STS: Complex Spatio-Temporal Sequence Mining in Flickr[⋆]

Chunjie Zhou and Xiaofeng Meng

School of Information, Renmin University of China, Beijing, China
{lucyzcj,xfmeng}@ruc.edu.cn

Abstract. Nowadays, due to the increasing user requirements of efficient and personalized services, a perfect travel plan is urgently needed. In this paper we propose a novel complex spatio-temporal sequence (STS) mining in Flickr, which retrieves the optimal STS in terms of distance, weight, visiting time, opening hour, scene features, etc.. For example, when a traveler arrives at a city, the system endow every scene with a weight automatically according to scene features and user's profiles. Then several interesting scenes (e.g., $o_1, o_2, o_3, o_4, o_5, o_6$) with larger weights (e.g., $w_1, w_2, w_3, w_4, w_5, w_6$) will be chosen. The goal of our work is to provide the traveler with the optimal STS, which passes through as many chosen scenes as possible with the maximum weight and the minimum distance within his travel time (e.g., one day). The difficulty of mining STS lies in the consideration of the weight of each scene, and its difference for different users, as well as the travel time limitation. In this paper, we provide two approximate algorithms: a local optimization algorithm and a global optimization algorithm. Finally, we give an experimental evaluation of the proposed algorithms using real datasets in Flickr.

Keywords: spatio-temporal, sequence, Flickr, approximate.

1 Introduction

With the rapid development of modern society, people are concentrating more on efficient and personalized services. In the tourist industry, a perfect traveling plan can help people to visit their favorite scenes as many as possible, and save a lot of time and energy. However, at present it is hard for people to make a proper and personalized traveling plan. Most of them follow other people's general travel trajectory, but do not consider their own profile and the best visiting order of scenes in this trajectory. So only after finishing their travel, do they know which scene is their favorite, which is not, and what is the perfect order of visits. Let's consider such a scenario: a person plans to travel on a holiday, but does not have a specific destination. In order to make a better plan, they scan the tourist routes on the Internet or they seek advice from travel companies. Then they choose a popular travel trajectory suggested by other people, but do not consider their own interests. As a result, this sightless travel plan may cause the following aftereffects: 1) waste a lot of time on the road among scenes; 2) waste lots of

⋆ This research was partially supported by the grants from the Natural Science Foundation of China (No.60833005, 61070055, 61003205); the National High-Tech Research and Development Plan of China (No.2009AA011904).

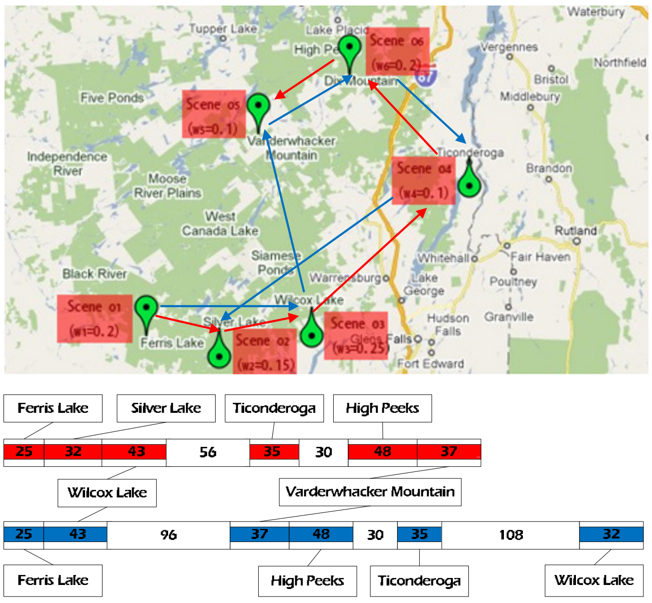

Fig. 1. Two different sequences (red, blue) including six chosen scenes with different weights on Google Maps and their timelines. Scenes of each sequence which have been visited are presented in the order of visitation. On the timelines, transitions between tourist scenes are depicted by rectangles and visiting time is given in minutes.

unnecessary money; 3) do not have enough time to visit their real favorite scenes, etc. However, with increasing interests in perfect travels and modern advanced services, more wonderful and personalized travel suggestions need to be supplied urgently.

In this paper, we propose a novel spatio-temporal sequence (STS) mining in Flickr. The goal of STS is to provide a user with the optimal STS, which minimizes the total traveling distance and maximizes the total weight within his limited travel time. In consequence a user can make a perfect and personalized travel plan based on his own profile before he starts to travel. The implementation of STS is based on two basic pieces of knowledge. 1) the user's profile. The methods of mining user's profile have been studied a lot [1]. Here we assume that every user's profile is already stored in his accompanied mobile devices. So when a user arrives at a city, the system can get his profile from his mobile devices directly. 2) scene features. The methods of mining scene features according to photos and tags in Flickr has been studied in our previous work [6]. So here we directly use the database in [6] that stores the features of each scene in each city. In this paper, we consider solutions for the STS problem as graphs where tourist scenes are nodes and paths between these scenes are edges. To the best of our knowledge, no prior work considers both point weight and edge distance together, which are inversely proportional.

Efficient STS evaluation could become a new important feature of advanced services in Flickr, and be useful for LBS (Location Based Services). The quality of these services can be greatly improved by supporting more advanced query types, like STS. An example of STS is shown in Figure 1. When a traveler arrives at a city, the system endows every scene with a weight automatically according to scene features and user's profiles. Then six interesting scenes (e.g., $o_1, o_2, o_3, o_4, o_5, o_6$) with larger weights (e.g., $w_1, w_2, w_3, w_4, w_5, w_6$) will be chosen. Given these six certain scenes, a database that stores the features of scenes will compute a proper STS (i.e., the red sequence) efficiently. The blue sequence is created by a user who follows others' general trajectory. These two different sequences are presented in Figure 1. Tourist visits cover not only the visiting time of scenes, but also the transitional cost among scenes (i.e., rectangles on the timelines)[16]. Because of the different visiting order of the scenes, the duration of these two sequences is variable, from under six hours (e.g., the red sequence) to over 9 hours (e.g., the blue sequence). Clearly, an ideal method is to propose an optimal tourist sequence that not only re-arranges the order in which these scenes are visited with the maximum value (see Section 2.1), but also should be within a limited travel time. For the red sequence, there are only two significant transitions (i.e., 56 min between Wilcox Lake and Ticonderoga, 30 min between Ticonderoga and High Peeks). The first three visited tourist scenes (Ferris Lake, Silver Lake and Wilcox Lake) and the last two ones (High Peeks and Varderwhacker) are adjacent and the tourist makes no pause between them. Some factors influencing visiting order, for example, personal preferences (i.e., scene weight) and travel time must be taken into account.

STS can be considered as a special case of the knapsack problem (KSP) which is NP-hard. The reduction from STS to KSP is straightforward. Given a set of m scenes from which we select some interesting scenes to be included in the spatio-temporal sequence in a limited travel time T_{total}. Each scene has a weight and a duration time. The objective is to choose the set of ordered scenes that optimize the STS and maximize the travel value (i.e., a minimum distance and a maximum tourist scene weight). By regarding distance as the multiplication of velocity and time, each scene of STS can be reduced to an item of KSP. There are also some differences between STS and KSP: 1) the goods in KSP is disordered, but the scenes in STS is strictly ordered; 2) KSP has only one objective function, but STS has two independent objective functions.

Contributions: This paper proposes a novel spatio-temporal sequence mining in Flickr and studies methods for solving it efficiently. Two approximate algorithms that achieve both local and global optimization are presented. In particular:

– We present a novel STS mining in Flickr, which can minimize the total traveling distance and maximize the total weight within a limited travel time. This type of mining has not been considered before.
– We give a formal definition of STS in road network. The weights of chosen scenes are specified according to personal preferences. This is more similar to the real world applications.
– We propose two types of algorithms for STS. Local optimization algorithms include approximation in terms of distance, weight and value respectively.
– We perform an extensive experimental evaluation of the proposed algorithms on real datasets in Flickr.

Paper Organization: The rest of the paper is organized as follows: Section 2 gives the problem definition and related works. The approximate algorithms are presented in Section 3. An experimental evaluation of the proposed algorithms using real datasets is presented in Section 4. Finally, we give the conclusion and future works.

2 Preliminaries

This section formally defines the STS problem and introduces the basic notation that will be used in the rest of the paper. Furthermore, a concise overview of related works is presented.

2.1 Problem Definition

Table 1 lists the main symbols we use throughout this paper. Following standard notation, we use capital letters for sets (e.g., P is a set of all scenes), and lowercase letters for vectors (e.g., o_i).

Table 1. Symbols

Symbol	Definition and Description
V	the value of the sequence $\{ o_1,...,o_k \}$
P	a set of all scenes $\{ P_1,...,P_m \}$
R	a chosen subset of scenes $\{ o_1,...,o_k \}$
Q	a travel sequence
N	the road network
T_{total}	the total travel time
$T(o_i,o_j)$	the time cost from the scene o_i to o_j
$T(o_i)$	the duration time of scene o_i
D_{dis_N}	the distance among scenes in road network
w_i	the weight of the chosen scene o_i
o_i	the i^{th} chosen scene
α	a balance factor
m	the number of all scenes
k	the number of chosen scenes

We consider solutions for the STS problem as graphs where tourist scenes are nodes (labeled with scene's name) and paths between these scenes are edges. Given a graph G(O,ξ) with n nodes O= $\{o_1,...,o_n\}$ and s edges ξ = $\{e_1,...,e_s\}$, each node in the graph has a weight denoting the interest percentage of the traveler. The value of traversing a scene sequence $(o_i,...,o_j)$ is expressed as V $(o_i,...,o_j) \geq 0$, which means the score of the sequence. As shown in Eq.(1), the value of the sequence is in proportion to the total weight of all chosen sences, but in contrast to the total distance. Here, we consider distance among scenes in road network, which is more meaningful in the real travel application scenario. Suppose that the average velocity of the traveler is υ, the distance $D_{dis_N}(o_i,o_j)$ can be denoted as $\upsilon * T(o_i,o_j)$. A balance factor α is defined between weight and distance, which may be changed in different situations.

$$V(o_i,\ldots,o_j) = \alpha * (w_i + \ldots + w_j) + (1-\alpha) * \frac{1}{\sum_{k=i}^{j} D_{dis_N}(o_k, o_{k+1})}$$
$$= \alpha * \sum_{k=i}^{j} w_k + (1-\alpha) * \frac{1}{\sum_{k=i}^{j} \upsilon * T(o_k, o_{k+1})} \quad (1)$$

Given a set of m scenes $P = \{P_1,\ldots,P_m\}$ (where $m \leq n$) and a mapping function π: $P_j \rightarrow o_i$ that maps each scene $P_j \in P$ to a node $o_i \in N$. So scenes can be regarded as special nodes, and be denoted by node symbols. In the rest of this paper, scenes and nodes will share the same symbols. The STS problem can be defined as follows:

Definition 1. *Given a set $R \subseteq P$ ($R = \{o_1, o_2, \ldots, o_k\}$), a source scene S and a destination scene E, identify the traveling scene sequence $Q = \{S, o_1, \ldots, o_k, E\}$ from S to E that visits as many scenes in R as possible (i.e., maximize the total weight of the sequence), and takes the minimum possible distance $D_{dis_N}(Q)$ (i.e., for any other feasible sequence Q' satisfying the condition $D_{dis_N}(Q) \leq D_{dis_N}(Q')$) in a limited travel time T_{total}.*

The time constraint condition is in the following, which includes not only the duration time of scenes, but also transitional cost among scenes. The duration time of scenes can be achieved by [16], which is not our focus in this paper. We mainly consider the temporal cost among scenes, namely, the distance between each couple of scenes.

$$\sum_{i=1}^{k} T(o_i) + \sum_{i=1, j=i+1}^{k} T(o_i, o_j) \leq T_{total} \quad (2)$$

2.2 Related Work

Rattenbury et al. [2] was an early attempt to discover both event and place names from Flickr geolocated textual metadata, resulting in an application [3] for geographic image retrieval, with representative and popular tags overlaid on a scalable map. Quack et al. [4] downloaded 200,000 georeferenced Flickr images from nine urban areas and clustered them using local image descriptors to discover place names and events, linking some places to their Wikipedia articles. In contrast to [4] and [5], we do not limit ourselves to geographic information of photographs since temporal information are also important for mining STS. Elsewhere [6], we detailed methods for mining the features of each scene in each city. Here we exploit these same results but shift our focus towards mining spatio-temporal sequence according to personal preferences.

Zheng et al. [7] recorded GPS tracks of 107 users during a year to determine the interestingness of tourist scenes. Cao et al. [8] presented techniques capable of extracting semantic locations from GPS data. The authors of [9] also focused on mining similar traveling sequences from multiple users' GPS logs while the authors of [10] retrieved maximum periodic patterns from spatio-temporal metadata. Girardin et al. [11] analyzed the tourist flows in the Province of Florence, Italy, based on a corpus of georeferenced Flickr photos and their results contribute to understanding how people travel.

Chen et al. [12,13] studied a problem of searching trajectories by locations, and the target was to find the K best-connected trajectories from a database such that it connected the designated locations geographically. None of these approaches considers scene features combined with user's profile, which are central pieces of our approach. Whereas [7] or [9] relied on accurate GPS traces for small scale regions and obtained from a relatively reduced number of users. Flickr data is noisy, but covers most interesting tourist regions of the world. As a result, we are able to propose itineraries in any region of the world that is sufficiently covered by Flickr data.

Visiting duration is an important characteristic of trips and it is classically estimated by domain experts [14]. The automatic extraction of visiting duration from Flickr metadata was only recently explored [15] but no separation between sightseeing and sightseeing + interior visits was proposed. Building on this latter work, Popescu et al. [16] used visual image classification to separate these two types of visits and to calculate typical visiting time of each case. In this paper, we use the method in [16] directly to get the visiting time of scenes, which is not our focus here. We mainly consider the time cost among scenes, namely, the distance between each couple of scenes.

Researches in spatial databases also address applications in spatial networks represented by graphs, instead of the traditional Euclidean space. Recent papers that extended various types of queries to spatial networks were [17]. Clustering in a road network database has been studied in [18], where a very efficient data structure was proposed based on the ideas of [19]. Li et al. [20] discussed a trip planning query in both Euclidean space and road network, which retrieved the best trip passing through at least one point from each category. However, they did not consider the point weight and the order of points. Likewise, we also study the STS problem in road network.

3 Approximation Algorithms

In this section we present two approximate algorithms for answering the spatio-temporal sequence mining.

3.1 Local Optimization Algorithms

Three local optimization algorithms in terms of distance, weight and value will be provided in the following.

Approximation in terms of distance: The most intuitive algorithm for solving STS is to form a sequence by iteratively computing the $\lceil m/2 \rceil$ nearest neighbor scenes of the current scene, comparing the value of them, choosing the scene whose value is maximum from all scenes that have not been visited yet. Then refresh the total time by adding this scene's duration time and the time cost between this scene and the last scene. If the total time is less than T_{total}, add this scene to the sequence; else, restore the total time and ignore this scene. Formally, given a partial sequence Q_k with $k<m$, Q_{k+1} is obtained by inserting the scene o_{k+1} whose value is larger than any scene in the $\lceil m/2 \rceil$ nearest neighbor of o_k. Meanwhile, this scene should not been covered yet and satisfy the time limitation. In the end, the final sequence is produced by connecting o_k to E. We call this algorithm d-LOA , which is shown in Algorithm 1.

Algorithm 1. d-LOA

```
Input:
    The start scene, $o = S$;
    The end scene, $E$;
    The set of scene IDs, $I = \{1, ..., m\}$;
    The initial spatio-temporal sequence, $Q_a = \{S\}$;
    The limited time, $T_{total}$;
Output:
    The d-local optimal spatio-temporal sequence, $Q_a$;
 1: $loc = S$;
 2: $t = T_{total}$;
 3: while $(I\ is\ not\ empty)\ and\ (t > 0)$ do
 4:     define an array $DS$ for storing the distances from $loc$ to other scenes;
 5:     for each $n \in I$ do
 6:         $DS(loc,n) = D_{dis_N}(loc,n)$;
 7:     end for
 8:     $HI$ = the set of $\lceil m/2 \rceil$ smallest $DS$ scene IDs;
 9:     define an array $V$ for storing the values from $loc$ to other scenes;
10:     for each $n \in HI$ do
11:         if $T(n)+T(loc,n) \leq t$ then
12:             $V(loc,n) = \alpha*(w_{loc}+w_n)+(1-\alpha)*\frac{1}{DS(loc,n)}$;
13:         end if
14:     end for
15:     $o$ = the scene whose value is maximum in $V$;
16:     $loc = o$;
17:     pop $o$ from $I$;
18:     put $o$ to $Q_a$;
19:     $t = t-(T(o)+T(loc,o))$;
20: end while
21: $Q_a \leftarrow \{$ E $\}$;
```

Algorithm 2. w-LOA

```
Input:
    The start scene, $o = S$;
    The end scene, $E$;
    The set of scene IDs, $I = \{1, ..., m\}$;
    The initial spatio-temporal sequence, $Q_a = \{S\}$;
    The limited time, $T_{total}$;
Output:
    The w-local optimal spatio-temporal sequence, $Q_a$;
 1: $loc = S$;
 2: $t = T_{total}$;
 3: while $(I\ is\ not\ empty)\ and\ (t > 0)$ do
 4:     for each $n \in I$ do
 5:         $HI$ = the set of $\lceil m/2 \rceil$ largest $w_i$ scene IDs;
 6:     end for
 7:     define an array $V$ for storing the values from $loc$ to other scenes;
 8:     for each $n \in HI$ do
 9:         if $T(n)+T(loc,n) \leq t$ then
10:             $V(loc,n) = \alpha*(w_{loc}+w_n)+(1-\alpha)*\frac{1}{DS(loc,n)}$;
11:         end if
12:     end for
13:     $o$ = the scene whose value is maximum in $V$;
14:     $loc = o$;
15:     pop $o$ from $I$;
16:     put $o$ to $Q_a$;
17:     $t = t-(T(o)+T(loc,o))$;
18: end while
19: $Q_a \leftarrow \{$ E $\}$;
```

Approximation in terms of weight: Another algorithm for solving STS is to form a sequence by iteratively performing the following operations. Choose the $\lceil m/2 \rceil$ maximum weight scenes, which connect to the current scene and have not been visited yet. Compare the value of these $\lceil m/2 \rceil$ scenes, and select the scene whose value is maximum. Then refresh the total time by adding this scene's duration time and the time cost between this scene and the last scene. If the total time is less than T_{total}, add this scene to the sequence; else, restore the total time and ignore this scene. We call this algorithm w-LOA, which is similar to d-LOA and shown in Algorithm 2.

Approximation in terms of value (i.e., distance and weight): A hybrid local optimization algorithm for solving STS is to form a sequence by iteratively performing the following operations. Compute the values between the current scene and every other scenes that have not been visited yet. Choose the scene whose value is maximum. Then refresh the total time by adding this scene's duration time and the time cost between this scene and the last scene. If the total time is less than T_{total}, add this scene to the sequence; else, restore the total time and ignore this scene. Formally, given a partial sequence Q_k with $k<m$, Q_{k+1} is obtained by inserting the scene o_{k+1} whose value is larger than any scene in R. Meanwhile, this scene should not been covered yet and satisfy the time limitation. In the end, the final sequence is produced by connecting o_k to E. We call this algorithm v-LOA, which is shown in Algorithm 3.

Algorithm 3. v-LOA (G,R,S,E)

```
Input:
    The start scene, $o = S$;
    The end scene, $E$;
    The set of scene IDs, $I = \{1,...,m\}$;
    The initial spatio-temporal sequence, $Q_a = \{S\}$;
    The limited time, $T_{total}$;
Output:
    The v-local optimal spatio-temporal sequence, $Q_a$;
 1: $loc = S$;
 2: $t = T_{total}$;
 3: while ($I$ is not empty) and $(t > 0)$ do
 4:     define an array $V$ for storing the values from $loc$ to other scenes;
 5:     for each $n \in I$ do
 6:         if $T(n)+T(loc,n) \leq t$ then
 7:             $V(loc,n)=\alpha*(w_{loc}+w_n)+(1-\alpha)*\frac{1}{DS(loc,n)}$;
 8:         end if
 9:     end for
10:     $o$ = the scene whose value is maximum in $V$;
11:     $loc = o$;
12:     pop $o$ from $I$;
13:     put $o$ to $Q_a$;
14:     $t = t-(T(o)+T(loc,o))$;
15: end while
16: $Q_a \leftarrow \{$ E $\}$;
```

3.2 Global Optimization Algorithm

This section introduces a novel heuristic algorithm, called GOA. This algorithm achieves a much better result in comparison with the previous algorithms. GOA can find the optimal sequence if the heuristic function never overestimates the actual minimal value of reaching the goal. Here we select the heuristic as the value in Euclidean space, as it is always less than or equal to the actual value in road network in this scenario (see Definition 2). This can guarantee the sequence optimality in terms of road network value.

Definition 2. *For two scenes u and v ($u, v \in N$), $D_{dis_N}(u,v)$ is the road network distance, and $D_{dis_E}(u,v)$ is the Euclidean distance. Correspondingly, $V_N\ (u,v)$ denotes the value of the sequence from u to v in road network, while $V_E\ (u,v)$ means that in Euclidean space. In this paper, we care more about personal preferences, namely, we set the balance factor α larger than 0.5. So according to Eq.(1), $V_E\ (u,v) \leq V_N\ (u,v)$.*

Maximum value sequence finding method. We can find the maximum value from the current scene to any scene in R using the heuristic algorithm GOA. This can be explicitly described by Theorem 1.

Theorem 1. *For an intermediary scene o along the sequence between u and v, the sequence with the maximum value $V_{NE}\ (u,o,v)$ is formalized by the sequence passing o. Then, the following Eq. (3) holds:*

$$V_{NE}(u,o,v) = V_N(u,o) + V_E(o,v) \tag{3}$$

Proof. Here $V_N\ (u,o)$ represents the value from the source scene u to the intermediary scene o, while $V_E\ (o,v)$ is the heuristic function that estimates the value from o to the destination scene v. According to the concept of naive heuristic algorithm, Eq. (3) holds. Then, $V_N\ (u,o)$ and $V_E\ (o,v)$ can be obtained by Lemma 1 and Lemma 2 respectively.

Lemma 1. *Assume that a source scene u:=o_0 and a destination scene v:=o_k. A road network traveling sequence ($o_0, o_1, \cdots, o_{k-1}, o_k$) is a sequence of $k+1$ interesting scenes. $V_N\ (o_0, o_k)$ denotes the value of the sequence from o_0 to o_k via $o_1, \cdots, o_{k-1}$:*

$$\sum_{i=1}^{k} V_N(o_{i-1}, o_i) = V_N(o_0, o_k) \tag{4}$$

Proof. The value function $V_N\ (o_0, o_k)$ accounts for a total value of the traveling sequence from o_0 to o_k in road network. That is, this value is the cumulative sequence value from the source scene o_0 to a destination scene o_k via as many scenes as possible from $o_1, \cdots, o_{k-1}$. So the total value $V_N\ (o_0, o_k)$ can be divided into $V_N(o_0, o_1)$ + $V_N(o_1, o_2)$ + $\cdots$ + $V_N(o_{k-1}, o_k)$, namely, $\sum_{i=1}^{k} V_N(o_{i-1}, o_i)$.

Lemma 2. *Let node o and v be the current scene and the destination scene respectively. $h(o)$ is the heuristic estimator. Then, for the value function $V_E\ (o,v)$ of a heuristic value, the following Eq.(5) holds:*

$$h(o) \leq V_E(o,v) \leq V_N(o,v) \tag{5}$$

Proof. The heuristic estimator can find an optimal traveling sequence to a destination scene if the destination scene is reachable. Hence, according to Definition 2, the heuristic employs the value in Euclidean space as a lower bound value of a sequence from o

to v. For that reason, $h(o) \leq V_E\ (o,v)$ holds when $h(o)$ as the estimator is approximately equal to the value in Euclidean space. Also, for the sequence value between o and v, Eq. (5) holds by $V_E\ (o,v) \leq V_N\ (o,v)$. Because although the Euclidean distance is less than or equal to the network distance, the balance factor is larger than 0.5, as follows:

$$V_N(o,v) = \alpha * (w_o + w_{o_i} + ... + w_{o_j} + w_v) +$$

$$(1-\alpha) * \frac{1}{D_{dis_N}(o,o_i) + \sum_{k=i}^{j-1} D_{dis_N}(o_k,o_{k+1}) + D_{dis_N}(o_j,v)}$$

$$V_E(o,v) = \alpha * (w_o + w_v) + (1-\alpha) * \frac{1}{D_{dis_E}(o,v)}$$

So

$$\Delta V = V_N(o,v) - V_E(o,v) = \alpha * (w_{o_i} + ... + w_{o_j}) + (1-\alpha) * \frac{1}{\Delta D_{dis}(o,v)} > 0$$

and this completes the proof.

Efficient optimal scene search. This paper employs the branch-and-bound technique [21] to search an optimal traveling sequence (i.e., a minimum distance and a maximum tourist scene weight). The technique is used to prune all of the unnecessary scenes from multiple neighbor scenes connected with a given current scene by Theorem 2. That is, to find the optimal STS whose value is maximum. This technique can select the tourist scene o_i, which has the minimum distance $D_{dis_{NE}}(u,\ o_i,\ v)$ out of the adjacent scenes $o_1, o_2, \cdots, o_i, \cdots, o_k$ emanating from u (i.e., $D_{dis_N}(u,\ o_1,\ v)$, $D_{dis_N}(u, o_2,\ v), \cdots$, $D_{dis_N}\ (u,\ o_i,\ v), \cdots, D_{dis_N}\ (u,\ o_k,\ v)$ in road network. Meanwhile, this technique can also select the tourist scene o_j, which has the maximum traveling weight $w\ (u,\ o_j,\ v)$ among all chosen scenes $o_1, o_2, \cdots, o_j, \cdots, o_k$. Hence, we define the optimal traveling sequence as follows:

Definition 3. *Let u and v be a source scene and a destination scene respectively. The optimal traveling sequence is a set of ordered scenes from u to v with the maximum value $V_{NE}\ (u,\ o_i,\ v)$ (i.e., a minimum distance $D_{dis_{NE}}\ (u,\ o_i,\ v)$, and a maximum weight $w(u,\ o_i,\ v)$), where o_i is a chosen scene adjacent to u.*

Theorem 2 presents how to find the optimal traveling sequence by the branch-and-bound technique and achieve $V^*_{NE}\ (u,\ o_i,\ v)$ from $V_{NE}\ (u,\ o_i,\ v)$.

Theorem 2. *Given two scenes u and v ($u,v \in N$) in road network, if there exist a set of chosen scenes $o_1, o_2, \cdots, o_i, \cdots, o_k$ connected to scenes u and v, Eq. (6) holds by Definition 3:*

$$V^*_{NE}(u,o_i,v) = \max_{1 \leq i \leq k} (V_{NE}(u,o_i,v)) \tag{6}$$

Proof. Let adjacent chosen scenes $o_i, ..., o_j$ be connected with a given scene u. If $V_{NE}\ (u,\ o_i,\ v)$ is larger than the value of any other adjacent chosen scenes, that is $V_{NE}\ (u,\ o_i,\ v) > V_{NE}\ (u,\ o_k,\ v)$, for $\forall\ k \in (i,j]$. Hence, $V_{NE}\ (u,\ o_i,\ v)$ can be the optimal traveling sequence via o_i among adjacent scenes connected to both u and v.

According to Theorem 2, the branch-and-bound technique can prone some scenes by pre-calculating values from the adjacent scenes to a destination scene.

Efficient traveling sequence finding. The result of STS query is the traveling sequence of the ordered scenes and the paths to them. Figure 2 shows an example of finding efficient STS from k scenes. First, we select a path from the source tourist scene S to each of other chosen scenes by Eq. (3). This procedure begins with the selection of the first scene with the maximum value to the source scene by Eq. (1) and then finds the path to it. In order to prevent the predetermined paths from being re-searched, we must allocate the heap for the optimal value of a scanned scene calculated by V_N (S,o_i) and the path from S to o_i. In Figure 2, we find STS starting from scanning the source scene S to scenes o_1, o_2 and o_3, choosing the scene o_2, whose value is maximum in the heap. Then calculate V_{NE} (S, o_2, o_1), $V_{NE}(S, o_2, o_3)$ and V_{NE} (S, o_2, o_4). In order to find the optimal scene, we calculate V^*_{NE} (S, o_2, o_4) by Eq. (6) and store the intermediary path into the heap. As a result, the procedure yields the sequence $S \rightarrow o_2 \rightarrow o_4$ whose value is maximum. Next, we refresh the sequence to other unvisited scenes with the maximum value to the destination scene by Eq. (3). In the same way, if the total time the sequence is less than or equal to the limited time T_{total}, we iteratively refresh the sequence to remaining scenes.

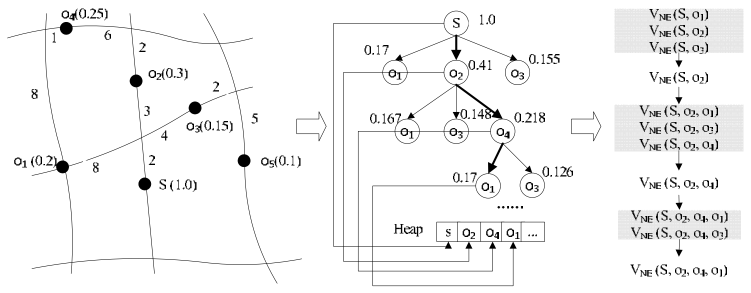

Fig. 2. Find the efficient traveling sequence

Algorithm 4 describes our GOA algorithm. At first, add the start scene to the OPEN list. Line 2 chooses a scene whose value is maximum from the OPEN list. We regard this scene as the current scene. From Line 3 to Line 6, if the destination scene is added to the CLOSE list, then the STS has been found, and the cycle stops. Else repeat the following operations from Line 7 to Line 28. Choose a scene whose value is maximum from the OPEN list. Then refresh the total time by adding the sum of this scene's duration time and the time cost between this scene and the last scene. If the total time is less than T_{total}, switch this scene to the CLOSE list; else, restore the total time and ignore this scene. For each of the other scenes adjacent to this current scene, if it is not walkable or it is in the CLOSE list, ignore it. Otherwise do the following operations. If it is not in the OPEN list, add it to the OPEN list. Regard the current scene as the parent of this scene, and calculate the value of the scene. If it is already in the OPEN list, check if there is other better path according to the value of the current sequence. If so, change the parent of the scene to the current scene, and recalculate the value of the scene.

Algorithm 4. GOA

```
Input:
    The start and end scene, $o = S, E$;
    The set of scenes ID, $I = \{1,\ldots,m\}$;
    The limited time, $T_{total}$;
Output:
 1: The global optimal spatio-temporal sequence, $Q_a$;
    The array $OPEN$ which stores all chosen but not visited scenes
 2: $OPEN = [S]$;
    The array $CLOSE$ which stores all visited scenes
 3: $CLOSE = []$;
 4: while $OPEN$ IsNotEmpty and $T_{total} > 0$ do
 5:   pop the first scene in $OPEN$ to $o$
 6:   put it into $CLOSE$;
 7:   if $o$ equal $E$ then
 8:     break;
 9:   end if
10:   $M$=the number of children scenes of o in $I$;
11:   for each $s \in M$ do
12:     if $T(o)$+$T(o,s) \leq T_{total}$ then
13:       calculate the estimated value, $EV(o,s)$=$\alpha$*($w_o$+$w_s$)+(1-$\alpha$)*$\frac{1}{D_{dis_N}(o,s)}$;
14:       if $s$ is not in OPEN or CLOSE then
15:         pop $s$ from $I$
16:         put s into $OPEN$
17:       else if $s$ is in OPEN then
18:         if $EV(o,s) > EV$(OPEN) then
19:           update the value of OPEN;
20:         end if
21:       else
22:         if $EV(o,s) < EV$(CLOSE) then
23:           update the value of CLOSE;
24:           pop $s$ from CLOSE
25:           put $s$ into OPEN;
26:         end if
27:       end if
28:     end if
29:   end for
30:   put $o$ into CLOSE;
31:   sort the scenes in $OPEN$ by the $EV$ descending
32:   $T_{total}$ = $T_{total}$-$T(o)$+$T(s)$);
33: end while
34: reverse $CLOSE$
35: $Q_a$ = $CLOSE$;
36: return $Q_a$;
```

4 Experimental Evaluation

This section presents a comprehensive performance evaluation of the proposed methods for STS using Flickr datasets.

Experimental Setup. We obtained the real dataset in the city of Beijing with 286 scenes and 658 edges. In this dataset, we generated some interesting scenes according to scene features combined with user's profile. Datasets with a varying number of interesting scenes, varying balance factor, as well as varying limited total time were generated. The total number of interesting scenes is in the range $m \in [1,20]$, the balance factor is in the range $\alpha \in [0,1]$, while the limited total time is in the range $T_{total} \in [1h,8h]$, where h denotes the time granularity "hour".

Performance Results. In this part we study the performance of the proposed four algorithms. In order to prove the advantage of our algorithms, we compared them with ARM (average random method). ARM is achieved by choosing scene randomly for 30 times, and taking the average value of them.

First, we study the effects of α and T_{total} on the value of STS. Figure 3 plots the value of STS as a function of α, when T_{total} =8h. Figure 4 plots the value of STS as a function of T_{total}, when α =0.7. In both cases, GOA outperforms v-LOA, d-LOA, w-LOA and ARM obviously. The value of ARM is the lowest. With the increase of α and T_{total}, the performance of all algorithms increases. The algorithm d-LOA is greatly affected by the relative locations of scenes, because it greedily follows the nearest $\lceil m/2 \rceil$ scenes from the remaining scenes irrespective of its direction with respect to the destination scene E. With the increase of α and T_{total}, the probability that d-LOA wanders off the correct direction increases. In Figure 4 the trends of algorithm w-LOA and v-LOA are almost the same. Because when α =0.7, the distance has little effect on the value of the sequence, and the value of both algorithms is similar.

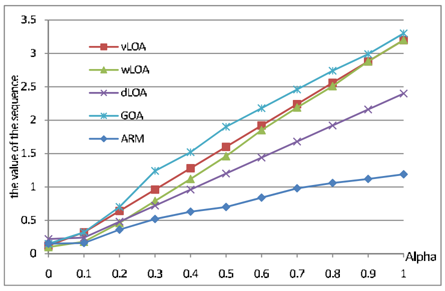

Fig. 3. The trend of V_N with different α

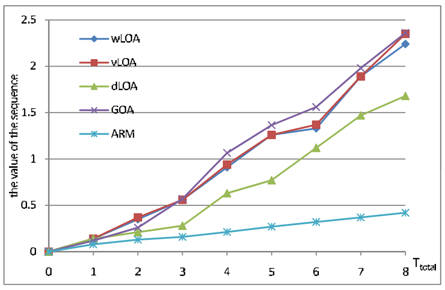

Fig. 4. The trend of V_N with different T_{total}

Figure 5 plots the results of road network distance of STS as a function of T_{total}, when α =0.7. Figure 6 plots the total weight of STS as a function of T_{total}, when α =0.7. With the increase of T_{total}, both the distance and the weight of STS in all algorithms increase. The distance of algorithm d-LOA is less than the other four algorithms, because it always greedily follows the nearest $\lceil m/2 \rceil$ scenes from the remaining scenes. The weight of algorithm GOA outperforms v-LOA, d-LOA and w-LOA in Figure 6. So our four algorithms can get better results than ARM.

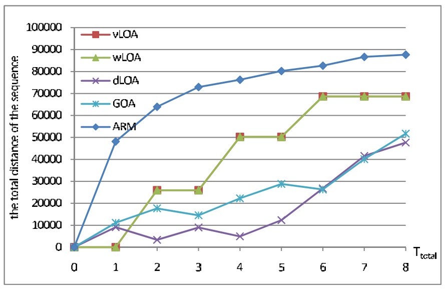

Fig. 5. The trend of D_{dis_N} with different T_{total}

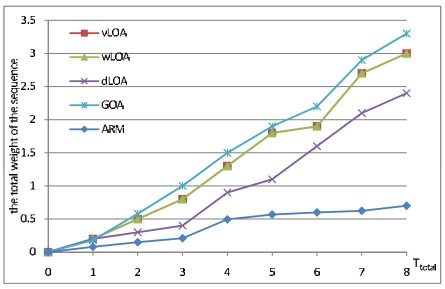

Fig. 6. The trend of w with different T_{total}

We also study the average length of STS as a function of T_{total}, when α =0.7 in Figure 7. In general, the algorithm GOA includes more scenes than the other four ones. The reason is that GOA can get the global optimal sequence in road network. The number of scenes in ARM is the smallest. From Figure 8 we can see the value of GOA is the maximum, v-LOA and w-LOA are almost the same, which depends on the choice of α. The value of ARM is the minimum.

We examine the trend of runtime with different number of scene set in Figure 9. When the number of scene set is more than 50000, the runtime of v-LOA, d-LOA, and w-LOA increase much faster. The trend of runtime with different T_{total} is examined in Figure 10, when the number of scene set is 10000. In both cases, with the increase of scene set, the runtime of all algorithms increase. The runtime of v-LOA is the maximum, and ARM's is the minimum. The difference is the increase speed of runtime in Figure 10 is slower than that in Figure 9.

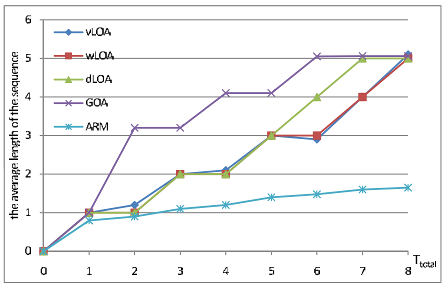

Fig. 7. The length of STS with different T_{total}

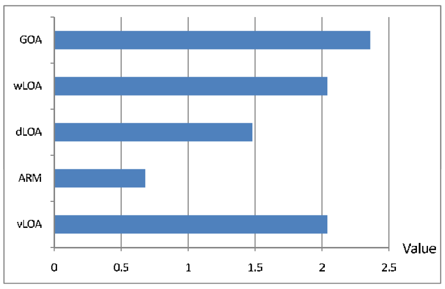

Fig. 8. The trend of V_N of different algorithms

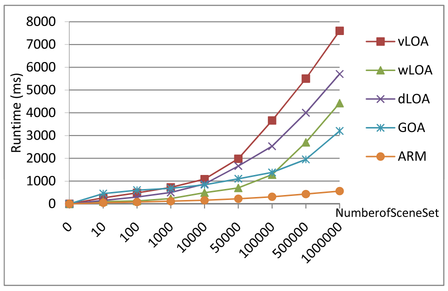

Fig. 9. Runtime with different NumSet

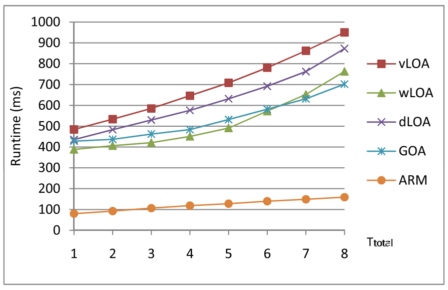

Fig. 10. The trend of runtime with different T_{total}

5 Conclusions and Future Work

The goal of this paper is to provide users with the optimal spatio-temporal sequence that passes through as many chosen scenes as possible with the maximum weight and the minimum distance within a limited travel time. We first argued that this problem is NP-hard, and gave a simple proof. Then formally defined the STS problem. We considered solutions for the STS problem as graphs where tourist scenes were nodes and paths between these scenes were edges. Two approximate algorithms: local optimization algorithms and a global optimization algorithm were provided. The experimental study using real datasets in Flickr demonstrated the effectiveness of our proposed algorithms.

Acknowledgement

We would like to thank Pengfei Dai of Beijing University of Posts and Telecommunications for his helpful comments in the experiments.

References

1. Nasraoui, O., Soliman, M., Saka, E., et al.: A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites. IEEE Transactions on Knowledge and Data Engineering (TKDE), 202–215 (2008)
2. Rattenbury, T., Good, N., Naaman, M.: Towards Automatic Extraction of Event and Place Semantics from Flickr Tags. In: Proceedings of the 30th Annual International ACM SIGIR Conference (2007)
3. Ahern, S., Naaman, M., Nair, R., Yang, J.: World Explorer: Visualizing Aggregate Data from Unstructured Text in Georeferenced Collections. In: Proceedings of the ACM IEEE Joint Conference on Digital Libraries, JCDL (2007)
4. Quack, T., Leibe, B., van Gool, L.: World-Scale Mining of Objects and Events from Community Photo Collections. In: Proceedings of the 7th ACM International Conference on Image and Video Retrieval, CIVR (2008)
5. Crandall, D., Backstrom, L., Hutternlocher, D., Kleinberg, J.: Mapping the World's photos. In: Proceedings of the 18th International World Wide Web Conference, WWW (2009)
6. Zhou, C., Meng, X.: Complex Event Detection on Flickr. In: Proceedings of the 27th National Database Conference of China, NDBC (2010)
7. Zheng, I., Zhang, L., Xie, X., Ma, W.Y.: Mining Interesting Locations and Travel Sequences from GPS Trajectories. In: Proceedings of the 18th International World Wide Web Conference, WWW (2009)
8. Cao, X., Cong, G., Jensen, C.: Mining Significant Semantic Locations From GPS Data. In: Proceedings of the VLDB Endowment, PVLDB, vol. 3(1) (2010)
9. Gonotti, F., et al.: Trajectory Pattern Mining. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 330–339 (2007)
10. Mamoulis, N., et al.: Indexing and Quering Historical Spatiotemporal Data. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 236–245 (2004)
11. Girardin, F., Dal Fiore, F., Blat, J., Ratti, C.: Understanding of Tourist Dynamics from Explicitly Disclosed Location Information. In: Proceedings of the 4th International Symposium on LBS and Telecartography (2007)
12. Chen, Z., Shen, H.T., Zhou, X., Yu, J.X.: Monitoring Path Nearest Neighbor in Road Networks. In: Proceedings of the 35th SIGMOD International Conference on Management of Data, SIGMOD (2009)
13. Chen, Z., Shen, H.T., Zhou, X., Zheng, Y., Xie, X.: Searching Trajectories by Locations-An Efficiency Study. In: Proceedings of the 36th SIGMOD International Conference on Management of Data, SIGMOD (2010)
14. Home and Abroad, `http://homeandabroad.com`
15. Popescu, A., Grefenstette, G.: Deducing Trip Related Information from Flickr. In: Proceedings of the 18th International World Wide Web Conference, WWW (2009)

16. Popescu, A., Grefenstette, G., Alain, P.: Mining Tourist Information from User-Supplied Collections. In: Proceedings of The 18th ACM Conference on Information and Knowledge Management, CIKM (2009)
17. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query Processing in Spatial Network Databases. In: Proceedings of 29th International Conference on Very Large Data Bases, VLDB (2003)
18. Yiu, M., Mamoulis, N.: Clustering Objects on a Spatial Network. In: Proceedings of the 30th SIGMOD International Conference on Management of Data, SIGMOD (2004)
19. Shekhar, S., Liu, D.: CCAM: A Connectivity Clustered Acccess Method for Networks and Network Computations. IEEE Transactions on Knowledge and Data Engineering (TKDE), 102–119 (1997)
20. Li, F., Cheng, D.: On trip planning queries in spatial databases. In: Anshelevich, E., Egenhofer, M.J., Hwang, J. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 273–290. Springer, Heidelberg (2005)
21. Tao, Y., Papadias, D., Shen, Q.: Continuous Nearest Neighbor Search. In: Proceedings of 28th International Conference on Very Large Data Bases (VLDB), pp. 287–298 (2002)