Supporting Multi-criteria Decision Support Queries over Time-Interval Data Streams

Nam Hun Park[1,⋆], Venkatesh Raghavan[2], and Elke A. Rundensteiner[2]

[1] Department of Computer Science, Anyang University, Incheon, Republic of Korea
nmhnpark@anyang.ac.kr
[2] Department of Computer Science, Worcester Polytechnic Institute, Massachusetts, USA
{venky,rundenst}@cs.wpi.edu

Abstract. Multi-criteria result extraction is crucial in many real-time stream processing applications, such as habitat and disaster monitoring. The ease in expressing user preferences makes *skyline* queries a popular class of queries. Skyline evaluation is computationally intensive especially over continuous time-interval streams where each object has its own individual expiration time. In this work, we propose *TI-Sky* – a continuous skyline evaluation framework. TI-Sky strikes a perfect balance between the costs of continuously maintaining the result space upon the arrival of new objects or the expiration of old objects, and the costs of computing the final skyline result from this space whenever a pull-based user query is received. This is achieved by incrementally maintaining a precomputed skyline result space at a higher level of abstraction and digging into the more expensive object-level processing only upon demand. Our experimental study demonstrates the superiority of *TI-Sky* over existing techniques.

1 Introduction

A wide array of applications including outpatient health care research and sensor monitoring need to provide real-time support for complex multi-criteria decision support (MCDS) queries. The intuitive nature of specifying a set of user preferences has made *skyline* queries a popular class of MCDS queries [1, 2]. Skyline result enable analysts to specify preferences along several different criteria and to learn about the trade offs between the different criteria.

The increased usage of sensors and RFID networks in various real-world scenarios has increased the availability of data streams [3]. Stream query processing typically deals with incoming data that are unbounded and time-variant (each with its own life span). Time-sensitive applications, such as sensor monitoring require query execution to keep up with incoming objects to produce real-time results.

State-of-the-Art Techniques. Skyline algorithms [1] over static databases are not viable for computing continuous skylines over time-interval data streams as they tend to assume that the data is static and rely on having all data a priori organized into indices to

⋆ This collaborative work was conducted as a research visiting scholar at Worcester Polytechnic Institute.

P. García Bringas et al. (Eds.): DEXA 2010, Part I, LNCS 6261, pp. 281–289, 2010.

facilitate query evaluation. In addition, they do not consider time-variant data as found in time-interval data streams.

Existing continuous skyline algorithms [3, 4] use a *sliding window model* that fixes the expiration times for *all* data objects to be the end of the window period. A new object o_{new} always expires after all objects in the current window and therefore can forever eliminate all existing objects that it dominates. In contrast, for time-interval streams an older object with a longer life span than o_{new} could revert back to becoming a skyline point in the future. This complicates the object dominance reasoning and worst yet may cause in some cases all objects to have to be retained. Thus a time-interval model is more general and complex than the sliding window model. Methods proposed for the time-interval model could also be used for the sliding window model, but not vice versa.

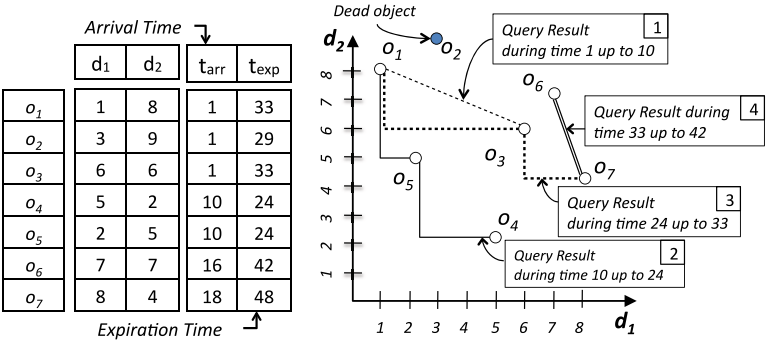
While [5] also handles skylines over time-interval streams, it suffers from the drawback of re-evaluating the skyline for each update and for each pull-based query. [3–5] work exclusively at the object granularity level - which is much more computation intensive for high volume data streams than our proposed higher-level abstraction.

Our Proposed Approach. We propose the *TI-Sky* a framework to evaluate continuous skylines over time-interval streams. Fully maintaining the skyline result space for each time instance makes answering user queries cheap. However, for high volume data streams the CPU resources needed to maintain this moving skyline result space are too prohibitive to be practical. Conversely, if we do not maintain this space, the results would need to be computed from scratch for each user pull request thereby negatively affecting performance. Therefore, our **optimization mantra** is *"to strike the perfect balance between the cost of maintaining the moving skyline result space and the cost of computing the final skyline result."* This is achieved by incrementally maintaining the partially precomputed skyline result space – however doing so efficiently by working at a higher level of abstraction. We introduce the notions of *Macro and Micro-TDominance* to model the time-based dominance relationships at the individual object- as well as at the abstract-granularity. We then design algorithms for insertion, deletion, purging and result retrieval that effectively exploit both layers of granularity. Our contributions include:

- We propose *TI-Sky* – an efficient execution framework to process multi-criteria decision support queries over time-interval data streams.
- We take the idea of time-based dominance to a new level by designing a two-layered model of time-based dominance, namely, *macro-* and *micro-* time-dominance.
- We propose efficient algorithms to handle real-time insertion, deletion, purging and result retrieval that effectively exploit both layers of our dominance model.
- Our experimental study demonstrates *TI-Sky*'s superiority over existing techniques.

2 Skylines over Time-Interval Streams

Let $o_i.t_{arr}$ and $o_i.t_{exp}$ denote the arrival and expiration time of an object o_i. A *strict partial order* $o_i \succ_{\mathbb{P}} o_j$ exists if for all attribute dimensions k, $o_i[k] \leq o_j[k]$ and there exist at least one dimension k such that $o_i[k] < o_j[k]$. Below we extend the concept of value-based dominance proposed in [4] to incorporate the notion of time-intervals.



	d_1	d_2	t_{arr}	t_{exp}
o_1	1	8	1	33
o_2	3	9	1	29
o_3	6	6	1	33
o_4	5	2	10	24
o_5	2	5	10	24
o_6	7	7	16	42
o_7	8	4	18	48

(a) Sample Input Stream (b) Continuous Skylines over Time-Interval Stream

Fig. 1. Motivating Example

Definition 1. Time-Interval Dominance. *Given a preference $\mathbb{P}$, the time-interval stream S, and $o_i, o_j \in S$ s.t. $o_i \succ_{\mathbb{P}} o_j$ and $o_j.t_{exp} > o_i.t_{exp}$. Then o_i* **TDominates** *o_j ($o_i \xrightarrow{T} o_j$) iff $\nexists o_k \in S$ s.t. $(o_k \succeq_{\mathbb{P}} o_j) \wedge (o_i.t_{exp} < o_k.t_{exp} < o_j.t_{exp})$.*

In Figure 1 we observe that object $o_4 \xrightarrow{T} o_7$ and $o_5 \xrightarrow{T} o_3$. This means objects o_3 and o_7 become part of the skyline when the objects o_5 and o_4 expire respectively.

Definition 2 (Skyline Over Time-Interval Stream). *Given a time-interval stream S and a preference $\mathbb{P}$, the skyline over S at time t_i (denoted as $\mathcal{S}_{\mathbb{P}}(S, t_i)$) is the set of all non TDominated objects $o_k \in S$ with $o_k.t_{arr} \leq t_i$ and $o_k.t_{exp} > t_i$.*

In Figure 1 at $t = 9$, o_1 and o_3 are the skyline objects. At $t = 10$, the new object o_5 $(2, 5)$ that dominates the older object o_3 $(6, 6)$ arrives. However, the newer o_5 is valid for a shorter time frame ($o_5.t_{exp} = 24$) than the older object o_3 which is valid even after o_5 expires ($o_3.t_{exp} = 33$). This is reflected in Figure 1.b where the older object o_3 contributes to the result between the time frame 24 to 33 when o_5 expires.

3 TI-Sky: Our Proposed Approach

In this work, we partition the input space composed of incoming streaming objects (see Definition 3). Similar in spirit to the principle of TDominance between two objects, henceforth referred to as Micro-TDominace. We propose to elevate the notion of time-based dominance to the granularity of abstractions called Macro-TDominance.

Definition 3. *A* **partition cell**[1] *abstracts the set of objects that map into a the d-dimensional bounding box ($P_k.RBoundary$) defined by its lower and upper bounds. P_k.***SkyHeap**(t) *is a list of objects not dominated by other objects in P_k at time instance t. P_k.***RestHeap**(t) *maintains the objects in P_k currently dominated by objects*

[1] Henceforth, a **partition cell** P_i is termed for short as a **partition** P_i.

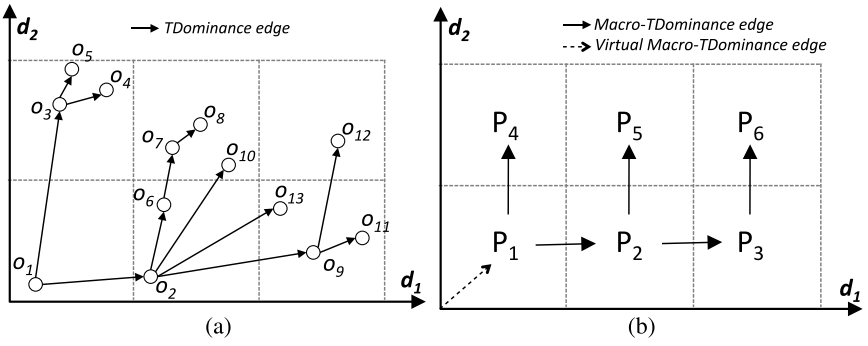


Fig. 2. Level of TDominance: (a) Object-Level (Micro) (b) Partition-Level (Macro)

in $P_k.SkyHeap$. P_k.**VBoundary** *denotes a vector of the min and max attribute values across all objects in* P_k. P_k.**TBoundary** *defines the minimum and maximum expiration time among objects in* P_k.

The concept of **Macro-TDominance** enables us to capture the time-based dominance relationship between partitions. The intuition is that non-empty root partitions of the Macro-TDominance tree are guaranteed to have skyline results and therefore should be investigated during the skyline data maintenance and result determination steps.

Definition 4. *Partition* P_f **Macro-TDominates** *partition* P_g*, denoted as* $P_f \xrightarrow{T} P_g$*, if and only if* $\exists\ o_i \in P_f \wedge \exists o_j \in P_g$ *such that* $o_i \xrightarrow{T} o_j$.

Example 1. In Figure 2 the Macro-TDominance relationships $P_1 \xrightarrow{T} P_4$ and $P_1 \xrightarrow{T} P_2$ exist since $o_1 \in P_1$ such that $o_1 \xrightarrow{T} o_3$, and $o_1 \xrightarrow{T} o_6$, where $o_3 \in P_4$ and $o_6 \in P_2$ respectively. Also, $P_{virtual} \xrightarrow{T} P_1$ is the *virtual Macro-TDominance* relationship.

In the Macro-TDominance tree, a partition P_i that is not Macro-TDominated by any other partition P_j is termed as a root partition and is guaranteed to at-least contain one skyline result. The arrival of new object o_{new} to one such root partition and the expiration of an older object from any root partition can affect the result space and therefore need to be processed immediately. In contrast operations in non-root partitions can be performed more lazily without affecting correctness.

During skyline result generation the Macro-TDominance tree aids us to identify partitions that have a higher likiehood of delivering results - without having to examine object-level dominance. To find the precise result set upon request, we begin by looking at objects within such root partitions and then iterate over partitions whose boundary values are not being dominated by the current skyline points. This approach of traversing the space avoids unnecessary object-level comparisons.

Definition 5. *Given a time-interval stream* S *and partition* P_k*, the* **Micro-TDominance** *relationship for the objects in* P_k *is simply the TDominance relationship among the objects in* P_k *rather than between all objects in* S.

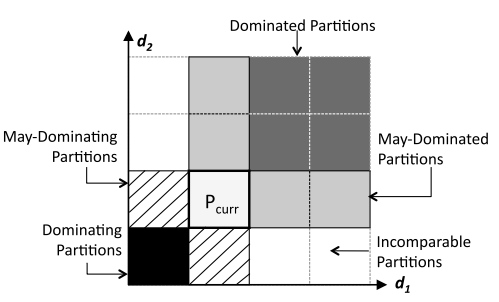


Fig. 3. Value-Based Categorization w.r.t. P_{curr}

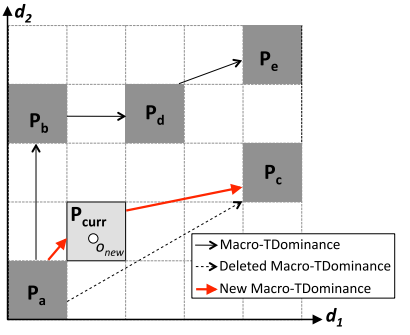


Fig. 4. Inserting o_{new} in P_{curr} and Traversing

4 Skyline Data Maintenance

We now describe the process of inserting a newly arriving object o_{new} into the abstract output skyline space. This is accomplished at two levels of granularity, namely both at the level of partitions as well as at the level of individual objects. For each new object o_{new}, we first map it to its corresponding partition. In Figure 4, o_{new} is mapped to the partition P_{curr}. Next, we perform local partition-level updates which encompass: (a) performing pairwise object-level comparisons against objects mapped to P_{curr} to determine whether o_{new} belongs to either *SkyHeap* or *RestHeap* of P_{curr}, and (b) updating the value- and temporal- bounds of P_{curr} (See Definition 3).

After local partition level operations, we determine the effects of o_{new} on the current and future skyline results. To avoid traversing the entire space as done in existing techniques [3–5], we propose first traverse the space at the higher abstraction of partitions and dig into the pairwise object comparisons as and when need. This is achieved by performing a breadth first search of Macro-TDominance tree starting with the *root partitions*. The goal of this exercise is to identify new Macro-TDominance relationships of the form $P_i \xrightarrow{T} P_{curr}$ and $P_{curr} \xrightarrow{T} P_i$ where P_i is a non-empty partition in the abstract output skyline space. To aid abstract-level decision making, we classify the visited partition P_i into four categories with respect to the partition P_{curr} as follows:

1. *Dominating Partition:* Partition P_i value-dominates P_{curr}.
2. *Dominated Partition:* Partition P_i is value-dominated by P_{curr}.
3. *May-Dominating Partition:* Objects mapped to the partition P_i can potentially value-dominate the new object $o_{new} \in P_{curr}$. But this cannot be determined with out performing pairwise object-level comparisons.
4. *May-Dominated Partitions:* Objects mapped to the partition P_i can potentially be value-dominated by the new object $o_{new} \in P_{curr}$. Similar to category 3 this can only be determined by performing pairwise comparisons.

Based on the above classification (as depicted in Figure 3) only partitions that are either in P_{curr}'s *Dominating Partition* or *May-Dominating* lists can Macro-TDominate P_{curr}.

Let us now consider the scenario where the currently visited partition P_i is in the May-Dominating list of P_{curr}. If the relationship $P_i \xrightarrow{T} P_{curr}$ already exists we merely update the time-stamp information of $P_i \xrightarrow{T} P_{curr}$. In contrast if the relationship $P_i \xrightarrow{T} P_{curr}$ does not exist - we first compare the VBoundary values of P_i and o_{new}. If upper bound of P_i's VBoundary dominates o_{new} then $P_i \xrightarrow{T} P_{curr}$, else there may still exist an object in P_i that dominated o_{new}. Next, we perform object-level comparisons to identify the object, say o_k, that dominated o_{new} that is last to expire. We then proceed to traverse the Macro-TDominance tree to check if there exist other partitions that can potentially Macro-TDominate P_{curr}.

Example 2. In Figure 4 initially, $P_a \xrightarrow{T} P_b$, $P_a \xrightarrow{T} P_c$, $P_b \xrightarrow{T} P_d$ and $P_d \xrightarrow{T} P_e$ are the Macro-TDominance relationships before the arrival of o_{new}. When P_{curr} is populated with o_{new}, we traverse the Marco-TDominance tree starting with P_a. Since P_a value dominates P_{curr} and there does not exist another partition in P_{curr}'s "*Dominating*" or "*May-Dominating*" list we add the Macro-TDominance $P_a \xrightarrow{T} P_{curr}$.

Next, we consider the relationship between P_{curr} and all partitions belonging to the *May-Dominated* and *Dominated* lists of P_{curr}. That is, we investigate if any $P_{curr} \xrightarrow{T} P_i$ relationships exist. Decisions can be made at the abstract level for *Dominated* lists and object-level comparisons are needed for partitions in the *May-Dominating* list.

Example 3. In Figure 4, $o_{new}.t_{exp} > P_a.TBoundary$. Thus we can delete the relationship $P_a \xrightarrow{T} P_c$ and insert the new Macro-TDominance relationship $P_{curr} \xrightarrow{T} P_c$

When building these Macro-TDominance relationships we want to avoid traversing the entire Macro-TDominance tree by exploiting the temporal properties of Macro-TDominance as in Theorem 1.

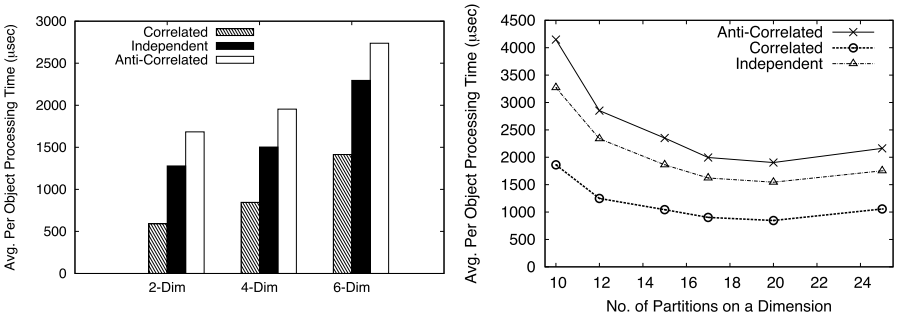
Theorem 1. *For the new object $o_{new} \in P_{curr}$ and a Macro-TDominance $P_i \xrightarrow{T} P_j$. If $\exists o_k \in P_i \nexists o_m, o_n \in P_j$ s.t. $(o_k \xrightarrow{T} o_m) \wedge (o_{new} \xrightarrow{T} o_n) \wedge (o_k.t_{exp} < o_{new}.t_{exp})$, then we do not need to investigate the Macro-TDomination sub-tree with P_j as its root.*

Example 4. In Figure 4, the Macro-TDominance $P_b \xrightarrow{T} P_d$ holds even after the arrival of the new object $o_{new} \in P_{curr}$. In such cases by Theorem 1 checking o_{new} against objects mapped to partitions with P_d as its root is unneccessary.

Detailed descriptions of the algorithms used to maintain the skyline result space due to object expiration and reducing memory usage by effective purging techniques are presented in the our technical report [6].

5 Skyline Result Determination

The *skyline result determination* phase piggybacks on the Macro-TDominance relationships gathered in the previous phase to determine the skyline results at any given time t_i. The root nodes (partitions) of the Macro-TDominance tree are guaranteed to have at least one query result. This translates into querying the Macro-TDominance tree [6].



(a) Avg. Per-Object Processing Time (b) Avg. Per-Object Processing Time

Fig. 5. Performance Evaluation of TI-Sky: (a) Effects of Dimensionality

6 Performance Study

State-of-the-Art Techniques Compared. We compare *TI-Sky* against *LookOut* [5] and the extension of the technique proposed in [4] here called *Stabbing-the-Sky+*.

Experimental Platform. All algorithms were implemented in C++. All measurements were obtained on a Windows machine with Intel 2.2GHz Dual Core and 2GB memory.

Data Sets. Generated by the *de-facto* standard in literature for stress testing skyline algorithms [1] contain three extreme attribute correlations: *independent*, *correlated*, or *anti-correlated*. We vary skyline dimensions d [2–6]. Since data sets generated by [1] have no associated time-stamps we add these time stamps. The validity period of each object is normally distributed with 100K as the mean (objects expire after 100K new objects have been processed) and a standard deviation of αK objects (denoted as 100K+/-αK). The cardinality of the data stream is 1 million data objects.

Experimental Analysis of TI-Sky. We study the robustness of *TI-Sky* by varying: (1) the number of partitions on each dimension, and (2) number of dimensions.

• *Number of Skyline Dimensions (d).* Figure 5.a varies d= 2 to 6 for all three distributions. As d increases more partitions are created to maintain and query the data.
• *Number of Partitions (k).* The number of partitions on each dimension affects how many objects will map to each partition. As k decreases more objects map to a partition cell increasing the partition-level memory and CPU usages. An increase in k will reduce local partition-level processing but increase abstract-level processing as the number of Macro-TDominance relations increases. Figures 5.b shows avg. per object processing time of *TI-Sky* when the number of partitions is varied from 10 to 25 across three distributions. In Figure 5.b the cross-over point for all distributions is $k = 20$.

Comparison of Alternative Techniques. In Figure 6, for correlated data *TI-Sky* is 1 and $\approx$ 2 fold faster than *Stabbing-the-Sky+* and *LookOut*. For independent data *TI-Sky* is 29% and 60% faster than *Stabbing-the-Sky+* and *LookOut* respectively. Lastly, for anti-correlated data *TI-Sky* has a performance benefit of being on an average of 38% and

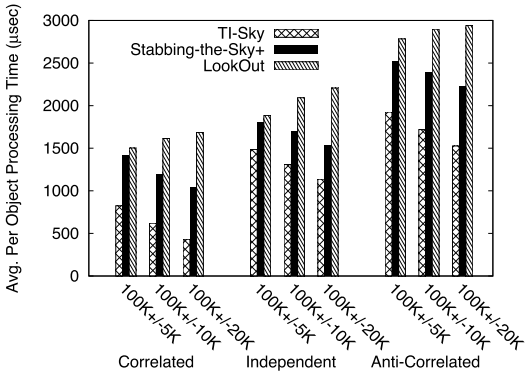


Fig. 6. Performance Comparison Against State-of-the-Art Algorithms ($d = 4$)

69% faster than *Stabbing-the-Sky+* [4] and *LookOut* [5] respectively. As the standard deviation increases the average per-object processing cost for both *TI-Sky* and *Stabbing-the-Sky+* decreases. This is because each object has a higher likelihood of remaining valid longer and can be effective in purging more dominated objects. However *LookOut* maintains all objects without purging and therefore requiring more time and space. These performances are consistent with those for other dimensions d[2–6].

7 Conclusion

Existing techniques that support skyline queries over streams either focus on the simpler model of continuous sliding window, or require to re-compute the skyline by performing multiple index scans for insertion and expiration of objects to handle time-interval streams. We present *TI-Sky* an efficient framework to evaluate skylines over time-interval streams. We propose a novel technique called *Macro-TDominance* to model the output dependencies between abstractions of the skyline result space. By analyzing the Macro-TDominance relationships, *TI-Sky* efficiently maintains the output space as well as deliver real-time results for user requests. Our experimental study confirms that *TI-Sky* has a superior performance compared to existing techniques.

Acknowledgment

This work was supported by National Science Foundation (IIS-0633930, ISS-0917017 and CRI-0551584) and the Korea Research Foundation (KRF-2008-357-D00214).

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
2. Raghavan, V., Rundensteiner, E.A.: Progressive result generation for multi-criteria decision support queries. In: ICDE, 733–744 (2010)

3. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. TKDE 18(2), 377–391 (2006)
4. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: ICDE, pp. 502–513 (2005)
5. Morse, M., Patel, J., Grosky, W.: Efficient continuous skyline computation. Inf. Sci., 3411–3437 (2007)
6. Park, N., Raghavan, V., Rundensteiner, E.: Supporting multi-criteria decision support queries over time-interval data streams. Technical Report WPI-CS-TR-10-12, Dept. of Computer Science, Worcester Polytechnic Institute (2010)