Discovering General Prominent Streaks in Sequence Data

GENSHENG ZHANG, The University of Texas at Arlington
XIAO JIANG, Shanghai Jiao Tong University
PING LUO, HP Labs China
MIN WANG, Google Research
CHENGKAI LI, The University of Texas at Arlington

This paper studies the problem of prominent streak discovery in sequence data. Given a sequence of values, a prominent streak is a long consecutive subsequence consisting of only large (small) values, e.g., consecutive games of outstanding performance in sports, consecutive hours of heavy network traffic, consecutive days of frequent mentioning of a person in social media, and so on. Prominent streak discovery provides insightful data patterns for data analysis in many real-world applications and is an enabling technique for computational journalism. Given its real-world usefulness and complexity, the research on prominent streaks in sequence data opens a spectrum of challenging problems.

A baseline approach to finding prominent streaks is a quadratic algorithm that exhaustively enumerates all possible streaks and performs pairwise streak dominance comparison. For more efficient methods, we make the observation that prominent streaks are in fact skyline points in two dimensions– streak interval length and minimum value in the interval. Our solution thus hinges upon the idea to separate the two steps in prominent streak discovery– candidate streak generation and skyline operation over candidate streaks. For candidate generation, we propose the concept of local prominent streak (LPS). We prove that prominent streaks are a subset of LPSs and the number of LPSs is less than the length of a data sequence, in comparison with the quadratic number of candidates produced by the brute-force baseline method. We develop efficient algorithms based on the concept of LPS. The non-linear LPS-based method (NLPS) considers a superset of LPSs as candidates, and the linear LPS-based method (LLPS) further guarantees to consider only LPSs. The proposed properties and algorithms are also extended for discovering general top-k, multi-sequence, and multi-dimensional prominent streaks. The results of experiments using multiple real datasets verified the effectiveness of the proposed methods and showed orders of magnitude performance improvement against the baseline method.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*data mining*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: computational journalism, sequence database, time-series database, skyline query

ACM Reference Format:
Zhang, G., Jiang, X., Luo, P., Wang, M., Li, C. 2013. Discovering General Prominent Streaks in Sequence

---

This material is based upon work partially supported by NSF Grants IIS-1018865, CCF-1117369 and HP Labs Innovation Research Award. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.
Author's addresses: G. Zhang, Department of Computer Science and Engineering, The University of Texas at Arlington; X. Jiang, Department of Computer Science, Shanghai Jiao Tong University; P. Luo, HP Labs China; M. Wang, the bulk of the research was done at HP Labs China, and she is currently affiliated with Google Research; C. Li, Department of Computer Science and Engineering, The University of Texas at Arlington.

DOI:http://dx.doi.org/10.1145/0000000.0000000

Data. *ACM Trans. Knowl. Discov. Data.* V, N, Article A (January YYYY), 37 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

This paper is on the problem of *prominent streak discovery* in sequence data. A piece of sequence data is a series of values or events. This includes time-series data, in which the data values or events are often measured at equal time intervals. Sequence and time-series data is produced and accumulated in a rich variety of applications. Examples include stock quotes, sports statistics, temperature measurement, Web usage logs, network traffic logs, Web clickstream, customer transaction sequence, social media statistics. Given a sequence of values, a *prominent streak* is a long consecutive subsequence consisting of only large (small) values. Examples of such prominent streaks include consecutive days of high temperature, consecutive trading days of large stock price oscillation, consecutive games of outstanding performance in professional sports, consecutive hours of high volume of TCP traffic, consecutive weeks of high flu activity, consecutive days of frequent mentioning of a person in social media, and so on.

It is insightful to investigate prominent streaks since they intuitively and succinctly capture extraordinary subsequences of data. Consider several example application scenarios: (1) Business analysts may be interested in prominent streaks in social media usage logs, e.g. streaks of re-tweeting a tweet, streaks of hashtagging a topic, and so on. (2) A security auditing may be performed after a streak of excessive login attempts is detected. (3) A cooling system can be started when a streak of days with high temperature has been discovered. (4) For disease outbreak detection, we can identify prominent streaks in time series of aggregated disease case counts. Previous works on outbreak detection focus on conventional data mining tasks such as clustering and regression [Wong 2004]. The concept of prominent streaks has not been studied before.

Prominent streak discovery can be particularly useful in helping journalists to identify newsworthy stories when data sequences evolve, investigators to find suspicious phenomena, and news anchors and sports commentators to bring out attention-seizing factual statements. Therefore it will be a key enabling technique for *computational journalism* [Cohen et al. 2011]. In fact, we witness the mentioning of prominent steaks in many real-world news articles:

- *"This month the Chinese capital has experienced 10 days with a maximum temperature in around 35 degrees Celsius – the most for the month of July in a decade."* (http://www.chinadaily.com.cn/china/2010-07/27/content_11055675.htm)
- *"The Nikkei 225 closed below 10000 for the 12th consecutive week, the longest such streak since June 2009."* (http://www.bloomberg.com/news/2010-08-06/japanese-stocks-fall-for-second-day-this-week-on-u-s-jobless-claims-yen.html)
- *"He (LeBron James) scored 35 or more points in nine consecutive games and joined Michael Jordan and Kobe Bryant as the only players since 1970 to accomplish the feat."* (http://www.nba.com/cavaliers/news/lbj_mvp_candidate_060419.html)
- *"Only player in NBA history to average at least 20 points, 10 rebounds and 5 assists per game for 6 consecutive seasons. (Kevin Garnett)"* (http://en.wikipedia.org/wiki/Kevin_Garnett)

The examples indicate that general prominent streaks can have a variety of constraints. A streak can be on multiple dimensions (e.g., $\langle$point, rebound, assist$\rangle$), its significance can be with regard to a certain period (e.g., "since June 2009") or a certain comparison group (e.g., "the month of July"), and we may be interested in not only the most prominent streaks but also the top-k most prominent ones (e.g., "LeBron James

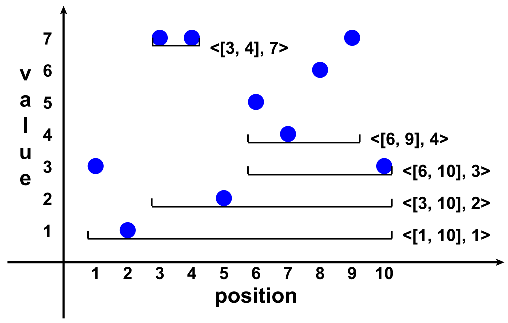


Fig. 1. A Data Sequence and its Prominent Streaks.

joined Michael Jordan and Kobe Bryant as the only players", which means LeBron James's scoring streak mentioned above is among the top-3 streaks.)

Given its real-world usefulness and variety, the research on prominent streaks in sequence data opens a spectrum of challenging problems. In an earlier work [Jiang et al. 2011], we proposed the concept of prominent streak and studied the problem of discovering the simplest kind of prominent streaks, i.e., those without the aforementioned constraints. In this paper, we extend the work to discovering general multi-dimensional and top-*k* prominent streaks from multiple sequences, which shall substantially broaden the applicability of our study in real-world scenarios, as evidenced by the above stories in news articles.

1.1. Problem Definition

Definition 1 (Streak and Prominent Streak)**.** Given an n-element sequence $P = (p_1, \cdots, p_n)$, a *streak* is an interval-value pair $\langle [l, r], v \rangle$, where $1 \leq l \leq r \leq n$ and $v = \min_{l \leq i \leq r} p_i$.

Consider two streaks $s_1 = \langle [l_1, r_1], v_1 \rangle$ and $s_2 = \langle [l_2, r_2], v_2 \rangle$. We say s_1 *dominates* s_2, denoted by $s_1 \succ s_2$ or $s_2 \prec s_1$, if $r_1 - l_1 \geq r_2 - l_2$ and $v_1 > v_2$, or $r_1 - l_1 > r_2 - l_2$ and $v_1 \geq v_2$. For example, $\langle [1, 2], 3 \rangle \prec \langle [4, 7], 6 \rangle$ and $\langle [1, 2], 3 \rangle \prec \langle [3, 4], 5 \rangle$, while $\langle [1, 2], 3 \rangle$ and $\langle [7, 8], 3 \rangle$ do not dominate each other.

With regard to $P = (p_1, ...p_n)$, the set of all possible streaks is denoted by $\mathcal{S}_P$. A streak $s \in \mathcal{S}_P$ is a *prominent streak* if it is not dominated by any streak in $\mathcal{S}_P$, i.e., $\nexists s'$ s.t. $s' \in \mathcal{S}_P$ and $s' \succ s$. The set of all prominent streaks in P is denoted by $\mathcal{PS}_P$.

Problem Statement: The *prominent streak discovery problem* is to, given a sequence P, produce $\mathcal{PS}_P$.

Figure 1 is our running example which shows the assists made by an NBA player in 10 consecutive games $P = (3, 1, 7, 7, 2, 5, 4, 6, 7, 3)$. There are 5 prominent streaks in P– $\langle [1, 10], 1 \rangle$, $\langle [3, 10], 2 \rangle$, $\langle [6, 10], 3 \rangle$, $\langle [6, 9], 4 \rangle$, $\langle [3, 4], 7 \rangle$. Each streak is represented by a horizontal segment, which crosses the minimal-value points in the streak and runs from the left end to the right end of the corresponding interval. For instance, $\langle [6, 9], 4 \rangle$ is a prominent streak of minimal value 4, whose interval is from p_6 to p_9. It captures the fact that the NBA player made at least 4 assists in 4 consecutive games (game 6 – game 9). $\langle [1, 10], 1 \rangle$, the whole data sequence, is also a trivial prominent streak because no other streak can possibly dominate the sequence itself. The streak $\langle [8, 9], 6 \rangle$ is an instance of non-prominent streaks because it is dominated by $\langle [3, 4], 7 \rangle$.

Definition 1 focuses on the simplest type of prominent streaks. The concept of prominent streak can be extended in several ways. First, we may be interested in top-*k* prominent streaks which are dominated by less than k other streaks. Second, we may need to compare streaks from not only the same sequence but also multiple different sequences (e.g., sequences corresponding to different NBA players, cities, stocks, etc.) Third, the data points in a sequence can be multi-dimensional, leading to the pursuit of multi-dimensional prominent streaks. We have seen examples of all such general prominent streaks at the beginning of Section 1 and their combinations naturally exist. The focus of our following discussion will first be on the simplest prominent streak discovery problem. In Section 5, we discuss how to discover general prominent streaks.

Definition 1 and the problem statement focus on finding streaks of large values. To find streaks of small values (e.g., a stock index below 10000 for 12 consecutive weeks, described in the aforementioned second news article), two changes should be made. First, a streak should be captured by its interval length and the maximal value (instead of the minimal value) in the interval, i.e., $v = \max_{l \leq i \leq r} p_i$. Second, the dominance relation between streaks should be defined to prefer smaller values. More specifically, s_1 *dominates* s_2 if $r_1 - l_1 \geq r_2 - l_2$ and $v_1 < v_2$ (instead of $v_1 > v_2$), or $r_1 - l_1 > r_2 - l_2$ and $v_1 \leq v_2$ (instead of $v_1 \geq v_2$). Given that the new definition would be exactly symmetric to Definition 1, finding streaks of large and small values become the same problem. Hence, we only consider finding streaks of large values in the rest of this paper.

1.2. Overview of the Solution

A brute-force method for discovering prominent streaks is not appealing. One can enumerate all possible streaks and decide if each streak is prominent by comparing it with every other streak. Given a sequence P with length n, there are $|\mathcal{S}_P| = \binom{n+1}{2}$ streaks in total. Thus the number of pair-wise streak comparison would be $\binom{|\mathcal{S}_P|}{2} = \frac{n^4+2n^3-n^2-2n}{8}$. Given a sequence of length 10000, the brute-force approach enumerates 10^8 streaks and performs 10^{16} comparisons. Many real-world sequences can be quite long. The sequence of daily closing prices for a stock with 40-year history has about 10000 values. A one-year usage log for a Web site has 8760 values at hourly interval.

Prominent streaks are in fact skyline points [Börzsönyi et al. 2001] in two dimensions– streak interval length ($r - l$) and minimum value in the interval (v). A streak is a prominent streak (skyline point) if it is not dominated by any point, i.e., there exists no streak that has both longer interval and greater minimum value.

Based on this observation, our solution hinges upon the idea to separate the two steps of prominent streak discovery– *candidate streak generation* and *skyline operation* over candidate streaks. In candidate generation, we prune a large portion of non-prominent streaks without exhaustively considering all possible streaks. For skyline operation, we apply efficient algorithms from the rich literature on this topic, e.g., [Börzsönyi et al. 2001; Tan et al. 2001; Kossmann et al. 2002; Papadias et al. 2005]. The effectiveness of pruning in the first step is critical to overall performance, because execution time of skyline algorithms increases superlinearly by the number of candidate points [Börzsönyi et al. 2001].

Candidate streak generation

We considered three methods with increasing pruning power in candidate generation– a baseline method, a non-linear LPS (local prominent streak)-based method, and a linear LPS-based method. The baseline method exhaustively enumerates $\mathcal{S}_P$, all possible streaks in a sequence P, by a nested-loop over the values in P. Thus, the baseline method does not have pruning power. The sketch of this method is in Algorithm 1. It produces quadratic ($\frac{n(n+1)}{2}$) candidate streaks. We then propose the concept of *local prominent streak* (LPS) for substantially reducing the number of candidate streaks (Section 3). The intuition is, given a prominent streak s, there can-

Algorithm 1: Baseline Method

Input: Data sequence P=$(p_1, ..., p_n)$
Output: Prominent streaks $skyline$

$skyline \leftarrow empty$
for $r = 1$ *to* n **do**
  $min\_value \leftarrow \infty$
  for $l = r$ ***downto*** 1 **do**
    $min\_value \leftarrow \min(p_l, min\_value)$
    $s \leftarrow \langle [l, r], min\_value \rangle$ // candidate streak
    $skyline \leftarrow skyline\_update(skyline, s)$

Algorithm 2: Update Dynamic Skyline ($skyline\_update$)

Input: Dynamic skyline $skyline$, new candidate streak $s = \langle [l, r], v \rangle$
Output: Updated dynamic skyline $skyline$

Find the largest i in $skyline$ s.t. $v_i \leq v$
if $s \prec s_i$ **or** $s \prec s_{i+1}$ **then**
  return $skyline$
while $s \succ s_i$ **and** $i > 0$ **do**
  Delete s_i from $skyline$
  $i \leftarrow i - 1$
Insert s into $skyline$
return $skyline$

not be a super-sequence of s with greater or equal minimal value. In other words, s must be locally prominent as well. Hence we only need to consider LPSs as candidates. The algorithm sequentially scans the data sequence and maintains possible LPSs. The non-linear LPS-based method finds a superset of LPSs as candidates, while the linear LPS-based method guarantees to find only LPSs.

Skyline operation

To couple candidate streak generation with skyline operation, Algorithm 1 maintains a dynamic skyline and updates it whenever a new candidate streak is produced. The updating procedure $skyline\_update$ is in Algorithm 2.

Our focus is not to compare various skyline algorithms. Many existing algorithms can be adopted. What matters is the number of candidate streaks produced by the candidate generation step. This is also verified by our experiments which show that, under various skyline algorithms, the candidate streak generation methods in Section 3 perform and compare consistently.

We can use a sorting-based method for finding the skyline points in a two-dimensional space [Börzsönyi et al. 2001]. If the candidate streak generation step does not prune streaks effectively, we cannot hold all candidate streaks in memory. The memory overflow can be addressed by external-memory sorting.

Another approach is to progressively update a dynamic skyline with candidate streaks, based on the nested-loop method in [Börzsönyi et al. 2001]. The outline of this approach is shown in Algorithm 2. We use $skyline$ to denote the dynamic skyline. When a new candidate streak s is generated, s is inserted into $skyline$ if it is not dominated by any point in $skyline$. The algorithm also checks if some points in $skyline$ are dominated by s and eliminates them from $skyline$.

The dominance relationship can be efficiently checked, given that the streaks have only two dimensions– interval length ($r - l$) and minimum value (v). The key idea is that the lengths of streaks monotonically decrease as their minimal values increase (except that there can be identical points, i.e., streaks with equal lengths and equal minimal values.) Hence the streaks in $skyline$ are ordered by v (or by $r-l$). Suppose the candidate streak is $s = \langle [l', r'], v' \rangle$. We find in $skyline$ a pivoting streak $s_i = \langle [l_i, r_i], v_i \rangle$ such that i is the largest index with $v_i \leq v'$, i.e., $v_i \leq v' < v_{i+1}$. The following Property 1 says that s must be dominated by s_i or s_{i+1} if it is dominated by any point in $skyline$ and Property 2 says that s can only dominate s_i and its immediate neighbors with smaller v values. (For concise presentation, in these properties, we omit the discussion of boundary cases, i.e., $i = 0$ or $i = |skyline|$.) For quickly finding s_i and its neighbors, we use a balanced binary search tree (BST) on v to store $skyline$. (Thus we call it the BST-based skyline method.)

Property 1. A candidate streak $s = \langle [l', r'], v' \rangle$ is dominated by some points in $skyline$ if and only if s is dominated by s_i or s_{i+1}, in which $s_i = \langle [l_i, r_i], v_i \rangle$ and i is the largest index such that $v_i \leq v'$, i.e., $v_i \leq v' < v_{i+1}$.

Proof. We first prove that, if there exists $j < i$ such that $s_j = \langle [l_j, r_j], v_j \rangle \succ s$, then $s_i \succ s$. Since i is the largest index such that $v_i \leq v'$, we have $v_j \leq v_i \leq v'$. Given that $s_j \succ s$, we know $v_j = v_i = v'$ and $r_j - l_j > r' - l'$. From $v_j = v_i$, we know that $r_j - l_j = r_i - l_i$, otherwise they cannot both exist in $skyline$. Therefore $s_i \succ s$.

We then prove that, if there exists $j > i+1$ such that $s_j = \langle [l_j, r_j], v_j \rangle \succ s$, then $s_{i+1} \succ s$. Since the points in $skyline$ are ordered by v, $v_{i+1} \leq v_j$ and $r_{i+1} - l_{i+1} \geq r_j - l_j$. We already know that $v' < v_{i+1}$ and $r_j - l_j \geq r' - l'$ (since $s_j \succ s$). Therefore $s_{i+1} \succ s$. ■

Property 2. If $s = \langle [l', r'], v' \rangle$ dominates totally k streaks in $skyline$, then the k streaks are s_i, s_{i-1}, ..., s_{i-k+1}.

Proof. Since the points in $skyline$ are ordered by v, we know that $v_i \leq v_j$ and $r_i - l_i \geq r_j - l_j$ if $i < j$. s cannot dominate any s_j such that $j > i$. The reason is that $v' < v_{i+1} \leq v_j$. If s dominates s_i, then $v' \geq v_i$ and $r' - l' \geq r_i - l_i$. Since v_i decreases by i and $r_i - l_i$ increases by i, the k streaks dominated by s must be consecutively ordered. ■

In comparison with the sorting-based method, the above BST-based skyline method saves both memory space and execution time. It avoids memory overflow because the number of streaks in the dynamic skyline in most cases remains small enough to fit in memory. Hence no streak needs to be read from/written to secondary memory. The small size of dynamic skyline in real data is verified by our experiments in Section 6. After all, prominent streaks (and skyline points in general) are supposed to be minority, otherwise they cannot stand out to warrant further investigation. Furthermore, even if the dynamic skyline grows large, a method such as the block nested-loop based method in [Börzsönyi et al. 2001] can be applied to fall back on secondary memory. The small size of dynamic skyline also means small number of streak comparisons. Intuitively, given c candidate streaks, a fast comparison-based sorting algorithm (say quicksort) requires $O(c \log c)$ comparisons, while the BST-based method only requires $O(c \log s)$ comparisons, where s is the maximal size of the dynamic skyline during computation. Experiments in Section 6 show that s is typically much smaller than c.

Monitoring Prominent Streaks

A desirable property of a prominent streak discovery algorithm is the capability of monitoring new data entries as the sequence grows continuously and always keeping the prominent streaks up-to-date. The aforementioned algorithms naturally fit into such monitoring scenario, with only minor modification. The details are given in Section 4.

1.3. Summary of Contributions and Outline

To summarize, our work makes the following contributions:

- We define the problem of prominent streak discovery. The simple concept is useful in many real-world applications. To the best of our knowledge, there has not been study along this line except our prior work [Jiang et al. 2011].
- We propose the solution framework to separate *candidate streak generation* and *skyline operation* during prominent streak discovery. Under this framework, we designed efficient algorithms for candidate streak generation, based on the concept of local prominent streak. Both the non-linear LPS-based method (NLPS) and the linear LPS-based method (LLPS) produce substantially less candidate streaks than the quadratic number of candidates produced by a baseline method. LLPS further guarantees a linear number of candidate streaks.
- We extend the solution framework to discovering general prominent streaks. While the extensions to top-k and multi-sequence prominent streaks are simple, the extension to multi-dimensional prominent streak is non-trivial. These extensions significantly broaden the real-world application scenarios of the work.
- We conduct experiments over multiple real datasets. The results verified the effectiveness of our methods and showed orders of magnitude performance improvement over the baseline method. We also showed some insightful prominent streaks discovered from real data, to highlight the practicality of this work.

The rest of the paper is organized as follows. In Section 2 we review related work. Section 3 presents the NLPS and LLPS methods for candidate streak generation. Section 4 discusses how to adapt the algorithms to monitor prominent streaks when data sequence continuously grows. Section 5 extends the concept of prominent streak and the algorithms for finding general prominent streaks. Experiment setup and results are reported in Section 6. Section 7 concludes the paper.

2. RELATED WORK

Data mining on sequence and time-series data has been an active area of research, where many techniques are developed for similarity search and subsequence matching in sequence and time-series databases [Agrawal et al. 1993; Faloutsos et al. 1993; Agrawal et al. 1995; Yi et al. 1998], finding sequential patterns [Agrawal and Srikant 1995; Srikant and Agrawal 1996; Zaki 2001; Pei et al. 2004; Yan et al. 2003], classification and clustering of sequence and time-series data [Smyth et al. 1997; Oates et al. 1999; Liao 2005; Shin and Fussell 2007], biological sequence analysis [Altschul et al. 1990; Rabiner 1989], etc. However, we are not aware of prior work on the prominent streak discovery problem proposed in this paper.

The skyline of a set of tuples is the subset of tuples that are not dominated by any tuple. A tuple dominates another tuple if it is equally good or better on every attribute and better on at least one attribute. The notion of skyline is useful in several applications, including multi-criteria decision making. Skyline query has been intensively studied over the last decade. Kung et al. [Kung et al. 1975] first proposed in-memory algorithms to tackle the skyline problem, which they called the “maximal vector problem”. Börzsönyi et al. [Börzsönyi et al. 2001] considered the problem in database context and integrated skyline operator into database system. They also invented a block-nested-loop algorithm (BNL) and extended the divide-and-conquer algorithm (DC) from [Kung et al. 1975]. Chomicki et al. presented the Sort-Filter-Skyline algorithm (SFS) [Chomicki et al. 2003], which improves upon BNL by pre-sorting tuples with a function compatible with the skyline criteria. We apply skyline algorithms over candidate streaks but our methods are orthogonal to specific choices of skyline algorithms.

A dataset may have too many skyline tuples, especially when the dimensionality of the data is high. Various approaches have been proposed to alleviate this problem. For example, Pei et al. [Pei et al. 2006] and Tao et al. [Tao et al. 2006] proposed to perform skyline analysis in subspaces instead of the original full space. Several methods were designed to find the representatives among a large number of skyline points [Zhang et al. 2005; Chan et al. 2006; Lin et al. 2007; Tao et al. 2009].

Progressive skyline algorithms optimize the efficiency in returning initial skyline points while producing more results progressively. Various algorithms developed along this line include the bitmap-based algorithm and the index-based algorithm [Tan et al. 2001], the nearest neighbor search algorithm [Kossmann et al. 2002], and the branch-and-bound skyline algorithm (BBS) [Papadias et al. 2005]. Other variants of skyline queries have also been studied, including skyline cube which aims to answer skyline queries over any combination of dimensions [Pei et al. 2006; Xia and Zhang 2006].

Jiang et al. [Jiang and Pei 2009] studied the problem of interval skyline queries on time-series. Given a set of time series and a time interval, they find the time series that are not dominated by others in the interval. A time series dominates another one if its value at every position is at least equal to the corresponding value in the other time series and it is at least larger at one position. The point-by-point equi-length interval comparison is clearly different from our problem.

The plateau of a time series is the time interval in which the vlaues are close to each other (within a given threshold) and are no smaller than the values outside the interval [Wang and Wang 2006]. The plateau problem is not concerned about comparing different intervals.

3. DISCOVERING PROMINENT STREAKS FROM LOCAL PROMINENT STREAKS

For an n-element sequence P, the baseline method (Algorithm 1) produces $\frac{n(n+1)}{2}$ candidate streaks. In this section, based on the concept of *local prominent streak* (LPS) we propose the non-linear LPS-based (NLPS) and linear LPS-based (LLPS) methods. Both drastically reduce the number of candidate streaks in practice. LLPS further guarantees only a linear number of candidate streaks.

3.1. Local Prominent Streak (LPS)

Definition 2 (Local Prominent Streak)**.** Given a sequence of data values $P = (p_1, \cdots, p_n)$, we say a streak $s = \langle [l, r], v\rangle \in \mathcal{S}_P$ is a *local prominent streak* (LPS) or *locally prominent* if there does not exist any other streak $s' = \langle [l', r'], v'\rangle \in \mathcal{S}_P$ such that $[l', r'] \supset [l, r]$ and $s' \succ s$. (I.e., there does not exist such s' that $[l', r'] \supset [l, r]$ and $v' \geq v$.) The symbol $\supset$ denotes the subsumption check between two intervals, i.e., $[l', r'] \supset [l, r]$ if and only if $l' \leq l \wedge r' > r$ or $l' < l \wedge r' \geq r$. We denote the set of local prominent streaks in sequence P as $\mathcal{LPS}_P$.

Figure 2 shows all the local prominent streaks found in our running example. All other streaks are not locally prominent. For example, $\langle [6, 8], 4\rangle$ is not locally prominent since it is dominated by $\langle [6, 9], 4\rangle$ and $[6, 9] \supset [6, 8]$. In the following we give several important properties of local prominent streaks.

Property 3. Every prominent streak is also a local prominent streak, i.e., $\mathcal{PS}_P \subseteq \mathcal{LPS}_P$.

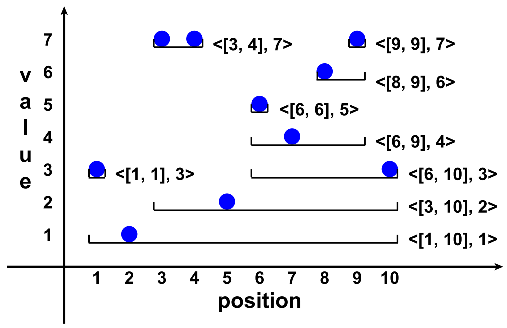


Fig. 2. Local Prominent Streaks.

Proof. Suppose there is a prominent streak that is not locally prominent, i.e., $\exists s \in \mathcal{PS}_P$ s.t. $s \notin \mathcal{LPS}_P$. By Definition 2, there exists some streak s' such that $[l', r'] \supset [l, r]$ and $s' \succ s$. That is contradictory to Definition 1 which says s is not dominated by any other streak. Therefore a streak cannot be prominent if it is not even locally prominent. ∎

The above Property 3 is illustrated by Figure 2, as all the prominent streaks in Figure 1 also appear in Figure 2. However, the reverse of Property 3 does not hold–local prominent streaks are not necessarily prominent streaks. For example, $\langle [8, 9], 6\rangle$ is an LPS but is dominated by $\langle [3, 4], 7\rangle$ and therefore is not in Figure 1.

Lemma 1. Suppose $s = \langle [l, r], v\rangle$ and $s' = \langle [l', r'], v'\rangle$ are two different local prominent streaks in P, i.e., $s, s' \in \mathcal{LPS}_P$, $l \neq l'$ or $r \neq r'$. For any $k \in argmin_{i\in[l,r]} p_i$ and $k' \in argmin_{i\in[l',r']} p_i$, we have $k \neq k'$. I.e., $argmin_{i\in[l,r]} p_i \cap argmin_{i\in[l',r']} p_i = \emptyset$.

Proof. If $[l, r] \cap [l', r'] = \emptyset$, i.e., the two intervals do not overlap, it is obvious that $k \neq k'$. Now consider the case when $[l, r] \cap [l', r'] \neq \emptyset$, i.e., $l \leq l' \leq r$ or $l' \leq l \leq r'$. By definition of $argmin$, $p_k = v = min_{i\in[l,r]} p_i$ and $p_{k'} = v' = min_{i\in[l',r']} p_i$. Suppose there exist such k and k' that $k = k'$. Thus $v = v' = p_k$. By Definition 1, we have $p_i \geq v$ for every $i \in [l, r]$ and every $i \in [l', r']$. Since the two intervals $[l, r]$ and $[l', r']$ overlap, their combined interval corresponds to a new streak $s'' = \langle [l, r] \cup [l', r'], v\rangle$. [1] It is clear $s'' \succ s$ and $s'' \succ s'$. That is a contradiction to the precondition that both s and s' are LPSs. Thus, this lemma holds. ∎

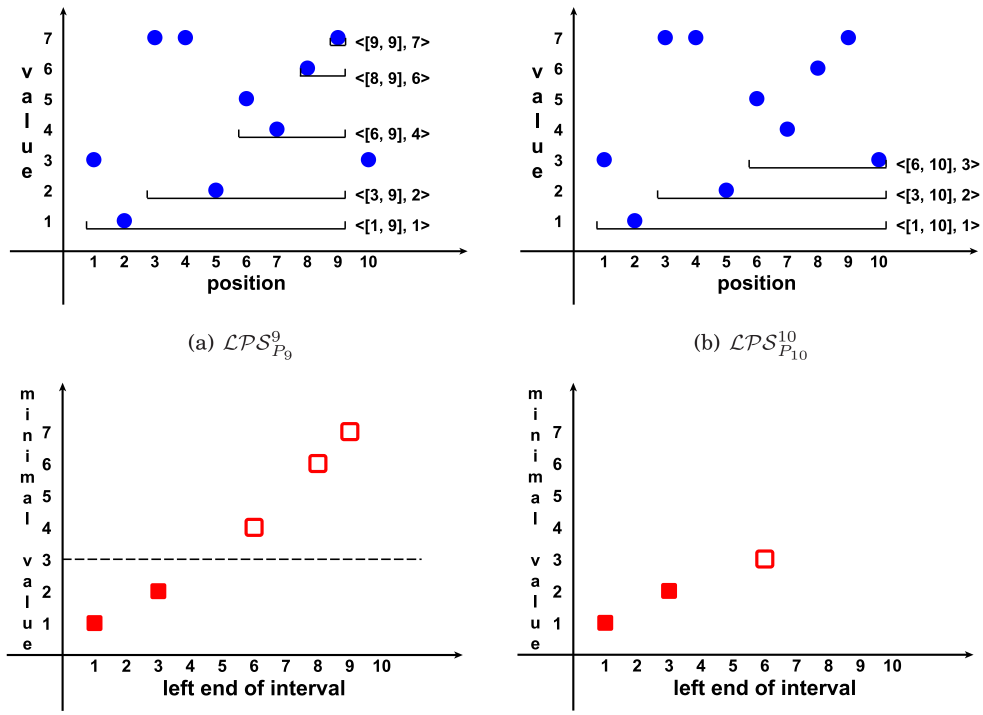
Lemma 1 indicates that two different LPSs cannot reach their minimal values at the same position. Therefore each value position in sequence P can correspond to the minimal value of at most one LPS. What immediately follows is that there are at most n LPSs in an n-element sequence. Formally we have the following property.

Property 4. $|\mathcal{LPS}_P| \leq |P|$.

From Property 3 we know that $\mathcal{LPS}_P$ is a sufficient candidate set for $\mathcal{PS}_P$, i.e., we can guarantee to find all prominent streaks if we only consider local prominent streaks. Property 4 further shows how small $\mathcal{LPS}_P$ is and thus how good it is as a candidate set. Specifically, the size of $\mathcal{LPS}_P$ is at most $|P|$, the length of the sequence, in contrast to the all $\frac{|P|(|P|+1)}{2}$ possible streaks considered by the baseline method (Algorithm 1). Thus, $\mathcal{LPS}_P$ helps to prune most streaks from further consideration. In the following sections we present efficient algorithms for computing a superset of $\mathcal{LPS}_P$ and $\mathcal{LPS}_P$ itself exactly.

[1]The two intervals can overlap in four different ways. Thus $[l, r] \cup [l', r'] = [l, r]$ or $[l, r']$ or $[l', r]$ or $[l', r']$.

3.2. $\mathcal{LPS}_P^k$ and $\mathcal{LPS}_{P_k}^k$



(a) $\mathcal{LPS}_{P_9}^9$ (b) $\mathcal{LPS}_{P_{10}}^{10}$

(c) l-v plot of $\mathcal{LPS}_{P_9}^9$ (d) l-v plot of $\mathcal{LPS}_{P_{10}}^{10}$

Fig. 3. From $\mathcal{LPS}_{P_9}^9$ to $\mathcal{LPS}_{P_{10}}^{10}$.

To facilitate our discussion, we first define a new notation, $\mathcal{LPS}_P^k$.

Definition 3. $\mathcal{LPS}_P^k$ is the set of local prominent streaks in P that end at position k, i.e., $\mathcal{LPS}_P^k = \{s | s \in \mathcal{LPS}_P \text{ and } s = \langle [l, k], v \rangle\}$.

There are two key components in the definition of $\mathcal{LPS}_P^k$. The first is the upper script k, which fixes the right end of every interval in the set. It is clear that $\mathcal{LPS}_P^1$, $\mathcal{LPS}_P^2$, $\ldots$, $\mathcal{LPS}_P^{|P|}$ is a natural partition of $\mathcal{LPS}_P$. We use this partition scheme in the design of our algorithms. Specifically, we show how each $\mathcal{LPS}_P^k$ in this partition is calculated in a sequential and progressive way.

The second key component in the definition of $\mathcal{LPS}_P^k$ is the lower script P, which provides the scope for local prominent streaks. By generalizing this component we define $\mathcal{LPS}_{P_k}^k$. We denote the sequence with the first k entries of P as P_k. Then $\mathcal{LPS}_{P_k}$ is the set of local prominent streaks with regard to sequence P_k (instead of P) and $\mathcal{LPS}_{P_k}^k$ are those LPSs in $\mathcal{LPS}_{P_k}$ that end at k. Due to the change of scope, $\mathcal{LPS}_{P_k}^k$ is a superset of $\mathcal{LPS}_P^k$. Formally, we have the following property.

Property 5. $\mathcal{LPS}_P^k \subseteq \mathcal{LPS}_{P_k}^k$.

Algorithm 3: Non-linear LPS Method (NLPS)

Input: Data sequence $P = (p_1, \cdots, p_n)$
Output: Prominent streaks $skyline$

$skyline \leftarrow empty$
for $k = 1$ **to** n **do**
  Compute $\mathcal{LPS}^k_{P_k}$ by Algorithm 4
  for each streak s in $\mathcal{LPS}^k_{P_k}$ **do**
    $skyline \leftarrow skyline\_update(skyline, s)$

Proof. Consider any streak $s \in \mathcal{LPS}^k_P$. By Definition 3, $s = \langle [l, k], v\rangle$ and $s \in \mathcal{LPS}_P$. Therefore by Definition 2, there does not exist any $s' = \langle [l', r'], v'\rangle$ in P such that $s' \succ s$ and $[l', r'] \supset [l, k]$. Since P_k is a prefix of P, i.e., the first k values in P, it follows that there does not exist any such s' in P_k either. Therefore $s \in \mathcal{LPS}^k_{P_k}$. ■

Consider the running example again. Figure 3(a) shows $\mathcal{LPS}^9_{P_9}$, including $\langle[1, 9], 1\rangle, \langle[3, 9], 2\rangle, \langle[6, 9], 4\rangle, \langle[8, 9], 6\rangle, \langle[9, 9], 7\rangle$. As shown in Figure 2, $\mathcal{LPS}^9_P$ contains $\langle[6, 9], 4\rangle, \langle[8, 9], 6\rangle, \langle[9, 9], 7\rangle$. Streaks $\langle[1, 9], 1\rangle$ and $\langle[3, 9], 2\rangle$ do not belong to $\mathcal{LPS}_P$, thus do not belong to $\mathcal{LPS}^9_P$, since they are locally dominated by $\langle[1, 10], 1\rangle$ and $\langle[3, 10], 2\rangle$, respectively. By contrast, $\langle[1, 9], 1\rangle$ and $\langle[3, 9], 2\rangle$ are part of $\mathcal{LPS}^9_{P_9}$ because they are not locally dominated by any streak of P_9, which only contains the first 9 values of P.

3.3. Non-linear LPS Method

By Property 5 and the fact that $\mathcal{LPS}^1_P, \cdots, \mathcal{LPS}^{|P|}_P$ is a partition of $\mathcal{LPS}_P$, we have

$$\mathcal{LPS}_P = \bigcup_{1\le k\le |P|} \mathcal{LPS}^k_P \subseteq \bigcup_{1\le k\le |P|} \mathcal{LPS}^k_{P_k} \tag{1}$$

Thus, we can use $\bigcup_{1\le k\le |P|} \mathcal{LPS}^k_{P_k}$ as our candidate set for prominent streaks. Although its size can be greater than that of $\mathcal{LPS}_P$, in practice it does substantially reduce the size of candidate streaks, verified by the experimental results in Section 6.

Along this line, Algorithm 3 presents the method to compute candidate streaks. Since the number of candidates may be super-linear to the length of data sequence, we call it the *non-linear LPS method* (NLPS). The algorithm iterates k from 1 to $|P|$, progressively computes $\mathcal{LPS}^k_{P_k}$ from $\mathcal{LPS}^{k-1}_{P_{k-1}}$ when the k-th element p_k is visited, and includes them into candidate streaks. The details of updating from $\mathcal{LPS}^{k-1}_{P_{k-1}}$ to $\mathcal{LPS}^k_{P_k}$ are in Algorithm 4, which is based on the following Lemma 2. For convenience of discussion, we first define the right-end extension of a streak and a streak set.

Definition 4. If $s = \langle [l, r], v\rangle$ is a streak in an n-element data sequence P and $r < n$, the right-end extension of s is streak $\langle [l, r + 1], v'\rangle$, where $v' = \min\{v, p_{r+1}\}$. The extension of a streak set S is the set which consists of extensions of all the streaks in S.

Lemma 2. If $s_1 = \langle [l, k], v_1\rangle \in \mathcal{LPS}^k_{P_k}$ and $l \neq k$, then the streak $s_2 = \langle [l, k - 1], v_2\rangle \in \mathcal{LPS}^{k-1}_{P_{k-1}}$.

Proof. First, note that $v_2 = \min_{l_1\le i\le k-1} p_i$ and $v_1 = \min\{v_2, p_k\}$. We prove by contradiction. Suppose $s_2 = \langle [l_1, k - 1], v_2\rangle \notin \mathcal{LPS}^{k-1}_{P_{k-1}}$. By Definition 3, $s_2 \notin \mathcal{LPS}_{P_{k-1}}$. Further by Definition 2, there exists $s_3 = \langle [l_3, r_3], v_3\rangle \in \mathcal{S}_{P_{k-1}}$ such that $[l_3, r_3] \supset [l_1, k - 1]$ and $s_3 \succ s_2$. Given any $s = \langle [l, r], v\rangle \in \mathcal{S}_{P_{k-1}}$, we have $r \le k - 1$. Therefore $r_3 =$

Algorithm 4: Progressive Computation of $\mathcal{LPS}^{k}_{P_k}$

Input: $\mathcal{LPS}^{k-1}_{P_{k-1}}$ and p_k
Output: $\mathcal{LPS}^{k}_{P_k}$
// When it starts, stack lps consists of streaks in $\mathcal{LPS}^{k-1}_{P_{k-1}}$.
$pivot \leftarrow null$
while ! $lps.isempty()$ **do**
  if $lps.top().v < p_k$ **then**
    break
  else
    $pivot \leftarrow lps.pop()$
if $pivot == null$ **then**
  $lps.push(\langle [k,k], p_k \rangle)$
else
  $pivot.v \leftarrow p_k$
  $lps.push(pivot)$
// Now, lps contains all the streaks in $\mathcal{LPS}^{k}_{P_k}$.

$k-1$, $l_3 < l_1$ and $v_3 \geq v_2$. The right-end extension of s_3 is $s_4 = \langle [l_3, k], v_4 \rangle$, where $v_4 = \min\{v_3, p_k\} \geq \min\{v_2, p_k\} = v_1$. Therefore $s_4 \succ s_1$, which contradicts with the precondition that $s_1 \in \mathcal{LPS}^{k}_{P_k}$. The property holds. ■

Lemma 2 indicates that, except $\langle [k,k], p_k \rangle$, for each streak in $\mathcal{LPS}^{k}_{P_k}$, its prefix streak is in $\mathcal{LPS}^{k-1}_{P_{k-1}}$. Hence, to produce $\mathcal{LPS}^{k}_{P_k}$, we only need to consider the right-end extension of $\mathcal{LPS}^{k-1}_{P_{k-1}}$. Beyond that, we only need to consider one extra streak $\langle [k,k], p_k \rangle$ since it may belong to $\mathcal{LPS}^{k}_{P_k}$ as well.

In order to articulate how to derive $\mathcal{LPS}^{k}_{P_k}$ from $\mathcal{LPS}^{k-1}_{P_{k-1}}$, we partition $\mathcal{LPS}^{k-1}_{P_{k-1}}$ into two disjoint sets, namely,

$$\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<} = \{s | s = \langle [l, k-1], v \rangle \in \mathcal{LPS}^{k-1}_{P_{k-1}}, v < p_k\}, \tag{2}$$

$$\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\geq} = \{s | s = \langle [l, k-1], v \rangle \in \mathcal{LPS}^{k-1}_{P_{k-1}}, v \geq p_k\}. \tag{3}$$

It is clear that $\mathcal{LPS}^{k-1}_{P_{k-1}}$ is the disjoint union of these two sets, i.e., $\mathcal{LPS}^{k-1}_{P_{k-1}} = \mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<} \cup \mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\geq}$, and $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<} \cap \mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\geq} = \emptyset$. Use the running example again. For $\mathcal{LPS}^{9}_{P_9}$ in Figure 3(a), since $p_{10} = 3$, the two sets are $\mathcal{LPS}^{9}_{P_9}{}^{<} = \{\langle [1,9], 1 \rangle, \langle [3,9], 2 \rangle\}$, $\mathcal{LPS}^{9}_{P_9}{}^{\geq} = \{\langle [6,9], 4 \rangle, \langle [8,9], 6 \rangle, \langle [9,9], 7 \rangle\}$.

We consider how to extend streaks in $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<}$ and $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\geq}$, respectively. For simplicity of presentation, we omit the formal proofs when we make various statements below.

- $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<}$: We use $S1$ to denote the right-end extension of $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<}$. Since every streak in $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{<}$ has a minimal value less than p_k, the corresponding extended new streak has the same minimal value. Hence all the new streaks belong to $\mathcal{LPS}^{k}_{P_k}$. For the running example, corresponding to $\mathcal{LPS}^{9}_{P_9}{}^{<}$, we have $S1 = \{\langle [1,10], 1 \rangle, \langle [3,10], 2 \rangle\}$.

- ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq}$: We use $S2$ to denote the right-end extension of ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq}$. Since every streak in ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq}$ has a minimal value greater than or equal to p_k, the minimal value of every streak in $S2$ equals p_k. Hence, the longest streak in $S2$, denoted as $S2^*$, dominates all other streaks in $S2$ and it is the only streak in $S2$ that belongs to $\mathcal{LPS}^{k}_{P_k}$. In other words, we only need to extend the longest streak in ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq}$ to form a new candidate streak. Furthermore, since every streak in $S2$ has the same r value (the right end of the interval), i.e., k, $S2^*$ is the streak with the minimal l value (the left end of the interval) in $S2$. Clearly there cannot be another streak in $S2$ with the same length. For the running example, corresponding to ${\mathcal{LPS}^{9}_{P_9}}^{\geq}$, we have $S2 = \{\langle[6,10],3\rangle, \langle[8,10],3\rangle, \langle[9,10],3\rangle\}$. The longest streak in $S2$ is $\langle[6,10],3\rangle$. It is clear that $\langle[6,10],3\rangle$ dominates other streaks in $S2$. Hence it belongs to $\mathcal{LPS}^{10}_{P_{10}}$.
- ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq} = \emptyset$: If ${\mathcal{LPS}^{k-1}_{P_{k-1}}}^{\geq}$ is empty, a new streak $\langle[k,k],p_k\rangle$ belongs to $\mathcal{LPS}^{k}_{P_k}$. (Otherwise, it is dominated by $S2^*$.)

The above discussion is captured by the following Property 6.

Property 6. $\mathcal{LPS}^{k}_{P_k} = S1 \cup \{S2^*\}$ if $S2 \neq \emptyset$ and $\mathcal{LPS}^{k}_{P_k} = S1 \cup \{\langle[k,k],p_k\rangle\}$ if $S2 = \emptyset$.

We use Figure 3 to explain the above procedure of producing $\mathcal{LPS}^{k}_{P_k}$ from $\mathcal{LPS}^{k-1}_{P_{k-1}}$. Figure 3(a) and 3(b) show $\mathcal{LPS}^{9}_{P_9}$ and $\mathcal{LPS}^{10}_{P_{10}}$, respectively. Figure 3(c) and 3(d) also show $\mathcal{LPS}^{9}_{P_9}$ and $\mathcal{LPS}^{10}_{P_{10}}$, by using a different presentation– l-v plot. All the streaks $\langle[l,r],v\rangle$ in $\mathcal{LPS}^{k-1}_{P_{k-1}}$ share the same value of r, which is $k-1$. Therefore we plot the streaks by l (x-axis) and v (y-axis). In Figure 3(c), the 5 points represent the 5 streaks in $\mathcal{LPS}^{9}_{P_9}$: $\langle[1,9],1\rangle$, $\langle[3,9],2\rangle$, $\langle[6,9],4\rangle$, $\langle[8,9],6\rangle$, $\langle[9,9],7\rangle$. The dotted line represents the 10-th data entry $p_{10}=3$. It bisects $\mathcal{LPS}^{9}_{P_9}$ into ${\mathcal{LPS}^{9}_{P_9}}^{\geq}$ (3 hollow points above the line) and ${\mathcal{LPS}^{9}_{P_9}}^{<}$ (2 filled points below the line). We produce new candidate streaks $\mathcal{LPS}^{10}_{P_{10}}$ by extending the right ends of streaks in $\mathcal{LPS}^{9}_{P_9}$ to 10. The streaks extended from ${\mathcal{LPS}^{9}_{P_9}}^{<}$ all belong to $\mathcal{LPS}^{10}_{P_{10}}$. They are the 2 filled points in Figure 3(d), corresponding to $\langle[1,10],1\rangle$ and $\langle[3,10],2\rangle$. Among the streaks extended from ${\mathcal{LPS}^{9}_{P_9}}^{\geq}$, only the one with the smallest l (the longest one) belongs to $\mathcal{LPS}^{10}_{P_{10}}$. It is the hollow point in Figure 3(d), corresponding to $\langle[6,10],3\rangle$. Hence $\mathcal{LPS}^{10}_{P_{10}} = \{\langle[1,10],1\rangle, \langle[3,10],2\rangle, \langle[6,10],3\rangle\}$.

The details of the algorithm are shown in Algorithm 4. We use a stack lps to maintain $\mathcal{LPS}^{k}_{P_k}$. Since the streaks $\langle[l,r],v\rangle$ in $\mathcal{LPS}^{k}_{P_k}$ have the same r value which equals k, we do not need to store r in lps. Hence each item in lps has two data attributes, v and l. The items in the stack are ordered by v (and l). More specifically, their v and l values both strictly monotonically increase, from the bottom of the stack to the top. The monotonicity on l is obvious since they are different streaks of the same r value. The monotonicity on v thus is also clear because their lengths monotonically decreases due to monotonically increasing l and they must not dominate each other. In fact, Figure 3(c) and 3(d) visualize all items in lps, before and after p_{10} is encountered, respectively. In each figure, the leftmost point denotes the bottom of the stack (with the smallest v), while the rightmost point denotes the top of the stack (with the largest v). After data entries p_1, ..., p_{k-1} are encountered, lps contains $\mathcal{LPS}^{k-1}_{P_{k-1}}$. Given data entry p_k, we popped from the stack all the streaks whose v values are greater than or equal to p_k. Among the popped streaks, the leftmost one (with the smallest l and v) is pushed back into the stack, with v value replaced by p_k and r extended from $k-1$ to

Algorithm 5: Linear LPS Method(LLPS)

Input: Data sequence $P = (p_1, \ldots p_n)$
Output: Prominent streaks $skyline$

$skyline \leftarrow empty$
for $k = 1$ **to** n **do**
  Compute $\mathcal{LPS}_P^{k-1}$ and $\mathcal{LPS}_{P_k}^k$ by Algorithm 6
  for each streak s in $\mathcal{LPS}_P^{k-1}$ **do**
    $skyline \leftarrow skyline\_update(skyline, s)$
$\mathcal{LPS}_P^n \leftarrow \mathcal{LPS}_{P_n}^n$
for each streak s in $\mathcal{LPS}_P^n$ **do**
  $skyline \leftarrow skyline\_update(skyline, s)$

Algorithm 6: Computing $\mathcal{LPS}_P^{k-1}$ and $\mathcal{LPS}_{P_k}^k$

Input: $\mathcal{LPS}_{P_{k-1}}^{k-1}$ and p_k
Output: $\mathcal{LPS}_P^{k-1}$ and $\mathcal{LPS}_{P_k}^k$

// Insert the following line before Line 1 in Algorithm 4.
$\mathcal{LPS}_P^{k-1} \leftarrow \emptyset$
// Insert the following two lines after Line 6 in Algorithm 4, in the same **else** branch as Line 6.
if $pivot.v > p_k$ **then**
  $\mathcal{LPS}_P^{k-1} \leftarrow \mathcal{LPS}_P^{k-1} \cup \{pivot\}$

k. (Again, the r value is not explicitly stored in the stack.) If no streak was popped, then $\langle [k, k], p_k \rangle$ is pushed into the stack. The remaining streaks in the original stack are kept, with their v and l values unchanged and r extended from $k - 1$ to k.

Algorithm 3 computes candidate streaks for an n-element sequence P. It invokes Algorithm 4 n times.[2] In each invocation, exactly 1 item is pushed into the stack. Therefore in total there are n insertions and thus at most n deletions. Hence, the amortized time complexity of Algorithm 4 is $O(1)$.

In each iteration of Algorithm 3, we compute $\mathcal{LPS}_{P_k}^k$ and include them into candidate streaks. Thus, for an n-element sequence, the total number of candidate streaks considered is $\sum_{k=1}^{n} |\mathcal{LPS}_{P_k}^k|$. In the worst case, we may have a strictly increasing sequence and the candidate streaks include all possible streaks. This is as bad as the exhaustive baseline method in Algorithm 1. For example, given sequence $(10, 20, 30)$, we have $\mathcal{LPS}_{P_1}^1 = \{\langle [1, 1], 10 \rangle\}$, $\mathcal{LPS}_{P_2}^2 = \{\langle [1, 2], 10 \rangle, \langle [2, 2], 20 \rangle\}$ and $\mathcal{LPS}_{P_3}^3 = \{\langle [1, 3], 10 \rangle, \langle [2, 3], 20 \rangle, \langle [3, 3], 30 \rangle\}$.

3.4. Linear LPS Method

Now we present the linear LPS (LLPS) method (Algorithm 5), which guarantees to produce a linear number of candidate streaks even in the worst case. Similar to Algorithm 3, this method iterates through the data sequence and computes $\mathcal{LPS}_{P_k}^k$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$ when the k-th data entry is encountered, for k from 1 to n. However, different

[2]With regard to the first data element p_1, $\langle [1, 1], p_1 \rangle$ is pushed into the stack. It is the only prominent streak and local prominent streak for P_1.

from Algorithm 3, it also computes $\mathcal{LPS}_P^{k-1}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$. Computation of both $\mathcal{LPS}_{P_k}^{k}$ and $\mathcal{LPS}_P^{k-1}$ is done in Algorithm 6, which is a simple extension of Algorithm 4. It is worth noting that, since $P_n = P$, $\mathcal{LPS}_P^n$ and $\mathcal{LPS}_{P_n}^n$ are identical.

To produce $\mathcal{LPS}_P^{k-1}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$ given the k-th entry p_k, Algorithm 6 is based on the following Property 7. Its intuition is as follows. Recall that the minimal value of any streak in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ (Equation (3)) is not smaller than p_k. It follows that if the minimal value of a streak in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ is greater than p_k, the streak cannot grow into a longer local prominent streak without changing the minimal value. Hence, the streak itself is a local prominent streak. To summarize, $\mathcal{LPS}_P^{k-1}$ is the same as $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$. The only exception is the longest streak in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$, i.e., the streak with the smallest l and thus the smallest minimal value v. If its minimal value is equal to p_k, then it does not belong $\mathcal{LPS}_P^{k-1}$, because it can be right-extended and included in $\mathcal{LPS}_P^{k'}$ for some $k' \geq k$.

Lemma 3. For an n-entry sequence P, a streak $s = \langle [l, r], v\rangle$ is a local prominent streak if and only if ($l = 1$ or $v > p_{l-1}$) and ($r = n$ or $v > p_{r+1}$).

Proof. We prove by contradiction. Consider $l > 1$. If $v \leq p_{l-1}$, then s is dominated by $\langle [l-1, r], v\rangle$, which contradicts with s being a local prominent streak. Consider $r < n$. Similarly if $v \leq p_{r+1}$, then s is dominated by $\langle [l, r+1], v\rangle$, which contradicts with s being locally prominent. ■

Property 7. Given an n-entry sequence P, for any position $1 < k \leq n$, $\mathcal{LPS}_P^{k-1} = \{s | s = \langle [l, k-1], v\rangle \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ and $v > p_k\}$.

Proof. Proof of the equality from left to right: Suppose streak $s = \langle [l, k-1], v\rangle \in \mathcal{LPS}_P^{k-1}$. By Property 5, $s \in \mathcal{LPS}_{P_{k-1}}^{k-1}$, and by Lemma 3 $v > p_k$. By the concept of $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ in Equation (3), $s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$.

Proof of the equality from right to left: Suppose streak $s = \langle [l, k-1], v\rangle$ satisfies $s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ and $v > p_k$. Then s is a local prominent streak in the scope of $\mathcal{LPS}_{P_{k-1}}^{k-1}$, which means, by Lemma 3, $l = 1$ or $v > p_{l-1}$. Since $v > p_k$, by Lemma 3 s is a local prominent streak in P. Therefore $s \in \mathcal{LPS}_P^{k-1}$. ■

Continue the running example. $\mathcal{LPS}_P^9 = \mathcal{LPS}_{P_9}^{9\ \geq} = \{\langle [6, 9], 4\rangle, \langle [8, 9], 6\rangle, \langle [9, 9], 7\rangle\}$. Note that $\mathcal{LPS}_{P_9}^{9\ \geq}$ and $\mathcal{LPS}_P^9$ are identical because the minimal values for the streaks in $\mathcal{LPS}_{P_9}^{9\ \geq}$ are all greater than p_{10}.

Similar to Algorithm 4, Algorithm 6 has an amortized time complexity of $O(1)$. With regard to candidate streaks, LLPS is different in that it only needs to consider the streaks in $\mathcal{LPS}_P^{k-1}$ as candidates. Consequently, LLPS reduces the total number of candidate streaks to $\sum_{k=1}^{n} |\mathcal{LPS}_P^k|$, i.e., $|\mathcal{LPS}_P|$ (Equation (1)). By Property 4, $|\mathcal{LPS}_P|$ is n at most, thus LLPS guarantees to produce only a linear number of candidate streaks even in worst case.

4. MONITORING PROMINENT STREAKS

One desirable property of a prominent streak discovery algorithm is the capability of monitoring new data entries as the sequence grows continuously and always keeping

Algorithm 7: Continuous Monitoring of Prominent Streaks

Input: The new data entry p_k

Compute $\mathcal{LPS}_P^{k-1}$ and $\mathcal{LPS}_{P_k}^{k}$ by Algorithms 6
if $last\_requested\_position < k-1$ **then**
  for each streak s in $\mathcal{LPS}_P^{k-1}$ **do**
    $skyline \leftarrow sklyine\_update(skyline, s)$
if $\mathcal{PS}_{P_k}$ is requested **then**
  for each streak s in $\mathcal{LPS}_{P_k}^{k}$ **do**
    $skyline \leftarrow sklyine\_update(skyline, s)$
  $last\_requested\_position \leftarrow k$
  // Now, $skyline$ contains all prominent streaks in $\mathcal{PS}_{P_k}$

the prominent streaks up-to-date. For example, a network administrator may check the prominent streaks in the network traffic of a Web server till any particular moment. Formally, given a continuously growing data sequence P (such as a data stream), the k-th data entry that has just come is denoted by p_k and the sequence so far is denoted by P_k. At this moment, if the user requests $\mathcal{PS}_{P_k}$, the prominent streaks of P_k, our method should efficiently discover them.

With regard to skyline operation, the BST-based method progressively updates the dynamic skyline with new candidate streaks, thus can be applied for monitoring prominent streaks without modification.

With regard to candidate streak generation, all three methods (baseline, NLPS, LLPS) use one-pass sequential scan of the data sequence, therefore they all naturally fit into the monitoring scenario. Specifically, the new data point p_k corresponds to the next iteration of the outer loop in Algorithm 1, 3, and 5. The baseline method exhaustively lists all streaks ending at p_k and updates the skyline with these streaks. The NLPS method updates $\mathcal{LPS}_{P_{k-1}}^{k-1}$ to $\mathcal{LPS}_{P_k}^{k}$, and updates the skyline with the streaks in $\mathcal{LPS}_{P_k}^{k}$.

The adaptation of LLPS is a bit more complex, as shown in Algorithm 7. This algorithm records the last position when the user requested the prominent streaks. When p_k arrives, $\mathcal{LPS}_P^{k-1}$ and $\mathcal{LPS}_{P_k}^{k}$ are dynamically computed by Algorithms 6. The skyline is updated with the candidate streaks in $\mathcal{LPS}_P^{k-1}$, only if $\mathcal{PS}_{P_{k-1}}$ was not requested by the user when p_{k-1} was visited. Note that if $\mathcal{PS}_{P_{k-1}}$ was requested, the skyline has already been updated with the streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}$. Since $\mathcal{LPS}_P^{k-1} \subseteq \mathcal{LPS}_{P_{k-1}}^{k-1}$, we do not need to update the skyline with $\mathcal{LPS}_P^{k-1}$ again. Finally, if the user requests $\mathcal{PS}_{P_k}$, the skyline has to be updated with $\mathcal{LPS}_{P_k}^{k}$ since all the local prominent streaks (with regard to P_k) ending at p_k must be considered. In Section 6 we will show the significant superiority of this adaptation of LLPS over other methods.

Note that this algorithm degrades to NLPS (Algorithm 3) if the user requests the prominent streaks at every data entry. On the other hand, if the prominent streaks are only requested at p_n, i.e., the last entry in the sequence, it becomes the same as LLPS (Algorithm 5).

5. DISCOVERING GENERAL PROMINENT STREAKS

In this section, we extend the concept of prominent streak and the algorithms introduced in previous sections to general cases. Specifically, we investigate how to discover top-k, multi-sequence, and multi-dimensional prominent streaks.

5.1. Top-k Prominent Streaks

Definition 5 (Top-k Prominent Streak)**.** With regard to a sequence $P=(p_1,...p_n)$ and its local prominent streaks $\mathcal{LPS}_P$, a streak $s \in \mathcal{LPS}_P$ is a *top-k prominent streak* if it is not dominated by k or more streaks in $\mathcal{LPS}_P$, i.e., $\big|\{s'|s' \in \mathcal{LPS}_P \text{ and } s' \succ s\}\big| < k$. The set of all top-$k$ prominent streaks in P is denoted by $\mathcal{KPS}_P$. Note that there can be more than k top-k prominent streaks.

Top-k prominent streaks are those local prominent streaks dominated by less than k other local prominent streaks, by Definition 5. This definition has two implications. First, a top-k prominent streak must be locally prominent. For instance, a streak does not qualify even if it is only dominated by 1 subsuming streak and $k > 1$. Second, a streak can qualify even if it is dominated by k or more other streaks, as long as less than k of those dominating streaks are local prominent streaks.

Consider a sequence $P = (20, 30, 25, 30, 5, 5, 15, 10, 15, 5)$, corresponding to the points made by a basketball player in all his games. The streak $\langle[3, 4], 25\rangle$, though only dominated by $\langle[2, 4], 25\rangle$, is a sub-streak of the latter, and hence is not a top-2 prominent streak. The intuitive explanation is that, $\langle[3, 4], 25\rangle$ is within the interval of $\langle[2, 4], 25\rangle$, therefore we do not consider it important. On the other hand, the streak $\langle[7, 9], 10\rangle$ is a top-2 prominent streak. Although it is dominated by 3 streaks $\langle[1, 4], 20\rangle$, $\langle[1, 3], 20\rangle$, and $\langle[2, 4], 25\rangle$, the dominating streaks are all from the same period and only 1 of the 3 is a local prominent streak.

The candidate streak generation methods discussed in previous sections are applicable in discovering top-k prominent streaks. We only need several small changes on skyline operation. For LLPS, since the candidates produced are guaranteed to be local prominent streaks only, we simply need to maintain a counter for each current skyline point in the dynamic skyline. The counter of a point records the number of its dominators in the skyline. When a candidate is compared against current skyline points, it is inserted into the skyline if it has less than k dominators. A current skyline point is removed if its counter reaches k. With regard to the baseline method and NLPS, they may produce candidates that are not local prominent streaks. A candidate must be pruned if another candidate streak dominates it and subsumes it. (Note that they both produce candidates with the same right-end of interval at the same time. Therefore a candidate cannot be locally dominated by existing points in the current skyline.)

5.2. Multi-sequence Prominent Streaks

Definition 6 (Multi-sequence Prominent Streak)**.** Given multiple sequences $\mathcal{P} = \{P^1, ..., P^m\}$ and their corresponding sets of streaks $\mathcal{S}_{P^1}$, ..., $\mathcal{S}_{P^m}$, a streak $s \in \mathcal{S}_{P^i}$ is a *multi-sequence prominent streak* in $\mathcal{P}$ if there does not exist a streak in any sequence that dominates s. More formally, $\nexists s', j$ s.t. $s' \in \mathcal{S}_{P^j}$, and $s' \succ s$. The set of all multi-sequence prominent streaks with regard to $\mathcal{P}$ is $\mathcal{PS}_{\mathcal{P}}$.

As an example, consider 3 sequences corresponding to the points made by 3 basketball players in all their games—$P_1 = (20, 30, 25, 30, 5, 5, 15, 10, 15, 5)$, $P_2 = (10, 5, 30, 35, 21, 25, 5, 15, 5, 25)$, and $P_3 = (5, 10, 15, 5, 25, 10, 20, 5, 15, 10)$. The streak $\langle[1, 4], 20\rangle$ of P_1 is a prominent streak within P_1 itself, but is dominated by $\langle[3, 6], 21\rangle$ in P_2. Hence it is not a multi-sequence prominent streak.

The extension from single-sequence algorithms (baseline, NLPS, LLPS) to multi-sequence algorithms is simple. We process individual sequences separately by the single-sequence algorithms and use a common dynamic skyline to maintain their prominent streaks. That is, when a local prominent streak within a sequence P_i is identified, it is compared with current streaks in the dynamic skyline, which contains prominent streaks from all sequences.

Algorithm 8: Update Dynamic Skyline for Multi-Dimensional Sequences ($skyline\_update$)

Input: Dynamic skyline $skyline$, new candidate streak $s = \langle [l, r], \vec{v} \rangle$
Output: Updated dynamic skyline $skyline$

```
dominating ← Find streaks in skyline that dominate s, by a range query on the
    KD-tree over skyline
if dominating ≠ ∅ then
    return skyline
dominated ← Find streaks in skyline that are dominated by s, by another range
    query on the KD-tree
Remove dominated from skyline
Insert s into skyline
return skyline
```

5.3. Multi-dimensional Prominent Streaks

Definition 7 (Multi-dimensional Prominent Streak)**.** In an n-entry d-dimensional sequence $P = (\vec{p}_1, \cdots, \vec{p}_n)$, a point $\vec{p}_i$ is a d-dimensional vector of data values. A streak s in P is an interval-vector pair $\langle [l, r], \vec{v} \rangle$, where

$$\vec{v} = (\min_{l \leq i \leq r} \vec{p}_i[1], \cdots, \min_{l \leq i \leq r} \vec{p}_i[d]), \tag{4}$$

$\vec{p}_i[j]$ is the j-th dimension of $\vec{p}_i$, and $1 \leq l \leq r \leq n$.

A d-dimensional vector $\vec{v} = (\vec{v}[1], \cdots, \vec{v}[d])$ dominates another vector $\vec{v'} = (\vec{v'}[1], \cdots, \vec{v'}[d])$, denoted by $\vec{v} \succ \vec{v'}$, if and only if $\vec{v}(1) \geq \vec{v'}[1], \cdots, \vec{v}[d] \geq \vec{v'}[d]$ and $\exists j$ such that $\vec{v}[j] > \vec{v'}[j]$. Moreover, we use $\vec{v} \succeq \vec{v'}$ to denote the case when $\vec{v}$ dominates or equals $\vec{v'}$.

A streak $s = \langle [l, r], \vec{v} \rangle$ dominates another streak $s' = \langle [l', r'], \vec{v'} \rangle$, denoted by $s \succ s'$, if and only if $r - l \geq r' - l'$ and $\vec{v} \succ \vec{v'}$, or $r - l > r' - l'$ and $\vec{v} \succeq \vec{v'}$.

The set of all possible streaks is denoted by $\mathcal{S}_P$. A streak $s \in \mathcal{S}_P$ is a *prominent streak* if it is not dominated by any streak in $\mathcal{S}_P$, i.e., $\nexists s'$ s.t. $s' \in \mathcal{S}_P$ and $s' \succ s$. The set of all multi-dimensional prominent streaks in P is denoted by $\mathcal{PS}_P$.

For a running example in this section, consider a two-dimensional sequence $P =$ ($(10, 10)$,$(40, 20)$,$(40, 30)$,$(30, 40)$,$(50, 30)$,$(20, 30)$). By the above definition, there are 8 prominent streaks in P– $\langle [1, 6], (10, 10) \rangle$, $\langle [2, 3], (40, 20) \rangle$, $\langle [2, 5], (30, 20) \rangle$, $\langle [2, 6], (20, 20) \rangle$, $\langle [3, 5], (30, 30) \rangle$, $\langle [3, 6], (20, 30) \rangle$, $\langle [4, 4], (30, 40) \rangle$, $\langle [5, 5], (50, 30) \rangle$. Other streaks are not prominent. For instance, $\langle [2, 4], (30, 20) \rangle$ is dominated by $\langle [3, 5], (30, 30) \rangle$.

In finding prominent streaks from a d-dimensional sequence, skyline operations perform dominance relationship test on $d + 1$ dimensions—d dimensions for data values and one special dimension for streak length. We maintain a KD-tree [Bentley 1975; Bentley 1979] on current skyline points. Given a candidate streak, we use a range query on the KD-tree to efficiently find its dominating points in the current skyline and another ranger query to find its dominated points in the current skyline. Specifically, Algorithm 2 is replaced by Algorithm 8 for multi-dimensional sequences. We do not further discuss how to answer range queries by multi-dimensional index structures such as KD-tree since it is well studied.

With regard to candidate streak generation, the brute-force baseline method does not require change, except that min_value and its calculation in Algorithm 1 are replaced according to the definition of vector $\vec{v}$ in Equation (4). Our focus in the rest of this section is to extend the concept of local prominent streak and its properties, in order to adapt NLPS and LLPS for multi-dimensional data sequence. Note that Property 3, Property 5 and Lemma 2 still hold, and can be proven in the same way as for single-dimensional sequence. We thus will use the result directly without tediously showing the proof. With the adaptation of NLPS and LLPS for multi-dimensional sequences, the continuous monitoring approach in Algorithm 7 works in the same way.

Definition 8. For a multi-dimensional sequence P, a streak $s = \langle[l,r],\vec{v}\rangle \in \mathcal{S}_P$ is a *local prominent streak* (LPS) if and only if there does not exist any other streak $s' = \langle[l',r'],\vec{v'}\rangle \in \mathcal{S}_P$, s.t. $[l',r'] \supset [l,r]$ and $s' \succ s$. (I.e., there does not exist such s' that $[l',r'] \supset [l,r]$ and $\vec{v'} \succeq \vec{v}$.) We use $\mathcal{LPS}_P$ to denote the set of all local prominent streaks in P.

For a multi-dimensional sequence, Property 3 still holds. Hence, every prominent streak in a multi-dimensional sequence P is also a local prominent streak, i.e., $\mathcal{PS}_P \subseteq \mathcal{LPS}_P$, and thus we can still find $\mathcal{LPS}_P$ and use it as the set of candidate streaks. Computing local prominent streaks in a multi-dimensional sequence is quite similar to that in a single-dimensional sequence. The concepts of $\mathcal{LPS}_P^k$ and $\mathcal{LPS}_{P_k}^k$ remain the same, except that the $\vec{v}$ in each streak $\langle[l,r],\vec{v}\rangle$ is a multi-dimensional vector instead of a single numeric value. Property 5 also holds. Therefore, the essential ideas of NLPS and LLPS algorithms remain unchanged. NLPS iterates k from 1 to $|P|$, progressively computes $\mathcal{LPS}_{P_k}^k$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$ when the k-th element $\vec{p}_k$ is visited, and includes $\mathcal{LPS}_{P_k}^k$ into candidate streaks. LLPS does not immediately include all of $\mathcal{LPS}_{P_k}^k$ into candidate streaks. Instead, it waits till seeing $\vec{p}_{k+1}$, then computes $\mathcal{LPS}_P^k$ (in addition to $\mathcal{LPS}_{P_{k+1}}^{k+1}$) from $\mathcal{LPS}_{P_k}^k$, and only includes $\mathcal{LPS}_P^k$ into candidate streaks. Hence, LLPS only considers local prominent streaks ($\mathcal{LPS}_P = \bigcup_{k=1}^n \mathcal{LPS}_P^k$) as candidates, while NLPS needs to consider more candidates ($\bigcup_{k=1}^n \mathcal{LPS}_{P_k}^k$), since $\mathcal{LPS}_P^k$ is subsumed by $\mathcal{LPS}_{P_k}^k$ according to Property 5.

5.3.1. Key Ideas.

Our following discussion focuses on how to compute $\mathcal{LPS}_{P_k}^k$ and $\mathcal{LPS}_P^{k-1}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$, when the k-th element $\vec{p}_k$ arrives. To facilitate the discussion, we partition $\mathcal{LPS}_{P_{k-1}}^{k-1}$ into two disjoint sets ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\preceq}$ and ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\npreceq}$, as shown below, which are similar to ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{<}$ and ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\geq}$ in Equations (2) and (3). ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\preceq}$ is the set of streaks, for which the value at any dimension of the vector $\vec{v}$ is not greater than the corresponding value in $\vec{p}_k$. ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\npreceq}$ is the set of streaks, for which $\vec{v}$ is greater than $\vec{p}_k$ on at least one dimension.

$$ {\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\preceq} = \{s | s = \langle[l,k-1],\vec{v}\rangle \in \mathcal{LPS}_{P_{k-1}}^{k-1}, \vec{v} \preceq \vec{p}_k\}, \quad (5) $$

$$ {\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\npreceq} = \{s | s = \langle[l,k-1],\vec{v}\rangle \in \mathcal{LPS}_{P_{k-1}}^{k-1}, \exists j \in [1,d] \ s.t. \ \vec{v}[j] > \vec{p}_k[j]\}. \quad (6) $$

For the running example, $\mathcal{LPS}_{P_5}^5$ is divided into ${\mathcal{LPS}_{P_5}^5}^{\preceq}$=$\{s_1$=$\langle[1,5],(10,10)\rangle\}$ and ${\mathcal{LPS}_{P_5}^5}^{\npreceq}$=$\{s_2$=$\langle[2,5],(30,20)\rangle$, s_3=$\langle[3,5],(30,30)\rangle$, s_4=$\langle[5,5],(50,30)\rangle\}$.

• *Compute $\mathcal{LPS}_P^{k-1}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$:*

We can prove that $\mathcal{LPS}_P^{k-1}$ is equivalent to ${\mathcal{LPS}_{P_{k-1}}^{k-1}}^{\npreceq}$, given by the following property.

Property 8. $\mathcal{LPS}_P^{k-1} = \mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$.

Proof. Since Property 5 still holds, $\mathcal{LPS}_P^{k-1} \subseteq \mathcal{LPS}_{P_{k-1}}^{k-1}$. Furthermore, $\mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$ disjointly partition $\mathcal{LPS}_{P_{k-1}}^{k-1}$, i.e., $\mathcal{LPS}_{P_{k-1}}^{k-1} = \mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq} \cup \mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq} \cap \mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq} = \emptyset$. Therefore we only need to prove that (1) none of the streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$ is in $\mathcal{LPS}_P^{k-1}$ and (2) all streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$ are in $\mathcal{LPS}_P^{k-1}$.

(1) $\forall s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$, $s \notin \mathcal{LPS}_P^{k-1}$. Suppose $s = \langle [l, k-1], \vec{v} \rangle$. Its right-end extension is $s' = \langle [l, k], \vec{v'} \rangle$, where $\vec{v'}[j] = min(\vec{v}[j], \vec{p}_k[j])$ for $j \in [1, d]$. Since $\vec{v} \preceq \vec{p}_k$ (by Equation (5)), it follows that $\vec{v'} = \vec{v}$ and thus $s' \succ s$. Hence, s cannot be a local prominent streak in P.

(2) $\forall s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$, $s \in \mathcal{LPS}_P^{k-1}$. We prove this by contradiction. Suppose $s = \langle [l, k-1], \vec{v} \rangle$. Assume $s \notin \mathcal{LPS}_P^{k-1}$, i.e., there exists $s' \succ s$ such that $s' = \langle [l', r'], \vec{v'} \rangle$, $[l', r'] \supset [l, k-1]$ and $\vec{v'} \succeq \vec{v}$. By Equation (6), $\exists j \in [1, d]$ such that $\vec{v}[j] > \vec{p}_k[j]$. Therefore $r' = k-1$, otherwise $r' = k$ and $\vec{v'}[j] <= \vec{p}_k[j] < \vec{v}[j]$, which contradicts with $\vec{v'} \succeq \vec{v}$. From $[l', r'] \supset [l, k-1]$ and $r' = k-1$, we get $l' < l$ which, along with $s' \succ s$, contradicts with $s \in \mathcal{LPS}_{P_{k-1}}^{k-1}$. The contradictions prove that $s \in \mathcal{LPS}_P^{k-1}$. ■

• *Compute $\mathcal{LPS}_{P_k}^{k}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$:*

We note that Lemma 2 still holds under multi-dimensional sequence, i.e., except $\langle [k, k], \vec{p}_k \rangle$, for each streak in $\mathcal{LPS}_{P_k}^{k}$, its prefix streak is in $\mathcal{LPS}_{P_{k-1}}^{k-1}$. Hence, to produce $\mathcal{LPS}_{P_k}^{k}$, we only need to consider the right-end extension of $\mathcal{LPS}_{P_{k-1}}^{k-1}$ and one extra streak $\langle [k, k], \vec{p}_k \rangle$ which may belong to $\mathcal{LPS}_{P_k}^{k}$ as well. Again, we consider the two disjoint partitions of $\mathcal{LPS}_{P_{k-1}}^{k-1}$, $\mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$, respectively.

(1) The right-end extensions of all streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$ belong to $\mathcal{LPS}_{P_k}^{k}$, by the property below.

Property 9. $\forall s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$, its right-end extension $s' \in \mathcal{LPS}_{P_k}^{k}$.

Proof. We prove by contradiction. Suppose $s = \langle [l, k-1], \vec{v} \rangle$. Its right-end extension is $s' = \langle [l, k], \vec{v'} \rangle$, where $\vec{v'}[j] = min(\vec{v}[j], \vec{p}_k[j])$ for $j \in [1, d]$. Since $s \in \mathcal{LPS}_{P_{k-1}}^{k-1\ \preceq}$, $\vec{v} \preceq \vec{p}_k$. Therefore $\vec{v'} = \vec{v}$. If $s' \notin \mathcal{LPS}_{P_k}^{k}$, then there exists $s'' = \langle [l'', k], \vec{v''} \rangle$ such that $s'' \succ s'$, i.e., $l'' < l$, and $\vec{v''} \succeq \vec{v'}$. Since s'' and s' have the same right end of interval and $l'' < l$, $\vec{v''} \preceq \vec{v'}$. Therefore $\vec{v''} = \vec{v'} = \vec{v}$. Consider $s''' = \langle [l'', k-1], \vec{v'''} \rangle$, i.e., s'' is the right-end extension of s'''. $\vec{v'''} \succeq \vec{v''}$ by definition of right-end extension. Therefore $\vec{v'''} \succeq \vec{v}$ and thus $s''' \succ s$ (since $l'' < l$). This contradicts with $s \in \mathcal{LPS}_{P_{k-1}}^{k-1}$. ■

(2) Given a streak in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \not\preceq}$, its right-end extension does not always belong to $\mathcal{LPS}_{P_k}^{k}$.

For a single-dimensional sequence, $\mathcal{LPS}_{P_{k-1}}^{k-1}$ was similarly partitioned into $\mathcal{LPS}_{P_{k-1}}^{k-1\ <}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$. Among the streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$, the right-end extension of the longest streak belongs to $\mathcal{LPS}_{P_k}^{k}$. If $\mathcal{LPS}_{P_{k-1}}^{k-1\ \geq}$ is empty, then $\langle [k, k], p_k \rangle$ belongs to $\mathcal{LPS}_{P_k}^{k}$.

For a multi-dimensional sequence, multiple but not necessarily all streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$ can be right-extended to streaks in $\mathcal{LPS}_{P_k}^{k}$. This can be simply proven by using the running example. Recall that $\mathcal{LPS}_{P_5}^{5}{}^{\preceq} = \{s_1 = \langle[1,5],(10,10)\rangle\}$ and $\mathcal{LPS}_{P_5}^{5}{}^{\npreceq} = \{s_2 = \langle[2,5],(30,20)\rangle,\ s_3 = \langle[3,5],(30,30)\rangle,\ s_4 = \langle[5,5],(50,30)\rangle\}$. Since $\vec{p_6} = (20,30)$, the right-end extensions of s_2, s_3 and s_4 are $s_2' = \langle[2,6],(20,20)\rangle$, $s_3' = \langle[3,6],(20,30)\rangle$ and $s_4' = \langle[5,6],(20,30)\rangle$, respectively. It is clear $s_2', s_3' \in \mathcal{LPS}_{P_6}^{6}$ and $s_4' \notin \mathcal{LPS}_{P_6}^{6}$ since $s_3' \succ s_4'$.

5.3.2. Efficient Computation.

Based on the discussion in Section 5.3.1, in computing $\mathcal{LPS}_{P_k}^{k}$ and $\mathcal{LPS}_{P}^{k-1}$ from $\mathcal{LPS}_{P_{k-1}}^{k-1}$, the key is to partition $\mathcal{LPS}_{P_{k-1}}^{k-1}$ into $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq}$ (which equals $\mathcal{LPS}_{P}^{k-1}$) and $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$. The right-end extensions of all streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq}$ belong to $\mathcal{LPS}_{P_k}^{k}$, and all remaining streaks in $\mathcal{LPS}_{P_k}^{k}$ are formed by right-end extensions of streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$. Below we discuss an efficient method of partitioning $\mathcal{LPS}_{P_{k-1}}^{k-1}$ and identifying streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$ that should be extended to streaks in $\mathcal{LPS}_{P_k}^{k}$.

• *Partition $\mathcal{LPS}_{P_{k-1}}^{k-1}$ into $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$:*

Suppose there are m streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}$, which are $s_1 = \langle[l_1, k-1], \vec{v}_1\rangle$, ..., $s_m = \langle[l_m, k-1], \vec{v}_m\rangle$, where $l_1 < \cdots < l_m$. We can prove that there exists t such that $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq} = \{s_1, \ldots, s_t\}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq} = \{s_{t+1}, \ldots, s_m\}$. (Two special cases are $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq} = \emptyset$ (i.e., $t = 0$) and $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq} = \emptyset$ (i.e., $t = m$).) The proof is sketched as follows. Since $l_1 < \cdots < l_m$, for any dimension j, the value $\vec{v}_i[j]$ monotonically increases by i (not necessarily strictly increasing), i.e., $\vec{v}_1[j] \le \vec{v}_2[j] \le \cdots \le \vec{v}_m[j]$. It follows that $\vec{v}_1 \prec \vec{v}_2 \prec \cdots \prec \vec{v}_m$. (Note that $\vec{v}_i \ne \vec{v}_{i+1}$ for any i, otherwise s_i would dominate s_{i+1}, which contradicts with that they all are local prominent streaks in P_{k-1}.) Given $s_{i_1} \in \mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq}$ and $s_{i_2} \in \mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$, it must be that $i_1 < i_2$, otherwise $i_1 > i_2$, $\vec{v}_{i_1} \succ \vec{v}_{i_2}$ and thus $\forall j, \vec{v}_{i_1}[j] \ge \vec{v}_{i_2}[j]$, which contradicts with Equations (5) and (6).

• *Identify streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$ that should be extended to streaks in $\mathcal{LPS}_{P_k}^{k}$:*

To find all those right-end extensions of streaks in $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$ that belong to $\mathcal{LPS}_{P_k}^{k}$, consider the aforementioned partitioning of $\mathcal{LPS}_{P_{k-1}}^{k-1}$ into $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\preceq} = \{s_1, \ldots, s_t\}$ and $\mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq} = \{s_{t+1}, \ldots, s_m\}$, where the m streaks $s_m, \ldots, s_1$ are decreasingly ordered by the left ends of their intervals. For each $s_i = \langle[l_i, k-1], \vec{v}_i\rangle \in \mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$, its right-end extension is $s_i' = \langle[l_i, k], \vec{v'}_i\rangle$. The following important property tells us that if $s_i' \not\prec s_{i-1}'$, then s_i' belongs to $\mathcal{LPS}_{P_k}^{k}$.

Property 10. For each streak $s_i = \langle[l_i, k-1], \vec{v}_i\rangle \in \mathcal{LPS}_{P_{k-1}}^{k-1}{}^{\npreceq}$, its right-end extension is $s_i' = \langle[l_i, k], \vec{v'}_i\rangle$. $s_i' \in \mathcal{LPS}_{P_k}^{k}$ if and only if $s_i' \not\prec s_{i-1}'$.

Proof. It is apparent that if $s_i' \prec s_{i-1}'$ then $s_i' \notin \mathcal{LPS}_{P_k}^{k}$. Thus our focus is to prove $s_i' \in \mathcal{LPS}_{P_k}^{k}$ if $s_i' \not\prec s_{i-1}'$, by contradiction. Assume $s_i' \not\prec s_{i-1}'$ but $s_i' \notin \mathcal{LPS}_{P_k}^{k}$. Hence,

$\exists j < i-1$ and $s'_j \succ s'_i$ (and thus $\vec{v'_j} \succeq \vec{v'_i}$). Since $l_j < l_i$, $\vec{v'}_j \preceq \ldots \preceq \vec{v'}_{i-1} \preceq \vec{v'}_i$. Therefore $\vec{v'}_j = \ldots = \vec{v'}_{i-1} = \vec{v'}_i$. Hence $s'_{i-1} \succ s'_i$ which contradicts with $s'_i \not\prec s'_{i-1}$. ■

Based on the properties discussed in Section 5.3.1 and 5.3.2 so far, we design an efficient method to compute $\mathcal{LPS}^k_{P_k}$ and $\mathcal{LPS}^{k-1}_P$ from $\mathcal{LPS}^{k-1}_{P_{k-1}}$. The current skyline points (prominent streaks) after the $(k-1)$-th element is encountered are stored in the aforementioned KD-tree index structure. The streaks in $\mathcal{LPS}^{k-1}_{P_{k-1}}$, $s_m, \ldots, s_1$, are stored in memory by the decreasing order of the left ends of their intervals. Since they have the same right ends of intervals, only the left ends and the corresponding vectors are stored. When the k-th element $\vec{p}_k$ arrives, this method considers the streaks s_i and their right-end extensions s'_i, starting from $i = m+1$ and iteratively decreasing i by 1. (For $i = m+1$, the special streak in consideration is $s'_{m+1} = \langle [k,k], \vec{p}_k \rangle$.) According to Property 10, the method only requires comparing s'_i with its predecessor s'_{i-1}. If $s'_i \prec s'_{i-1}$, then s_i is removed from the memory. Otherwise s'_i belongs to $\mathcal{LPS}^k_{P_k}$ and thus s_i is updated to s'_i in memory. More specifically, the vector $\vec{v}_i$ of s_i needs to be updated to $\vec{v'}_i$, by $\vec{v'}_i[j] = min(\vec{v}_i[j], \vec{p}_k[j])$ for $j \in [1,d]$. The method goes on until $i = t$ such that $\vec{v}_t \preceq \vec{p}_k$. At that moment, the method will take the following actions.

- The streaks scanned so far $(s_m, \ldots, s_{t+1})$ form $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\not\preceq}$ which is equivalent to $\mathcal{LPS}^{k-1}_P$. All remaining streaks in $\mathcal{LPS}^{k-1}_{P_{k-1}}$ $(s_t, \ldots, s_1)$ form $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\preceq}$.
- The streaks in $\mathcal{LPS}^{k-1}_P$ are candidate prominent streaks. They are compared with current skyline points by the aforementioned range queries over the KD-tree on the skyline points. Non-dominated candidates are inserted into the KD-tree.
- For all remaining streaks in the memory (i.e., $\mathcal{LPS}^{k-1}_{P_{k-1}}{}^{\preceq}$), their right-end extensions belong to $\mathcal{LPS}^k_{P_k}$. Since their vectors are all dominated by or equivalent to $\vec{p}_k$, their corresponding vectors do not need to be updated. At this moment, all streaks of $\mathcal{LPS}^k_{P_k}$ are stored in memory by the decreasing order of the left-ends of their intervals.

More concretely, Algorithms 3 and 5 remain unchanged, and Algorithms 4 and 6 are replaced by Algorithms 9 and 10, respectively.

5.3.3. A Note on "Curse of Dimensionality".
For a single-dimensional sequence with n elements, LLPS produces at most n candidates (i.e., local prominent streaks), according to Property 4. This upper-bound guarantees LLPS to be an efficient linear-time algorithm. However, the same property does not hold for multi-dimensional sequences. Consider an extreme case which is a 2-dimensional n-element sequence $(\vec{p}_1, \ldots, \vec{p}_n)$, where $\vec{p}_i = (i, n-i)$. It is not hard to prove that all $\frac{n(n+1)}{2}$ possible streaks in this sequence are prominent streaks and thus automatically local prominent streaks. This represents the worst case, in which nothing beats the brute-force baseline method.

While the worst case indicates the rather notorious "curse of dimensionality", our empirical results on multiple datasets are much more encouraging. The results show that the number of prominent streaks and the execution time of LLPS do not increase exponentially by the dimensionality of data. This is mainly due to that data values fluctuate and are correlated. We investigate these results in more details in Section 6.

Algorithm 9: Progressive Computation of $\mathcal{LPS}^{k}_{P_k}$ on Multi-Dimensional Sequences

Input: $\mathcal{LPS}^{k-1}_{P_{k-1}}$ and $\vec{p}_k$
Output: $\mathcal{LPS}^{k}_{P_k}$
// When it starts, stack lps consists of streaks in $\mathcal{LPS}^{k-1}_{P_{k-1}}$.
1 $temp\_stack \leftarrow$ an empty stack
2 **while** ! $lps.isempty()$ **do**
3   **if** $lps.top().\vec{v} \preceq \vec{p}_k$ **then**
4   **break**
5   **else**
6         $s = \langle [l_s, k-1], \vec{v}_s \rangle \leftarrow lps.pop()$
7         $s' \leftarrow \langle [l_s, k], \vec{v'}_s \rangle$, where $\vec{v'}_s = (min(\vec{v}_s[1], \vec{p}_k[1]), \ldots, min(\vec{v}_s[d], \vec{p}_k[d]))$ //right-end extension of s
8   **if** $lps.isempty()$ **then**
9             $temp\_stack.push(s')$
10   **else**
11             $q = \langle [l_q, k-1], \vec{v}_q \rangle \leftarrow lps.top()$
12             $q' \leftarrow \langle [l_q, k], \vec{v'}_q \rangle$, where $\vec{v'}_q = (min(\vec{v}_q[1], \vec{p}_k[1]), \ldots, min(\vec{v}_q[d], \vec{p}_k[d]))$ //right-end extension of q
13   **if** $q' \not\succ s'$ **then**
14                 $temp\_stack.push(s')$
15 **while** ! $temp\_stack.isempty()$ **do**
16     $lps.push(temp\_stack.pop())$
17 **if** $lps.isempty()$ *or* $lps.top().\vec{v} \not\succeq \vec{p}_k$ **then**
18     $lps.push(\langle [k, k], \vec{p}_k \rangle)$
// Now, lps contains all the streaks in $\mathcal{LPS}^{k}_{P_k}$.

Algorithm 10: Computing $\mathcal{LPS}^{k-1}_{P}$ and $\mathcal{LPS}^{k}_{P_k}$ on Multi-Dimensional Sequences

Input: $\mathcal{LPS}^{k-1}_{P_{k-1}}$ and $\vec{p}_k$
Output: $\mathcal{LPS}^{k-1}_{P}$ and $\mathcal{LPS}^{k}_{P_k}$
// Insert the following line before Line 1 in Algorithm 9.
$\mathcal{LPS}^{k-1}_{P} \leftarrow \emptyset$
// Insert the following line after Line 6 in Algorithm 9.
$\mathcal{LPS}^{k-1}_{P} \leftarrow \mathcal{LPS}^{k-1}_{P} \cup \{s\}$

6. EXPERIMENTS

We report and analyze experimental results in this section. The algorithms were implemented in Java. The experiments were conducted on a server with four 2.00GHz Intel Xeon E5335 CPUs running Ubuntu Linux. The limit on the heap size of Java Virtual Machine (JVM) was set at 512MB. We discuss the results on basic and general prominent streak discovery in Section 6.1 and Section 6.2, respectively.

Table I. Data Sequences Used in Experiments on Basic Prominent Streak Discovery.

name	length	# prominent streaks	description
Gold	1074	137	Daily morning gold price in US dollars, 01/1985-03/1989.
River	1400	93	Mean daily flow of Saugeen River near Port Elgin, 01/1988-12/1991.
Melb1	3650	55	The daily minimum temperature of Melbourne, Australia, 1981-1990.
Melb2	3650	58	The daily maximum temperature of Melbourne, Australia, 1981-1990.
Wiki1	4896	58	Hourly traffic to http://en.wikipedia.org/wiki/Main_page, 04/2010-10/2010.
Wiki2	4896	51	Hourly traffic to http://en.wikipedia.org/wiki/Lady_gaga, 04/2010-10/2010.
Wiki3	4896	118	Hourly traffic to http://en.wikipedia.org/wiki/Inception_(film), 04/2010-10/2010.
SP500	10136	497	S&P 500 index, 06/1960-06/2000.
HPQ	12109	232	Closing price of HPQ in NYSE for every trading day, 01/1962-02/2010.
IBM	12109	198	Closing price of IBM in NYSE for every trading day, 01/1962-02/2010.
AOL	132480	127	Number of queries to AOL search engine in every minute over three months.
WC98	7603201	286	Number of requests to World Cup 98 web site in every second, 04/1998-07/1998.

6.1. Experimental Results on Basic Prominent Streak Discovery

We used multiple real-world datasets, including time series data library,[3] Wikipedia traffic statistics dataset,[4] NYSE exchange data,[5] AOL search engine log,[6] and FIFA World Cup 98 web site access log.[7] These datasets cover a variety of application scenarios, including meteorology, hydrology, finance, web log, and network traffic. Table I shows the information of 12 data sequences from these data sets that we used in experiments. For each data sequence, we list its name, length, and the number of prominent streaks in the sequence. Each data sequence was stored in a data file.

Examples of Interesting Prominent Streaks Discovered:

From 1985 to 1989, there had been more than one thousand consecutive trading days with morning gold price greater than $300. During this period, there had been a streak of four hundred days with price more than $400, though the $500 price only lasted two days at most.

In Melbourne, Australia, during the years between 1981 and 1990, the weather had been pleasant. There had been more than two thousand days with minimal temperature above zero, and the streak was not ending. (We do not have data beyond 1990.) The longest streak during which the temperature hit above 35 degrees Celsius is six days. It was in the summer of the year 1981.

More than half of the prominent streaks we found in the traffic data of the Lady Gaga Wikipedia page were around September 12th, when she became a big winner in the MTV Video Music Awards (VMA) 2010. During that time, the page had been visited by at least 2000 people in every hour for almost four days.

Number of Candidate Streaks:

The three algorithms for candidate streak generation, namely Baseline (Algorithm 1), NLPS (Algorithm 3), and LLPS (Algorithm 5), differ by the ways they produce candidates and thus the numbers of produced candidates. Table II shows the total number of candidate streaks considered by each algorithm on each data sequence. The baseline algorithm produces an extremely large number of candidates since it enumerates all possible streaks, e.g., $\binom{7603202}{2}$=2.89×10^{13} for WC98. By contrast, NLPS only needs to consider $\bigcup_{1\leq k\leq |P|} \mathcal{LPS}^{k}_{P_k}$, which is a superset of the real prominent streaks $\mathcal{PS}_P$ but a much smaller subset of all possible streaks $\mathcal{S}_P$. For instance, the number of candidate streaks by NLPS is 1.78×10^8 for WC98, which is 5 orders of magnitude

[3]http://robjhyndman.com/TSDL/

[4]http://dammit.lt/wikistats/

[5]http://www.infochimps.com/datasets/nyse-daily-1970-2010-open-close-high-low-and-volume

[6]http://gregsadetsky.com/aol-data/

[7]http://ita.ee.lbl.gov/html/contrib/WorldCup.html

Table II. Number of Candidate Streaks, Basic Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
Gold	5.77×10^5	6.04×10^4	1.05×10^3
River	9.81×10^5	2.18×10^4	1.33×10^3
Melb1	6.66×10^6	4.47×10^4	3.50×10^3
Melb2	6.66×10^6	4.28×10^4	3.49×10^3
Wiki1	1.20×10^7	7.16×10^4	4.79×10^3
Wiki2	1.20×10^7	5.77×10^4	4.75×10^3
Wiki3	1.20×10^7	7.31×10^4	4.70×10^3
SP500	5.14×10^7	1.69×10^6	9.98×10^3
HPQ	7.33×10^7	5.24×10^5	1.08×10^4
IBM	7.33×10^7	6.97×10^5	1.13×10^4
AOL	8.78×10^9	3.53×10^6	1.20×10^5
WC98	2.89×10^{13}	1.78×10^8	6.69×10^6

Table III. Execution Time (in Milliseconds), Basic Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
Gold	183	122	13
River	126	84	19
Melb1	385	101	36
Melb2	384	101	35
Wiki1	670	105	46
Wiki2	646	97	46
Wiki3	632	126	48
SP500	4453	789	116
HPQ	6285	338	101
IBM	4228	377	135
AOL	290744	752	201
WC98	>1 hour	38999	3012

smaller than what Baseline considers. LLPS further significantly educes the number of candidates by only considering LPSs. For example, there are 6.69×10^6 LPSs in WC98, which is about 30 times smaller than 1.78×10^8. Note that the number of LPSs for LLPS is bounded by sequence length (Property 4), which is verified by Table II.

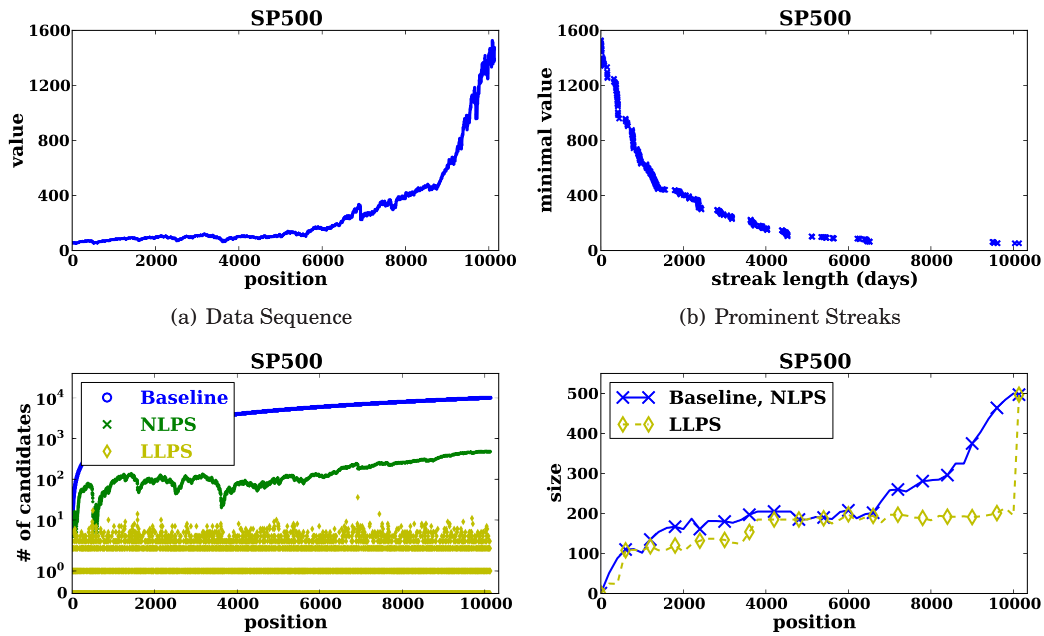
Execution Time:

The number of candidate streaks directly determines the efficiency of our algorithms. In Table III we report the execution time of our algorithms using the three candidate streak generation methods (Baseline, NLPS, LLPS), for all 12 data sequences. For skyline operation, we implemented the sorting-based, external-memory sorting-based, and BST-based skyline methods mentioned in Section 1. Under these different skyline methods, Baseline, NLPS, and LLPS perform and compare consistently. Therefore in Table III we only report the results for implementations based on the BST-based skyline method, due to space limitations. The reported execution time is in milliseconds and is the average of five runs.

When reporting the execution time of these algorithms, we excluded data loading time, i.e., the time spent on just reading each data file. This is because data loading time is dominated by processing time of the algorithms once the data file gets large. In our experiments, WC98 cost 1 second to load while the loading time of all other datasets was below 30ms.

In Table III we use '$>$1 hour' to denote the execution time when an algorithm could not finish within one hour (i.e., 3600000ms). This lower bound is sufficient in showing the performance difference of the various algorithms.

With regard to the comparison of Baseline, NLPS, and LLPS, it is clear from Table III that LLPS outperforms NLPS, and both NLPS and LLPS are far more efficient



(a) Data Sequence

(b) Prominent Streaks

(c) Number of Candidate Streaks

(d) Size of Dynamic Skyline

Fig. 4. Detailed Results on SP500, Basic Prominent Streak Discovery.

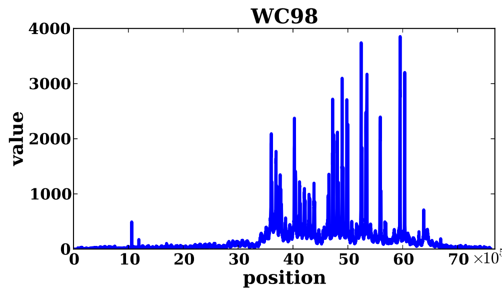
than Baseline. This is exactly due to the large gap in the number of candidate streaks (shown in Table II), which in turn determines the number of comparisons performed during skyline operations.

A Closer Look:

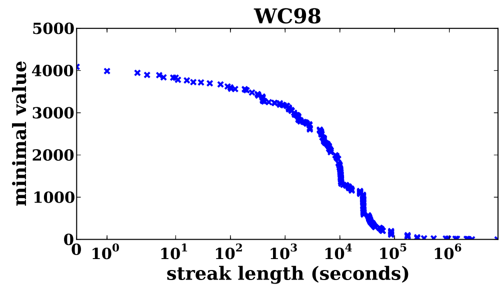
To have a better understanding of the experimental results, we take a close look at the SP500 data sequence. Figure 4(a) shows the data sequence itself. We see that the sequence is almost monotonically increasing at the coarse grain level. Due to that, the number of prominent streaks found in SP500 (497, as shown in Table I) is the most among all the data sequences. We also visualize the prominent streaks in SP500 in Figure 4(b), where the x-axis is for interval length and the y-axis is for minimal value in the interval.

In Table II we have seen the huge difference among Baseline, NLPS and LLPS in total number of candidate streaks. These three algorithms all generate candidates progressively. Therefore in Figure 4(c) we show, for each algorithm, the number of new candidate streaks produced at every value position of the data sequence. The figure clearly shows the superiority of LLPS since it always generates orders of magnitude less candidates at each position.

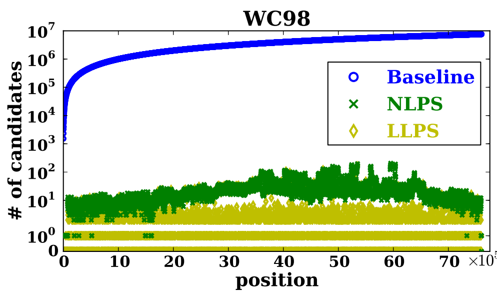
The BST-based skyline method maintains a dynamic skyline, as a binary search tree, in memory. The size of this tree affects the efficiency of tree operations, such as inserting and deleting a streak. Figure 4(d) shows the size of the dynamic skyline along the sequence of SP500 by each algorithm. The curves for Baseline and NLPS overlap since they both store $\mathcal{PS}_{P_k}$, at every position k, in the dynamic skyline. On the contrary, LLPS does not need to store some streaks in $\mathcal{PS}_{P_k}$, hence the tree size is much smaller than that for Baseline/NLPS when the sequence is almost constantly growing in the second half of SP500.



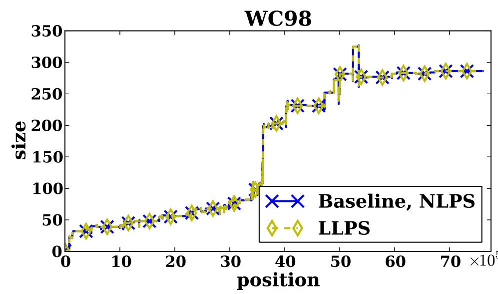
(a) Data Sequence



(b) Prominent Streaks

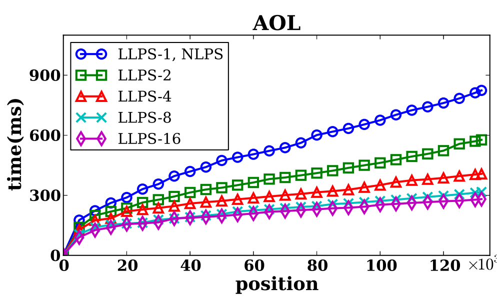


(c) Number of Candidate Streaks

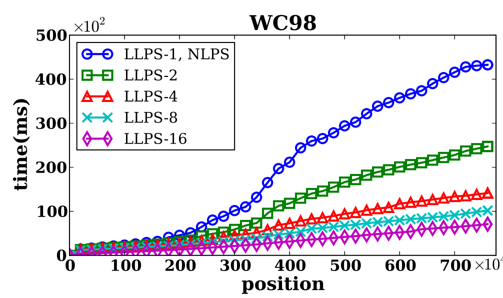


(d) Size of Dynamic Skyline

Fig. 5. Detailed Results on WC98, Basic Prominent Streak Discovery.



(a) AOL



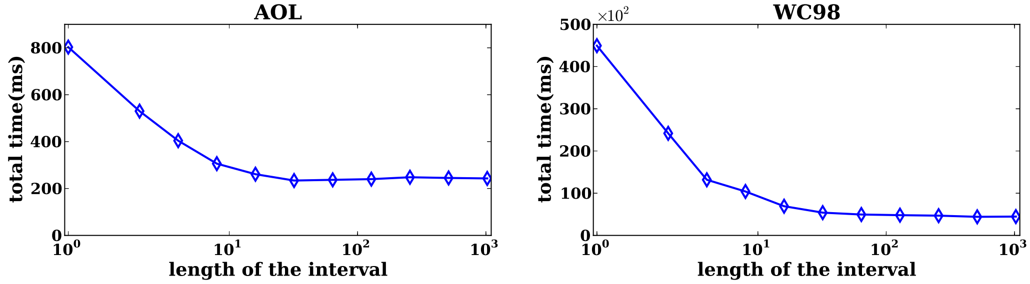
(b) WC98

Fig. 6. Cumulative Execution Time at Various Positions, for Different Reporting Frequencies, Basic Prominent Streak Discovery.

In Figure 5 we show the detailed results on WC98 data, which are similar to the results on SP500 but are also different on several aspects. The data sequence fluctuates. Hence there are less candidate streaks by NLPS and LLPS, which makes the gap between them and Baseline much bigger. For the same reason, the size of the dynamic skyline is almost identical for the three algorithms. Note that Figure 5(b) uses logarithmic scale on x-axis, because the very long streaks would otherwise make most other streaks cluttered to the left if linear scale is used.

Monitoring Prominent Streaks:

In Section 4 we discussed how to monitor the prominent streaks as a data sequence evolves and new data values come. The adaptation of LLPS for monitoring purpose was



(a) AOL (b) WC98

Fig. 7. Total Execution Time by Reporting Frequencies, Basic Prominent Streak Discovery.

shown in Algorithm 7. This algorithm can control at which positions the prominent streaks (so far) need to be reported.

Take AOL and WC98 as examples. Figure 6 shows the execution time of Algorithm 7. The x-axis represents the sequence position, and the y-axis is for the total execution time by that position. There are five curves in each figure, corresponding to five different frequencies of reporting prominent streaks. For instance, LLPS-1 means that, whenever a new data entry comes, all the prominent streaks so far are reported; LLPS-16 means the prominent streaks are requested at every 16 data entries. As discussed in Section 4, LLPS-1 is identical to NLPS (Algorithm 3), and LLPS-n is identical to LLPS (Algorithm 5), where n is the sequence length when it does not evolve anymore. Figures 6(a) and 6(b) clearly show that the total execution time of LLPS-i increases as the reporting frequency increases (i.e., reporting interval i decreases). Figures 7(a) and 7(b) further show how the total execution time changes along different reporting intervals. We can see that the execution time drops rapidly at the beginning and quickly reaches near-optimal value even when the frequency is still fairly high (e.g., reporting the prominent streaks at every 16 entries.)

6.2. Experimental Results on General Prominent Streak Discovery

In this section, we discuss the results on top-*k*, multi-sequence, and multi-dimensional prominent streak discovery. At the end of this section, we also present the results from an experiment that put together these different extensions.

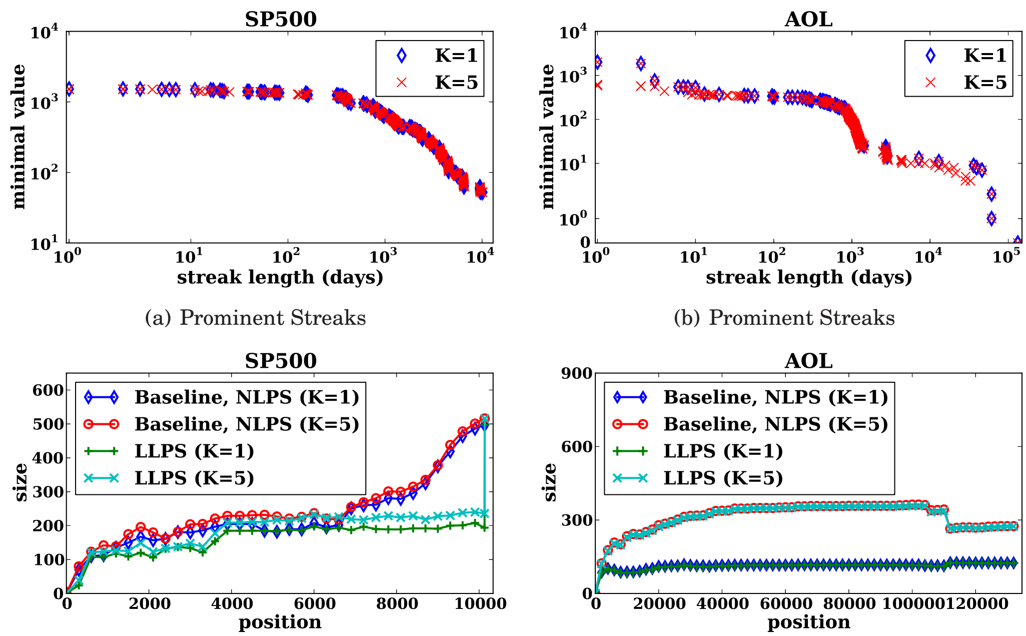
Top-k Prominent Streaks:

The experiments on top-k prominent streaks were conducted on the same datasets discussed in Section 6.1. For each dataset, Table IV shows the number of top-*5* prominent streaks (i.e., $\mathcal{KPS}_P$ in Definition 5) and the execution time of the extended Baseline, NLPS and LLPS algorithms. Note that the number of candidate streaks shown in Table II remains the same, since the same candidate streak generation methods are used for top-*k* prominent streaks, as discussed in Section 5.1.

As Table IV shows, in comparison with the execution time in Table III (i.e., the time of discovering top-1 prominent streaks), the execution time of Baseline increased by one or more orders of magnitude, while the performance of NLPS and LLPS was degraded by less than one order of magnitude in most cases. This is explained as follows. Finding top-*k* prominent streaks incurs higher cost of skyline operation than finding top-1 prominent streaks. More specifically, the cost of skyline operation is determined by the number of dominance comparisons between candidate streaks and streaks in the dynamic skyline. Therefore the number of comparisons increases by both the number of candidate streaks and the size of the dynamic skyline. In comparison with top-*1*,

Table IV. Number of Prominent Streaks and Execution Time (in Milliseconds), Top-5 Prominent Streaks.

name	# prominent streaks	Baseline	NLPS	LLPS
Gold	147	1884	348	44
River	144	6.81×10^4	275	57
Melb1	160	6.06×10^7	572	96
Melb2	160	3.01×10^6	445	150
Wiki1	181	3.68×10^7	1369	140
Wiki2	115	1.88×10^7	565	172
Wiki3	172	1.05×10^6	473	136
SP500	516	7.09×10^6	13700	270
HPQ	251	$>$ 10 hours	3211	178
IBM	232	$>$ 10 hours	5914	229
AOL	250	$>$ 10 hours	26000	798
WC98	409	$>$ 10 hours	$>$ 10 hours	13300



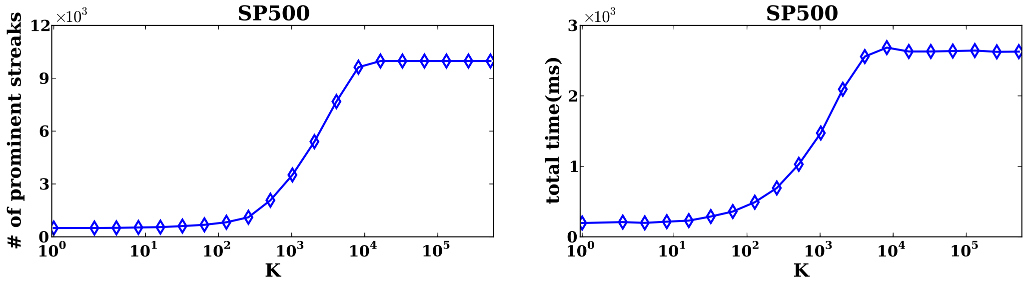
(a) Prominent Streaks (b) Prominent Streaks

(c) Size of Dynamic Skyline (d) Size of Dynamic Skyline

Fig. 8. Detailed Results on SP500 and AOL, Top-1 vs. Top-5 Prominent Streaks.

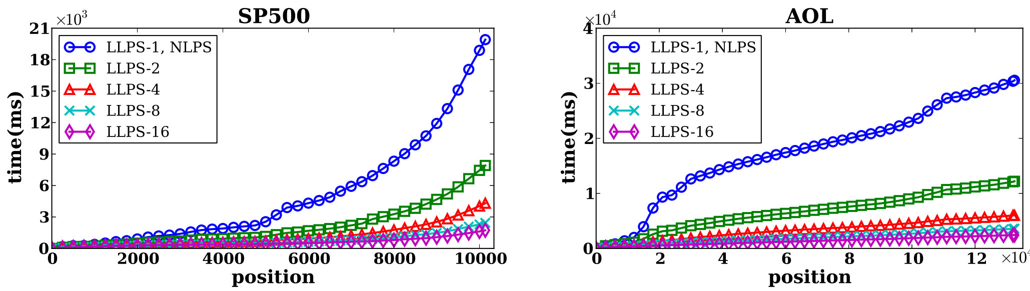
finding top-k prominent streaks requires maintaining local prominent streaks with as many as $k-1$ dominators, which increases the size of dynamic skyline and thus incurs larger cost. For example, a dominator search for a candidate cannot terminate until the number of dominators reaches k, whereas the search terminates immediately in top-1 algorithms once a dominator is found. The more candidate streaks there are, the larger the increment of skyline operation cost (from top-1 to top-k) grows. This further explains why the performance of Baseline was degraded the most.

Figure 8 shows some interesting detailed results on two different sequences. Since sequence SP500 increases almost monotonically, a local prominent streak that is not globally prominent most likely has a relatively large number of dominators. Hence, the size of dynamic skyline in top-*5* prominent streak discovery is only slightly larger than that in top-*1*. This explains Figure 8(c). On the contrary, for sequence AOL, the size of dynamic skyline for top-*5* is about twice the size for top-*1* (igure 8(d)). This is



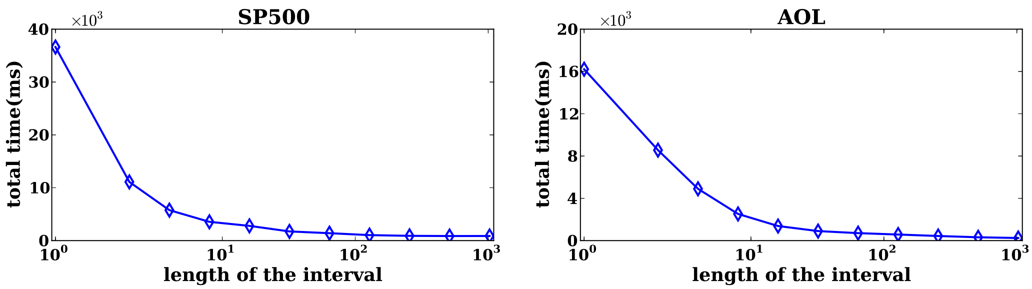
(a) Number of Prominent Streaks (b) Execution Time

Fig. 9. Number of Prominent Streaks and Execution Time, LLPS on SP500, Top-k Prominent Streaks, Varying k.



(a) SP500 (b) AOL

Fig. 10. Cumulative Execution Time at Various Positions, for Different Reporting Frequencies, Top-5 Prominent Streak Discovery.



(a) SP500 (b) AOL

Fig. 11. Total Execution Time by Reporting Frequencies, Top-5 Prominent Streak Discovery.

because sequence AOL fluctuates. The prominent streaks have different right ends of intervals due to the fluctuation. This also explains why the sizes of dynamic skylines in Baseline, NLPS and LLPS do not differ much from each other in this sequence. However, as Table IV shows, their differences on execution time are still significant because NLPS and LLPS generate much less candidates than Baseline does.

By Definition 5, $\mathcal{PS}_P \subseteq \mathcal{KPS}_P$, i.e., all prominent streaks are also top-k prominent streaks. This is clearly shown in Figures 8(a) and 8(b). Furthermore, $\mathcal{KPS}_P \subseteq \mathcal{LPS}_P$, i.e., top-$k$ prominent streaks must be local prominent streaks too. Therefore, the set

Table V. Data Sequences Used in Experiments on Multi-sequence Prominent Streak Discovery.

name	# sequences	average length	# prominent streaks	description
NBA1	1225	281	28	Points scored by all NBA players from 1991-2004
Wiki	8	14454	59	Hourly traffic to the Wikipedia pages of Ivy League universities

Table VI. Number of Candidate Streaks, Multi-sequence Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
NBA1	9.41×10^7	1.23×10^6	3.31×10^5
Wiki	8.36×10^8	1.23×10^6	1.86×10^5

Table VII. Execution Time (in Milliseconds), Multi-sequence Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
NBA1	3436	310	292
Wiki	33537	275	190

$\mathcal{KPS}_P$ grows by k and stops growing after k reaches a certain value, when all streaks in $\mathcal{LPS}_P$ are included in $\mathcal{KPS}_P$. This is demonstrated by Figure 9(a) in which the number of prominent streaks in sequence SP500 increases by k until k reaches about $10,000$. As a result, total execution time also changes in sync with the number of prominent streaks, as shown in Figure 9(b).

We also experimented with monitoring top-*k* prominent streaks. The results are shown in Figure 10 and Figure 11, which exhibit patterns of execution time similar to the patterns in Figure 6 and Figure 7 for monitoring top-1 prominent streaks.

Multi-sequence Prominent Streaks:

We used two datasets for experiments on multi-sequence prominent streak discovery. One (Wiki) is the hourly traffic to Ivy League universities' Wikipedia pages,[4] one sequence per university. The other dataset (NBA1) contains 1225 sequences, one sequence per NBA player. Each sequence lists the scores of a player in all the games he played from 1991 to 2004. [8] The characteristics of these two datasets are shown in Table V, including the number of sequences, average sequence length and the number of prominent streaks. Tables VI and VII show the number of candidate streaks and the execution time, respectively, for Baseline, NLPS and LLPS. The results are very similar to that in Tables II and III. This is because the process of multi-sequence prominent streak discovery is not very different from its single-sequence counterpart.

For dataset NBA1, Table VIII shows the distribution of players by the number of prominent streaks contributed by them. All 29 prominent streaks, i.e., NBA scoring records in the period of 1991 to 2004, come from merely 10 different players. Table IX shows the detailed records. One interesting observation from the table is that Karl Malone and John Stockton, two of the healthiest NBA players, had scored in two longest streaks of games. Another example is that Allen Iverson is the only one who scored at least 20 points in more than 50 consecutive games.

Multi-dimensional Prominent Streaks:

We used three datasets for experiments on multi-dimensional prominent streak discovery, listed in Table X. The first dataset is the game log of NBA player Karl Malone, from 1991 to 2004 seasons.[8] This is a sequence of 986 elements, each of which represents Malone's performance in a game on 6 performance dimensions. The second dataset is the 2003-2011 Texas Motor Vehicle Crash Statistics,[9] a 5-dimensional se-

[8]http://www.databasebasketball.com/index.htm
[9]http://www.dot.state.tx.us/txdot_library/drivers_vehicles/publications/crash_statistics/default.htm

Table VIII. Distribution of Players by Number of Prominent Streaks.

number of prominent streaks	number of players
0	1215
1	6
3	1
4	1
5	1
10	1

Table IX. Multi-sequence Prominent Streaks in Datast NBA1.

length	minimal value	players
1	71	David Robinson
2	51	Allen Iverson; Antawn Jamison
4	42	Kobe Bryant
9	40	Kobe Bryant
13	35	Kobe Bryant
14	32	Kobe Bryant
16	30	Kobe Bryant
17	27	Michael Jordan
27	26	Allen Iverson
34	24	Tracy McGrady
45	21	Allen Iverson
57	20	Allen Iverson
74	19	Shaquille O'Neal
94	18	Shaquille O'Neal
96	17	Karl Malone
119	16	Karl Malone
149	15	Karl Malone
159	14	Karl Malone
263	13	Karl Malone
357	12	Karl Malone
527	11	Karl Malone
575	10	Karl Malone
758	7	Karl Malone
858	6	Shaquille O'Neal
866	2	Karl Malone
932	1	John Stockton
1185	0	Jim Jackson

Table X. Data Sequences Used in Experiments on Multi-dimensional Prominent Streak Discovery.

name	length	# prominent streaks	# dimensions	description
Malone	986	640	6	1991-2004 game log of Karl Malone (minutes, points, rebounds, assists, steals, blocks)
Crashes	3287	1493	5	2003-2011 Texas Motor Vehicle Crash Statistics (Crashes and Injuries by Date)
AAPL	6411	2616	3	NASDAQ stock data of Apple Inc. from 1970 to 2010, on daily values of opening price, change ratio ($(open - close)/open \times 100\%$) and trading volume

quence of 3287 elements, where each element is for one day and represents the daily counts of crashes, injuries, fatalities, and so on. The last dataset is the historical NASDAQ stock data of Apple Inc. from 1970 to 2010.[10] In this 6411-element sequence, each element is for a trading day and contains the opening price, change ratio, and trading volume of the stock of Apple Inc. on that day.

The number of candidate streaks and the execution time by Baseline, NLPS, and LLPS are shown in Tables XI and XII. Figure 12 further shows detailed experimental results on dataset AAPL. The observations made on these results are similar to those for basic, top-k, and multi-sequence prominent streak discovery.

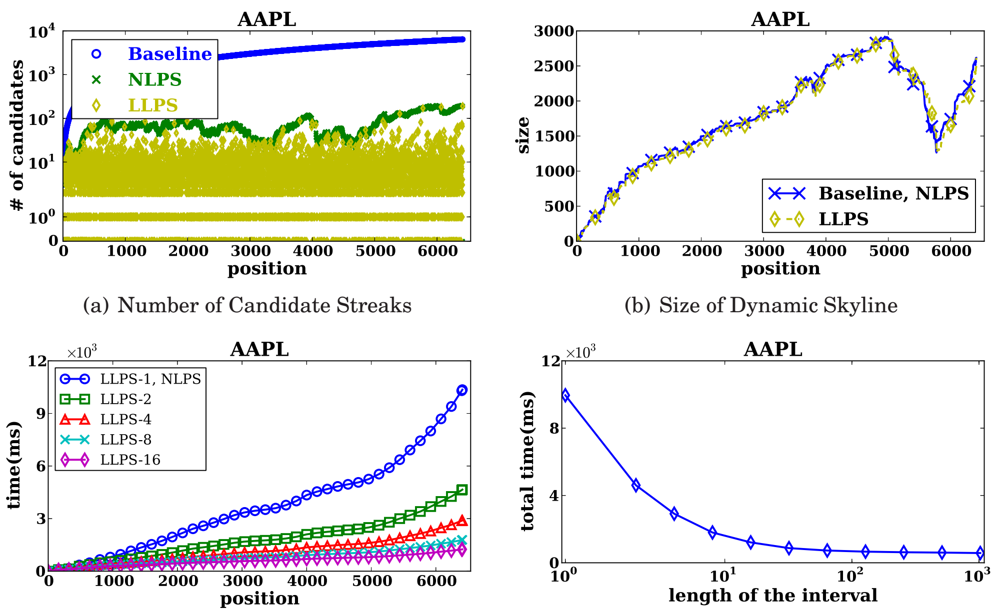
[10]http://www.infochimps.com/datasets/nasdaq-exchange-daily-1970-2010-open-close-high-low-and-volume

Table XI. Number of Candidate Streaks, Multi-dimensional Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
Malone	4.87×10^5	1.27×10^4	4.47×10^3
Crashes	5.40×10^6	6.95×10^5	1.82×10^4
AAPL	2.06×10^7	4.77×10^5	4.08×10^5

Table XII. Execution Time (in Milliseconds), Multi-dimensional Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
Malone	4575	336	180
Crashes	1.08×10^5	1113	326
AAPL	5.65×10^5	9997	557



(a) Number of Candidate Streaks

(b) Size of Dynamic Skyline

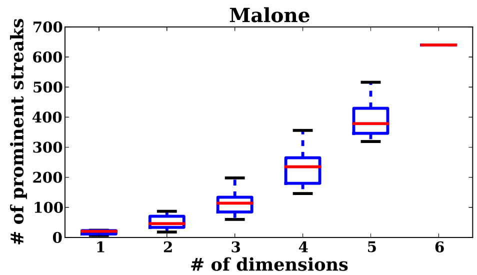
(c) Cumulative Execution Time at Various Positions, for Different Reporting Frequencies

(d) Total Execution Time by Reporting Frequencies

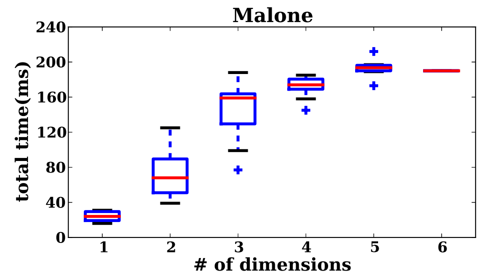
Fig. 12. Detailed Results on AAPL, Multi-dimensional Prominent Streak Discovery.

We also investigated how number of prominent streaks and total execution time of LLPS increase by the dimensionality of data, using dataset Malone. As the boxplots in Figure 13 show, these measures do not increase exponentially by data dimensionality, at least under small dimensionality such as 6. This indicates that, while the “curse of dimensionality” can raise concerns, the empirical results are much more encouraging. It is partly due to that data values fluctuate and thus the appearance of a small value terminates many prominent streaks. Furthermore, data values are correlated, which practically reduces data dimensionality. Finally, the results are for at most 6 dimensions. We note that arguably the prominent streaks found in the real world, such as the ones in Section 1, mostly would not have more than 6 dimensions.

Putting it Together: Top-k Prominent Streaks on Multiple Multi-dimensional Sequences:
We also used dataset NBA2 (Table XIII) for experiments on discovering top-k prominent streaks from multiple multi-dimensional sequences. This dataset contains 1185



(a) Number of Prominent Streaks



(b) Execution Time of LLPS

Fig. 13. Experiments on Increasing Dimensionality.

Table XIII. Data Sequences Used in Experiments on Top-5 Multi-sequence Multi-dimensional Prominent Streak Discovery.

name	# sequences	average length	# dimensions	# prominent streaks	description
NBA2	1185	290	6	10867	1991-2004 game log of all NBA players (minutes, points, rebounds, assists, steals, blocks)

Table XIV. Number of Candidate Streaks, Top-5 Multi-sequence Multi-dimensional Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
NBA2	9.41×10^7	2.98×10^6	8.76×10^5

6-dimensional sequences, each of which corresponds to the game log of an NBA player from 1991 to 2004. One of the sequences is the aforementioned dataset Malone.

Figure 14 shows that distribution of prominent streaks by length. It is clear that the distribution follows the power law. The reason is that the minimal value vector for a streak takes the minimal value on each dimension from all elements. The longer a streak is, the smaller the values in its minimal value vector become. Therefore it is difficult for a long streak to stand out as prominent. Figure 15 shows detailed experimental results on this dataset which show similar patterns to those observed for aforementioned experiments.

7. CONCLUSION AND FUTURE WORK

In this paper, we study the problem of discovering prominent streaks in sequence data. A prominent streak is a long consecutive subsequence consisting of only large (small) values. We propose efficient methods based on the concept of local prominent streak (LPS). We prove that prominent streaks are a subset of LPSs and the number of LPSs is less than the length of a data sequence. Our linear LPS-based method guarantees to consider only local prominent streaks, thus achieving significant reduction in candidate streaks. The proposed properties and algorithms are also extended for discovering general top-k, multi-sequence, and multi-dimensional prominent streaks. The results of experiments over multiple real datasets verified the effectiveness of the proposed methods.

Prominent streak discovery provides insightful data patterns for data analysis in many real-world applications and is an enabling technique for computational journalism. Given its real-world usefulness and complexity, the research on prominent streaks in sequence data opens a spectrum of challenging problems. Here we briefly outline several future directions. (1) More general concept of prominent streak can be pursued. For instance, finding conditional prominent streaks is about discovering constrains that make streaks prominent, e.g. “since June 2009” and “the month of

Table XV. Execution Time (in Milliseconds), Top-5 Multi-sequence Multi-dimensional Prominent Streak Discovery.

name	Baseline	NLPS	LLPS
NBA2	1.39×10^7	4.33×10^5	1.14×10^5

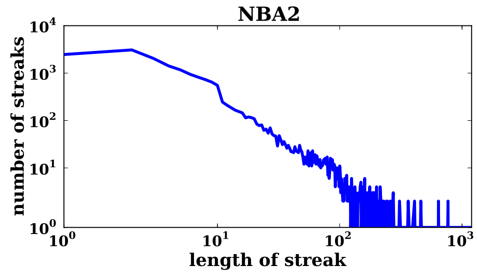
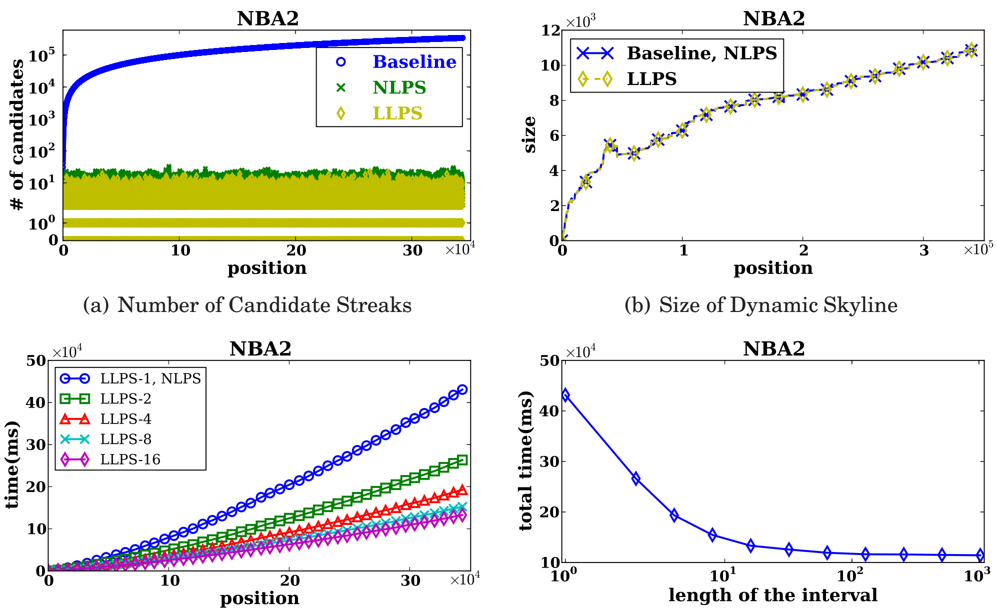


Fig. 14. Distribution of Prominent Streaks by Length.



(a) Number of Candidate Streaks (b) Size of Dynamic Skyline

(c) Cumulative Execution Time at Various Positions, for Different Reporting Frequencies (d) Total Execution Time by Reporting Frequencies

Fig. 15. Detailed Results on NBA2, Top-5 Multi-sequence Multi-dimensional Prominent Streak Discovery.

July" for the motivating example streaks in Section 1. (2) Prominent streaks can be incorporated with the model of data cube [Gray et al. 1997]. Specifically, given a multi-dimensional sequence, the goal is to discover prominent streaks in not only the full space but also all possible subspaces. For example, given the NBA2 dataset used in our experiments, we may want to find prominent streaks in spaces (points, rebounds), (points, assists, blocks), and so on. (3) When there are many prominent streaks, it is important to rank them by their interestingness, so that a user can focus on the top-ranked prominent streaks. Some important ranking criteria to consider include streak length, number of similar prominent streaks in the dataset, and so on.

Acknowledgements: We thank Jun Yang for discussion of the initial ideas of this paper when Chengkai Li and Jun Yang were both visiting HP Labs in Beijing, China in the summer of 2010.

REFERENCES

Rakesh Agrawal, Christos Faloutsos, and Arun Swami. 1993. Efficient similarity search in sequence databases. (1993), 69–84. DOI:http://dx.doi.org/10.1007/3-540-57301-1_5

Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. 1995. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 490–501. http://dl.acm.org/citation.cfm?id=645921.673155

R. Agrawal and R. Srikant. 1995. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*. 3–14.

S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. 1990. Basic local alignment search tool. *Journal of molecular biology* 215, 3 (1990), 403–410.

Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (Sept. 1975), 509–517. DOI:http://dx.doi.org/10.1145/361002.361007

Jon Louis Bentley. 1979. Multidimensional binary search trees in database applications. *Software Engineering, IEEE Transactions on* SE-5, 4 (1979), 333–340. DOI:http://dx.doi.org/10.1109/TSE.1979.234200

Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*. 421–430. DOI:http://dx.doi.org/10.1109/ICDE.2001.914855

Chee-Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. 2006. On high dimensional skylines. In *Proceedings of the 10th international conference on Advances in Database Technology*. Springer-Verlag, Berlin, Heidelberg, 478–495. DOI:http://dx.doi.org/10.1007/11687238_30

J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. 2003. Skyline with presorting. In *Data Engineering, 2003. Proceedings. 19th International Conference on*. 717–719. DOI:http://dx.doi.org/10.1109/ICDE.2003.1260846

Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational Journalism: A Call to Arms to Database Researchers. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR)*. 148–151.

Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. 1993. Fast subsequence matching in time-series databases. University of Maryland at College Park, College Park, MD, USA.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Min. Knowl. Discov.* 1, 1 (Jan. 1997), 29–53. DOI:http://dx.doi.org/10.1023/A:1009726021843

Bin Jiang and Jian Pei. 2009. Online Interval Skyline Queries on Time Series. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE '09)*. IEEE Computer Society, Washington, DC, USA, 1036–1047. DOI:http://dx.doi.org/10.1109/ICDE.2009.70

Xiao Jiang, Chengkai Li, Ping Luo, Min Wang, and Yong Yu. 2011. Prominent streak discovery in sequence data. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*. ACM, New York, NY, USA, 1280–1288. DOI:http://dx.doi.org/10.1145/2020408.2020601

Donald Kossmann, Frank Ramsak, and Steffen Rost. 2002. Shooting stars in the sky: an online algorithm for skyline queries. In *Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02)*. VLDB Endowment, 275–286. http://dl.acm.org/citation.cfm?id=1287369.1287394

H. T. Kung, F. Luccio, and F. P. Preparata. 1975. On Finding the Maxima of a Set of Vectors. *J. ACM* 22, 4 (Oct. 1975), 469–476. DOI:http://dx.doi.org/10.1145/321906.321910

T. Warren Liao. 2005. Clustering of time series data-a survey. *Pattern Recogn.* 38, 11 (Nov. 2005), 1857–1874. DOI:http://dx.doi.org/10.1016/j.patcog.2005.01.025

Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. 2007. Selecting Stars: The k Most Representative Skyline Operator. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. 86–95. DOI:http://dx.doi.org/10.1109/ICDE.2007.367854

T. Oates, L. Firoiu, and P.R. Cohen. 1999. Clustering time series with hidden markov models and dynamic time warping. In *Proceedings of the IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*. 17–21.

Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2005. Progressive skyline computation in database systems. *ACM Trans. Database Syst.* 30, 1 (March 2005), 41–82. DOI:http://dx.doi.org/10.1145/1061318.1061320

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Trans. on Knowl. and Data Eng.* 16, 11 (Nov. 2004), 1424–1440. DOI:http://dx.doi.org/10.1109/TKDE.2004.77

Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang, Yufei Tao, Jeffrey Xu Yu, and Qing Zhang. 2006. Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.* 31, 4 (Dec. 2006), 1335–1381. DOI:http://dx.doi.org/10.1145/1189769.1189774

L. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286. DOI:http://dx.doi.org/10.1109/5.18626

Young-In Shin and Donald Fussell. 2007. Parametric kernels for sequence data analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1047–1052. http://dl.acm.org/citation.cfm?id=1625275.1625445

Padhraic Smyth and others. 1997. Clustering sequences with hidden Markov models. *Advances in neural information processing systems* (1997), 648–654.

Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. (1996), 1–17. DOI:http://dx.doi.org/10.1007/BFb0014140

Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. 2001. Efficient Progressive Skyline Computation. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 301–310. http://dl.acm.org/citation.cfm?id=645927.672217

Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. 2009. Distance-Based Representative Skyline. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, USA, 892–903. DOI:http://dx.doi.org/10.1109/ICDE.2009.84

Yufei Tao, Xiaokui Xiao, and Jian Pei. 2006. SUBSKY: Efficient Computation of Skylines in Subspaces. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*. IEEE Computer Society, Washington, DC, USA, 65–. DOI:http://dx.doi.org/10.1109/ICDE.2006.149

Min Wang and X. Sean Wang. 2006. Finding the plateau in an aggregated time series. In *Proceedings of the 7th international conference on Advances in Web-Age Information Management (WAIM '06)*. Springer-Verlag, Berlin, Heidelberg, 325–336. DOI:http://dx.doi.org/10.1007/11775300_28

Weng-Keen Wong. 2004. *Data mining for early disease outbreak detection*. Ph.D. Dissertation. Pittsburgh, PA, USA. Advisor(s) Moore, Andrew.

Tian Xia and Donghui Zhang. 2006. Refreshing the sky: the compressed skycube with efficient support for frequent updates. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD '06)*. ACM, New York, NY, USA, 491–502. DOI:http://dx.doi.org/10.1145/1142473.1142529

X. Yan, J. Han, and R. Afshar. 2003. CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of SIAM International Conference on Data Mining*. 166–177.

B.-K. Yi, H.V. Jagadish, and C. Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *Data Engineering, 1998. Proceedings., 14th International Conference on*. 201–208. DOI:http://dx.doi.org/10.1109/ICDE.1998.655778

MohammedJ. Zaki. 2001. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning* 42, 1-2 (2001), 31–60. DOI:http://dx.doi.org/10.1023/A:1007652502315

Zhenjie Zhang, Xinyu Guo, Hua Lu, Anthony K. H. Tung, and Nan Wang. 2005. Discovering strong skyline points in high dimensional spaces. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM '05)*. ACM, New York, NY, USA, 247–248. DOI:http://dx.doi.org/10.1145/1099554.1099610