Reading:
Dominant Graph: An Efficient Indexing Structure to Answer Top-K Queries

Yu-Pin Liang

What is the problem?

Background

- Given a record set D and a query function F, a top-k preference query (top-k query for short) returns k records from D, whose values of function F on their attributes are the **highest**.

- *For example*: Top-k queries are very popular in many applications. A job seeker may want to find the best jobs fit to her preferences, such as near to her home, high salary, and short working time. For different applicants, they may have their own ranking by assigning different weights. Some may put a high weight on "salary", while others care more about "reputation" of the company. No matter what their preferences are, all these queries **belong to top-k preference queries**.

What are the existing techniques and their limitations?

Existing techniques

1. **Sorted lists-based:** The methods in this category sort the records in each dimension and aggregate the rank by parallel scanning each list until the top-k results are returned, such as **TA** and **CA**.

2. ***Layer-based :***The algorithms in this category organize all records into consecutive layers, such as **Onion** and **AppRI**.

3. **View-based:** Materialized views are used to answer top-k query, such as PREFER and LPTA.

What are the existing techniques and their limitations?

Limitations

1. **Sorted lists-based:** The algorithms in the first category need to merge different sorted lists, and their query performance is not as good as the rest two categories.

2. **Layer-based and view-based approaches:**
   - 1) all existing algorithms in these two categories only support linear query function
   - 2) high maintenance cost.

What makes it possible for the authors to address these limitations?

Dominant Relationships - use of dominant relationship to answer top-k query

The intuition is that for the aggregate monotonic function, F, of a top-k query, when record X dominates record Y, the query score of X must be less than that of Y. It also means that X can be pruned safely when we know Y is not in top-k answer. Therefore, top-k answers can be derived from **offline** generated dominant relationships among all the records.

How is the problem actually solved in this paper?

1. Represent the **dominant relationships** among the records into a partial order graph, called **Dominant Graph (DG).** In DG index, records are linked together according to the dominant relationship. Based on offline built DG, Traveler algorithm be proposed, which transforms top-k query into a graph traversal problem.

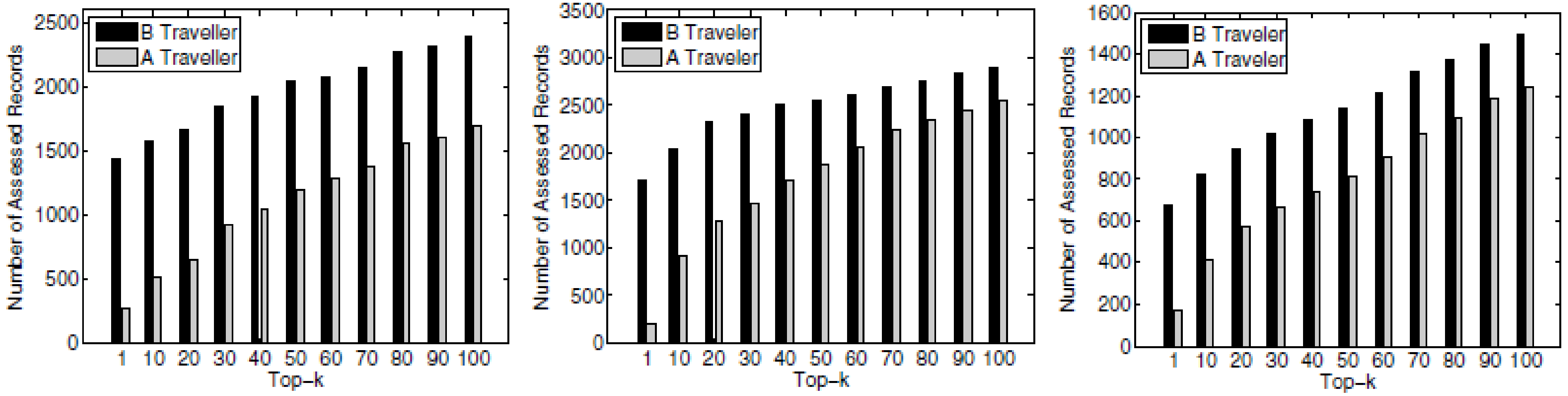
2. An analytic model to the query cost of Traveler. To improve the query performance, ***pseudo records***, and **extend Traveler** to its advanced version, Advanced ***Traveler*** algorithm also be introduced.

3. In order to handle high-dimensional datasets, N-Way Travel algorithm also be proposed for top-k query.

How is the performance of the proposed technique compared to that of the existing ones?

Experiment 1. (Compare Basic and Advanced Traveler)

- in 5 dimension data sets, Uniform (U 5), Gaussian (G 5) and Correlated (R 5)
- the number of all accessed records in Advanced Traveler are much less than that in Basic Traveler,



(a) U_5, $|D| = 1M$, $|m|$=5 (b) G_5, $|D| = 1M$, $|m|$=5 (c) R_5, $|D| = 1M$, $|m|$=5

Fig. 5. *The Number of Accessed Records During Top-K Query*

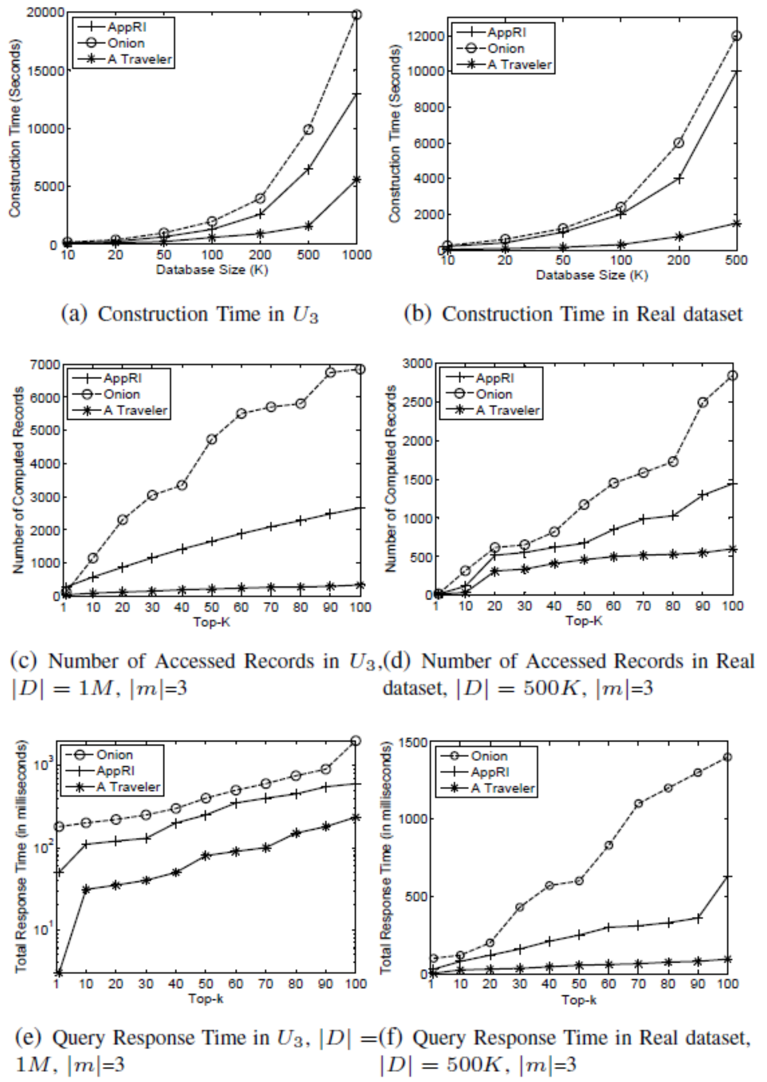
How is the performance of the proposed technique compared to that of the existing ones?

Experiment 2. (Comparison with Existing Top-K algorithms)

- Performance of Traveler (that is Advanced Traveler) is compared with two representative layer-based top-k algorithms, such as **Onion** and **AppRI.**

- Number of accessed records in Traveler algorithm is much smaller than that in AppRI and Onion.

- The query response time in Fig. 6(e) and Fig. 6(f) also confirm the performance of Traveler



(a) Construction Time in U_3 (b) Construction Time in Real dataset

(c) Number of Accessed Records in U_3, $|D| = 1M$, $|m|$=3 (d) Number of Accessed Records in Real dataset, $|D| = 500K$, $|m|$=3

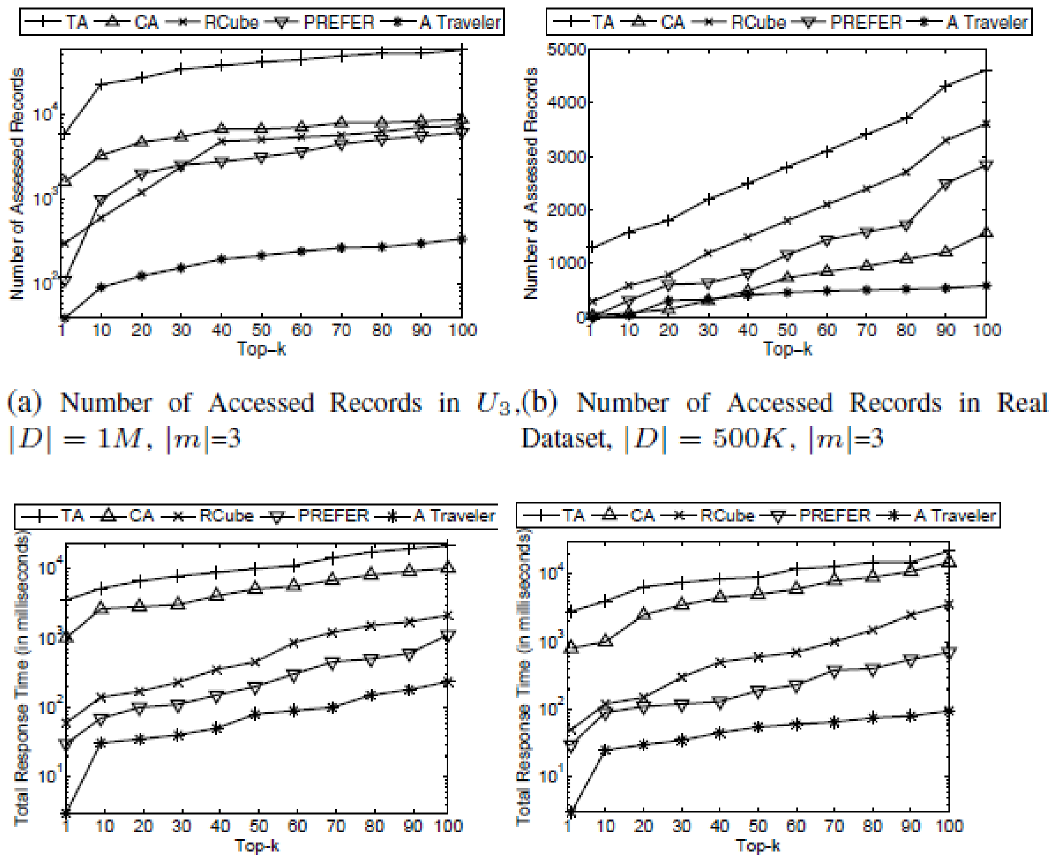
(e) Query Response Time in U_3, $|D| = 1M$, $|m|$=3 (f) Query Response Time in Real dataset, $|D| = 500K$, $|m|$=3

Fig. 6. *Comparison With Layer-Based Algorithms*

How is the performance of the proposed technique compared to that of the existing ones?

Experiment 2. (Comparison with Existing Top-K algorithms)

- Compare traveler with some non-layer based top-k algorithms, that are TA, CA, PREFER and RankCube.
- Traveler outperforms other algorithms in both the number of accessed records and query response time



(a) Number of Accessed Records in U_3, $|D| = 1M$, $|m|$=3

(b) Number of Accessed Records in Real Dataset, $|D| = 500K$, $|m|$=3

(c) Query Response Time in U_3, $|D| = 1M$, $|m|$=3

(d) Query Response Time in Real dataset, $|D| = 500K$, $|m|$=3

Fig. 7. *Comparison With Non-Layer Based Algorithms*

How is the performance of the proposed technique compared to that of the existing ones?

Experiment 3. (DG maintenance)

- In the experiment, we evaluate the DG maintenance algorithms and compare them with other layer-based algorithms.

- DG maintenance algorithm is suitable in the online process to handle insertion and deletion operations, since the running times are less than 140 seconds and 500 seconds for 10K insertions and deletions.

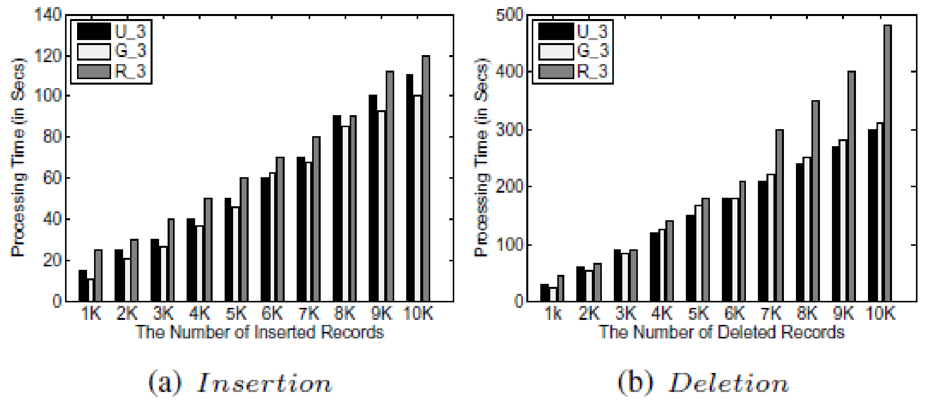
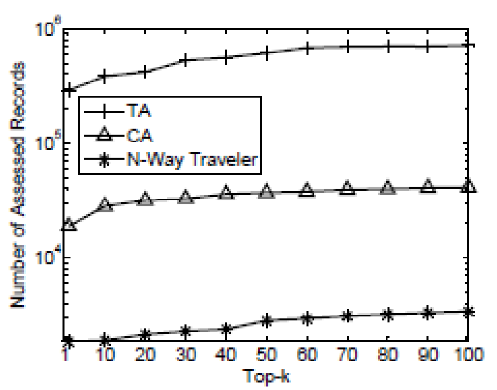


Fig. 8. *Evaluating DG Maintenance*

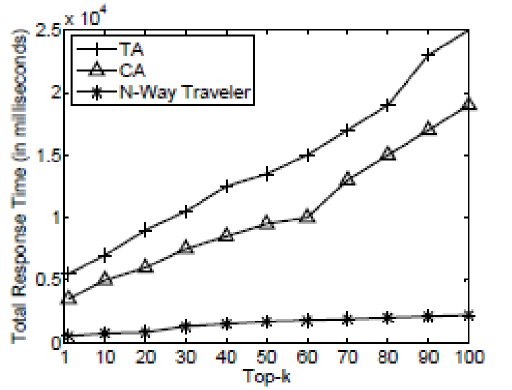
How is the performance of the proposed technique compared to that of the existing ones?

Experiment 4 (High Dimension and Worst Case Testing)

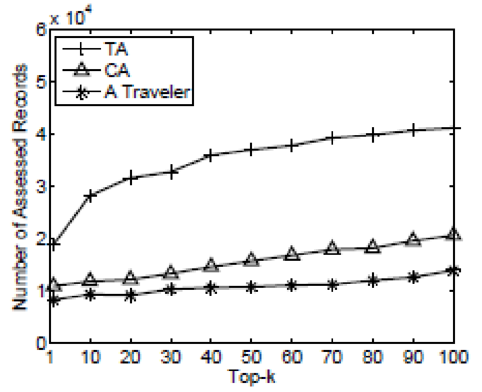
- Use TA and CA algorithms here for comparison with N-Way Traveler
- Advanced Traveler still works well in the worst case.



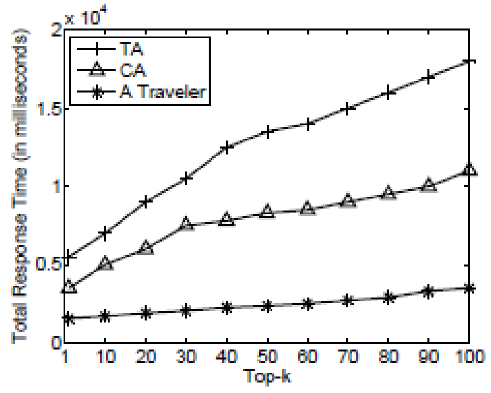
(a) Number of Accessed Records, $|m| = 10$, $|D|$= 1M



(b) Query Response Time, $|m| = 10$, $|D|$= 1M



(c) Number of Accessed Records in the worst case, $|D|$=100K, $|m| = 5$



(d) Query Response Time in the worst case, $|D|$=100K, $|m| = 5$

Fig. 9. *Evaluating Traveler in High-Dimension and the Worst Case*

Conclusion

1. Based on the intrinsic connection between **top-k** problem and **dominant relationships**, we propose an efficient **indexing structure DG and a top-k query algorithm Traveler**
2. An analytic cost model be proposed for our basic query algorithm and prove that the query cost is directly related to the cardinality of skylines in the record set.
3. Extend Basic Traveler to its advanced version by introducing pseudo records to reduce the cost, and design N-way Traveler algorithm to handle the high dimension problem.
4. Extensive experiments confirm that new approaches have significant improvement over the previous methods.
5. Furthermore, due to advantage of DG indexing structure, the “light” DG maintenance algorithm be proposed to handle insertion and deletion in the online process.

What do you think about this paper?

- Authors have very clear central idea and each paragraph deliver a clear main point or topic sentence.
- The topic is very practical and can be used in many way to reduce the time and resource to solve the top-k problem.
- The solution is innovative with an efficient layer-based indexing structure, Dominant Graph (DG).
- The authors provides many detailed information (pseudo codes) and graphs to make readers better understand each intermediate steps.
- This study can bring a huge impact to the top-k related problems due to the innovative approach and excellent performance of the ***Traveler*** algorithm.