Trip Planning by an Integrated Search Paradigm

Sheng Wang, Mingzhao Li, Yipeng Zhang,
Zhifeng Bao, David Alexander Tedjopurnomo
School of Science, RMIT University
firstname.surname@rmit.edu.au

Xiaolin Qin
College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics
qinxcs@nuaa.edu.cn

ABSTRACT

In this paper, we demonstrate a trip planning system called TISP, which enables users' interactive exploration of POIs and trajectories in their incremental trip planning. At the back end, TISP is able to support seven types of common queries over spatial-only, spatial-textual and textual-only data, based on our proposed unified indexing and search paradigm [7]. At the front end, we propose novel visualization designs to present the result of different types of queries; our user-friendly interaction designs allow users to construct further queries without inputting any text.

ACM Reference Format:
Sheng Wang, Mingzhao Li, Yipeng Zhang, Zhifeng Bao, David Alexander Tedjopurnomo and Xiaolin Qin. 2018. Trip Planning by an Integrated Search Paradigm. In *Proceedings of 2018 International Conference on Management of Data (SIGMOD'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3183713.3193543

1 INTRODUCTION

In this paper, we would like to demonstrate a system that we have developed, TISP — a Trip planning system by an Integrated Search Paradigm. TISP helps users (even those without any prior knowledge of the target city) interactively discover a city and incrementally plan a unique trip. A preliminary version of the system[1] and a demonstration video[2] are available for public access.

Planning a trip usually involves a series of search processes, where users may issue several queries of the same type (with different settings), or even different types of queries, until the desired points of interest (POIs) and trajectories are found. In particular, for POI search, it involves the keyword query [4], k-Nearest Neighbor (k**NN**) query [5], Top-k Spatial Keyword (**TkSK**) query [9], Aggregate Nearest Neighbour (**ANN**) query [3], and Aggregate Textual Nearest Neighbour (**ATNN**) query [8]. For trajectory search, it involves the k-Best-Connected-Trajectory (k**BCT**) query [2] and Top-k Spatial-Textual Trajectory (**TkSTT**) query [6]. Detailed scenarios demonstrating the query usage are elaborated in Section 3.

Our prior study [7] has shown that standalone index or processing method for each type of query is expensive in both storage and computation cost, thus failing to provide interactive search. Therefore, we first proposed a general Top-k query called *Monotone Aggregate Spatial Keyword query* - MASK, which can cover all the above queries. Then we proposed a unified index and query processing paradigm to answer various types of location-based queries on both spatial point data and spatial trajectory data. This results in the first feature of TISP.

The second feature of TISP is an interactive data exploration. In particular, we design an interactive visualization procedure to help users explore various data objects and plan their trips. On one hand, we propose a series of novel visual encoding designs to visualize the result of different types of queries. E.g., we design an enhanced transfer graph with careful retinal channel mapping of vertices and edges (Figure 2) to visualize popular attractions and attraction pairs. On the other hand, we design user-friendly interactions upon multiple visualization views, which heuristically guide users to interact with different visualization results, construct further queries without inputting any text, and gradually complete their planned trip (Section 4).

Since there is no system from academia serving similar purposes to TISP, we particularly compare TISP with commercial trip planning platforms. After trying to plan trips using several popular commercial trip planning platforms, we have two main observations. (1) Some systems like TripAdvisor[3] only allow users to search for POIs (e.g., attractions, restaurants, hotels) rather than trips. Moreover, TripAdvisor only supports searching hotels in a region or near a single attraction, but users cannot search hotels close to multiple attractions in their planned trip. (2) Some systems (such as TripHobo[4] and Google Trips[5]) maintain a number of popular journeys for each city, and allow users to search by the city name and duration of travel, then existing trips that meet the requirements are presented to users; however, they do not support searching by specified preferences, such as attractions that are covered by any single previous journey, or activities that users want to do around the attractions. Hence, TripHobo and Google Trips can only support trip search partially. More details about the supported functions of these three platforms are presented in Table 1.

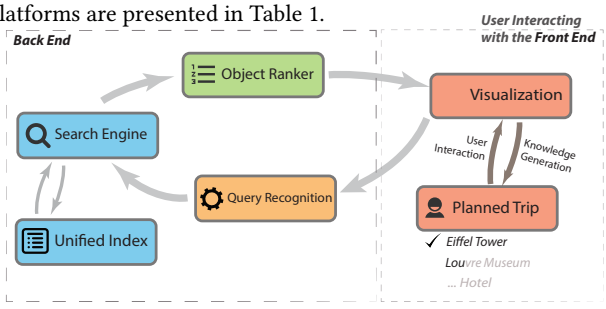

Figure 1: System architecture of TISP.

[1] https://sites.google.com/site/shengwangcs/tisp
[2] https://youtu.be/5vB22ZR8kvk
[3] https://www.tripadvisor.com/
[4] https://www.triphobo.com/
[5] https://www.google.com/trips/

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3193543

2 SYSTEM ARCHITECTURE

As shown in Figure 1, TISP includes a front end of visual interface and a back end of query processing. At the front end, TISP presents information to users in a visually effective way (Figure 4). Each new user interaction over the interface will be recognised as a specific query type at the back end (see Figure 3 for classification of queries), and a search engine is employed to answer the query on top of a unified index. With the query result returned by the *Object Ranker*, different views of the visualization results at the front end will be partially refreshed to provide new information, which allows users to acquire new knowledge in completing their trips.

3 BACK-END TECHNIQUES

Here we describe how our unified indexing and query processing paradigm facilitate answering various queries, and data structures for visual explorations. Please refer to our work [7] for details.

3.1 Preliminaries

Data Model. We maintain two kinds of data objects in our system. One is the points of interest (*POIs*), which include attractions, hotels and restaurants. Each POI $p = (\rho, \psi)$ is a pair consisting of the location ρ and a set of associated terms $\psi = (t_1, t_2, \ldots, t_i)$ which describes ρ and/or users' activities at ρ. The other one is a trip T which is a set (or sequence) of attractions, i.e., $T = \{p_1, p_2, \cdots, p_l\}$.

Transfer Graph. A transfer graph $G = (V, E)$ is composed of an attraction vertex set V and an edge set E. Each edge $e \in E$ connects two attraction vertices $v_1, v_2 \in V$, and it is attached with the weight to indicate the popularity of the edge. The weight is computed based on the number of historical trips that cross v_1 and v_2. Each attraction vertex v is also attached with a weight indicating how many trips cover the vertex.

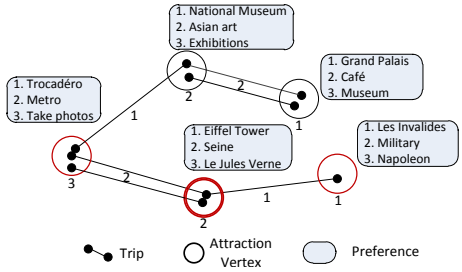

Figure 2: The transfer graph composed by vertices and edges.

Example 3.1. Figure 2 shows a graph built from the historical trips. There are 6 vertices, each one is attached with a preference list. A user can click the vertex to check the associated activities. The linked vertices are highlighted if these two attractions are frequently visited by a majority of tourists.

3.2 An Integrated Search Paradigm

We describe the explorations using three common scenarios in trip planning, i.e., exploring attractions, restaurants and hotels. Table 1 shows the main query types in each scenario. Note that, more query types [1] can be supported in TISP incrementally, while this paper focuses on seven most common types of query which can be answered efficiently by our integrated search paradigm.

Table 1: An overview of queries, their matching scenarios (S1, S2, S3) and supported systems: TH (TripHobo), TA (TripAdvisor), GT (Google Trips), ★: partially supported. (KS: Keyword Search, kBCT: k Best Connect Trajectories, TkSTT: Top-k Spatial Textual Trajectory, kNN: k Nearest Neighbor, TkSK: Top-k Spatial Keyword, ANN: Aggregate Nearest Neighbor, ATNN: Aggregate Textual Nearest Neighbor.)

	Exemplar Demand	Query Type	Input	Output	TH	TA	GT
S1	Cultural Attractions	**KS** [4]	Keywords	POIs	✓	✓	★
	Top-k Trips	**kBCT** [2]	Locations	Trips	★	✗	★
	Top-k Activity Trips	**TkSTT** [6]	Locations, keywords	Trips	✗	✗	✗
S2	k Nearest Restaurants	**kNN** [5]	Location	POIs	✓	✓	✓
	k French Restaurants	**TkSK** [9]	Location, keywords	POIs	✗	✓	✓
S3	k Nearest Hotels to My Trip Attractions	**ANN** [3]	Locations	POIs	✗	✗	✗
	k Nearest Hotels with Pool to My Trip Attractions	**ATNN** [8]	Locations, keywords	POIs	✗	✗	✗

Query & Similarity Model. According to the user input in Table 1, we deduce an appropriate type of query to be triggered. Specifically, a query Q is in form of $\{Q_s, Q_t\}$, where $Q_s = \{\rho_1, \rho_2, \cdots, \rho_m\}$ is the query location set, and $Q_t = \bigcup_{i=1}^{m} \psi_i$ is the query keyword set, i.e., $m = |Q_s|$ and $n = |Q_t|$. To judge whether a POI or a trip is related to the user query, without loss of generality, we employ a monotone aggregate similarity function [7] as below:

$$\hat{S}(Q, o) = \alpha \cdot \sum_{\rho \in Q_s} SPr(\rho, o.\rho) + (1 - \alpha) \cdot \sum_{\psi \in Q_t} TRe(\psi, o.\psi) \quad (1)$$

where $SPr(\rho, o.\rho)$ is the spatial proximity between $o.\rho$ and ρ, $TRe(\psi, o.\psi)$ is the textual relevance between ψ and $o.\psi$, and $\alpha \in [0, 1]$ is a query preference parameter that balances the spatial proximity and textual relevance. Figure 3 shows the query recognition process according to the key parameters in the query and the data model.

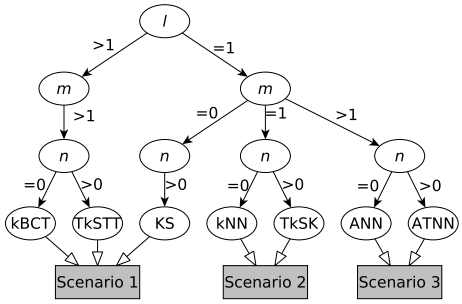

Figure 3: Query recognition according to user input (l, m, n), and the relationships between the queries and scenarios.

Unified Index. The index is composed by a grid index $\mathcal{I}_G$, a textual inverted list $\mathcal{I}_T$, and a point list $\mathcal{I}_P$.

1) $\mathcal{I}_T$ sorts the keyword list in a descending order of term weight, and divides the list into blocks of fixed size. TextualIterator $(q_i, \mathcal{I}_T)$ is defined as an operator to scan the list.

2) In $\mathcal{I}_G$, the whole space is divided into grids of equal size, and each leaf cell is assigned with a unique id using the z-curve cross-coding [9]. SpatialIterator $(q_i, \mathcal{I}_G)$ is defined to access nearby points from near to distant.

3) A trip is a set (or sequence) of points, and the proposed query processing method is built at point granularity. Therefore, we create an inverted index $\mathcal{I}_P$ to store a mapping from an attraction POI to its corresponding trips. Based on CoveredObjects $(R(q_i), \mathcal{I}_P)$, the trips that a point belongs to will be returned efficiently.

Algorithm 1: Search(D, Q, k)

Output: Top-k result set $\mathcal{R}$

```
1  R ← ∅; D_c ← ∅;                          // Result, Scanned object set
2  while |D_c| < |D| do
3      foreach q_i ∈ Q do
4          R(q_i) ← TextualIterator(q_i, I_T);
5          R(q_i) ← R(q_i) ∪ SpatialIterator(q_i, I_G);
6      D_c ← D_c ∪ ⋃_{i=1}^{|Q|} CoveredObjects(R(q_i), I_P);
7      if |D_c| > k then
8          if Ŝ↓(Q, R) > Ŝ↑(D − D_c) then
9              R ← RefineCandidate(D_c);
10             return R;
```

Search Engine. The main idea is to conduct filtering to get all the candidates first, followed by a refinement on them to get the final top-k results. With $\mathcal{I}_G$ and $\mathcal{I}_T$, we keep scanning for new points which are related to the query keywords and are close to the locations mentioned in the query. Then, we find the related trips based on $\mathcal{I}_T$. Based on the similarity model, we compute the upper bound similarity $\hat{S}^{\uparrow}(D - D_c)$ for all unseen trips or POIs, then compare it with the top-k results' lower bound similarity $\hat{S}^{\downarrow}(Q, \mathcal{R})$, which is a common filter-refine framework employed by almost all existing studies [2, 6, 9]. If the unseen upper bound is smaller than the current lower bound, we can terminate the whole search process and return the top-k results. A complete solution is presented in Algorithm 1. More technical details on bound computations can be found in our previous work [6, 7].

4 DEMONSTRATION SCENARIOS

We will demonstrate TISP based on real-world datasets and scenarios. We crawled POIs (restaurants and hotels) from TripAdvisor Paris[6], and acquired 3,249 trips from TripHobo Paris[7], which cover 197 attractions. A large-scale data collection and integration for more other cities are available at onsite demonstration.

4.1 An Overview of User Interface

TISP is initialized with a transfer graph on top of the map to present an overview of popular tourist attractions in the target city. As shown in Figure 4(b), each circle represents an attraction, where the size and color of the circle represent the popularity and type of the attraction, respectively. A pair of attractions (i.e. other attractions B that travellers also visit together with attraction A) are linked with edges, with the width of each edge illustrating the popularity of the attraction pair. Users can also select preferred attraction types (e.g. "*cultural*") from the selection panel (Figure 4(a)). Next we will describe four scenarios when *Grace* uses TISP to plan a trip.

4.2 Scenario 1: Exploring Attractions

Grace can click on an attraction in the transfer graph, and a detailed view (Figure 4(c)) will appear with a word cloud describing the popular activities at the selected attraction. The clicked activity in the word cloud will be added to her favourite activity list. Grace can also filter attractions by the attraction filter: a keyword search (**KS**) will be conducted at the back end, and the transfer graph will be refreshed based on the results after filtering.

Clicking an attraction in the transfer graph will also trigger the *pair pattern view* (Figure 4(d)), which visualizes what other attractions that travellers often visit together with the selected attraction. In this view, the selected attraction is placed in the center, and other attractions are placed in a clockwise fashion based on the popularity of the attraction pair. Grace can iteratively double click to select another attraction and add it to the centre of the view. The *pair pattern view* will be updated, showing the attractions that travellers normally visit together with the two currently selected attractions. As we can see from Figure 4(d), among all the attractions, "River Seine" is the most popular attraction that travellers often visit together with "Eiffel Tower".

However, the pair pattern view is not enough to discover all related attractions as it only shows the neighbor vertices of current attraction. Alternatively, Grace can search for previous travellers' trips by the attractions and activities in her own favourite list. The k**BCT** [2] and **T**k**STT** query [6] will be conducted and a list of trips will be shown in the *trip list view* (Figure 4(e)). Each line is a different trip ranked by the query result, and each trip is visualized with linked attractions: the length and the width of the edge are defined by the travel time between two adjacent attractions, and the popularity of the attraction pairs, respectively; the opacity of the vertex is determined by whether the attraction is in Grace's list.

4.3 Scenario 2: Exploring Facilities Nearby

In the *detailed view* of an attraction (Figure 4(c)), Grace can also click on the "*Nearby restaurants*" button; then a k**NN** search [5] over the restaurant dataset will be conducted at the back end. Afterwards, nearby restaurants will be shown on top of the map. Meanwhile, a word cloud is generated to describe popular features (e.g. "French cuisine", "gluten free") of the nearby restaurants. Grace can choose the features by directly clicking on the individual feature in the word cloud. Next, an advanced query **T**k**SK** [9] will be generated to find the restaurants which contain the chosen features.

4.4 Scenario 3: Exploring Hotels

After Grace has added some attractions/restaurants into her favorite list, she might consider finding a hotel that is convenient for her to visit most of the attractions. Grace can click on the "*Find hotels*" button, an aggregate nearest neighbor (**ANN**) [3] query will be conducted to find those hotels with the minimum aggregate distance to all selected attractions (Figure 6). At the front end, a list of hotels will be shown on top of the map. Meanwhile, a word cloud will be generated based on popular keywords related to hotels (e.g. "kitchen", "pool"). If Grace selects some keywords, she will trigger an aggregate textual nearest neighbor (**ATNN**) [8] query. Since price is an important factor for users to choose an ideal hotel, TISP also provides a slider (Figure 4(a)) for users to define the price range.

4.5 Scenario 4: Optimal Travel Route

Once Grace has gathered a series of discrete attractions and activities, restaurants, and hotels related to each attraction, the last step is

[6] https://www.tripadvisor.com/Tourism-g187147-Paris_Ile_de_France-Vacations.html

[7] https://www.triphobo.com/tripplans/paris

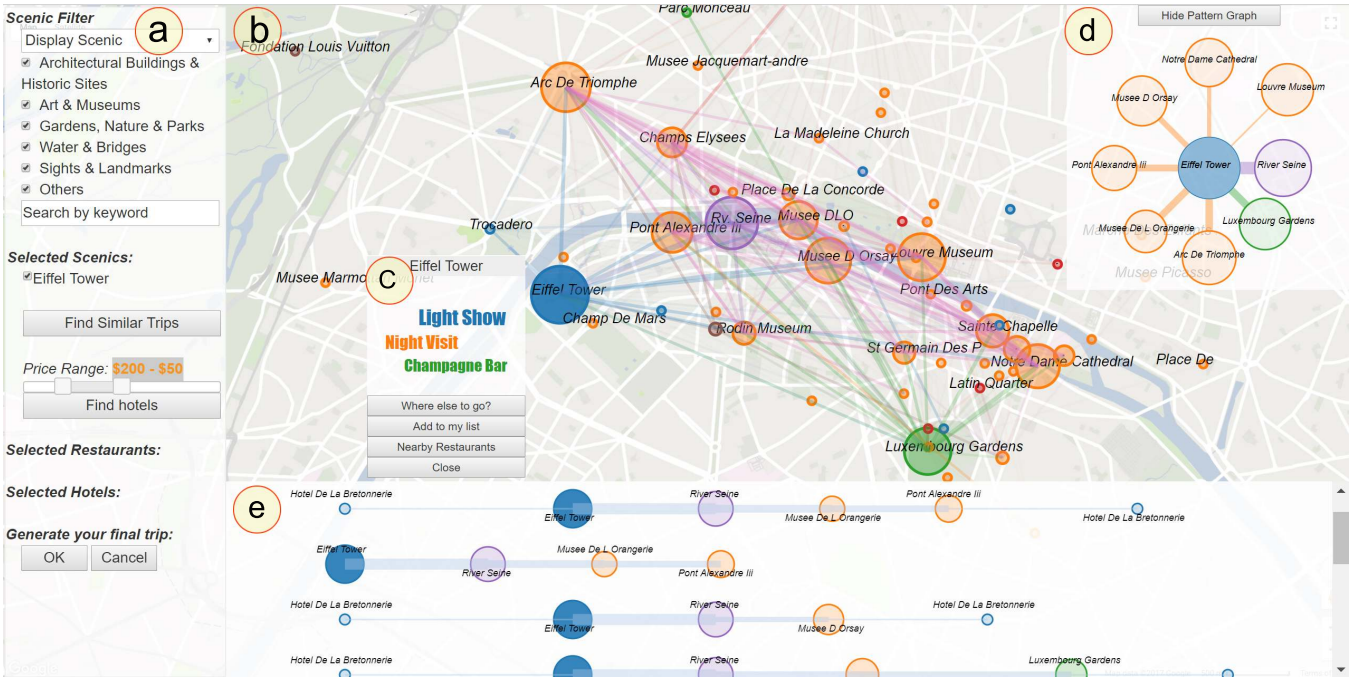

Figure 4: An overview of the main user interface of TISP: (a) the selection panel, which allows users to filter attractions and manage their favorite list; (b) the map view with an enhanced transfer graph; (c) the detailed view initialized when the user click on an attraction; (d) the pair pattern view to display popular attraction pairs of a selected attraction; (e) the trip list view.

Figure 5: Scenario 2: Searching the 5 nearest restaurants to the *Eiffel Tower*.

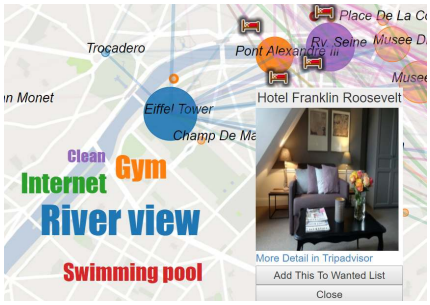

Figure 6: Scenario 3: Searching the top five hotels near to the planned trip.

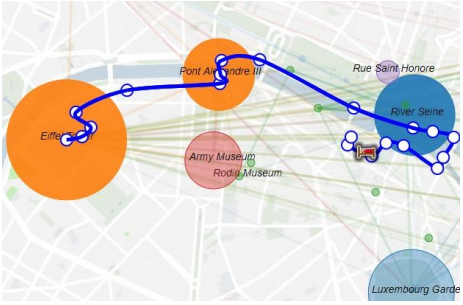

Figure 7: Scenario 4: Optimal route that connects the selected attractions and hotel.

to connect them and recommend the optimal routes, which aims to achieve the minimum travel time or distance, which is similar to the *travelling salesman problem*. Grace can select it as a final planned trip from the suggested options, or modify it by dragging the line. As shown in Figure 7, TISP suggests a route of the shortest path on the road network based on the Google Map API. The route covers all her favored attractions. Moreover, connecting all the searched POIs as an optimal trip can be a novel optimizing problem rather than a shortest path search, such as minimizing the trip duration (days) or the budget which is also supported.

ACKNOWLEDGMENTS

This work was partially supported by ARC DP170102726, DP180102050, and National Natural Science Foundation of China (NSFC) 61728204, 91646204. Zhifeng Bao is supported by a Google Faculty Award.

REFERENCES

[1] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. *PVLDB*, 6(3):217–228, 2013.
[2] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations-an efficiency study. In *SIGMOD*, pages 255–266, 2010.
[3] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database Systems*, 30(2):529–576, 2005.
[4] L. Qin, J. X. Yu, and L. Chang. Keyword Search in Databases: The Power of RDBMS. In *SIGMOD*, pages 681–693, 2009.
[5] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: a dynamic index for multi-dimensional objects. In *VLDB*, pages 507–518, 1987.
[6] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, M. Sanderson, and X. Qin. Answering top-k exemplar trajectory queries. In *ICDE*, pages 597–608, 2017.
[7] S. Wang, Z. Bao, S. Huang, and R. Zhang. A unified processing paradigm for interactive location-based web search. In *WSDM*, pages 601–609, 2018.
[8] K. Yao, J. Li, G. Li, and C. Luo. Efficient group top-k spatial keyword query processing. In *APWeb*, pages 153–165, 2016.
[9] D. Zhang, C.-Y. Chan, and K.-L. Tan. Processing spatial keyword query as a top-k aggregation query. In *SIGIR*, pages 355–364, 2014.