Human Computation and Crowdsourcing: Works in Progress Abstracts:
An Adjunct to the Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing

LoRUS: A Mobile Crowdsourcing System for Efficiently Retrieving the Top-k Relevant Users in a Spatial Window

Anirban Mondal, Gurulingesh Raravi, Amandeep Chugh, Tridib Mukherjee
Xerox Research Center India, Bengaluru, India
{Anirban.Mondal, Gurulingesh.Raravi, Amandeep.Chugh, Tridib.Mukherjee}@xerox.com

Abstract

The prevalence of mobile devices and applications strongly motivate mobile crowdsourcing for facilitating location-dependent services. We propose LoRUS, a **Lo***cation-based* **R***elevant* **U***ser determination* **S***ystem for* efficiently *retrieving the top-k relevant mobile users in a given spatial window.*

1 Introduction

Given the ever-increasing popularity and proliferation of mobile devices, it has now become practically feasible to enable people to share information about dynamic events (e.g., trees fallen on roads due to a storm, sudden truck breakdowns and unscheduled processions) in their current location. This strongly motivates facilitation of various kinds of location-dependent information retrieval services and applications using mobile devices. Such mobile applications have an inherently *spatio-temporal* aspect in the sense that only those mobile users, who are present near a given location at that time, are able to respond to queries arising from other users. Consider a user Alice, who just heard from her colleagues about an unscheduled procession in a route that she uses daily for driving home. She may have queries such as *"How many cars are currently stuck due to the procession?"*, *"How many hours is the route likely to remain blocked?"* and so on. Here, only the mobile users in the vicinity of the procession can answer her queries.

In such above mentioned applications, it is critical to determine the most **relevant users** (i.e., those who are more likely to respond to such queries in a relatively accurate manner) in a given spatial window within a valid **time-duration**. For example, since many users are likely to be shopping at a large retail store for at least an hour, the time-duration for queries related to offers in that store could be set to 60 minutes. In effect, this means that only those users, who have been present in the store within the past 60 minutes would be considered as relevant for answering queries. Interestingly, the time-duration appropriate for each kind of query can vary. Thus, the ability to handle multiple time-durations enables *personalization* towards the diverse querying needs of users.

In this work, the relevance of a user w.r.t. a given query is based on the user score, which depends upon the user's level of engagement in responding relatively accurately to queries in the past. Thus, the scoring can be based on the percentage of queries answered by a user, the timeliness of the response, as well as the quality of the response. Hence, *given a spatial window, we address the problem of efficiently retrieving the top-k relevant mobile users who are moving in that window within valid (possibly multiple) time-duration(s).*

Existing broadcast approaches flood the network by sending queries to all the users since they do not keep track of user scores. Hence, they do not address mobile resource constraints (e.g., energy, bandwidth) and also result in unnecessary spam. On the other hand, multi-cast approaches *randomly* send the queries to some of the users to preserve mobile resources, but they do not ensure the direction of queries to the most relevant users. Furthermore, spatial indexes, such as the R-tree (Guttman 1984), the quadtree (Finkel and Bentley 1974) and the Q+R-tree (Y.Xia and Prabhakar 2003), cannot *efficiently* retrieve information about the most relevant mobile users within a given spatial region, let alone at varied spatial granularities and for different time-durations.

We propose LoRUS, a **Lo**cation-based **R**elevant **U**ser identification **S**ystem. LoRUS can *efficiently* retrieve the top-k relevant mobile users in a given spatial window. The key contribution of LoRUS is the τR-tree, a novel R-tree-based index augmented with an array for indexing different time-durations and fixed-length arrays for indexing the most relevant users in space for each pre-defined time-duration.

2 τR-tree: An efficient index for retrieving relevant users in a spatial window

The τR-tree is a novel R-tree-based spatial index in which each node of the R-tree is augmented with (a) an array for indexing the recent different time-durations; and (b) fixed-length arrays for indexing the most *relevant* users in the Minimum Bounding Rectangle (MBR) of the region covered by a given R-tree node for each time-duration.

Non-leaf nodes of the τR-tree are of the form (ptr, mbr, AR), where ptr is a pointer to a child node in the τR-tree, and mbr is the MBR, which covers all the MBRs in that child node. Here, AR is an array, each entry of which corresponds to a given time-duration. The size of each time-duration is a constant T. Suppose T=10 minutes. Here, $AR[1]$ corresponds to information about the relevant users in mbr for the past 10 minutes; $AR[2]$ concerns the past 20 minutes and so on. Furthermore, the array has a fixed n number of entries for indexing n different time-durations. Here, the values of T and n are both application-dependent.

Each entry of AR contains a pointer to a fixed-length array A, which contains information about relevant users for the given time-duration in the region covered by the node. Each such fixed-length array A contains entries of the form (U_{id}, U_{score}), where U_{id} represents the unique identifier of a given user, while U_{score} refers to her score. The elements of A are kept sorted in descending order of the user scores for facilitating quick retrieval of the most relevant users. Each fixed-length array structure (for indexing user information) at each τR-tree node has a maximum size limit λ.

Leaf nodes of the τR-tree are of the form (pt, $rect$, AR), where pt is a pointer to a rectangle in the spatial database, while $rect$ is the MBR of that rectangle. Here, the significance of AR is essentially the same as explained above. Creation, search and updates in the τR-tree use standard R-tree algorithms (Guttman 1984) with the only exception being the handling of the time-duration arrays and the fixed-length array structures that store user information. Using the τR-tree, we obtain top-k relevant users (with user scores) corresponding to each of the queried time-durations. Then we compute a net user score across the time-durations.

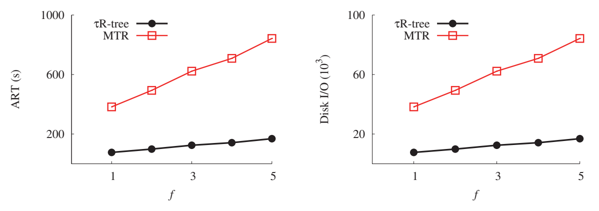
3 Performance Evaluation

In our experiments, we simulate the positions of mobile users by attaching a given user to each spatial point (representing a landmark) in the dataset; then the user moves within a fixed radius of that point. In consonance with practice, this ensures that regions with a high number of points (landmarks) have a higher density of users. Each user was randomly assigned a score between 0 to 100.

For generating a synthetic dataset with 1 million spatial points, we divide the universe into a grid of $10X10$ equal-sized cells. The points are assigned to the cells using a Zipf distribution with zipf factor ZF_S of 0.7, which represents a highly skewed spatial distribution. Thus, there would be a relatively few **dense cells** containing a large number of points, while the other cells would be **sparse**. Within a given cell, the location of each point is decided randomly.

We considered a total of 8 time-durations; queries were selected to be across the recent 5 time-durations. We set the value of k to 10 for all our experiments i.e., all experiments aimed at retrieving the top-10 relevant users in a given spatial window. We set k' to 15 i.e., each query was sent to 15 users. The query window scaling factor f scales the area of a square window query. We set the default area of a given query window (i.e., for the case when f = 1) to 1% of the universal space. Moreover, we set the maximum number M of entries in the τR-tree node to 40. The maximum size λ for each fixed-length (user) array in the τR-tree is set to 50 for all our experiments. We used the disk page size as 4 KB.

For performance comparison, we adapt and combine variants of the approaches proposed in (Y.Xia and Prabhakar 2003) and (Papadias, Mamoulis, and Theodoridis 1999) to our scenario. We adapt the approach in (Y.Xia and Prabhakar 2003) by building an R-tree on the mobile users to index the users who are in a given region at a point in time. Similar to the τR-tree, each R-tree node in this reference approach is augmented with fixed-length array structures for indexing user IDs for a specific time-duration. Similar to (Papadias, Mamoulis, and Theodoridis 1999), our reference approach creates and maintains multiple R-trees i.e., one R-tree for each time-duration. After traversing the corresponding R-trees for each of the time-durations, it collates the results and sends the query to any **randomly** selected k users. We designate the reference approach as **MTR (Multiple Temporal R-Trees)**. MTR does not consider user scores.



(a) ART (b) Disk I/O

Figure 1: Effect of variations in the query window scaling factor f for Dense Regions (Synthetic spatial dataset)

Our performance metrics are **average response time (ART)** and **number of disk I/Os** incurred by the queries. Figure 1 depicts the results of the experiment in which we selected a query window randomly from one of the relatively *dense* cells and then varied the query window scaling factor f. As the value of f increases, the query window area also increases. Hence, the query window intersects more index nodes, thereby resulting in higher disk I/Os and consequently higher ART. The predominant cost here is the disk I/O cost due to index node retrievals. Hence, ART follows a similar trend to that of disk I/O for both approaches. Recall that we use the information across five time-durations for determining the top-k users for both approaches. Thus, unlike the τR-tree, MTR traverses five R-trees (one for each time-duration), thereby incurring higher ART and disk I/Os.

4 Conclusion

We have proposed the LoRUS mobile crowdsourcing system for *efficiently* retrieving the top-k relevant mobile users in a given spatial window. In the future, we plan to explore using LoRUS in conjunction with different incentive models.

References

Finkel, R., and Bentley, J. 1974. Quad trees: A data structure for retrieval on composite keys. In *Acta Informatica 4 (1)*.

Guttman, A. 1984. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*.

Papadias, D.; Mamoulis, N.; and Theodoridis, Y. 1999. Processing and optimization of multiway spatial joins using R-trees. In *Proc. PODS*.

Y.Xia, and Prabhakar, S. 2003. Q+Rtree: Efficient indexing for moving object databases. In *Proc. DASFAA*.