TR 09-006

# Discovering Flow Anomalies: A SWEET Approach

James Kang, Shashi Shekhar, Christine Wennen, and Paige Novak

March 09, 2009

# Discovering Flow Anomalies: A SWEET Approach

James M. Kang[1] Shashi Shekhar[1] Christine Wennen[2] Paige Novak[2]

[1]Department of Computer Science, [2]Department of Civil Engineering,
University of Minnesota, MN, USA
[1]{jkang, shekhar}@cs.umn.edu, [2]{wenne052,novak010}@umn.edu

## Abstract

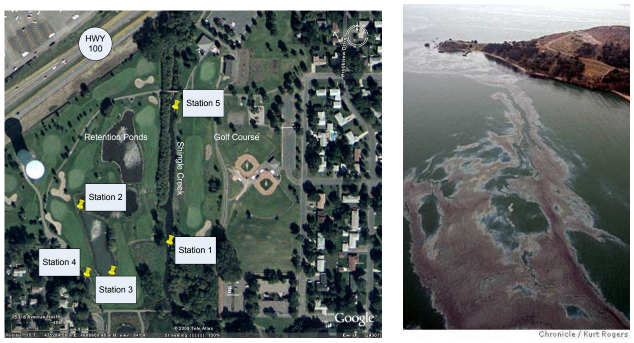
*Given a percentage-threshold and readings from a pair of consecutive upstream and downstream sensors, flow anomaly discovery identifies dominant time intervals where the fraction of time instants of significantly mis-matched sensor readings exceed the given percentage-threshold. Discovering flow anomalies (FA) is an important problem due to applications such as environmental flow monitoring networks and early warning detection systems for water quality problems. However, mining FAs is computationally expensive because of the large (potentially infinite) number of time instants of measurement and potentially long delays due to stagnant (e.g. lakes) or slow moving (e.g. wetland) water bodies between consecutive sensors. Traditional outlier detection methods (e.g. t-test) are suited for detecting transient FAs (i.e., time instants of significant mis-matches across consecutive sensors) and cannot detect persistent FAs (i.e., long variable time-windows with a high fraction of time instant transient FAs) due to a lack of a pre-defined window size. In contrast, we propose a Smart Window Enumeration and Evaluation of persistence-Thresholds (SWEET) method to efficiently explore the search space of all possible window lengths. Computation overhead is brought down significantly by restricting the start and end points of a window to coincide with transient FAs, using a smart counter and efficient pruning techniques. Analytical evaluation show that the proposed method is correct and complete. Experimental evaluation using synthetic and real datasets shows our proposed approach outperforms Naïve alternatives.*

## 1. Introduction

**Motivation.** Mining flow anomalies (FA) is important in several spatio-temporal application domains such as environmental monitoring networks for detection of potential flood conditions, chemical spills, and/or pollutants entering river networks. Maintaining sufficient water of high quality for the world population is one of our greatest global challenges [15]. Several recent articles from the popular press report that many dangerous contaminants are entering our water from unknown sources and at unknown times, resulting in expensive and experimental manual investigations (e.g. [12]).

Currently, hydrologists and environmental engineers are placing advanced sensors in water bodies around the United States to understand the behavior of river networks and lakes [11]. Figure 1a shows a satellite image of Shingle Creek, MN where there are five sensors monitoring various water quality parameters [6] (e.g., turbidity, dissolved oxygen, specific conductivity, nitrate levels, etc.). Two sensors (water flow from 5 to 1) are placed in the creek and three others are placed in neighboring ponds. Figure 1a illustrates that the water from neighboring ponds can flow into the creek between sensors 5 and 1. Because of the large amount of data collected continuously, a pollutant entering the river between two monitoring sensors could easily go unnoticed. For example, a measured variable in the Shingle Creek, MN study site is the amount of nitrate in the water at a given time. It is known that the large consumption of nitrate by infants or pregnant women may cause methemoglobenemia, or Blue Baby syndrome, which can cause death [10]. A robust method for identifying such a contamination event and pin-pointing the time intervals at which it occurred while minimizing false alarms could vastly improve not only warning systems but the ability of scientists and engineers to understand the cause of the event.

**Problem Statement.** Given pairs of time-series of measured variables from consecutive upstream and downstream sensors, flow anomaly discovery flags pollution events occurring between the sensor pair by finding time-periods with a (user-defined) high fraction of time-instants having significantly different readings across consecutive upstream and downstream sensors. In absence of contamination between sensors, the observations are similar across both upstream and downstream sensors. If a phenomenon is seen only at one sensor and not the other, a transient flow anomaly has occurred. A single persistent flow anomaly



(a) Shingle Creek, MN (Source: Google Maps) (b) Oil Spill in San Francisco (Courtesy: [7])

**Figure 1. Study Site and Motivational Example (Best Viewed in Color)**

may consist of several transient flow anomalies with a few time-instants without flow-anomalies. A persistent flow anomaly is dominant (e.g., entire oil spill event) if its time-interval is not a subset of time intervals of any other persistent flow anomaly. For example, Figure 1b shows one of the largest oil spills in San Francisco, CA history in November of 2007. As can be seen, the spill does not flow as a single unit; rather there are several gaps within the spill [7].

**Challenges.** Mining FA patterns is computationally challenging for several reasons. First, a single persistent FA pattern may consist of subsets that may or may not be anomalies. Second, size of the time-interval for the longest dominant persistent FAs may not be known in advance. Third, the temporal length of each persistent FA pattern may be different. Fourth, the match between the observations at the upstream and downstream sensors may be one-to-one, one-to-many, or many-to-many. Finally, the datasets of time instants may be potentially infinite in size.

**Related Work.** Related work to the discovery of flow anomalies may be categorized into two groups: string matching and data stream correlations. In string matching, relationships are created based on the similarity or dissimilarly between two strings. Amir et al. proposed an inverse string matching technique that finds a pattern between two strings that maximizes or minimizes the number of mismatches [1]. Lee et al. proposed a similar method to inverse pattern matching that included wild cards [9]. However, there are several main differences with string matching and the discovery of FA patterns. First, several string matching techniques use an exact matching technique between multiple strings, whereas the FA problem needs a statistical measure because an exact match may not exist between data streams. Second, string matching uses a discrete alphabet whereas multiple time series analysis uses a continuous alphabet.

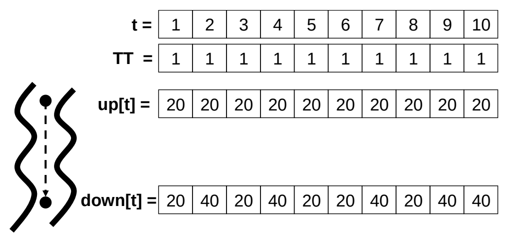
In data streams, relationships are made using a fixed sliding window and a correlation measure. Chan et al. found local correlations between multiple data streams using a sliding window [3]. Sayal found global relationships between data streams and a single siding window to summarize a single data stream [13]. Balut introduced an incremental approach to find correlations of a query for multiple pre-defined time windows [2]. Datar and Muthukrishnan identified rarity and similarity between data streams using a fixed window size [4]. However, these approaches focus on finding correlations using a fixed size sliding window and some correlation measure. Using a fixed window size may assume that the domain specialist knows the duration of the anomalous events which may not be known in environmental systems (e.g. rain events). In addition, a large window size may miss several smaller yet interesting events, whereas a smaller window may miss larger events. Correlation measures are used to find associations between data streams whereas this work proposes a statistical interest measure to find interesting relationships.

To some extent, basic outlier detection techniques (e.g. t-test [5]) detecting transient FAs may discover transient and some persistent FAs (with a 100% mismatched time-interval) (e.g. [8, 14]). However, these methods are not designed to detect persistent flow anomalies with less than 100% mismatched time-instants and may miss many persistent flow anomalies.

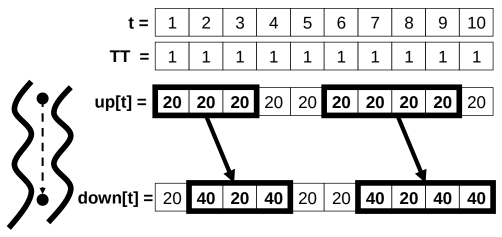
**Contributions.** In this paper, we propose a Smart Window Enumeration and Evaluation of persistent-Thresholds (SWEET) approach that utilizes several key insights into this problem to efficiently identify persistent FAs between the upstream and downstream sensors. First, to reduce the the search space for different sized windows, space of interesting persistent FAs are constrained to start and end at transient FAs, i.e. time-instants with significantly different measurements. Second, a smart counter is used to reduce the computation required to evaluate a candidate time-interval for the interest measure, from a linear function of time-interval-size to a constant function. Finally, a pruning strategy is used to reduce the number of persistent FAs (pFAs) to discover the dominant pFAs (dpFAs).

In summary, this paper makes the following contributions: First, we define Flow Anomalies (FAs) and the FA mining problem. Second, we propose a new interest measure to discover and mine FAs. Third, we characterize the computational structure of the FA discovery problem and propose a novel and computationally efficient SWEET algorithm by restricting the start and end points of a window to coincide with persistent FAs, using a smart counter and efficient pruning techniques. Fourth, we show that the proposed algorithm is correct and complete. Finally, we experimentally evaluate our proposed method using synthetic and real datasets.

**Scope.** The following issues are beyond the scope of this


t = 1 2 3 4 5 6 7 8 9 10
TT = 1 1 1 1 1 1 1 1 1 1
up[t] = 20 20 20 20 20 20 20 20 20 20
down[t] = 20 40 20 40 20 20 40 20 40 40

(a) FA Example Input


t = 1 2 3 4 5 6 7 8 9 10
TT = 1 1 1 1 1 1 1 1 1 1
up[t] = 20 20 20 20 20 20 20 20 20 20
down[t] = 20 40 20 40 20 20 40 20 40 40

(b) FA Example Output

**Figure 2. Example Input and Output for the Flow Anomaly Discovery Problem**

paper: (i) anomalies occurring beyond the two consecutive upstream and downstream sensors, (ii) inferring the travel time from the dataset, that is, the travel time is given as part of the input for the FA discovery problem, and (iii) one-to-many and many-to-many matches of time instants of measurements across sensors, that is, only one-to-one matches is considered for the FA discovery problem.

**Organization.** The rest of the paper is organized as follows. Section 2 presents the basic concepts to provide a formal model of FAs and the problem statement of discovering FAs. Section 3 presents our proposed SWEET method. Analytical analysis of our proposed approach is given in Section 4. Section 5 gives the experimental evaluation and Section 6 concludes the paper and discusses future work.

## 2. Basic Concepts and Problem Statement

In this section, we first introduce several key concepts to model Flow Anomalies (FAs) and then, we give a formal problem statement. Figure 2, referenced throughout this section, illustrates an input and output example of discovering Flow Anomalies. The input consists of time series of 10 time instants for the upstream and downstream sensors with a constant travel time (TT) of 1 for simplicity. The output contains two dominant persistent FAs, using the time instants at the upstream sensor, the periods are 1-3 and 6-9.

### 2.1 Basic Concepts

This section presents several relevant definitions to our problem statement and proposed approaches.

**Definition 1** *An observation $o$ is a measured reading of a variable at every time instant $t$ at the upstream $up$ and a downstream $down$ sensors.*

For example, Figure 2a, $up$ sensor has an observation of 20 at time-instants 1,2,. . .,etc. and $down$ sensor has an observation of 40 at time instant 2. Figure 1a gives an example of two sets of $up$ and a $down$ sensors observing a phenomena. Sensors 5 and 1 are the $up$ and $down$ sensors respectively within the river. Sensors 2 and 4 are the $up$ and $down$ sensors respectively within the lakes.

**Definition 2** *Travel Time, $TT$, is the duration between the time (t) when an observation $o$ is made at the up sensor and the time (t+TT[t]) of the corresponding observation at the down sensor.*

Figure 2a gives an example where the travel time (TT) is 1. For time instant 2, up[2] and down[2+1] have corresponding observations of 20 and 20 respectively.

**Definition 3** *An Instant Pair, $IP$, is two observations of a common phenomenon made at the $up$ and $down$ sensors having a temporal length of $TT$ between them.*

Definition 3 can be formally expressed in Equation 1.

$$IP[t] = (up[t], down[t + TT]) \tag{1}$$

Figure 2a gives an example where the instant pair (IP) at time instant of 1 is IP[1]=(20,40) where the TT=1. We note that this definition assumes a very small temporal footprint for the phenomenon of interest at each sensor. This assumption is made to simplify the discussion in this paper. It is revisited in the last section on future work.

**Definition 4** *An upstream contiguous set of instant pairs are IPs that occur one right after the other for some temporal length $k$.*

Definition 4 can be formally expressed in Equation 2.

$$S = \{< up[t+i], down[(t+i)+TT[t+i]] > | i \in 0, \dots, k\} \tag{2}$$

Figure 2a gives an example of an upstream contiguous set of instant pairs for the period of 1-3 containing $\{< 1, 2 >, < 2, 3 >, < 3, 4 >\}$ where the travel time, TT, is 1.

**Definition 5** *A transient Flow Anomaly, $tFA$ is when the difference between the corresponding observations across each sensor in the $IP$, is larger than the given error threshold, $\Theta_e$.*

The term “flow” refers to anomalies found between the flow of two sensors (e.g. sensors 5 and 1). Definition 5 can be formally expressed in Equation 3.

$$tFA[t] \iff (|up[t] - down[t+TT]| > \Theta_e) \tag{3}$$

Suppose the error threshold is zero (or 10 or 15); then Figure 2a gives an example of a flow anomaly at time interval 1 where the IP[1]=(20,40) because the difference in observations is greater than zero.

**Definition 6** *A persistent Flow Anomaly, $pFA$, is an upstream contiguous set of $IP$s where a high fraction ($\geq$ persistent threshold $\Theta_p$) are transient flow anomalies. Also, the start $s$ and end $e$ $IP$ in a $pFA$ must be a transient flow anomaly, $tFA$.*

Definition 6 can be formally expressed in Equation 3.

$$pFA[s \dots e] \iff (tFA[s])\ \&\ (tFA[e])\ \&\ \left( \frac{\sum_{i=s}^{e} tFA[i]}{(e-s)+1} \geq \Theta_p \right) \tag{3}$$

Figure 2b gives an example of a $pFA$ pattern when the persistent threshold is 0.6 and the error threshold is 0. The period range of 1 to 3 time intervals of the $up$ sensor is a $pFA$ pattern where the first and ending IPs are transient flow anomalies and the interest measure has a value of 2/3, which satisfies the persistent threshold. The reason that the $up$ time intervals from 1 to 9 is not a $pFA$ is that it does not satisfy the persistence threshold of 0.6 (i.e., five transient flow anomalies by the temporal length of 9 is less than 0.6).

**Definition 7** *A dominant persistent Flow Anomaly, $dpFA$, is a $pFA$ that is not a subset of any other $dpFA$.*

Figure 2b gives an example of a $dpFA$ pattern as shown by the $up$ sensor time intervals from 1 to 3 and 6 to 9. The time interval for 6 to 9 is dominant because there are smaller $pFAs$ such as 8 to 9 and it is not a subset of the other $dpFA$ from 1 to 3.

## 2.2 Problem Statement

The FA discovery problem can be formally expressed as follows:

**Given:**

- An upstream, $up$, and a downstream, $down$ sensor
- Direction of flow between the $up$ and $down$ sensor
- An upstream contiguous set of Instant Pairs, $IP$ at time intervals $t = 1...n$ where $n$ is the length of the time series for $up$ sensor
- The travel time, $TT[t]$, between the $up$ and $down$ sensors at every $t$
- An error threshold, $\Theta_e$
- A persistent threshold, $\Theta_p$

**Find:** All dominant persistent Flow Anomalies (dpFAs).
**Objective:** Minimize the computational costs.
**Constraints:** A single directional flow between sensors and a single one to one match between observation upstream and downstream.

**Example.** Figure 2a gives an example of an input time series from the $up$ and $down$ sensors where the travel time is the temporal length when an observation is expected to be seen once at each sensors. Figure 2b gives an example output of the types of FA patterns when the error threshold is zero and the persistence threshold is 0.6. The dominant persistent Flow Anomalies (dpFA) in this example are the time instants at the $up$ sensor of 1 to 3 and 6 to 9. The dpFA of 1 to 3 satisfies the pFA condition as follows: (1) the ratio of two anomalies and the period length of three is greater than 0.6 and (2) the first and last IP is a tFA. Likewise, the dpGA of 6 to 9 also satisfies the pFA condition as follows: (1) the ratio of three anomalies and the period length of 4 is greater than the persistence threshold of 0.6 and (2) the first and last IP is a tFA. Also, periods 1-3 and 6-9 are not subsets of each other satisfying Definition 7. Even though the period length of 1 to 9 satisfies the second condition where both the first and last IPs are tFAs, the ratio of 5 tFAs and the period length of 9 does not satisfy the persistence threshold. Thus, the period of 1-9 is not a dpFA.

A key computational challenge for this problem is that a dpFA may satisfy $\Theta_p$ but its subsets may not. Thus, no apriori or other pruning strategies can be applied. Also, it may be hard to prune any IPs apriori because they may be part of a larger dpFA. For example, in Figure 2b, there are several IPs within a dpFA pattern that do not qualify to be a tFA (Definition 5), but which are still important for the entire dpFA pattern.

## 3. Mining Flow Anomalies

In this section, we first discuss the Naïve method and then propose a novel Smart Window Enumeration and Evaluation of persistence-Thresholds (SWEET) method to mine dominant persistent flow anomalies.

### 3.1 Naïve Method

**Algorithm 1** Pseudo code for the Naïve Method

**Inputs:**
- $up$ and $down$ time series of $o$ at $t = 1 \ldots n$
- Travel Time, $TT$, of $o$ at $t = 1 \ldots n$
- Error Threshold, $\Theta_e$
- Persistence Threshold, $\Theta_p$

**Outputs:**
- Dominant Persistent Flow Anomalies (dpFA)

**Algorithm**

```
    {Phase I: Identity pFAs}
 1: tFA_count = 0
 2: pFA ← $\emptyset$
 3: for each window size $i = 1 \ldots n$ do
 4:     for each shift of window $j = 1 \ldots n$ do
 5:         for each element $k$ in window of size $i$ do
 6:             if (window exists in both $up$ and $down$) AND
                (diff(up[k],down[k+TT[k]]) $> \Theta_e$) then
 7:                 tFA_count++
 8:             end if
 9:         end for
10:         if (first and last IP is a tFA) AND
            ((tFA_count/i) $\geq \Theta_p$) then
11:             pFA ← period $j$ to $j + i$
12:         end if
13:         tFA_count = 0
14:     end for
15: end for
    {Phase II: Identify Dominant pFAs}
16: dpFA ← $\emptyset$
17: while pFA set size $\neq \emptyset$ do
18:     Identify pFA $c$ with largest temporal length
19:     Remove $c$ from pFA
20:     dpFA ← $c$
21:     Remove any pFA that is a subset of $c$
22: end while
23: return dpFA
```

This section presents the Naïve method to find all the dominant persistent Flow Anomalies (dpFAs). In general, the Naïve method has two main phases, namely, the *identification of the persistent FAs* and *identification of the dominant pFAs*. The main idea of the first phase is to exhaustively search through the entire dataset for every possible size window and determine if each window satisfies the persistence threshold, $\Theta_p$, based on the number of transient Flow Anomalies (tFAs) found within the window using the error threshold, $\Theta_e$. In the second phase, the pFAs having the largest temporal length are kept as the dpFAs, while any other pFA that is a subset of the answers is removed and the final result is returned.

Algorithm 1 presents the pseudo code for the Naïve method. The inputs of the Naïve method includes the following: (1) $up$ and $down$ time series of a single observation $o$ at $t = 1 \ldots n$, where n is the number of observations in each time series, (2) the Travel Time, $TT$, of $o$ at $t = 1 \ldots n$, (3) the error threshold, $\Theta_e$, and (4) the persistent threshold, $\Theta_p$. The output of the Naïve method consists of the dpFAs in terms of the start and end time intervals for the $up$ sensor.

**Phase 1: Identify Persistent FAs**. This phase is concerned with identifying all the possible windows between the $up$ and $down$ time series that satisfy the *pFA* definition (Definition 6). Initially, the counter for the number of tFAs in a window is set to zero and the set pFA is set empty (Lines 1-2 of Algorithm 1). An exhaustive search is completed by examining each window size to the length of the time series and then shifting each window size through the entire series (Lines 3-4 of Algorithm 1). As each window size slides throughout the time series, each instant pair is checked against the error threshold, $\Theta_e$ (Lines 5-6 of Algorithm 1). If a tFA is found, then the counter (tFA_counter) in this window is incremented (Line 7 of Algorithm 1). Once each instant pair in the window of size $i$ is checked for a transient flow anomaly, then the method determines whether the window satisfies the *pFA* definition and if it does, the period is added to the *pFA* set (Lines 10-12 of Algorithm 1). This process continues until all possible windows have been analyzed.

**Phase II: Identify Dominant pFAs**. This phase is concerned with identifying all the dominant pFAs from the *pFA* set found in Phase 1 with the largest temporal length such that it is not a subset of any other *dpFA* (Definition 7). Initially, the *dpFA* set to empty (Line 16 of Algorithm 1). The *pFA* with the largest temporal length is found, removed from the *pFA* set, and added to the *dpFA* set (Lines 18-20 of Algorithm 1). All *pFAs* that are a subset of this *dpFA* answer are removed from the *pFA* set (Line 21 of Algorithm 1). This process continues until there are no more *pFAs* available. Finally, the dpFAs are returned (Line 23 of Algorithm 1).

## 3.2 SWEET Method

This section presents the SWEET (Smart Window Enumeration and Evaluation of persistence-Thresholds) method to find all the dominant persistent Flow Anomalies (dpFA). The SWEET method has two main phases, namely, *identification of the persistent FAs* and *identification of the dominant pFAs*. Unlike the Naïve approach, the main ideas in the first phase of the SWEET method is to initially perform a single scan and examine potential pFAs starting and ending with a transient Flow Anomaly (tFA). The second phase is the same as that of the Naïve approach to find the dpFA patterns. Algorithm 2 presents the pseudo code for the SWEET method. The inputs and outputs of the SWEET method are the same as those of the Naïve approach.

**Phase 1: Identify pFAs**. This phase is concerned with identifying all the possible windows between the $up$ and $down$ time series that satisfy the *pFA* definition (Definition 6). Initially, the number of tFAs in a window (tFA_count) for a period is set to zero, the set *tFA* that contains the time interval at the $up$ sensor when a transient flow anomaly has occurred is set empty, and the *pFA* is set empty.

**Algorithm 2** Pseudo code for the SWEET Method

```
Inputs:
  • $up$ and $down$ time series of $o$ at $t = 1 \dots n$
  • Travel Time, $TT$, of $o$ at $t = 1 \dots n$
  • Error Threshold, $\Theta_e$
  • Persistence Threshold, $\Theta_p$
Outputs:
  • Dominant Persistent Flow Anomalies (dpFA)
Algorithm
   {Phase 1: Identity pFAs}
 1: tFA_count = 0
 2: tFA ← $\emptyset$
 3: pFA ← $\emptyset$
 4: for each $o$ at $i = 1 \dots n$ do
 5:   if (diff(up[i],down[i+TT[i]]) $> \Theta_e$) then
 6:     tFA ← i
 7:     for each tFA, j = 1... tFA.size() do
 8:       for each element $k$ in window $tFA[j]$ to $i$ do
 9:         if diff(up[k],down[k+TT[k]]) $> \Theta_e$ then
10:           tFA_count++
11:         end if
12:       end for
13:       if (tFA_count/i) $\geq \Theta_p$ then
14:         pFA ← period $j$ to $j + i$
15:       end if
16:     end for
17:     tFA_count = 0
18:   end if
19: end for
   {Phase II: Identify Dominant pFAs}
20: dpFA ← $\emptyset$
21: while pFA set size $\neq \emptyset$ do
22:   Identify pFA $c$ with largest temporal length
23:   Remove $c$ from pFA
24:   dpFA ← $c$
25:   Remove any pFA that is a subset of $c$
26: end while
27: return dpFA
```

Unlike for the Naïve method, only a single scan of the entire time series is performed (Line 4 of Algorithm 2). The time series is scanned until a tFA is found for an Instant Pair (IP) and then the time interval at the $up$ sensor is added to the *tFA* set (Lines 5-6 of Algorithm 2). Once the tFA is found, a scan from the first tFA in the *tFA* set to the current tFA is checked for any additional anomalies (Lines 7-12 of Algorithm 2). Then, this period is added to the pFA if it satisfies the persistence threshold, $\Theta_p$ (Lines 13-15 of Algorithm 2). If it does not satisfy, any remaining tFAs are checked with the current tFA for any pFAs. This process continues until no more tFAs are found.

**Phase II: Identify Dominant pFAs**. This phase is the same as for the Naïve method, that is, to identify all the dominant FAs (dFAs) from the *pFA* set found in Phase 1 with the longest temporal length such that it is not a subset of any other dFAs (Definition 7). Initially, *dFA* is set to empty (Line 20 of Algorithm 2). The *pFA* with the largest temporal length is found, removed from the *pFA* set, and added to the *dpFA* set (Lines 22-24 of Algorithm 2). All

**Table 1. Execution Trace for SWEET**

| tFA | Period | Int. Meas. | Satisfy? | pFA |
|---|---|---|---|---|
| 1 | 1-1 | 1/1 | YES | 1-1 |
| 3 | 1-3 | 2/3 | YES | 1-3 |
| 6 | 1-6 | 3/6 | NO | |
| | 4-6 | 2/4 | NO | |
| | 6-6 | 1/1 | YES | 6-6 |
| 8 | 1-8 | 4/8 | NO | |
| | 3-8 | 3/6 | NO | |
| | 6-8 | 2/3 | YES | 6-8 |
| 9 | 1-9 | 4/9 | NO | |
| | 3-9 | 4/7 | NO | |
| | 6-9 | 3/4 | YES | 6-9 |

pFAs that are a subset of this *dpFA* answer are removed from the *pFA* set (Line 25 of Algorithm 2). This process continues until there are no more pFA patterns available. Finally, the dpFAs are returned (Line 27 of Algorithm 2).

**Design Decisions**. Several additional design decisions can be applied to the SWEET approach to improve the overall execution time. First, a **smart counter** can be added after a tFA has been found to keep track of the total number of transient flow anomalies in the time series (Line 6 of Algorithm 2). This smart counter will allow the method to identify the number of transient flow anomalies between the first tFA in the *tFA* set and the current tFA without scanning the window and determine if the window satisfies the persistence threshold. This smart counter will remove the re-scan of the window between the two tFAs (Line 8 of Algorithm 2).

The second design decision is a **pruning** strategy that can be applied as soon as a pFA candidate has been found (Line 14 of Algorithm 2). If a larger period has been found to satisfy the pFA definition, then any other periods that are a subset of the larger period do not need to be checked. As will be shown in the experimental evaluation in Section 5, this pruning strategy results, on average, in far fewer *pFA* patterns than the Naïve method.

**Execution Trace.** Table 1 gives the execution trace of the SWEET algorithm as it is applied to the dataset in Figure 2 when the error threshold, $\Theta_e = 0$ and the persistent threshold, $\Theta_p = 0.60$. Although our SWEET method can handle variable travel times, for simplicity, this example uses a constant time for *TT*. Initially, the SWEET method will start scanning the $up$ and $down$ time series until it finds an instant pair that has a tFA. The first tFA is at time interval 1 for the $up$ sensor where the $down$ sensor reads an observation value of 40 based on the travel time of 1. Since this is the first tFA, the period of 1-1 (both values refer to the time intervals of only the $up$ sensor) is added as one of the pFA patterns and the smart counter will be incremented. The SWEET method will continue scanning the two time series until the next tFA at time interval 3 is found and increments the smart counter. Based on the previously found tFAs and the smart counter, the period 1-3 is analyzed to see if it satisfies $\Theta_p$. The interest measure obtains a value of 2/3 and is greater than 0.60 and adds it to pFA. It is important to note that due to the pruning strategy incorporated into the SWEET method, there is no reason to add period 3-3 will be removed anyway in the second phase, resulting in one less pFA than the Naïve approach.

The next flow anomaly is found at time interval 6 where $diff(up[6], down[6+1]) = diff(20, 40) < \Theta_e$ and increments the smart counter. The period of the largest possible among the previous found tFAs is examined (period 1-6). Based on the smart counter, there are 3 tFAs in this period. This period having an interest measure of 3/6 does not satisfy the $\Theta_p$ of 0.6. Then the next smaller period of 4-6 is analyzed which again does not satisfy the pFA definition. Finally, 6-6 is reached and adds this to the pFAs. The fourth tFA is at time interval 8 and starts checking the largest period first of 1-8. This period has an interest measure of 4/8, which is less than $\Theta_p$. This process continues until its finds the period 6-8 which is successful and adds it to the pFA. Finally, the last anomaly is found at time interval 9, which again starts with the periods 1-9 and 3-9, and is finally successful at period 6-9. There is no need to go any further because any smaller period that may satisfy the pFA definition will be removed in the second phase since 6-9 will always be larger.

The second phase of the SWEET algorithm finds the dominant pFA (dpFA) patterns. First, pattern 6-9 is found having a temporal length of 4 and any other pFA that is a subset of 6-9 is removed. The removed pFAs are 6-8 and 6-6. The next largest pFA is 1-3, which causes 1-1 to be removed. Thus, the dpFA answers are 1-3 and 6-9.

## 4. Analytical Evaluation

In this section, we present the analytical evaluation of the SWEET method and prove that: (1) SWEET is correct, i.e., each pattern is dominant and satisfies the dpFA definition, (2) SWEET is complete, i.e., all patterns satisfying the dpFA definition are found, and (3) SWEET has lower asymptotic computational costs than the Naïve method.

**Theorem 1** *SWEET is correct if the pattern $p$ satisfies the dpFA definition and the temporal length is dominant.*

**Proof** A pattern $p$ is dominant if it first satisfies the two conditions in the pFA definition (Definition 6). The first condition is that the first and last Instant Pairs (IPs) are a transient Flow Anomaly (tFA) based on the error threshold, $\Theta_e$. In the SWEET method, this condition will always be handled since only periods starting and ending with a transient flow anomaly will be analyzed (Line 9 in Algorithm 2). The second condition is that the period must satisfy the persistence threshold, $\Theta_p$. Each period having a first and last IP that is a tFA will be checked within the SWEET algorithm before it is added to the *pFA* set (Lines 13-15 of Algorithm 2). Thus, only patterns satisfying the pFA definition will be found in Phase 1 of the SWEET method. The dpFA will be found in Phase 2 when the pattern with the largest temporal length is found. Since a pattern $p$ will not be a subset of any other pattern, dominant pFA patterns will be found in the SWEET approach (Lines 20-26 of Algorithm 2). □

**Theorem 2** *SWEET is complete if all dominant pFA patterns are found.*

**Proof** In the first phase, only periods where the first and last instant pairs have a tFA are checked against $\Theta_p$ (Lines 4-19 in Algorithm 2). In the pruning design decision, the number of tFAs is monotonic as the size of a period decreases. This is because as the temporal length decreases to the next possible period, the original number of tFAs is reduced by one using the smart counter design decision. Based on this property, once a pFA pattern is found, smaller pFA patterns that are a subset of the larger one are pruned and do not need to be analyzed because they would be removed anyway in the second phase. Thus, no dpFA will be missed in Phase I of the SWEET method. In the second phase, only the pFA patterns with the largest temporal length among the remaining pFAs will be found as the dpFA answers. Thus, no dpFA patterns will be missed in phase 2 in SWEET. □

**Theorem 3** *SWEET has lower asymptotic computational costs than the Naïve method.*

**Proof** Let $n$ be the number of time instances for each upstream and downstream sensor, $m$ be the number of transient flow anomalies, $I$ be the average temporal size of candidate time-intervals where $m \leq n$ and $1 \leq I \leq n$. The worst case complexity of each approach can be characterized by O(number of windows evaluated * average cost of evaluating each window). In the the Naïve approach, it exhaustively searches through all possible window sizes having the complexity of $n^2$ and the cost of evaluating each window is $I$. Then, the worst case complexity for the Naïve approach is $O(n^2 * I)$. In the SWEET approach, the number of windows evaluated is based on the number of transitive flow anomalies having the complexity of $m * m$ and because a smart counter is used, the evaluation of each candidate is a 1 time cost. Thus, the complexity of the SWEET approach is $O(m^2 * 1) = O(m^2)$. The pruning strategy will also reduce the number of pFAs by identifying only the largest qualifying windows. Thus, even though phase 2 of both approaches is the same, the pruning technique may have fewer pFAs to examine than in the Naïve case. The experimental evaluation shows that, on average, there are fewer pFAs

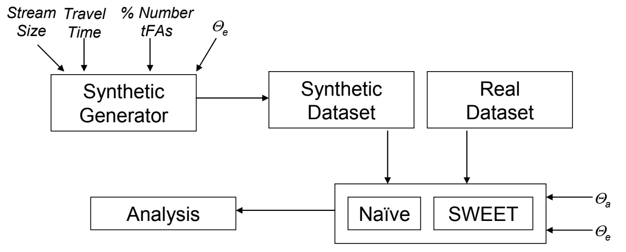


**Figure 3. Experimental Setup**

found in the SWEET approach using the pruning technique than with the Naïve method. □

## 5. Experimental Evaluation

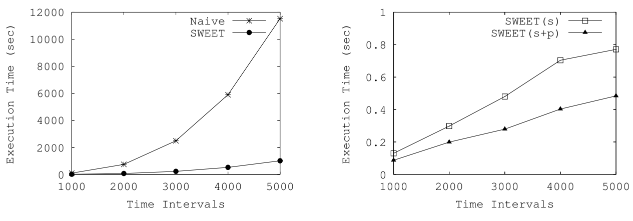
In this section, we present our experimental evaluations of several design decisions and workload parameters for the proposed SWEET method. We evaluated the Naïve approach, the SWEET approach, SWEET using the smart counter (denoted as "s"), and SWEET using both the smart counter and pruning technique (denoted as "s+p") by varying the number of time intervals and the persistent threshold using both synthetic and real datasets. Figure 3 shows the experimental setup to compare the Naïve method and SWEET method along with its design decisions. The synthetic generator takes four inputs (stream size, TT, % number of tFAs, and $\Theta_e$) to create the synthetic datasets (see Section 5.1). The Naïve and the SWEET approach and its design decisions was analyzed using the generated synthetic dataset and two real datasets of measurable variables (turbidity and specific conductivity). All approaches were compared in terms of execution time and the number of pFAs found and results presented in plots with experimental values. All experiments were performed on an Intel P4 2 GHz 1.2 GB RAM.

### 5.1 Experiments using Synthetic Data

The synthetic dataset was generated based on the following: (1) the size of the time series for both $up$ and $down$ sensors, (2) the travel time, (3) the percent number of transient flow anomalies (tFA), and (4) the error threshold, $\Theta_e$. Based on these parameters, the generator creates a single time series of equal length that was randomly generated and used for each sensor. The observations in the $down$ time series were shifted by the specified $TT$. The location of each tFA was chosen randomly and ensured that there will be exactly the percent number of anomalies specified in the input.

#### 5.1.1 Effect on the Size of Time Intervals

The parameters used by the synthetic generator in this experiment are as follows: (1) the size from 1000 to 5000,



(a) Naïve vs. SWEET (b) SWEET(s) vs. SWEET(s+p)

**Figure 4. Synthetic: Execution Time**

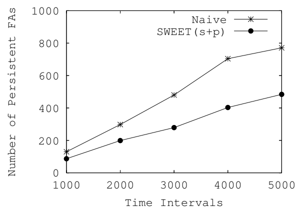
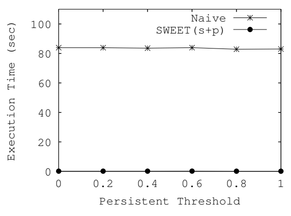


**Figure 5. Synthetic: Number of Persistent FA**

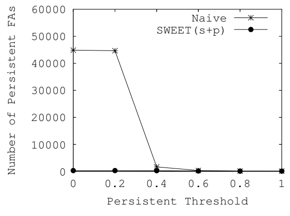
(2) TT=10, (3) 30% or 300 anomalies, and (4) $\Theta_e = 10$. The same TT and $\Theta_e$ was used in this experiment. Figure 4 gives the execution times of all four methods: Naïve, SWEET, SWEET using the smart counter (SWEET(s)), and SWEET using the smart counter and pruning technique (SWEET(s+p)), as the number of time intervals increase. Figure 4a shows that the Naïve approach performs more poorly than the SWEET approach due to the exhaustive search required to check for all possible period lengths. The SWEET(s) method (Figure 4b) results in a significant reduction in execution time by not having to check each period again as in the SWEET method (Figure 4a). Also, in Figure 4b, the SWEET(s+p) outperforms all of the methods. The main reason is due to the removal of pFAs that are subsets of larger pFAs before the dominance can be completed in phase 2. This is evident by the significant decrease in the number of pFAs (Figure 5).

#### 5.1.2 Effect on the Size of the Persistent Threshold

The parameters used by the synthetic generator in this experiment are as follows: (1) size of 1000, (2) TT=10, (3) 30% or 300 anomalies, and (4) varied from 0 to 1 at 0.2 increments. The same size and TT was used in the experiment. The size of 1000 was chosen because it was the most competitive between Naïve and SWEET methods. Figure 6 gives the comparison between the Naïve approach and the SWEET(s+p) method in terms of the execution time (Figure 6a) and the number of pFAs as the $\Theta_p$ varies from 0 to 1 (Figure 6b). SWEET (s+p) is the only proposed method compared against the Naïve approach because it is the only one using any pruning strategy. It is expected that as the persistent threshold increases, fewer patterns will satisfy the persistent threshold. However, the Naïve approach still needs to perform an exhaustive search of the whole dataset.



(a) Execution Time



(b) Number of pFAs

| | Execution Time (sec) | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 83.9 | 0.1 |
| 0.2 | 83.8 | 0.1 |
| 0.4 | 83.5 | 0.1 |
| 0.6 | 83.9 | 0.2 |
| 0.8 | 82.8 | 0.1 |
| 1 | 83.0 | 0.1 |

(c) Exp. Data for (a)

| | Number of pFAs | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 44850 | 300 |
| 0.2 | 44666 | 300 |
| 0.4 | 1665 | 257 |
| 0.6 | 325 | 160 |
| 0.8 | 130 | 88 |
| 1 | 107 | 86 |

(d) Exp. Data for (b)

**Figure 6. Synthetic: Varying $\Theta_p$**

Whereas, the SWEET (s+p) will always scan the time series once and check for periods as tFAs are identified. Thus, the execution time for SWEET is much better than for Naïve (Figure 6a).

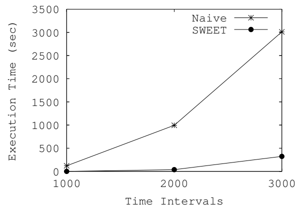
Figure 6b shows a significant drop around the persistent threshold of 0.30. This is due to the fact that the input parameter of the synthetic generator for the percent number of tFAs is also set to 0.30. Thus, when the threshold is below 0.30, the Naïve method will take into account all of the tFAs within the dataset. In this experiment, the temporal length is 1000 time intervals and there are exactly 300 tFAs. Based on the pFA definition and an persistent threshold of zero, there will be 300 choose 2 combinations, or 44,850 pFAs for the Naïve method (Figure 6b). The significant drop in the number of generated pFAs may influence the execution time in the second phase of both methods, but the main costs are due to the exhaustive search by Naïve.

## 5.2 Experiments Using Real Data

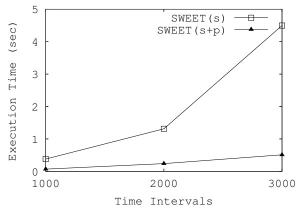
The real datasets were obtained from the study site shown in Figure 1b between sensors 5 and 1. Two datasets representing different measurable variables were used in the experiments, turbidity (approx. 3000 time intervals) and dissolved oxygen (approx. 5000 time intervals). The travel time was taken from the real dataset using discharge and depth measured at sensor 5, the width of Shingle Creek, and the length between sensors 5 and 1.

### 5.2.1 Effect on the Size of Time Intervals

Figure 7 gives the execution time for the real dataset using turbidity. As the time intervals increase, Naïve is more expensive than SWEET. Since both methods use the same phase 2 method and no pFAs are pruned, this experiment gives an accurate comparison to search for pFA patterns.

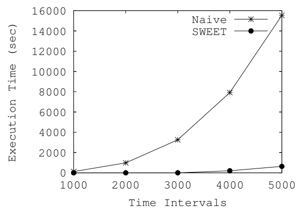


(a) Naïve vs. SWEET

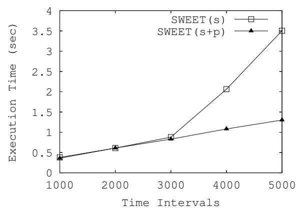


(b) SWEET(s) vs. SWEET(s+p)

**Figure 7. Turbidity: Execution Time**



(a) Naïve vs. SWEET



(b) SWEET(s) vs. SWEET(s+p)

| | Execution Time (sec) | | | |
|---|---|---|---|---|
| Size | Naïve | FA | FA(s) | FA(s+p) |
| 1000 | 124.7 | 0.4 | 0.38 | 0.36 |
| 2000 | 982.3 | 0.6 | 0.611 | 0.611 |
| 3000 | 3258.2 | 0.8 | 0.881 | 0.831 |
| 4000 | 7917.8 | 1.1 | 2.063 | 1.082 |
| 5000 | 15531.8 | 1.3 | 3.505 | 1.302 |

(c) Experimental Values

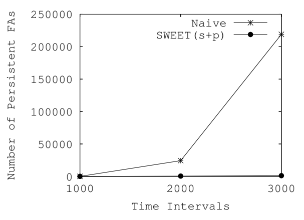
**Figure 8. Dissolved Oxygen: Execution Time**

Figure 7b gives the execution time between SWEET(s) and SWEET(s+p). Since the number of pFAs for SWEET(s+p) is lower than SWEET(s) (Figure 9a), the performance of SWEET(s+p) is much better than SWEET(s) (Figure 7b).

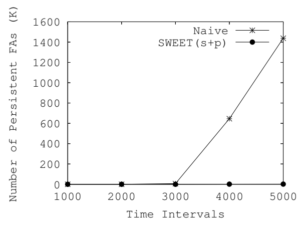
Figure 8 gives the execution time for the real dataset using dissolved oxygen as the number of Time Intervals increase. In Figure 8a, SWEET performs better than Naïve due to the single scan and checking for periods when a tFA has occurred. Figure 8b shows that the execution time for both methods is similar up to the 3000th time interval because the number of pFAs for both approaches is close to zero (Figure 9b). Thus, the pruning strategy is not effective due to very few number of pFAs. As the number of temporal intervals increases from 3000 to 5000, a decrease in execution time occurs for the pruning technique (Figure 8b) as it is evident in the number of pFAs found (Figure 9b).

### 5.2.2 Effect on the Size of the Persistent Threshold

Figure 10 shows the comparison between Naïve and SWEET(s+p) as the $\Theta_p$ is varied for turbidity. As $\Theta_p$ increases, the number of pFAs and the execution times are reduced in both approaches in the second phase. However, this will not affect phase 1 of Naïve due to its exhaustive search of every window size whereas SWEET(s+p) is dependent on the number of tFAs in the dataset when the pe-



(a) Turbidity



(b) Dissolved Oxygen

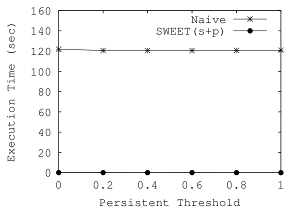
| | Number of pFAs | |
|---|---|---|
| Size | Naïve | SWEET(s+p) |
| 1000 | 10 | 5 |
| 2000 | 24235 | 447 |
| 3000 | 218913 | 1076 |

(c) Exp. Data for (a)

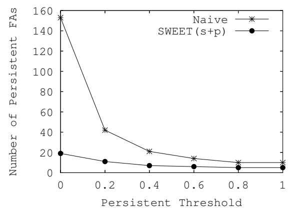
| | Number of pFAs (K) | |
|---|---|---|
| Size | Naïve | SWEET(s+p) |
| 1000 | 0.0 | 0.0 |
| 2000 | 0.0 | 0.0 |
| 3000 | 6.5 | 0.2 |
| 4000 | 645.5 | 1.2 |
| 5000 | 1438.0 | 1.7 |

(d) Exp. Data for (b)

**Figure 9. Real: Number of pFAs**



(a) Execution Time



(b) Number of pFAs

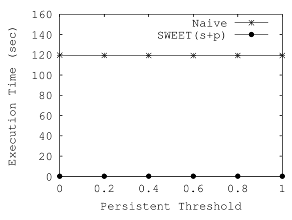
| | Execution Time (sec) | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 121.9 | 0.1 |
| 0.2 | 120.6 | 0.1 |
| 0.4 | 120.5 | 0.1 |
| 0.6 | 120.5 | 0.1 |
| 0.8 | 120.7 | 0.1 |
| 1 | 120.9 | 0.0 |

(c) Exp. Data for (a)

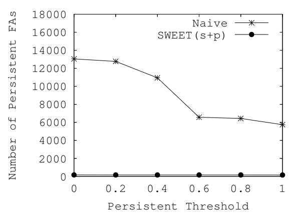
| | Number of pFAs | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 153 | 19 |
| 0.2 | 42 | 11 |
| 0.4 | 21 | 7 |
| 0.6 | 14 | 6 |
| 0.8 | 10 | 5 |
| 1 | 10 | 5 |

(d) Exp. Data for (b)

**Figure 10. Turbidity: Varying $\Theta_p$**



(a) Execution Time



(b) Number of pFAs

| | Execution Time (sec) | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 119.6 | 0.1 |
| 0.2 | 119.4 | 0.1 |
| 0.4 | 119.2 | 0.2 |
| 0.6 | 119.2 | 0.2 |
| 0.8 | 119.3 | 0.2 |
| 1 | 119.3 | 0.1 |

(c) Exp. Data for (a)

| | Number of pFAs | |
|---|---|---|
| $\Theta_p$ | Naïve | SWEET(s+p) |
| 0 | 13041 | 161 |
| 0.2 | 12762 | 160 |
| 0.4 | 10940 | 160 |
| 0.6 | 6562 | 159 |
| 0.8 | 6429 | 158 |
| 1 | 5741 | 155 |

(d) Exp. Data for (b)

**Figure 11. Dissolved Oxygen: Varying $\Theta_p$**

riods are validated against the pFA definition.

Figure 11 gives the comparison between Naïve and SWEET(s+p) as the persistent threshold is varied for dissolved oxygen. Figure 11a gives the execution time of Naïve and performs worse than SWEET(s+p) due to its exhaustive search. Figure 11b gives the number of pFAs and shows that the number of pFAs for SWEET(s+p) is less than for Naïve due to its pruning strategy.

## 6. Conclusion and Future Work

In this paper, we have introduced a novel problem of finding all dominant persistent Flow Anomalies. This problem has applications for environmental monitoring networks to aid environmentalists in their search for clues on how potential contaminants may enter a river network. Several new concepts and interest measures are introduced. We characterized the computational structure of the FA discovery problem and proposed a novel and computationally efficient SWEET (Smart Window Enumeration and Evaluation of persistent-Thresholds) algorithm by restricting the start and end points of a window to coincide with persistent FAs, using a smart counter and efficient pruning techniques. The proof of correctness and completeness is shown. Finally, experimental evaluation was performed on both synthetic and real datasets.

We plan to explore additional real datasets using nitrate and specific conductivity. Further studies may be needed to discover the path of a FA across more than two sensors. Also, multiple sensors would create further challenges such as multiple inputs and outputs to discover FAs. Finally, the effect of FAs for one-to-many and many-to-many matchings will be explored.

## 7. Acknowledgements

This work has been supported by NSF IGERT, NSF IIS-CXT, NSF grant EAR-0607138, USDOD, CUASHI grant EAR-0326064, and the University of Minnesota. We would like to thank Kim Koffolt for her comments.

## References

[1] A. Amir, A. Apostolico, and M. Lewenstein. Inverse pattern matching. *Journal of Algorithms*, 24(2):325–339, 1997.

[2] A. Bulut and A. K. Singh. A unified framework for monitoring data streams in real time. In *ICDE*, pages 44–75, 2005.

[3] A. Chen, C. Tang, C. an Yuan, J. Peng, and J. Hu. Mining Correlations Between Multi-streams Based on Haar Wavelet. In *ASIAN*, pages 270–271, 2005.

[4] M. Datar and S. Muthukrishnan. Estimating Rarity and Similarity over Data Stream Windows. In *ESA*, pages 323–334, 2002.

[5] M. DeGroot and M. J. Scheverish. *Probability and Statistics, 3rd. Edition*. Addison Wesley, 2002.

[6] EPA. Drinking water contaminents, 2008, http://www.epa.gov/safewater/contaminants/index.html.

[7] R. Galbraith. Bay spill, san francisco chronicle, 2007, http://www.sfgate.com/cgi-bin/news/oilspill/busan.

[8] E. Knorr and R. Ng. A Unified Notion of Outliers: Properties & Computation. In *KDD*, 1997.

[9] H. Lee, R. T. Ng, and K. Shim. Estimating Rarity and Similarity over Data Stream Windows. In *VLDB*, pages 195–206, 2007.

[10] F. Lorna. Drinking-water nitrate, methemoglobinemia, and global burden of disease: A discussion. *Environmental health perspectives*, 112(14):1371–1374, 2004.

[11] D. A. Matthews, S. W. Effler, C. T. Driscoll, S. M. O'Donnell, and C. M. Matthews. Electron budgets for the hypolimnion of a recovering urban lake, 1989-2004. *Limnology and Oceanography*, 53(2):743–759, 2008.

[12] S. Saulny. Fish-killing virus spreading in the great lakes, new york times, 2007.

[13] M. Sayal. Detecting time correlations in time-series data streams. *Hewlett-Packard Company*, 2004.

[14] S. Shekhar, C. T. Lu, and P. Zhang. A unified approach to spatial outliers detection. *GeoInformatica*, 7(2):139–166, 2003.

[15] WFUNA. Millenium project: Global challenges facing humanity, 2007.