P-LAG: Location-Aware Group Recommendation for Passive Users

Yuqiu Qian[1(✉)], Ziyu Lu[1], Nikos Mamoulis[2], and David W. Cheung[1]

[1] The University of Hong Kong, Hong Kong, Hong Kong
{yqqian,zylu,dcheung}@cs.hku.hk
[2] University of Ioannina, Ioannina, Greece
nikos@cs.uoi.gr

Abstract. Consider a group of users who would like to meet to a place in order to participate in an activity together (e.g., meet at a restaurant to dine). Such meeting point queries have been studied in the context of spatial databases, where typically the suggested points are the ones that minimize an aggregate traveling distance. Recently, meeting point queries have been enriched to take as input, besides the locations of users, also some preference criteria (e.g., expressed by some keywords). However, in many applications, a group of users may require a meeting point recommendation without explicitly specifying any preferences. Motivated by this, we study this scenario of group recommendation for such passive users. We use topic modeling to infer the preferences of the group on the different points of interest and combine these preferences with the aggregate spatial distance of the group members to the candidate points for recommendation in a unified search model. Then, we propose an extension of the R-tree index, called TAR-tree, that indexes the topic vectors of the places together with their spatial locations, in order to facilitate efficient group recommendation. We propose and compare three variants of the TAR-tree and a compression technique for the index, that improves its performance. The proposed techniques are evaluated on real data; the results demonstrate the efficiency and effectiveness of our methods.

1 Introduction

With the proliferation of smart mobile devices, recommendation services are becoming location-aware; advertisements and suggestions to users are not generated only based on the (explicitly expressed or implicitly derived) user preferences, but also based on the user locations. For example, a mobile user who likes green tea would be recommended a nearby green tea shop (if she wants to take a break), instead of a more famous one, which is too far.

In this paper, we consider the scenario of providing recommendations to a *group* of users who would like to meet and enjoy an activity or event together. By considering only the locations of the individual users in the group, a meeting point query can be modeled and solved as an *group nearest neighbor* search [23],

© Springer International Publishing AG 2017
M. Gertz et al. (Eds.): SSTD 2017, LNCS 10411, pp. 242–259, 2017.
DOI: 10.1007/978-3-319-64367-0_13

where the suggested places are the ones that minimize the average or maximum distance to the users of the group. In addition, *spatial skyline* queries [27] can be used to suggest the places which are not dominated by other places in the space defined by their distances from the users in the group.

Given the fact that the recommended places by spatial-only criteria may not be consistent with the preferences of the group, recent work has considered additional, non-spatial search criteria that are explicitly expressed by the group. For example, the group may provide a set of keywords and require that the recommended places are textually relevant to the keywords in addition to being spatially near the group members. In this direction, the *textually relevant spatial skyline* was proposed in [28], which includes all objects that are not dominated by others with respect to their aggregate spatial distance to the group users and their similarity to a set of given keywords. A similar, *location-aware group preference* query was proposed in [17]; the objective is to find a nearby destination that belongs to a specific category (e.g., a bar), which is also close to places that satisfy the explicitly given preferences of each user.

In a real scenario of recommending a place to a group of users, the users could be *passive*, i.e., they might not express their preferences explicitly. For example, it might be hard for the group to reach a consensus for the set of keywords to use in the search, or some users in the group might find it difficult to express their preferences by keywords or by ranking functions on the features of the places. Hence, previous work on meeting point recommendation largely ignores the preferences of silent users. In this paper, we *learn* these preferences by analyzing the check-in/reviewing history of users and use them in meeting point recommendation. It is worth mentioning that, although geographic information has been considered in previous work on recommending venues to groups of passive users (e.g., see [20]), recommender systems do not consider the case where the users of the group are presently at different locations.

Hence, in this paper, we consider the problem of location-aware group recommendation for *passive users* (P-LAG for short), assuming that the members of the group are spatially dispersed and they have to meet in order to enjoy an activity or event at the recommended place. Although the locations of the users are known, the preferences of the group should be *inferred* based on any previously collected information related to the activities of the users on the candidate places (e.g., check-in history in a geo-social network such as Foursquare, place visits derived by analyzing GPS trajectories, posted reviews that indicate place visits and preferences). For this purpose, we use a *topic modeling* approach [5], which represents each user by a topic vector that models his/her preferences to each topic. A topic is a hidden property of the places; accordingly, the candidate places are also represented by topic vectors. Thus, for each user and each place we can model the preference of the user to the place by the similarity between the corresponding vectors. For the group of users, we compute an aggregate topic vector and use it to model the preferences of the group.

Our second contribution is an extension of the classic R-tree, called TAR-tree, to index the topic vectors of the places together with their spatial locations.

We propose three variants of the TAR-tree and a compression technique for the index, that greatly improves search performance. The effectiveness of the TAR-tree is largely due to the skew in the topic vectors and the spatial auto-correlation of the topics, which we analyze experimentally. Finally, we conduct a comparative study, using two real datasets, which demonstrates the efficiency and effectiveness of our proposed methods.

The remainder of this paper is organized as follows. Section 2 provides a formal definition of the top-k location-aware group recommendation problem in a passive user setting and discusses a naive solution as well as our framework. In Sect. 3, we present our approach for extracting the topic vectors of users and places, in a preprocessing phase. Online query evaluation is studied in Sect. 4. Section 5 evaluates the effectiveness and efficiency of the proposed framework. In Sect. 6, we discusses related work. Finally, Sect. 7 concludes the paper.

2 Problem Definition

The P-LAG problem we are focusing on in this paper is to find k venues (i.e., places) which are (i) spatially close to the current location of each group user and (ii) consistent to the preferences of the group. Without loss of generality, Euclidean distance $dist(p_i, q_j)$ is used as spatial distance in this work. A formal definition of the P-LAG query is given below:

Definition 1 (*P-LAG query*). *Consider a set of venues* $P = \{p_1, p_2, ..., p_n\}$, *each with a topic vector* $p_i.\psi$. *Given a group* $Q = \{q_1, q_2, .., q_m\}$ *of users having an aggregate topic vector* $Q.\psi$, *the P-LAG query finds the* k *venues* $p_i \in P$ *which minimize the distance function* $Dist(p_i, Q) = f(\sum_{q_j \in Q} dist(p_i, q_j), \omega(p_i.\psi, Q.\psi))$, *where* $dist(p_i, q_j)$ *is the spatial distance between* p_i *and the location of user* q_j, $\omega(p_i.\psi, Q.\psi)$ *is the similarity between topic vectors* $p_i.\psi$ *and* $Q.\psi$.

Notice that the ranking function $Dist(p_i, Q)$ can be any monotonic aggregate function $f(\cdot, \cdot)$ which considers both spatial information and user preferences, such as weighted sum $\alpha \times \sum_{q_j \in Q} dist(p_i, q_j) + (1 - \alpha) \times (1 - \omega(p_i.\psi, Q.\psi))$ [30] and weighted distance $\sum_{q_j \in Q} dist(p_i, q_j)/\omega(p_i.\psi, Q.\psi)$ [28]. In this paper, we use the latter definition, because it is parameter-free and it does not require normalization. Still, the approaches proposed in this paper are independent of how $f(\cdot, \cdot)$ is defined.

To support P-LAG queries, we follow the two-step framework illustrated in Fig. 1. First, the topic vectors of both users and venues (i.e., user preference vectors and venue topic vectors) are automatically extracted with the help of a model that uses previous trajectory/check-in information. This is an offline process executed once based on the available historical data. The process can be repeated in regular time intervals in order to keep the topic vectors up-to-date. The second component of our framework addresses online query evaluation. Once a group of users are formed, the top-k venues for the group are identified and recommended within milliseconds.

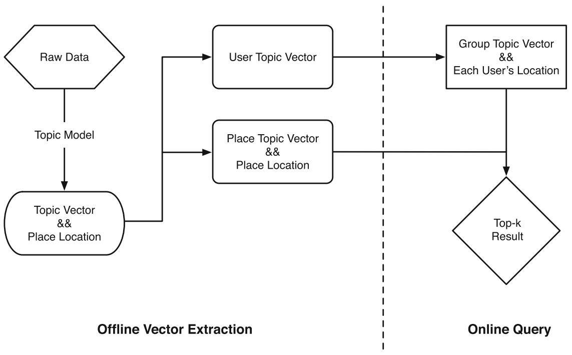

Raw Data
Topic Model
Topic Vector && Place Location
User Topic Vector
Place Topic Vector && Place Location
Group Topic Vector && Each User's Location
Top-k Result
Offline Vector Extraction
Online Query

Fig. 1. Overall framework

3 Vector Extraction

3.1 Topic Vector Extraction

Our framework first extracts topic vectors for all users and places. For this purpose, we apply Latent Dirichlet Allocation (LDA) [5] on the history of user visits to places, which can be obtained by past check-in or trajectory records. Hence, the raw input data is a collection of $\langle u, l, t\rangle$ records, where u represents a user, l is a location (i.e., place) visited by u, and t is the time of the visit.

For each user $u \in U$, we synthesize a *document* D_u by aggregating all records of u, and regard each location $l \in L$ visited by u, as a word in D_u. Hence, for each user u, we have a document $D_u = \{u, L_u\}$, where L_u includes the identifiers of all places visited by u. Overall, we obtain a collection D of documents D_u, one for each $u \in U$.

Figure 2 shows the graphical representation of the Location-LDA model we have designed, which uses two distributions: a user-topic distribution θ and a topic-location distribution ϕ. The topic-location distribution represents the topic vector for each location, e.g., ϕ_l for location l. The user-topic distribution represents the topic vector for each user, e.g., θ_u for user u. The latent topics Z are extracted based on the check-in preferences of users (i.e., they are not related to the spatial features of the places). α, β are prior parameters for the two distributions; we use $\alpha = 50/K$ and $\beta = 0.01$, where K is the number of topics.

3.2 Topic Vector Analysis

In this section, we conduct two spatial analysis studies on the extracted topic vectors that lead us to important observations. The first study shows the correlation between the geographical distance and topic similarity of two locations. The second one shows that different topics have different spatial distributions. We use the topic vectors extracted from Yelp-USA, which is a subset of the Yelp

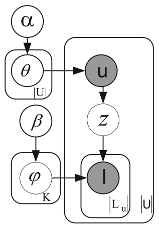


Fig. 2. Graphical representation of our Location-LDA model

Dataset Challenge dataset[1]. The vectors have a dimensionality $K = 20$, which is the default value in our experiments (see Sect. 5).

Correlation between Spatial Distance and Topic Proximity. Intuitively, locations that are geographically close to each other should have overlapping topics with high probability. To confirm this, we sampled 2000×2000 locations pairs from the check-in data of Yelp-USA. For each pair of locations (places) $(p_i,\ p_j)$, we compute their distance $dist(p_i, p_j)$ in the Euclidean space and the similarity between their topic vectors $w(p_i.\psi, p_j.\psi)$. The topic similarity is computed as the dot product of the vectors.

We divided the location pairs into several groups based on their geographic distances and calculated the average topic similarity for each group. As shown in Fig. 3, topic similarity is positively correlated to spatial proximity. The size of each point in the plot represents the number of location pairs that fall in the corresponding distance range. Note that location pairs that are within 1 km to each other are significantly more related than others.

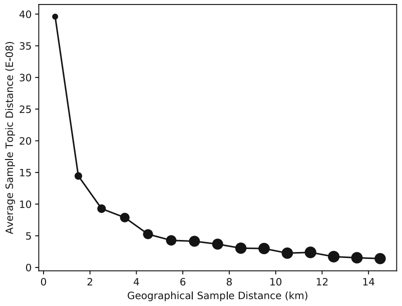
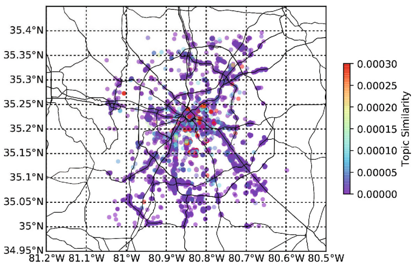


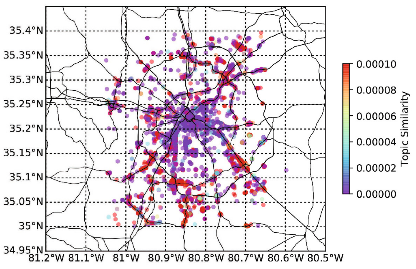
Fig. 3. Correlation between spatial distance and topic proximity

Geographical Topic Heatmap. By visualization analysis, we observed that the spatial distributions of different topics differ significantly. In other words, different topics form different spatial clusters even in a map of relatively small scale.

[1] http://www.yelp.com/dataset_challenge.



(a) Topic 0



(b) Topic 3

Fig. 4. Topic value heatmap in Charlotte

For example, in Fig. 4, we use heatmaps to visualize the spatial distribution of different topics in locations of Charlotte. Each place is colored based on its value of topic 0 (Fig. 4(a)) and topic 3 (Fig. 4(b)). Observe that different topics cover different spatial regions and different spatial regions have relevance to a given topic.

4 Indexing and Search for P-LAG

Deriving topic vectors for venues and users as described in Sect. 3 allows us to search for the best places to recommend to a given group of users, based on spatial distance and topic similarity (see Definition 1). Specifically, for a group $Q = \{q_1, q_2, .., q_m\}$ of spatially dispersed users, we should compute the k places $p_i \in P$ that minimize the objective function $Dist(p_i, Q) = \sum_{q_j \in Q} dist(p_i, q_j)/\omega(p_i.\psi, Q.\psi)$.

A naive algorithm (NA) traverses all venues $p_i \in P$ one by one, computes the objective function for each of them, and maintains the set of k venues with smallest $Dist(p_i, Q)$. This approach is generally expensive because it computes $Dist(p_i, Q)$ exhaustively for each place p_i. In addition, the topic vector similarity component of $Dist(p_i, Q)$ is more expensive to compute compared to the spatial distance component. Therefore, a natural approach for P-LAG queries is to access the candidate venues for recommendation in a spatial order, while deriving bounds for the k-th distance; this enables us to stop search early by avoiding examining the places whose spatial distances are too large.

4.1 Basic R-tree Approach

An intuitive approach for supporting P-LAG queries efficiently is to spatially index all venues $p_i \in P$, e.g., by an R-tree [4,11]. With the help of the R-tree, we can derive a lower bound for the group spatial distance $\sum_{q_j \in Q} dist(e, q_j)$ to each entry e in the R-tree (see [23]). However, we can access the topic vector of each place, only after accessing the place at leaf level of the tree, which gives us no information about the vector similarity between the places under an R-tree entry e and $Q.\psi$ during search. This makes it hard to prune sets of candidate places at the non-leaf levels of the R-tree. A basic approach for alleviating this problem is to compute and use the maximum value of $\omega(p_i.\psi, Q.\psi), \forall p_i \in P$, i.e. $\omega(p^*.\psi, Q.\psi)$, where $p^*.\psi$ is a vector having the maximum values of each dimension in the topic space. It is easy to prove $\omega(p^*.\psi, Q.\psi)$ as a upper bound of all $\omega(p_i.\psi, Q.\psi), p_i \in P$ since $\omega(p_i.\psi, Q.\psi) = \sum_j p_i.\psi_j \cdot Q.\psi_j \leq \sum_j p^*.\psi_j \cdot Q.\psi_j = \omega(p^*.\psi, Q.\psi)$. Therefore, $\omega(p^*.\psi, Q.\psi)$ is also a upper bound and $Dist(e, Q) = \sum_{q_j \in Q} dist(e, q_j)/\omega(p^*.\psi, Q.\psi)$ can be utilized as a lower bound to prune non-qualifying R-tree nodes.

Algorithm 1 illustrates our algorithmic framework for P-LAG queries. Line 17 computes and uses a lower distance bound of each R-tree entry. This bound for the basic R-tree approach (i.e., $\sum_{q_j \in Q} dist(e, q_j)/\omega(p^*.\psi, Q.\psi)$) is too loose, motivating us to study additional ways for tightening the bound and speeding up P-LAG search.

Algorithm 1. P-LAG Algorithmic Framework

```
1:  procedure RTree Approach($P, Q, k$)
2:      MinHeap $H \leftarrow \varnothing$                          ▷ nodes to visit
3:      $R \leftarrow \varnothing$                                  ▷ top-$k$ result set
4:      $kDist \leftarrow Inf$                               ▷ $k$-th $Dist(p_i, Q)$ in $R$
5:      Add $root$ of RTree to $H$
6:      while $H$ is not empty do
7:          $e \leftarrow$ deHeap($H$)
8:          if $e.dist > kDist$ then
9:              return $R$
10:         $N \leftarrow readNode(e)$
11:         if $e$ is an leaf node then
12:             for each object $p'$ in $N$ do
13:                 $p'.dist \leftarrow \frac{\sum_{q_j \in Q} dist(p', q_j)}{\omega(p'.\psi, Q.\psi)}$
14:                 Update $R$ and $kDist$ with $p'$
15:         else
16:             for each entry $e'$ in $N$ do
17:                 $e'.dist \leftarrow$ lower bound of $Dist(e', Q)$
18:                 if $e'.dist < kDist$ then
19:                     Add $e'$ to $H$
20:     return $R$
```

4.2 TAR-tree Approach: Topic-Aware R-tree

Our approach for improving our framework is to embed vector information into the R-tree, making it a TAR-tree (i.e. Topic-aware R-tree), which helps deriving tighter bounds for pruning the search space effectively.

B-TAR-tree: Ball Bounding. One way of doing so is to borrow the idea behind the Ball-tree [22], and use ball bounding to group multi-dimensional topic vectors $p_i.\psi$, such that $p_i \in P$. For each entry e in the R-tree, we need to find the bounding ball with the minimum radius that contains all places in the subtree pointed by e. The minimum balls are computed in a bottom up fashion similar to the ball tree, but following the original structure of the R-tree.

Following the proof in [25], it is easy to get an upper bound for vector similarity for each entry e in the B-TAR-tree, as $\omega(e.\psi, Q.\psi) \leq \omega(e.\mu, Q.\psi) + e.\lambda \cdot \|q\|$, where $e.\mu$ is the center of minimum ball and $e.\lambda$ is the radius of the minimum ball of entry e. Therefore, we can derive a bound for $Dist(e, Q)$ as $Dist(e, Q) = \sum_{q_j \in Q} dist(e, q_j)/\omega(e.\psi, Q.\psi) \geq \sum_{q_j \in Q} dist(e, q_j)/(\omega(e.\mu, Q.\psi) + e.\lambda \cdot \|q\|)$ to be used in line 17 of Algorithm 1. However, the ball bounding approach has the side effect of storing extra information at the non-leaf levels of TAR-tree: a vector $e.\mu$ as well as a number $e.\lambda$. Hence, each non-leaf node has a much smaller fanout compared to the original R-tree and the tree's height becomes larger.

N-TAR-tree: Norm Bounding. As an alternative of the ball bounding technique, which requires the storage of one extra vector for each MBR, we suggest a second approach (norm bounding), which only needs to store one number per MBR. Specifically, for each non-leaf entry e in the R-tree pointing to a node S with entries $\{s_1, s_2, ..., s_f\}$, it is easy to show that $\omega(s_i.\psi, Q.\psi) = \|s_i.\psi\| \cdot \|Q.\psi\| \cdot \cos(s_i.\psi, Q.\psi) \leq \|s_i.\psi\| \cdot \|Q.\psi\|, \forall s_i \in S$. Let $e.\phi$ be the maximum $\|s_i.\psi\|, \forall s_i \in S$. We can get a tighter bound for $\omega(e.\psi, Q.\psi)$ compared to $\omega(p^*.\psi, Q.\psi)$ of the basic R-tree approach, since $\omega(e.\psi, Q.\psi) \leq e.\phi \cdot \|Q.\psi\|$, where $e.\phi$ is the maximum norm value of entries $s_i \in S$ (i.e., a single number).

R-TAR-tree: Rectangle Bounding. The third proposed TAR-tree variant uses rectangle bounding. For each entry e in a R-tree node, we store a vector $e.\psi$, which stores for each topic dimension the maximum value of that dimension in the subtree pointed by e. Then, we have $\omega(e.\psi, Q.\psi) \geq \omega(s_i.\psi, Q.\psi), \forall s_i \in S$, where $S = \{s_1, s_2, ..., s_f\}$ is the set of entries in the node pointed to by e.

As an illustration, in Fig. 5, M_1 and M_2 are two entries of a TAR-tree, each pointing to a leaf node containing a triple of venues; each venue v_1 to v_6 has a topic distribution vector (shown next to each v_i), whereas the corresponding vector of each M_i has the maximum of values for each topic of the entries in M_i. By using the minimum spatial distances of the user locations to each M_i and the vector of M_i, we can derive bounds for the spatial distances and the vector

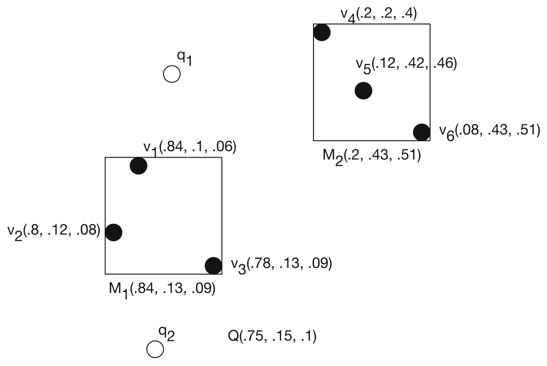


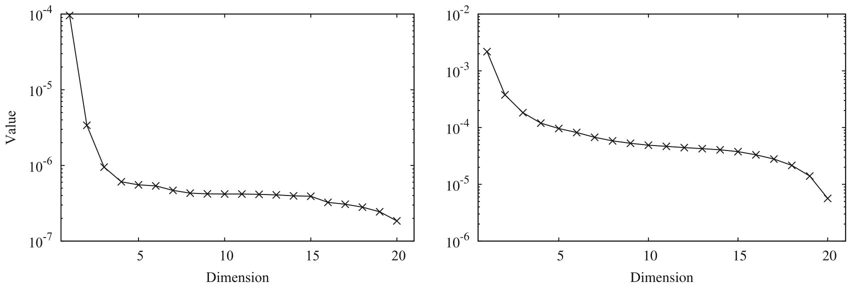
Fig. 5. Example of TAR-tree

similarities of each venue in M_i. These bounds can be used to prune M_i, if it is worse than the k-th venue found so far.

4.3 Vector Compression in the TAR-tree

All three TAR-tree variants we have discussed extend the R-tree to store extra topic information into its entries. However, the potential benefit of getting tighter bounds can be outweighed by the side effect of higher storage requirements, leading to a slower method. In view of this, we propose a compression technique that exploits the skew and the spatial autocorrelation of topic vectors.

To demonstrate the skew of topic vectors in practice, we first conduct an experiment on the Foursquare [10] dataset we used in our experiments. For 20-dimensional topic vectors, Fig. 6(a) shows the average distribution of their values after ranking them from larger to smaller. Note that the largest 2–3 scalars are significantly larger than the remaining ones. We repeat this experiment, this time for the non-leaf entries of the above-leaf level of the TAR-tree (Fig. 6(b)). The skew at the vectors of the non-leaf entries indicates that the vectors in the same leaf node are correlated. This is due to the spatial autocorrelation of the topics (see our analysis in Sect. 3.2).



(a) whole topic vector (b) topic vector of first level

Fig. 6. Skewness of topic vectors stored in the TAR-tree

Recall that, in the R-TAR-tree, we store a vector $e.\psi$ for each entry e in an R-tree node, whose value at each dimension is the maximum of the corresponding dimensions in all its children entries. Then, we use $\omega(e.\psi, Q.\psi)$ as the largest approximate vector topic value of all its children nodes. To reduce the storage cost for these vectors, which is linear to their dimensionality, we propose a compression technique. Take $e.\psi = (v_0, v_1, ..., v_{19})$ whose vector size is 20 as a example (shown in Fig. 7). After rearranging the scalars in decreasing value, we get a list $v_{t_0}, v_{t_1}, ..., v_{t_{19}}$, where v_{t_i} is the value of dimension t_i in the original $e.\psi$ vector and $v_{t_i} \geq v_{t_{i+1}}$ for each $i \in [0, 18]$. Instead of storing the entire 20-dimensional vector $e.\psi$, we can reduce the storage size by only storing the top-m (m is called compression factor) largest values in it together with their indexes t_i and the $(m+1)$-th largest value. This way, the storage cost of each topic vector can be reduced to $2m+1$. With the compressed vector, we have a looser upper bound for vector similarity as $w(e.\psi, Q.\psi) = \sum_j v_j \cdot Q.\psi_j = \sum_{j=0}^{m-1} v_{t_j} \cdot Q.\psi_{t_j} + \sum_{j=m}^{|e.\psi|-1} v_{t_j} \cdot Q.\psi_{t_j} \leq \sum_{j=0}^{m-1} v_{t_j} \cdot Q.\psi_{t_j} + \sum_{j=m}^{|e.\psi|-1} v_{t_m} \cdot Q.\psi_{t_j}$, where v_{t_m} denotes the $(m+1)$-th largest value.

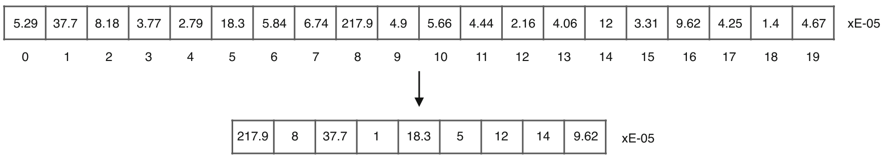


Fig. 7. Example of compression

This vector compression technique is used only at the non-leaf entries of the tree. We denote this approach by R-TAR-tree+. However, non-leaf nodes are significantly fewer than leaf-nodes, hence the space saved by the compression is limited. Therefore, we consider also applying this compression to place topic vectors at the leaves of the tree. Note that after this change the tree only indexes the vectors approximately. We denote this approach by R-TAR-tree-C. Since topic vector extraction is an approximation technique by nature, we do not expect much loss of accuracy in practice.

5 Experimental Evaluation

We evaluate the performance of all approaches proposed in this paper, namely the basic R-tree approach (Sect. 4.1), the three variants of TAR-tree approach (Sect. 4.2), and the two compression approaches (Sect. 4.3). All tested methods were implemented in Java and the experiments were conducted on an Intel(R) Core(TM) i5-3570 CPU @ 3.40 GHz machine, with 16 GB of RAM.

5.1 Datasets

We used two real datasets, Foursquare [10] and Yelp-USA[2], in our evaluation. Foursquare is a popular geo-social network with more than 50 million users and 10 billion check-ins[3]. Yelp-USA is a subset of the dataset used in the Yelp Dataset Challenge, containing all USA located records. The details of the datasets are shown in Table 1. In the topic vector extraction phase, for each dataset, we mark 80% of the data as the training set and use the remaining data to test the accuracy of our Location-LDA model. To generate the user groups for the queries, we sample users by bounding the average distance to each other; we use the real learnt topic vectors for the users.

[2] http://www.yelp.com/dataset_challenge.
[3] https://foursquare.com/about/.

Table 1. Statistics of the datasets

Dataset	Yelp-USA	Foursquare
# of records	1,525,924	2,290,997
# of users	432,536	11,326
# of venues	38,694	187,218

5.2 Efficiency Analysis

We evaluate the performance of different methods under various parameter settings, i.e. varying the result set size k, the topic vector dimensionality K, the group size, the average distance between the users in the group measured in degrees of latitude and longitude, and the compression factor m. The details of the parameters are listed in Table 2.

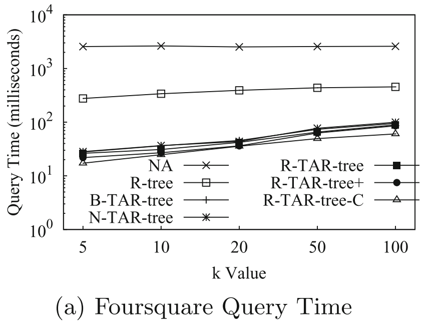
Table 2. Parameters (default values in bold)

Parameter	Value
k	5, 10, **20**, 50, 100
Dimensionality of topic vectors (K)	5, 10, **20**, 50, 100
Group size	3, 5, **10**, 15, 20
User maximum distance	0.05, 0.1 **0.2**, 0.5, 1
Compression parameter (m)	1, 2, **4**, 6, 8

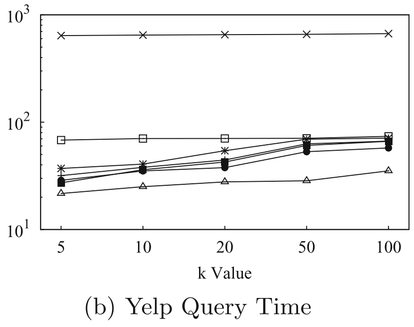
Effect of k. To study the effect of the result set size k, we fix the other parameters to their default values, and vary k from 5 to 100. As shown in Fig. 8, the cost increases with k, which is consistent with our expectation. We observe that the basic R-tree method is much faster compared to the naive sequential scan approach and the TAR-tree methods further reduce the query time and I/O. Different TAR-tree methods do not have big differences and generally R-TAR-tree+ performs best among all exact methods. R-TAR-tree-C improves the efficiency of R-TAR-tree+ up to 2 times; as we will see in Sect. 5.3 its accuracy is very high.

Effect of vector dimensionality. The second parameter of which the effect we investigate is the vector dimensionality. As shown in Fig. 9, on both datasets, the average query time increases as the dimensionality increases. The I/O cost shows a similar trend (we omit the plots due to space constraints). At the same time, we observed (in experiments not shown here) that the topic extraction quality does not improve much when we increase the dimensionality to above 20. Therefore, 20 is a good enough value in terms of quality.

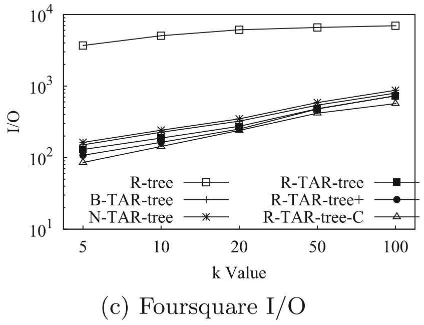
Effect of group size. The third parameter that we study is the group size $|Q|$; in general, the groups can be formed by any number of users. Observe from



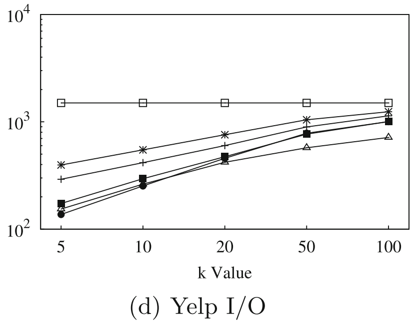
(a) Foursquare Query Time



(b) Yelp Query Time

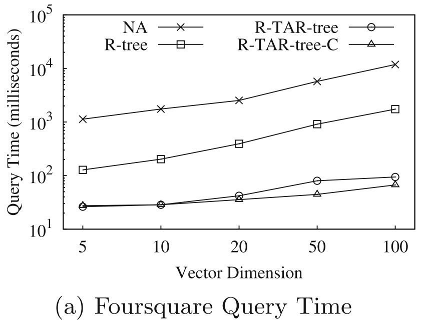


(c) Foursquare I/O

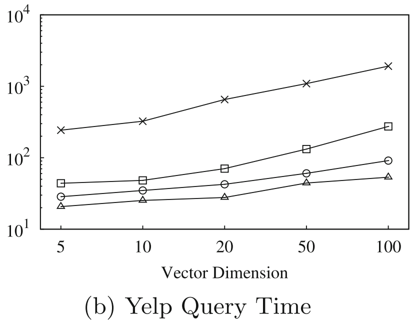


(d) Yelp I/O

Fig. 8. Comparison when varying the value of k



(a) Foursquare Query Time

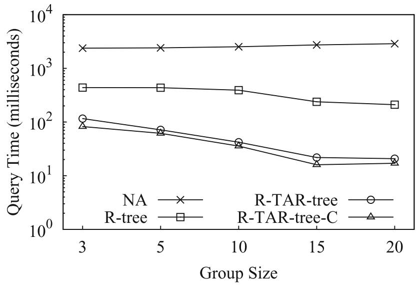


(b) Yelp Query Time

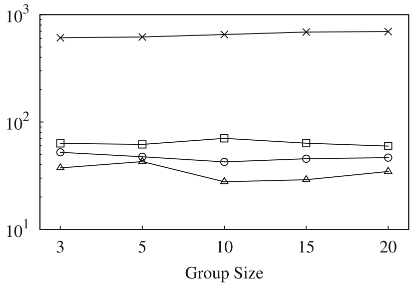
Fig. 9. Comparison when varying vector dimensionality

Fig. 10 that the query time decreases when the group size increases. The reason is that the increase in the group size increases the importance of the spatial distance component, which helps to prune the search space faster.

Effect of user maximum distance. The fourth parameter is the maximum distance between the users in the group, which we vary from 0.05 to 1° of latitude and longitude and sample instances randomly so that their distances satisfy this bound. Similar to the effect of the group size, the query time and I/O increases when distance increases as shown in Fig. 11.

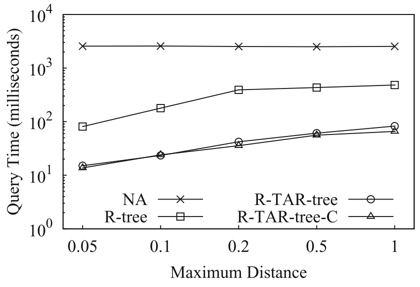


(a) Foursquare Query Time

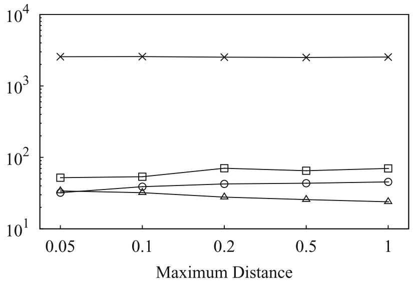


(b) Yelp Query Time

Fig. 10. Comparison when varying user group size



(a) Foursquare Query Time



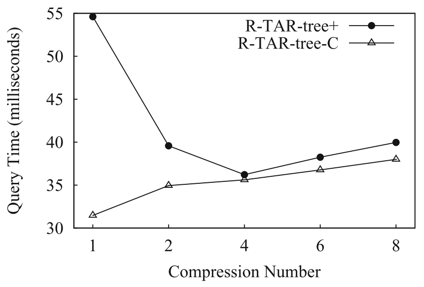
(b) Yelp Query Time

Fig. 11. Comparison when varying group average distance

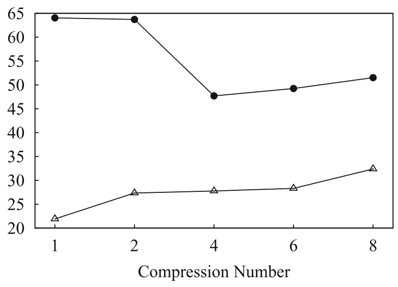
Effect of compression size. The compression techniques used in R-TAR-tree+ and R-TAR-tree-C are different in nature. For R-TAR-tree+, only vectors in non-leaf nodes of the tree are compressed, rendering it an exact method, while R-TAR-tree-C is an approximate method since it compresses all vectors. The compression factor m therefore plays different role in them as shown in Fig. 12: the query time of R-TAR-tree-C increases with compression size, while the query time of R-TAR-tree+ first decreases and then increases. When m is too small, the approximate vectors in non-leaf nodes are too loose to be useful, resulting in higher I/O cost. However, when m increases, the approximation becomes better, which brings the final kinked pattern.

5.3 Effectiveness

In this section, we study the performance of R-TAR-tree-C since it is an approximate method. We use Precision@k $= |E_k \cap R_k|/k$ for measuring the accuracy of the result R_k computed by R-TAR-tree-C compared to the exact top-k result E_k. Table 3 shows that when the compression factor m equals 4, the result set is nearly same as that of exact search for topic vector dimensionality of 20.



(a) Foursquare Query Time



(b) Yelp Query Time

Fig. 12. Comparison when varying the compression parameter m

Table 3. Precision@k when varying the compression parameter m

Dataset	Foursquare					Yelp				
Precision@k	1	2	4	6	8	1	2	4	6	8
1	0.110	0.290	1.0	1.0	1.0	0.0	0.010	0.990	1.0	1.0
5	0.322	0.612	1.0	1.0	1.0	0.018	0.070	0.998	1.0	1.0
10	0.563	0.781	1.0	1.0	1.0	0.123	0.270	0.999	1.0	1.0
20	0.841	0.937	1.0	1.0	1.0	0.494	0.653	1.0	1.0	1.0

5.4 Storage Requirements

Although the TAR-tree method is more efficient than the basic R-tree, it needs more storage. Table 4 shows the space occupied by the TAR-tree versions and the R-tree for different vector dimensionality. R-TAR-tree has higher storage size compared to R-tree, but R-TAR-tree+ reduces the cost a little using compression. The differences are not big because the R-tree also stores the topic vectors of places in the leaf nodes to avoid random accesses for fetching them. Finally, R-TAR-tree-C compresses all vectors and therefore has the least storage requirement among all methods.

Table 4. Storage size when varying vector dimension

Dataset	Foursquare					Yelp				
Vector dimensionality	5	10	20	50	100	5	10	20	50	100
R-tree	47.0M	70.2M	118.6M	263.1M	492.5M	9.8M	14.6M	24.5M	54.3M	101.6M
R-TAR-tree	47.4M	71.4M	122.3M	284.0M	578.0M	10.0M	14.9M	25.5M	59.0M	119.5M
R-TAR-tree+	47.2M	70.2M	119.7M	269.1M	514.9M	9.8M	14.7M	24.6M	55.1M	106.3M
R-TAR-tree-C	35.4M	42.6M	56.5M	100.2M	182.1M	7.4M	8.9M	11.8M	20.8M	37.7M

6 Related Work

Group Recommendation. Recommender systems [1] have been successfully deployed in a wide range of applications, such as inferring the preferences of a given user on a set of items (e.g., products, services, events, venues) and recommending the items with the highest probability to be liked by the user. Group recommendation is also supported, with the input of a set of users, recommending the items that are most likely to be favored by the input group. Early approaches [15,21], for each item, combine the predicted ratings of all group members to derive a single representative rating for the group and then suggest to the group the items with the highest representative ratings. In [3] the agreement between group members is also considered. More recently [16], the social relationships between members are used to enhance the quality of group recommendations. Topic-based group recommendation models [18,35] estimate the impact (i.e., influence) that every user in the group has on the other group members for different topics (i.e., aspects) of the objects to be recommended. For the case where the recommended items are venues, the spatial preferences of the group have to be considered. In [35], the distances between candidate venues and those visited in the past by the users of the group are used as a factor in the group recommendation model. To our knowledge, no previous work considers the current locations of users in the target group.

Group Spatial Search. Nearest neighbor (NN) search [26] is one of the most common spatial queries, well studied in both Euclidean [12] and spatial network spaces [24]. Given a query location q and a collection P of spatial points of interest, NN query retrieves the nearest spatial object in P to q (or the k nearest ones in kNN search). NN queries, were extended to aggregate NN (ANN) queries in [23]. An ANN query takes as input a group $Q = \{q_1, q_2, ..., q_m\}$ of m query locations (e.g., representing different users who want to dine together). The objective is to find the object p in P that minimizes an aggregation of the distances from each $q_i \in Q$ to p (e.g., $\sum_{q_i \in Q} dist(q_i, p)$ or $\max_{q_i \in Q} dist(q_i, p)$. The distance function can be Euclidean distance [23] or spatial network distance [34]. Papadias et al. [23] propose an algorithm for ANN queries, which applies on the original Euclidean space and assumes that P is indexed by an R*-tree. Yiu et al. [34] present algorithms which adapt the top-k retrieval methods of Fagin et al. [9] to compute ANN queries in spatial networks.

Later, the spatial skyline query (SSQ) [27] has been proposed as an alternative of ANN search, which utilizes the concept of skyline query [6]. Similar to ANN search, the input is a set P of points of interest and a set Q of m query locations, representing the locations of a group of users who want to meet. A point $p_i \in P$ is said to spatially dominate another point $p_j \in P$ if for each $q \in Q$, $dist(q, p_i) \leq dist(q, p_j)$. Intuitively, in this case, p_i would be a better point to meet compared to p_j in the eyes of every user in the group. SSQ reports as the spatial skyline the largest subset of P which contains only points that are not spatially dominated by others in P. Voronoi Diagrams are precomputed and indexed to facilitate the efficient computation of spatial skylines in [27,29]. [8] extended the techniques of [27] to apply on a road network, where Euclidean distance is replaced by shortest path distance. Later, [36] proposed a novel index and a filter-and-refine approach for this problem. Dynamic skyline queries in general metric spaces have been investigated in [7]. However, both ANN or SSQ do not consider the explicit or implicit preferences of the users.

Preference-based Meeting Point Search. [28] extends SSQ to a Spatio-Textual Skyline (STS) query, which allows users to find places that are both close to the locations of the group users and relevant to a set of user-defined keywords. Three different models for integrating textual relevance into spatial skylines are proposed. Among them, model STD (Spatio-Textual Dominance), which replaces the spatial distance measure of the derived dimensions by a combined spatio-textual distance, is experimentally shown to be the best one. Another recent work [17] proposed a Location-aware Group Preference (LGP) query, which suggests a place labeled with a specified category feature (e.g., hotel) to a group of users. Each user in the group has a location and a set of additional preferences. The query result should belong to the specified category and should be near the current location of the users and close to places satisfying the additional preferences of users. In this project, we assume that the group members are spatially dispersed and study methods for venue recommendation that consider (i) the traveling cost of the group members to the suggested venues based on their locations and (ii) the implicit preferences of users for the venues.

Spatial Topic Modeling. Spatial topic models have been extensively studied in the past decade. Yin et al. [33] proposed a model based on Probabilistic Latent Semantic Indexing (PLSI), after observing that geographical distributions can help model topics while topics group different geographical regions. Hong et al. [13] proposed a sparse generative model to uncover geographical topic patterns on Twitter. Liu et al. [19] proposed the spatio-temporal topic model to capture microscopic and macroscopic patterns of check-ins. Ahmed et al. [2] presented a hierarchical topic model which captures regional variations of topics. Spatial topic models are typically used for location recommendation [14,31,32]. For example, Yin et al. [31] proposed a system (LCARS) that exploits both local preferences and item content information for spatial item recommendation. Since our main focus is to model the implicit user preferences with spatial topic modeling, we adapted the most commonly used Latent Dirichlet Allocation (LDA) model [5] for simplicity.

7 Conclusion

This paper is the first study on location-aware group recommendation queries for passive users (P-LAG problem), which builds on previous work on meeting point recommendation. We follow a two-step framework: offline topic vector extraction and online querying based on appropriate indexing. Three variants of a TAR-tree approach are proposed, as well as a vector compression technique that improves search performance and reduces the storage requirements. In the future, we plan to test alternative approaches for preference extraction or elicitation and apply our framework with alternative definitions of spatial distance (e.g., travel distance on spatial networks).

Acknowledgements. We thank the reviewers for their valuable comments. This work is partially supported by GRF Grants 17201414 and 17205015 from Hong Kong Research Grant Council. It has also received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 657347.

References

1. Aggarwal, C.C.: Recommender Systems: The Textbook. Springer, Switzerland (2016)
2. Ahmed, A., Hong, L., Smola, A.J.: Hierarchical geographical modeling of user locations from social media posts. In: WWW, pp. 25–36. ACM (2013)
3. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: semantics and efficiency. PVLDB **2**(1), 754–765 (2009)
4. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The r*-tree: an efficient and robust access method for points and rectangles. In: SIGMOD, vol. 19, pp. 322–331. ACM (1990)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. JMLR **3**, 993–1022 (2003)
6. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430. IEEE (2001)
7. Chen, L., Lian, X.: Dynamic skyline queries in metric spaces. In: EDBT, pp. 333–343. ACM (2008)
8. Deng, K., Zhou, X., Tao, H.: Multi-source skyline query processing in road networks. In: ICDE, pp. 796–805. IEEE (2007)
9. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci. **66**(4), 614–656 (2003)
10. Gao, H., Tang, J., Liu, H.: gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In: CIKM, pp. 1582–1586. ACM (2012)
11. Guttman, A.: R-trees: a dynamic index structure for spatial searching, vol. 14. ACM (1984)
12. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. TODS **24**(2), 265–318 (1999)
13. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsioutsiouliklis, K.: Discovering geographical topics in the twitter stream. In: WWW, pp. 769–778. ACM (2012)
14. Hu, B., Ester, M.: Spatial topic modeling in online social media for location recommendation. In: ACM RecSys, pp. 25–32. ACM (2013)
15. Jameson, A., Smyth, B.: Recommendation to groups. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 596–627. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72079-9_20
16. Li, K., Lu, W., Bhagat, S., Lakshmanan, L.V., Yu, C.: On social event organization. In: SIGKDD, pp. 1206–1215. ACM (2014)
17. Li, M., Chen, L., Cong, G., Gu, Y., Yu, G.: Efficient processing of location-aware group preference queries. In: CIKM, pp. 559–568. ACM (2016)

18. Liu, X., Tian, Y., Ye, M., Lee, W.-C.: Exploring personal impact for group recommendation. In: CIKM, pp. 674–683. ACM (2012)
19. Liu, Y., Ester, M., Qian, Y., Hu, B., Cheung, D.W.: Microscopic and macroscopic spatio-temporal topic models for check-in data. TKDE (2017)
20. Lu, Z., Li, H., Mamoulis, N., Cheung, D.W.: Hbgg: a hierarchical Bayesian geographical model for group recommendation. In: SDM (2017)
21. O'connor, M., Cosley, D., Konstan, J.A., Riedl, J.: Polylens: a recommender system for groups of users. In: Prinz, W., Jarke, M., Rogers, Y., Schmidt, K., Wulf, V. (eds.) ECSCW 2001. Springer, Dordrecht (2001). doi:10.1007/0-306-48019-0_11
22. Omohundro, S.M.: Five balltree construction algorithms. International Computer Science Institute Berkeley (1989)
23. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, pp. 301–312. IEEE (2004)
24. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB, pp. 802–813. VLDB Endowment (2003)
25. Ram, P., Gray, A.G.: Maximum inner-product search using cone trees. In: ACM SIGKDD, pp. 931–939. ACM (2012)
26. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD, vol. 24, pp. 71–79. ACM (1995)
27. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: VLDB, pp. 751–762. VLDB Endowment (2006)
28. Shi, J., Wu, D., Mamoulis, N.: Textually relevant spatial skylines. TKDE **28**(1), 224–237 (2016)
29. Son, W., Lee, M.-W., Ahn, H.-K., Hwang, S.: Spatial skyline queries: an efficient geometric algorithm. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) SSTD 2009. LNCS, vol. 5644, pp. 247–264. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02982-0_17
30. Wu, D., Cong, G., Jensen, C.S.: A framework for efficient spatial web object retrieval. PVLDB **21**(6), 797–822 (2012)
31. Yin, H., Sun, Y., Cui, B., Hu, Z., Chen, L.: Lcars: a location-content-aware recommender system. In: SIGKDD, pp. 221–229. ACM (2013)
32. Yin, H., Zhou, X., Shao, Y., Wang, H., Sadiq, S.: Joint modeling of user check-in behaviors for point-of-interest recommendation. In: CIKM, pp. 1631–1640. ACM (2015)
33. Yin, Z., Cao, L., Han, J., Zhai, C., Huang, T.: Geographical topic discovery and comparison. In: WWW, pp. 247–256. ACM (2011)
34. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. TKDE **17**(6), 820–833 (2005)
35. Yuan, Q., Cong, G., Lin, C.-Y.: Com: a generative model for group recommendation. In: SIGKDD, pp. 163–172. ACM (2014)
36. Zou, L., Chen, L., Özsu, M.T., Zhao, D.: Dynamic skyline queries in large graphs. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part II. LNCS, vol. 5982, pp. 62–78. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12098-5_5