TravelBuddy : Interactive Travel Route Recommendation with a Visual Scene Interface

Cheng-Yao Fu[1], Min-Chun Hu[2], Jui-Hsin Lai[1], Hsuan Wang[3], and Ja-Ling Wu[1]

[1] Dept. of CSIE, National Taiwan University
[2] Dept. of CSIE, National Cheng Kung University
[3] Dept. of Civil Engineering, National Central University

Abstract. In this work, we propose a convenient system for trip planning and aim to change the behavior of trip planners from exhaustively searching information to receiving useful travel recommendations. Given the essential and optional user inputs, our system automatically recommends a route that suits the traveler based on a real-time route planning algorithm and allows the user to make adjustment according to their preferences. We construct a traveling database by collecting photos taken around famous attractions and analyzing these photos to extract each attraction's travel information including popularity, typical stay time, available visiting time in a day, and visual scenes of different time. All the extracted travel information are presented to the user to help him/her efficiently know more about different attractions so that he/she can modify the inputs to obtain a more favorable travel route. The experimental results show that our system can effectively help the user to plan the journey.

1 Introduction

Besides becoming a millionaire, traveling around the world is the most popular answer when people are asked about their fantasy. Traveling can always relax our mind, increase our knowledge and experience, widen our perspective, and create invaluable memories. With limited money and time, people would like to choose escorted tours planned by the travel agent and arranged to satisfy the average customer demand. Escorted tours are convenient but usually more expensive and with less flexibility than independent tours. With the popularity of internet and world wide web, people tend to search traveling information and plan suitable schedule on their own. However, this task has become more and more difficult due to internet information overflow. Trip planning websites/forums are then established for people to share their traveling experience so that one can find more reliable information and reorganize a new trip plan based on others' experiences. Unfortunately, the discussions on these websites/forums are usually disordered and without proper summarization.

The proliferation of social media websites and mobile devices motivate us to change the behavior of trip planners from exhaustively searching information to receiving useful recommendation. In this work, we propose a convenient system

with user friendly interface for trip planning. The proposed system framework is illustrated in Figure 1. We first construct a traveling attraction database containing famous attractions of each given city (e.g., Seville Cathedral in Barcelona and Eiffel Tower in Paris). These attractions are collected from TripAdvisor and Yahoo!Travel, which are the most two popular travel websites with helpful traveling information shared by experienced tourists [1]. Note that the GPS location of each attraction is also stored in the database. Moreover, photos of each attraction are retrieved from social media websites (e.g. Flickr) based on the attraction name and GPS information. We then analyze these photos to extract each attraction's travel information including popularity, typical stay time, available visiting time in a day, and visual scenes of different time.

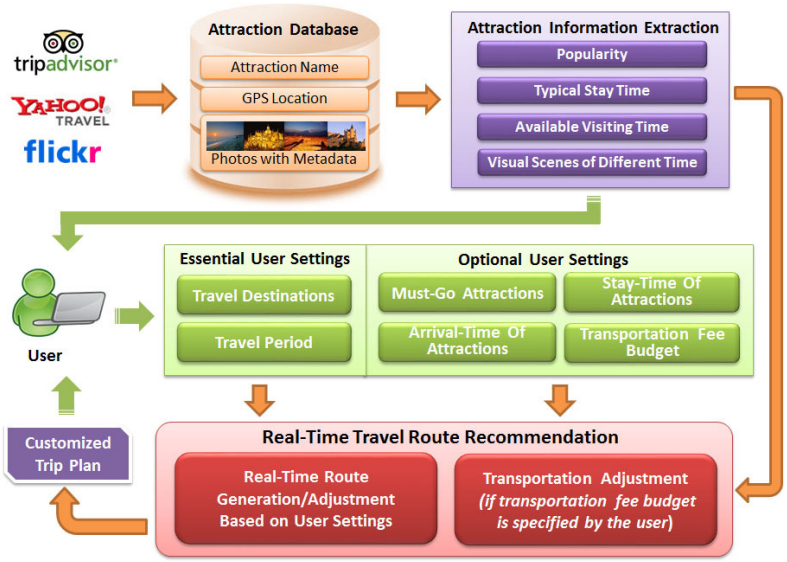

Fig. 1. The proposed system framework.

When the user wants to plan a trip, he/she could simply input concise traveling constraints which are essential or optional. The essential user setting includes a sequence of city destinations (e.g. Barcelona→Paris→Roma) and the travel period (defined by the start time and the end time of the trip), while the optional user setting involves must-go attractions, stay-time of specific attractions, arrival-time of specific attractions, and transportation fee budget. The system presents attraction information of the input destinations to the user with a designed visual scene interface[4] so that he/she could further select must-go

[4] Visual Scene Interface: Each attraction may look different at different time in different seasons, and we design an interface to show representative photos of various time period for each attraction based on Flickr photos.

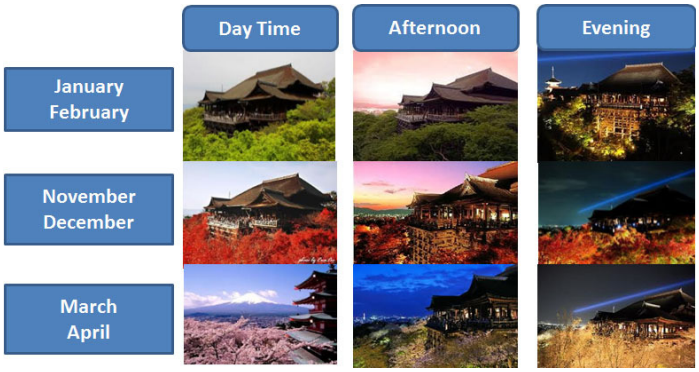

Fig. 2. Visual scenes of Kiyomizu Temple (Kyoto, Japan) at different time.

(or must-not-go) attractions and arrival/stay-time of specific attractions according to his/her preference. The travel route recommendation component then customizes a more customized and favorable trip route strictly including (or excluding) these selected attractions in real time with the transportation information between two adjacent attractions. For example, after browsing visual scenes of Kiyomizu Temple taken at different time (as shown in Figure 2), the user would specify to go to Kiyomizu Temple in the afternoon if his/her travel is scheduled in November or December because he/she is attracted by the visual scenes of Kiyomizu Temple taken in the afternoon. Besides recommending a suitable travel route after the user first inputs essential settings, our system also allows the user to interactively adjust the recommended route according to their preference on each suggested attraction or transportation between two adjacent attractions. The system will re-customize the route according to the new feedback (i.e. the optional user settings) until the user is satisfied.

The main contributions of this work are threefold. 1) To the best of our knowledge, our system is the first travel recommendation system that considers the available visiting time of attractions and transportation budget for trip planning. To be more precise, we tackle the query of finding a best trip route such that the traveler will arrive at some attractions at some specified time subject to that the total travel time and the total transportation budget is constrained. This kind of query is very common in real-world trip planning scenarios, but none of the existing travel recommendation system is designed to solve this problem. 2) We designed a convenient visual scene interface to present travel information of each attraction, so that the user could quickly have an impression on each attraction. 3) Our system deals with the problem of effectively re-adjusting the travel route and the transportation way according to the users feedback.

The remainder of this paper is organized as follows: Section 2 expounds how we extract travel information of each attraction and Section 3 introduces the proposed real-time travel route recommendation algorithm. The experimental

results and discussions are shown in Section 4, and conclusions are given in Section 5.

2 Attraction Information Extraction

As mentioned in Section 1, we construct the traveling database by searching famous attractions of each city on TripAdvisor and Yahoo!Travel and collecting Flickr photos taken around these attractions. All the collected Flickr photos (including the EXIF information) are used for extracting each attraction's travel information including popularity, typical stay time, available visiting time in a day, and visual scenes of different time. The extracted travel information will be presented to the user with the designed visual scene interface to help him/her efficiently know more about different attractions.

2.1 Estimation of Popularity

To estimate the popularity of each attraction, we assume that the more people had been to the attraction, the more popular the attraction is. This assumption is widely used by previous works [2, 3]. For each attraction A, we take the number of people who had taken photos at that attraction (i.e. the number of Flickr users who had uploaded photos taken at attraction A) as the corresponding popularity (popular score), denoted as $PS(A)$.

2.2 Estimation of Typical Stay Time

The estimation of typical stay time can be realized by the internal path discovering (IPD) method proposed by Lu et al. [2]. In general, one can utilize the information of photo taken time (embedded in the EXIF) to compute the time length of a photo sequence. For each photo sequence s_i composed of photos taken at attraction A by the same user on the same date, we set the corresponding time length, denoted as $TL(s_i)$, as the stay time of attraction A based on the sequence s_i. The estimated typical stay time of attraction A can be then defined by the average of all $TL(s_i)'s$. Yet photo sequences uploaded by users are usually sparse and incomplete. Hence the time length of photo sequence would often be much shorter than the real stay time of the attraction. IPD can concatenate multiple incomplete photo sequences to a complete one by discovering the internal path, and therefore the estimated typical stay time would be much more precise. However, for outdoor attractions, there is usually no specific visiting path, and therefore IPD may not successfully concatenate incomplete photo sequences. In this case, we apply the method proposed by Xie et al. [4] instead. That is, the typical stay time of an attraction A would be $(\omega * area\ size\ of\ A)$, where ω depends on the popularity of the attraction. For instance, popular attractions are with higher ω than unpopular ones since people would like to spend more time in more popular attraction per unit area.

2.3 Estimation of Available Visiting Time

In order to automatically retrieve the available visiting time in a day of an attraction A, we exploit the photo taken time to gather statistics about when this attraction is available for visiting. If there are enough amount of photos taken around attraction A at certain time, then this time should belong to the available visiting time of attraction A. Practically, we quantize a day into 48 time intervals (i.e. each interval equals 30 mins) and calculate the amount of photos taken around attraction A during each time interval instead of at a specific time. If a time interval has few photos, it implies that A's open time may not include that time interval.

2.4 Visual Scenes Generation

We modify the representative photo extraction method proposed in [3] to generate visual scenes at different time for each given attraction. The flowchart of our method is illustrated in Figure 3. All photos of an attraction are clustered into N_D Date Clusters by their taken date to distinguish different scenes in a year. Photos in each Date Cluster are further clustered into N_T Time Clusters to distinguish different scenes in a day. Finally, for each Time Cluster we use the visual features to cluster all involved photos into N_V Visual Clusters [3]. Therefore, photos in the same Visual Cluster would look similar and photos in different Visual Clusters would look more different. We select the centroid photo from each Visual Cluster as the representative scene of the attraction at a certain time. In our implementation, N_D is set to 24, which means photos within the same half month interval would be clustered together. N_T is set to 3 since there are usually three quite different scenes (morning, afternoon, and evening) in a day of an attraction. N_v is set to $\sqrt{n/2}$ as suggested in [3], where n denotes the number of photos being clustered.

3 Real-time Travel Route Recommendation

After gathering travel information of all attractions and the user inputs, we can compute and generate a suitable travel route for the user. In this section, we define the route recommendation problem and introduce how to solve it in real-time.

3.1 Problem Statement

The target travel route should fulfill the following criteria:

- **Plentiful.** The travel route should contain as many popular attractions as possible since in general tourists would like to visit as many popular and representative attractions as they can [5, 6]. The total popular score of a travel route R can be defined as:

$$TPS(R) = \sum_{i=1}^{RouteSize-1} PS(i), \tag{1}$$

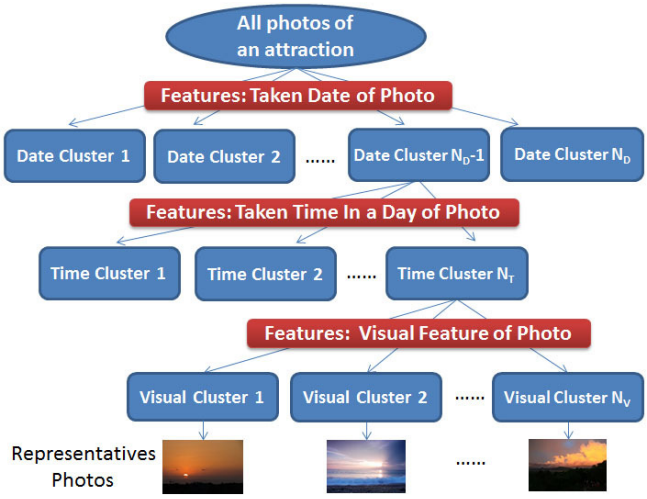

Fig. 3. Flowchart of Visual Scene Generation.

where i denotes the ith attraction on the trip route, $i = 0$ means the start location, and $i = 1$ means the first visited attraction.

- **Smooth.** The trip route should be smooth to avoid back and forth from one location to another. The non-smoothness of a trip route R is defined by

$$NS(R) = \sum_{i=1}^{RouteSize-1} [dist(i-1,i) + dist(i,i+1) - dist(i-1,i+1)], \quad (2)$$

where $dist(i-1,i)$ denotes the distance between the $i-1$th attraction and the ith attraction calculated based on the GPS locations. Besides, this criterion also implies that users would spend more time on staying in attractions instead of driving between attractions [5, 6].

- **Feasible.** Every attraction in the generated route should be visited or arrived at its available visiting time.
- **Customized.** Every users requirement should be satisfied, e.g. "the trip route must consist of attraction A", "arrives at attraction B in the evening of day 2", and "stay at attraction C for 3 hours".

Based on the criteria described above, the problem of generating a good trip route is to maximize $TPS(R)$ and minimize $NS(R)$ subject to the constraint that every attraction should be visited at its available visiting time interval and all user settings should be satisfied. The problem can be transformed to the Vehicle Routing Problem with Time Window (VRPTW) [7], which had been proven to be an NP-hard problem. However, in our application, to compute the travel route in real-time is essential. Inspired by [8], we propose an effective heuristic algorithm which can be executed in real-time but also yield good results.

3.2 The Proposed Heuristic Algorithm

As proposed in [8], in the beginning the travel route only consists of the start location and the end location, then the algorithm starts to insert one attraction at a time to the travel route until no more attraction can be inserted. To decide which attraction to insert, we first examine every of the rest attractions and determine which position in the current trip route is most proper to insert the examined attraction. The non-smoothness of inserting attraction A to position i is defined by

$$NS(A) = \min_i(dist(i, A) + dist(A, i + 1) - dist(i, i + 1)), \tag{3}$$

where $i = 0, ..., CurrentRouteSize-1$. Hence, the most proper inserting position of an attraction A can be determined by

$$MPIP(A) = \arg\min_i(dist(i, A) + dist(A, i + 1) - dist(i, i + 1)), \tag{4}$$

where $i = 0, ..., CurrentRouteSize - 1$. After calculating the most proper inserting position for each of the rest attractions, we decide which attraction is selected to be inserted first. Since our goal is to maximize $TPS(R)$ and minimize $NS(R)$, we define the inserting priority score of an attraction A as

$$IPS(A) = \alpha * PS(A) - \beta * NS(A), \tag{5}$$

where $\alpha, \beta \geq 0$. Each time, the attraction with the highest IPS is inserted to the original trip route if it is appropriate to be inserted into its most proper inserting position in the current route. An attraction is said to be appropriate if the new trip route still fulfills all the constraints after inserting it. The attraction insertion process will be terminated until no appropriate attractions is left, and then we obtain the final travel route. To customize the travel route, user feedback should be taken into account. With the proposed heuristic algorithm, user feedback can be easily integrated by inserting the user-specified must-go attractions prior to other attractions into appropriate position of the travel route to fullfill user-specified arrival-time (if any). Considering the real-time computation issue, we further apply the local search algorithm [7] to improve the original solution, as shown in Algorithm 1.

3.3 Transportation Budget Control

There usually exist multiple transportation choices from one attraction to another attraction, and each transportation choice may differ in the transportation duration and transportation fee. In most cases, the shorter the transportation duration is, the more expensive the transportation fee is. In other words, when users travel time and transportation budget are both limited, then the generated trip route should compromise between taking rapid but expensive transportation and taking slow but cheaper transportation. The problem becomes how to

Algorithm 1 Real-time Travel Route Recommendation Based on Local Search

Input: OTR(Original Travel Route), k(number of attractions being exchanged at a time), RA(set of the rest attractions).
Output: UTR(Updated Travel Route)

Initialize $UTR = OTR$
do
  $OTR = UTR$;
  $UTR = ExchangeWithinTripRoute(OTR, k)$;
  $ExchangeBetweenTripRouteAndRestAttractions(UTR, k, RAset)$;
while $IsImproved(OTR, UTR)$
function ExchangeWithinTravelRoute(R_o,k,R_u)
  $R_u = R_o$;
  $R_{tmp} = R_o$;
  for *every* $k-$*combination of attractions in* R_o **do**
    for *every possible exchanges among them* **do**
      $R_{tmp} =$ *travel route after exchanging*
      if $IsFeasible(R_{tmp})$ *and* $IsImproved(R_o, R_{tmp})$ **then**
        $R_u = R_{tmp}$;
        $R_o = R_u$;
      end if
      break;
    end for
  end for
  return R_u;
end function
function Exchange(R_o,k,$RAset$,R_u)
  $R_u = R_o$;
  $R_{tmp} = R_o$;
  for *every* $k-$*combination of attractions in* R_o *and* $RAset$ **do**
    for *every possible exchanges among them* **do**
      $R_{tmp} =$ *travel route after exchanging*
      if $IsFeasible(R_{tmp})$ *and* $IsImproved(R_o, R_{tmp})$ **then**
        $R_u = R_{tmp}$;
        $R_o = R_u$;
      end if
      break;
    end for
  end for
  return R_u;
end function
function IsFeasible(R)
  if *all attractions in* R *satisfy all the constrains* **then**
    return *true*;
  else
    return *false*;
  end if
end function
function IsImproved(R_o,R_u)
  if $(TPS(R_u) > TPS(R_o)$ *and* $NS(R_u) \leq NS(R_o))$ *or* $(TPS(R_u) \geq TPS(R_o)$ *and* $NS(R_u) < NS(R_o))$ **then**
    return *true*;
  else
    return *false*;
  end if
end function

Table 1. Execution time for a 150-hours-long trip plan with different number of candidate attractions.

Number of Candidate Attractions	50	100	150	200	250
Average Execution Time (secs)	0.273	0.655	1.039	1.589	2.319

choose every transportation way between every adjacent attractions pair in the trip route properly so that the generated trip route can be better?

To address the problem stated above, we propose the following algorithm. Each time we insert a new attraction into current trip route, we have to determine which transportation to take. However, at this time it is hard to decide which can yield better result in the end. Therefore, our algorithm would always choose the shortest duration but most expensive one, and keep inserting attractions regardless of whether user-specified transportation budget is exceeded until no more attractions can be inserted. Next, we start to reduce the total transportation fee to meet the transportation budget specified by the user. There are two ways to reduce the total transportation fee: (1) Replace one of the transportation way of the current trip route with another transportation choice which has longer duration but is cheaper. (2) Remove the attraction with lowest IPS from the current trip route. Our algorithm applies the first way. However, if there is no solution that meets the budget constraint, the second way will be utilized.

4 Experimental Results

In this section, we present and discuss the experiment results including the execution time of the proposed trip route planning algorithm, effectiveness of the proposed trip route planning algorithm.

4.1 Execution Time of the Proposed Trip Route Planning Algorithm

In order to evaluate whether the proposed trip route planning algorithm can compute a good trip route in real-time, we test our algorithm on the PC with Intel Dual Core i5-3210M CPU of 2.50 GHz. To simulate the user settings, we randomly generate a number of possible user queries. Then we average all the queries results to examine the average execution time. As Table 1 shows, although lots of constraints should be satisfied in order to achieve a suitable trip route, the proposed algorithm is still able to be accomplished in real-time even if the total traveling time is 150 hours long (a 7-days trip plan) and the number of attractions is 200 large.

4.2 Effectiveness of Proposed Trip Route Planning Algorithm

We crawled photos of attractions in popular travel area such as Eastern Area of Taiwan, Southern Europe and Kansai Area of Japan from Flickr to extract

Table 2. Results of trip route planning evaluated by objective criteria.

	NAV	ETR	STR	TRPSR
Popular Insert First	17.47	0.98	66.16	0.73
Proposed Algorithm	22.22	0.97	77.67	0.78
Improvements Rate	1.27	0.99	1.17	1.06

travel information of attractions. Then we facilitate Google Maps API to fetch the distance and the duration of different transportation between attractions in the same travel area. We conduct subjective test to evaluate the quality of a planned trip route. However, there are some criteria which can objectively measure the quality of a trip plan, including Number of Attractions Visited (NAV), Elapsed Time Ratio (ETR, Total Travel Duration/Users Travel Time Budget), Stay Time Ratio (STR, Time Spent on Attraction/ Total Travel Duration), and Trip Route Popular Score Ratio (TRPSR, Total Popular Score of Attractions being Visited/Maximum Total Popular Score of All Generated Trip Route) [6]. When the other three criteria are the same for the two trip routes, the trip route with one of the criteria higher than the other trip routes would be considered better.

Because there doesnt exist trip plan generating algorithm which also takes the available visiting time of attractions into account to compare with, here we implement a baseline algorithm called PopularInsertFirst which at each round would select the attraction with highest popular score to insert into the trip route. We randomly generate a number of different user settings to simulate possible users queries and then average the results. The results in Table 2 show that proposed algorithm on average performs better than Baseline Algorithm except for the ETR. However, the difference between the two algorithms is quite small and both algorithms ETR are high enough. Proposed algorithm improved the most in NAV and its very reasonable because when smoothness is taken into account, the total travel time would be reduced and therefore more attractions can be added to the trip route. For the STR, the proposed algorithm also outperforms a lot. This is because that a smooth trip route also implies that there would be less redundant path and hence the transportation duration would be shorter. It is interesting that proposed algorithm also surpassed in IDR despite that at each round the attraction chosen by PopularInsertFirst has higher popular score than the one selected by proposed algorithm. The reason is that the improvements of NAV and STR are high enough to compensate for the slightly lower popular score of each rounds selection.

Except for the objective evaluation, we also recruit 20 people who had travel experience or knowledge in the travel destination to help evaluate the quality of the trip route generated by proposed algorithm. In this part, we also want to verify the importance of taking available visiting time into consideration. Therefore we ask users to compare the algorithm with previous works that does not consider available visiting time (denoted as NCA, NotConsiderAvailability). Besides, we also compare with the RT (RandomTrip) algorithm which randomly choose at-

tractions and paths to generate a trip route. The questions we ask including: 1) Are the attractions being visited representative, interesting or popular enough? 2) Is the trip route smooth enough? 3) Is the stay time ratio reasonable? 4) Is the arrival time of attractions being visited reasonable? 5) Overall, how do you think the quality of the trip route? Users are asked to rate 1- 5 score to the above questions. For questions 1 to 4, score 1 means not agree at all and score 5 means absolutely agree. For the fifth question, score 1 means the quality is very bad, and score 5 means very good. The results are shown in Figure 4.

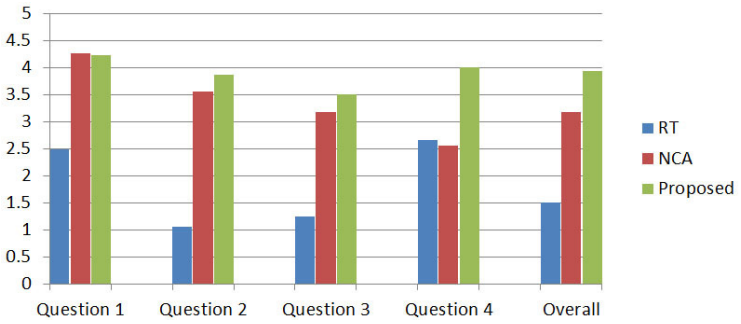

Fig. 4. Results of trip route planning evaluated by subjective questions.

5 Conclusions

A convenient system for trip planning is proposed to change the behavior of trip planners from exhaustively searching information to receiving useful travel recommendations. A traveling database is constructed by collecting photos taken around famous attractions and analyzing these photos to extract each attractions travel information including popularity, typical stay time, available visiting time in a day, and visual scenes of different time. All the extracted travel information are presented to the user to help him/her efficiently know more about different attractions in terms of visual scenes. Given the start location, destinations, traveling dates, and transportation budget information as the input, the system automatically recommends a route that suits the traveler based on a real-time routing planning algorithm and allows the user to make adjustment according to their preferences.

References

[1] Top 15 Most Popular Travel Websites, March 2013. http://www.ebizmba.com/articles/travel-websites
[2] Lu, X., et al. 2010. Photo2Trip: Generating Travel Routes from Geo-Tagged Photos for Trip Planning. ACM MM10.

[3] Jiang et al. 2011. ContextRank: Personalized Tourism Recommendation by Exploiting Context Information of Geotagged Web Photos. IEEE ICIG11.
[4] Xie et al. 2011. CompRec-Trip: a Composite Recommendation System for Travel Planning. ICDE11.
[5] Yoon H. et al. 2010. Smart Itinerary Recommendation Based on User-Generated GPS Trajectories. UIC10.
[6] Yoon H. et al. 2012. Social itinerary recommendation from user-generated digital trails. Journal of Personal and Ubiquitous Computing.
[7] Bräysy O. and Gendreau M. 2005. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. Transportation Science.
[8] Solomon, M. M. 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints, Operations Research 35, 254-265.