation cost of permutation can be saved if we do not need to generate the permutations if the lower bound of a trip already exceeds the user’s budget constraint. Hence, we put them into \(VT\). Two valid 2-sets are joined as candidate 3-set. There are 3 candidate 3-sets. \(|\{a_1, a_2, a_3\}|\) can be pruned since the trip cost exceeds the user’s budget constraint. \(|\{a_1, a_2, a_3\}|\) also can be pruned by the aforementioned lower bound checking mechanism. Only \(|\{a_1, a_2, a_3\}|\) needs to generate permutations and check the travel time and trip score. Notice that we only need to generate permutation of attractions with prefix \(a_i\), i.e., \(<a_1, a_2, a_3>\) and \(<a_1, a_2, a_4>\), since the prefix attraction has already fixed. In thread 3 and 4, we repeat the same processes and obtain \(VT_3, VT_4\), and \(VT_7\) which stored with valid trips generated in Thread 2, 3, and 4, respectively. Finally, we combine all valid trip sets into a global valid trip set \(VT\). The ranking list of trip can be obtained after sorting \(VT\) by trip scores.

Figure 4. The process of Trip-Mine\(^+\) algorithm.

### 4.4.2.2 Lower Bound Checking Mechanism
In Trip-Mine\(^+\), we apply one of optimization mechanisms proposed in Trip-Mine, namely lower bound checking, to reduce the number of permutation generations. If we calculate the lower bound of travel time for each candidate set, we do not check all the permutations if the lower bound of a trip already exceeds the travel time constraint. To explain the lower bound checking mechanism, we take Figure 1 and Figure 4 as an example. In Trip-Mine\(^+\), for the candidate set \(\{a_1, a_2, a_3\}\) in Figure 4, we need to generate all permutations of the set and check whether every trip is valid. The computation cost of permutation can be saved if we check the travel time lower bound of \(|\{a_1, a_2, a_3\}|\). As Figure 5 shows, the sum of stay time for the three attractions is 30+20+50=100. The routes \((a_2, a_3)\) and \((a_1, a_3)\) will be selected as the minimal moving travel time between attractions which is 30+50=80, and the routes \((a_2, a_3)\) and \((a_3, a_1)\) will be selected as the minimal moving travel time between start location to attractions which is 30+30=60. Finally the travel time lower bound of \(|\{a_1, a_2, a_3\}|\) is 100+80+60=240, which is greater than the travel time constraint. Hence, we do not need to generate the permutations of \(|\{a_1, a_2, a_3\}|\) and check their trip time.

Figure 5. Lower bound checking for attraction set \(|\{a_1, a_2, a_3\}|\).

### 4.4.2.3 Parallel Trip-Mine\(^+\) Algorithm
Although Trip-Mine\(^+\) is efficient, the problem of trip planning is still a NP-hard problem. To improve the efficiency of trip planning, we integrate the parallel computing techniques into Trip-Mine\(^+\) as a Parallel Trip-Mine\(^+\) algorithm. We first take each attraction as prefix and generate the candidate set based on the prefix attraction. Hence, we can separate the whole candidate set into several parts. With the characteristic of parallel computing techniques, the processes of permutation and pruning can be executed simultaneously. All of the valid trips can be obtained when all parallel processes are completed. Hence, we sort all the valid trips by trip scores. Finally we can get a ranking list of trip.

To explain Parallel Trip-Mine\(^+\), we use the same example shown in Figure 6. In Thread 1, we pick up attraction \(a_1\) as a prefix. \(VT\) stores all valid trips generated in Thread 1. Initially, \(VT_1 = \emptyset\). There are 3 candidate 2-sets with prefix \(a_1\), i.e., \(<a_1, a_3>\), \(<a_1, a_3>\) and \(<a_1, a_4>\). Their permutations do not need to be generated since their prefixes have already fixed in \(a_1\). All the 3 candidates are valid. Hence, we put them into \(VT\). Two valid 2-sets are joined as candidate 3-set. There are 3 candidate 3-sets. \(|\{a_1, a_2, a_3\}|\) can be pruned since the trip cost exceeds the user’s budget constraint. \(|\{a_1, a_2, a_3\}|\) also can be pruned by the aforementioned lower bound checking mechanism. Only \(|\{a_1, a_2, a_3\}|\) needs to generate permutations and check the travel time and trip score. Notice that we only need to generate permutation of attractions with prefix \(a_i\), i.e., \(<a_1, a_2, a_3>\) and \(<a_1, a_2, a_4>\), since the prefix attraction has already fixed. In thread 2, 3, and 4, we repeat the same processes and obtain \(VT_2, VT_3\), and \(VT_7\) which stored with valid trips generated in Thread 2, 3, and 4, respectively. Finally, we combine all valid trip sets into a global valid trip set \(VT\). The ranking list of trip can be obtained after sorting \(VT\) by trip scores.

Figure 6. The process of Parallel Trip-Mine\(^+\) algorithm.

### 4.4.3 Diversity-based Trip Re-ranking
Although the Trip-Mine\(^+\) algorithm can find the trip with the highest score, users may not satisfy if there are many attractions with the same category in the trip. We take Figure 7 as an example. Trip A and B have very close trip score while the score of A is a little bit higher than that of B. If PTR just consider the trip score, PTR may recommend A for users. However, people may prefer B than A since the attraction categories of B are more diverse than those of A. To avoid recommending users a lot of attractions with the same category, PTR re-ranks the trips considering not only the trip score but also trip diversity.

Figure 7. Two sample trips with different diversities.
describe the experimental real data crawled from time and a planned trip, respectively.

\[ \text{Entropy}(p) = -\sum_{c \in c(p)} \frac{n(c)}{|p|} \times \log_2 \left( \frac{n(c)}{|p|} \right) \]  

Finally, we pick up the top-\(k\) trips with the highest trip scores to ensure that the recommended candidate trips must satisfy the user preferences and time properties. Then, a parameter \(\beta\) is set by users to control the weight between trip score and trip entropy. The trip score considering trip diversity, denoted as \(\text{TPS}_{\beta}(u, t, tp)\), is re-scored by (17), where \(u, t\) and \(tp\) indicate a user, a specific time and a planned trip, respectively.

\[ \text{TPS}_{\beta}(u, t, tp) = \beta \times \text{Entropy}(tp) + (1 - \beta) \times \text{TPS}(u, t, tp) \]  

5. Experimental Evaluations

In this section, we conducted a series of experiments to evaluate the effectiveness and efficiency of our proposed PTR. We first describe the experimental real data crawled from Gowalla and the experimental settings. Next, the experimental evaluations are divided into three parts: 1) Comparison of accuracy by various attraction scoring strategies; 2) Comparison of efficiency between Parallel Trip-Mine' and Trip-Mine'; and 3) Comparison of effectiveness by various trip planning methods. The experiments are implemented in Java running Windows 7 64 bits OS at Intel Core i7-2600 clock of 3.4 GHz with 8 cores, memory of 8 GB.

5.1 Experiment Settings

We first describe a real check-in dataset obtained from Gowalla. We collect the dataset from January 2009 to April 2011. The collected dataset consists of check-in data in New York (NY) and other cities. For the whole dataset, the number of users, attractions and attraction check-ins is 2,192, 977,983 (3,007 in NY) and 1,422,762 (153,997 in NY), respectively. Each user averagely checks-in 61.13 different attractions in NY and 587.94 different attractions in other cities. For each attraction, the stay time and attraction cost are estimated based on the category of attraction. The travel time between attractions is proportionally estimated based on the distance of them. Both the user-based attraction score and temporal-based attraction score are estimated by our proposed strategies in Section 4.3. However, the size of real dataset we can crawl is limited by Gowalla and the scalability of datasets we can crawl is limited by Gowalla and the proposed strategies in Section 4.3. Therefore, a complete synthetic dataset is established.

5.2 Accuracy of Attraction Scoring

To compare the effectiveness of various attraction scoring strategies, we equally divide users into 5 groups (4 groups of users for training and 1 group of users for testing). For the 80% training users (4 groups), the preference vectors defined in Section 4.3.1.1 are extracted by their whole check-in data. For the 20% testing users, we remove their check-ins in NY as the ground truths and extract their preference vectors by their check-in data in other cities. Based on these preference vectors, the similarity between training users and testing users can be further calculated. Based on the similarity or the friendship between users, the scores of attractions in NY can be estimated for testing users. These scores can be used to recommend a ranking list of attractions in NY to the testing user. The recommended effectiveness can be measured by comparing the ranking list with the ground truth check-ins in NY which we removed previously. We compare our proposed strategies RBP, FCF, SCF and SFCF with Random recommendation and the three existed strategies HITS, RBF and RBC [16] in terms of two well-known measurements, namely nDCG (normalized discounted cumulative gain) [7] and precision. nDCG is used to compute the ranking result relative to the ideal ranking list. For each attraction, we take the check-in frequency as rating. The discounted cumulative gain is computed as (18), where \(\text{rel}_i\) is the check-in frequency of attraction at rank \(i\) in the recommended ranking list. Given the ideal discounted cumulative gain \(\text{IDCG}_p\), then DCG at \(p\)-th position can be computed as (19).

\[ \text{DCG}_p = \text{rel}_1 + \sum_{i=2}^{p} \frac{\text{rel}_i}{\log_2(i)} \]  

\[ \text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p} \]  

precision is often used to measure the prediction performance. In our experiments, we examine the precision of recommended ranking list. The precision at \(p\)-th position can be computed as (20), where \(TP_p\) and \(FP_p\) indicate the \(p\)-th attraction in the ranking list that the testing user has checked-in and the user has never checked-in, respectively.

\[ \text{precision}_p = \frac{TP_p}{TP_p + FP_p} \]  

Table 3 summarizes the major parameters and the default values in the simulation model. In the base experiment model, we use a \(|\mathcal{W}| \times |\mathcal{W}|\) network to model the travel region. All of the attractions are randomly located on the travel region. The number of attractions is determined by a given parameter \(N_a\). For each attraction, the user-based attraction score, the temporal-based attraction score, the attraction cost and the stay time are determined based on uniform distributions within given ranges \(R_{CS}, R_{TS}, R_{CS}\), respectively. For each pair of attractions, the travel time is proportionally determined based on the distance of them. Thus, a complete synthetic dataset is established.

Finally, we simulate tourists query the trip with a travel time constraint, a budget constraint and a departure time. For each tourist, the start location is randomly selected in NY (real dataset) or in the travel region (synthetic dataset). For each pair of start location and attractions, the travel time is proportionally determined based on the distance of them. The travel time constraint, the budget constraint and the departure time are determined by the parameter \(C_T\), \(C_B\) and \(D\), respectively.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>\mathcal{W}</td>
<td>)</td>
</tr>
<tr>
<td>(N_a)</td>
<td>The number of attractions in the region</td>
<td>400</td>
</tr>
<tr>
<td>(R_{CS})</td>
<td>The range of stay time for each attraction</td>
<td>30–240 mins</td>
</tr>
<tr>
<td>(R_{TS})</td>
<td>The range of attraction cost for each attraction</td>
<td>0–50 USD</td>
</tr>
<tr>
<td>(R_{CS})</td>
<td>The range of user-based attraction score for each attraction</td>
<td>0–1</td>
</tr>
<tr>
<td>(R_{TS})</td>
<td>The range of temporal-based attraction score for each attraction</td>
<td>0–1</td>
</tr>
<tr>
<td>(C_T)</td>
<td>The travel time constraint</td>
<td>400 mins</td>
</tr>
<tr>
<td>(C_B)</td>
<td>The budget constraint</td>
<td>100 USD</td>
</tr>
<tr>
<td>(D)</td>
<td>The departure time</td>
<td>AM 08:00</td>
</tr>
</tbody>
</table>

Figure 8 shows that the our proposed SFCF outperforms various existed attraction recommendation strategies in terms of nDCG and precision at rank 5, 10 and 15. As the results show, both of the nDCG and precision of Random recommendation strategy are very low since there are large number of attractions in NY, i.e., 3,007, but each user only averagely checks-in 61.13 attractions. Hence, it is very difficult to recommend the right attractions users have checked-in. HITS is also not good since the hypothesis of HITS is that experienced users will visit interesting attractions and
interesting attractions will be visited by experienced users too. However, the experienced users in Gowalla dataset usually check-in attractions around their neighborhoods or attractions nearby their companies. Hence, the recommended attractions may not be interesting. RBC and RBF performs well since popular attractions usually attract most of people. However, RBC and RBF do not consider user preference. Hence, the attraction scores calculated by RBC and RBF may not be satisfied by everyone. For our proposed strategies, RBP is not good since RBP only consider the categories of attractions. Hence, RBP can not differentiate which attraction is better when there are a number of attractions with the same category. FCF is not good too since users’ friends may not have the same interests. We can observe that SCF is better than RBC and RBF in terms of nDCG and precision. The reason is that SCF captures the user preferences based on the user check-in similarity. SCF outperforms SCF and FCF since we consider not only the friendship but also the check-in behaviors.

Figure 8. nDCG and precision of various scoring strategies.

5.3 Efficiency of Trip Planning

We conducted an experiment for comparing the efficiency of our proposed two algorithms, i.e., Trip-Mine’ and Parallel Trip-Mine’, on Gowalla dataset and synthetic dataset.

5.3.1 Impact of Number of Attractions $N_A$

This experiment analyzes the execution time of Trip-Mine’ and parallel Trip-Mine’ on Gowalla dataset and synthetic dataset when the number of attractions ($N_A$) is varied from 200 to 700. Figure 9 shows that Parallel Trip-Mine’ outperforms Trip-Mine’ in terms of execution time with varied the number of attractions. We observe that the execution time increase significantly with the number of attractions increases. The reason is that the number of possible trips exponentially increases when the number of attractions increases. Due to our problem is to find the optimal trip, our proposed algorithms need more execution time to generate and check the more candidate attraction sets.

5.3.2 Impact of Travel Time Constraint $C_{TT}$

This experiment analyzes the execution time of Trip-Mine’ and Parallel Trip-Mine’ on Gowalla dataset and synthetic dataset when the travel time constraint ($C_{TT}$) is varied from 240 to 480 minutes. Figure 10 shows that Parallel Trip-Mine’ outperforms Trip-Mine’ in terms of execution time with varied the travel time constraint. We observe that the execution time significantly increase by increasing the travel time constraint. The reason is that the number of valid trips increases, when the travel time constraint increases. Hence, our proposed algorithms need to spend more time to find all valid trips.

5.3.3 Impact of Budget Constraint $C_{BD}$

This experiment analyzes the execution time of Trip-Mine’ and Parallel Trip-Mine’ on Gowalla dataset and synthetic dataset when the budget constraint ($C_{BD}$) is varied from 0 to 100 and from 50 to 200 USD. Figure 11 shows that Parallel Trip-Mine’ outperforms Trip-Mine’ in terms of execution time with varied the budget constraint. In synthetic dataset we observe that the execution time significantly increase at first while keep steady after the budget constraint higher than 150 USD. The reason is that the number of candidate sets will be bounded by the number of attractions and the travel time constraint. Hence, the number of valid trips will not increase even if we keep increasing the budget constraint.

5.4 Effectiveness of Planned Trip

We conducted an experiment for comparing the planed results between our proposed Parallel Trip-Mine’ and a series of methods, i.e., d-LOA, v-LOA and GOA, proposed by Zhou et al. [17]. The strategy of d-LOA is to form a sequence by iteratively computing the half nearest neighbor attractions of the current attraction, comparing the scores of them, choosing the attraction whose score is maximal from all attractions that have not been visited yet. The strategy of v-LOA is to form a sequence by
iteratively computing other attractions that have not been visited yet and choosing the attraction whose score is maximal. The strategy of GOA is to form a sequence by iteratively computing other attractions that have not been visited yet, choosing an attraction which the value fused by attraction score and the inverse of distance to the destination is maximum. All these three methods try to find trips with higher trip score, while keep lower transition time cost.

5.4.1 Impact of Travel Time Constraint \( C_{TT} \)
This experiment compares the planed trip scores of Parallel Trip-Mine\(^+\) with d-LOA, v-LOA and GOA on Gowalla data when the travel time constraint \( (C_{TT}) \) is varied from 240 to 480 minutes. Figure 12 (a) shows that the trips planed by Parallel Trip-Mine\(^+\) always achieve the highest score. For all methods, the trip scores increase by increasing the travel time constraint. The reason is that trips can contain more attractions when the travel time constraint increases. The results of d-LOA and v-LOA are very similar since d-LOA usually choose the same next attraction as v-LOA if the attraction with the highest score is located in the half nearest neighbor attractions set. The result of GOA is not good since GOA may choose the attractions which close to the destination but with relative lower score.

5.4.2 Impact of Budget Constraint \( C_{BD} \)
This experiment compares the planed trip scores of Parallel Trip-Mine\(^+\) with d-LOA, v-LOA and GOA on Gowalla data when the budget constraint \( (C_{BD}) \) is varied from 0 to 100 USD. Figure 12 (b) shows that the trips planed by Parallel Trip-Mine\(^+\) always achieve the highest score. For Parallel Trip-Mine\(^+\), d-LOA and v-LOA, trip scores increase at first while keep steady after the budget constraint higher than 40 USD. The reason is that the number of attractions in a trip will be bounded by the travel time constraint. The results of d-LOA and v-LOA are very similar. The reason is the same with Section 5.4.1. The trip score of GOA slightly decreases with the budget constraint increases since GOA may chooses the attractions which close to the destination but with relative lower score.

![Figure 12. Comparison of various trip planning methods](image)

6. Conclusions and Future Work
In this paper, we have proposed a novel approach named Personalized Trip Recommendation (PTR) for recommendation of personal trip with multiple constraints by mining users’ check-in behaviors. In the proposed PTR, we have proposed i) an automatic module to estimate the scores of attractions by considering both of user-based preferences and temporal-based properties, and ii) Parallel Trip-Mine\(^+\) to efficiently plan the trip which satisfies multiple user-specific constraints. To our best knowledge, this is the first work on travel recommendation that considers the issues of multiple constraints, social relationship, temporal property and parallel computing simultaneously. Through a series of experiments by the real dataset Gowalla, we have validated our proposed PTR and shown that PTR has excellent performance under various conditions. In the future work, we will consider more constraints and collect more real datasets to improve and verify our proposed PTR, respectively.

7. REFERENCES