SCREEN: Stream Data Cleaning under Speed Constraints

Shaoxu Song[§] Aoqian Zhang[§] Jianmin Wang[§] Philip S. Yu[‡]

[§]KLiss, MoE; TNList; School of Software, Tsinghua University, China
{sxsong, jimwang}@tsinghua.edu.cn zaq13@mails.tsinghua.edu.cn

[‡]Department of Computer Science, University of Illinois at Chicago, USA &
Institute for Data Science, Tsinghua University, China psyu@cs.uic.edu

ABSTRACT

Stream data are often dirty, for example, owing to unreliable sensor reading, or erroneous extraction of stock prices. Most stream data cleaning approaches employ a smoothing filter, which may seriously alter the data without preserving the original information. We argue that the cleaning should avoid changing those originally correct/clean data, a.k.a. the *minimum change principle* in data cleaning. To capture the knowledge about *what is clean*, we consider the (widely existing) constraints on the speed of data changes, such as fuel consumption per hour, or daily limit of stock prices. Guided by these semantic constraints, in this paper, we propose SCREEN, the first constraint-based approach for cleaning stream data. It is notable that existing data repair techniques clean (a sequence of) data *as a whole* and fail to support stream computation. To this end, we have to relax the global optimum over the entire sequence to the local optimum in a window. Rather than the commonly observed NP-hardness of general data repairing problems, our major contributions include (1) polynomial time algorithm for global optimum, (2) linear time algorithm towards local optimum under an efficient *Median Principle*, (3) support on out-of-order arrivals of data points, and (4) adaptive window size for balancing repair accuracy and efficiency. Experiments on real datasets demonstrate that SCREEN can show significantly higher repair accuracy than the existing approaches such as smoothing.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: General

Keywords

Data repairing; speed constraints

1. INTRODUCTION

Dirty values commonly exist in data streams, e.g., in traditional sensor data due to the unreliable readers [10]. Even in the domains of Stock and Flight where people believed data are reliable, a large amount of inconsistent data are surprisingly observed [15]. According to the study, the accuracy of Stock in *Yahoo! Finance* is 0.93, and the Flight data accuracy in *Travelocity* is 0.95. Reasons for imprecise values include ambiguity in information extraction, unit error or pure mistake. For instance, the price of SALVEPAR (SY) is misused as the price of SYBASE, which is denoted by SY as well in some sources. (See more examples of data errors below.) Such inaccurate values, e.g., taken as the 52-week low price, may seriously mislead business investment.

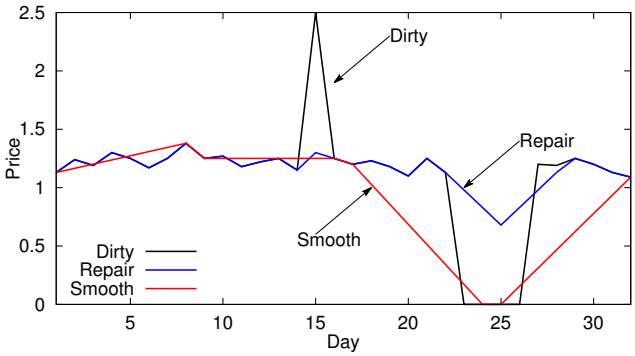

Figure 1: Smoothing filter seriously alters the original correct data, while the minimum repair under speed constraints aim to preserve the original information as much as possible

A temporal smoothing filter, e.g., via segmentation in sliding windows [13], may modify almost all the data values, most of which are originally correct/clean. It thus seriously damages the precision of individual data points (such as daily stock prices). Indeed, in order to preserve the original clean information as much as possible, the *minimum change principle* is widely considered in improving data quality [1].

To capture the knowledge about *what is clean*, we notice that the "jump" of values in a stream is often constrained, so called *speed constraints*. In financial and commodity markets, prices are only permitted to rise or fall by a certain number of ticks per trading session. In environment monitoring, the temperature difference of any two days in a week should not be greater than 20 degrees. The fuel consumption of a crane should not be negative and not exceed 40 liters per hour. We believe that with these meaningful constraints on value change speed, cleaning could be more accurate.

Example 1. *Consider the prices of a stock in 32 trading days, in Figure 1. As illustrated, large spikes appear in the*

SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

http://dx.doi.org/10.1145/2723372.2723730.

dirty data (in black), e.g., in day 15, owing to ambiguity in information extraction as discussed or pure mistake. It may also be raised by temporary loss of data (days from 23 to 26) and the subsequent coding of these missing values as zero by the data collection system.

The smoothing method (in red) modifies almost all the price values, most of which are indeed accurate. Without preserving the original clean price of each day, the modified data values become useless. It is obviously not the best way for cleaning the stream data.

The speed constraints derived from price limit[1] *state that the price difference of two consecutive trading days should not be greater than 0.5. The maximum speed* $s_{\max} = 0.5$ *specifies that the increase amount is no larger than 0.5 in a single trading day from the previous day's settlement price. The minimum speed* $s_{\min} = -0.5$ *indicates that the decrease should be within 0.5.*

With speed constraints, the imprecise value of day 15 can be detected. It obviously increases too much from the price of the previous day 14. As shown, the speed constraint-based repair (proposed in this study) preserves more originally clean price values (in blue).

Challenges. Unlike the existing techniques on smoothing time series [13], we propose to minimally modify the data values such that the declared speed constraints are satisfied. This constraint-based cleaning, however, is non-trivial and challenging especially in the following aspects:

(1) *Soundness.* Owing to the inherent hardness of general data repair problems, a greedy strategy is employed in the existing repair [4]. It modifies values to eliminate currently observed violations (w.r.t. the given constraints) in each round which may introduce new violations to other data points, and thus evokes another round of repairing. In particular, the greedy repair could be trapped in local optima, and thus cannot eliminate all the violations. In other words, the soundness w.r.t. satisfaction of (speed) constraints is not guaranteed. According to our empirical study (see details in Section 6), 40-80% values in the greedily repaired results may still be in violation w.r.t. the speed constraints.

(2) *Online Computing.* Typically, data repair techniques consider a global optimization function on modifying the entire data [1]. It has to first collect all the data, and then repair them as a whole. Online cleaning on the streaming data is not supported. To enable streaming computation, we have to decompose the global optimum into a list of local optimum on each data point, respectively. Integral cleaning can thus be applied by incrementally computing the local optimal repair on every data point of the sequence in turn.

(3) *Out-of-order Arrival.* Network latencies or device failures may cause data to arrive out-of-order [16]. To avoid output blocking, it is necessary to carry on the cleaning with the absence of some data, and update the results later when the delayed data come. The repair on a data point depends not only on the previous data points, but also on the future data points. That is, the arrival of each delayed data point may cause some of the previous data points to be repaired again. How to efficiently determine the data points affected and the amount to be adjusted is the main challenge.

[1]In some markets, the price limit is specified by a certain percentage. See Section 6.4 for obtaining max/min speed from such price limit.

(4) *Throughput.* The speed constraint is often meaningful in a period with certain lengths. For instance, it is meaningless to consider the constraint on temperature of two days in different years. Such a large window size requires to compare more data points w.r.t. speed constraints, and thus may result in huge system latencies and memory resource overflow. On the other hand, with a small window size, while the time efficiency is improved, the power of speed constraints could be limited in repairing (as illustrated in Example 2). Even worse, such trade-off on window sizes may vary with the evolving of the arrival rate (the number of data points arrived in a period). To increase system throughput, it is promising to devise adaptive window sizes that can automatically balance the cleaning accuracy and efficiency.

Contributions. To the best of our knowledge, this is the first study on constraint-based stream data cleaning. The proposed SCREEN (Speed Constraint-based stREam data clEaNing) is a linear time, constant space cleaning approach. Our main contributions are summarized as follows.

(1) We formalize the repair problem under speed constraints (in Section 2). By considering the entire sequence as a whole, the monolithic cleaning finds a repaired sequence that minimally differs from the input. Unlike NP-hardness of general data repair problems [17, 14], we show that stream data cleaning under speed constraints can be modeled as a linear programming problem, i.e., polynomial time solvable.

(2) We devise an online cleaning algorithm (in Section 3). To support integral cleaning (i.e., incrementally repair one data point a time in the sequence rather than monolithic cleaning as a whole), we relax the global optimum over the entire sequence to the local optimum in a window. The main idea is to locally compute a data point repair, which is minimal w.r.t. the upcoming data points in a window and also compatible with the previously repaired data points. In particular, to efficiently compute the local optimum, we propose a novel *Median Principle*, following the intuition that a solution with the minimum distance (i.e., as close as possible to each point) probably lies in the middle of the data points. It is notable that soundness w.r.t. speed constraint satisfaction is guaranteed in the devised algorithm.

(3) We extend the algorithm for out-of-order data arrival (in Section 4). An update of previously repaired results is performed when the delayed data comes. We further reduce the latency by heuristically applying the updates.

(4) We propose a sampling-based, adaptive window (in Section 5). By modeling data points as random samples of approaching the speed constraints, the criteria in distribution approximation can be employed to suggest increasing or reducing the window sizes for acquiring more or less samples.

Finally, experiments on real data demonstrate that our proposed SCREEN achieves significantly higher repair accuracy than the smoothing method [13]. Moreover, compared to the state-of-the-art data repair method [4], SCREEN with local optimum shows up to 4 orders of magnitude improvement in time costs without losing much accuracy.

Table 2 in the Appendix lists the frequently used notations. Proofs of major results can be found in the long version technique report [6].

2. MONOLITHIC CLEANING

First, considering a sequence as a whole, we perform monolithic repair towards the globally minimum repair distance.

2.1 Preliminary

Consider a sequence $x = x[1], x[2], \ldots,$ where each $x[i]$ is the value of the i-th data point. Each $x[i]$ has a timestamp $t[i]$. For brevity, we write $x[i]$ as x_i, and $t[i]$ as t_i.

A *speed constraint* $s = (s_{\min}, s_{\max})$ with window size w is a pair of minimum speed $s_{\min}$ and maximum speed $s_{\max}$ over the sequence x. We say that a sequence x *satisfies* the speed constraint s, denoted by $x \vDash s$, if for any x_i, x_j in a window, i.e., $0 < t_j - t_i \leq w$, it has $s_{\min} \leq \frac{x_j - x_i}{t_j - t_i} \leq s_{\max}$.

The window w denotes a period of time. In real settings, speed constraints are often meaningful within a certain period. For example, it is reasonable to consider the maximum walking speed in hours (rather than the speed between two arbitrary observations in different years), since a person usually cannot keep on walking in his/her maximum speed for several years without a break. In other words, it is sufficient to validate the speed w.r.t. two points x_i, x_j in a window $w =$ 24 hours, i.e., whether $s_{\min} \leq \frac{x_j - x_i}{t_j - t_i} \leq s_{\max}, 0 < t_j - t_i \leq w$. In contrast, considering the speed w.r.t. two points in an extremely large period (e.g., two observation points in different years) is meaningless and unnecessary. Similar examples include the speed constraints on stock price whose daily limit is directly determined by the price of the *last* trading day, i.e., with window size 1.

The speed constraint s can be either positive (restricting value increase) or negative (on decrease). In most scenarios, the speed constraint is natural, e.g., the fuel consumption of a crane should not be negative and not exceed 40 liters per hour, while some others could be derived. (See Section 6.4 for a discussion on obtaining speed constraints.)

A *repair* x' of x is a modification of the values x_i to x'_i where $t'_i = t_i$. Referring to the minimum change principle in data repairing [1], the repair distance is evaluated by the difference between the original x and the repaired x',

$$\Delta(x, x') = \sum_{x_i \in x} |x_i - x'_i|. \tag{1}$$

Example 2 (Speed constraints, violations, and repairs)**.** *Consider a sequence $x = \{12, 12.5, 13, 10, 15, 15.5\}$ of six data points, with timestamps $t = \{1, 2, 3, 5, 7, 8\}$. Figure 2(a) illustrates the data points (in black). Suppose that the speed constraints are $s_{\max} = 0.5$ and $s_{\min} = -0.5$.*

For a window size $w = 2$ in the speed constraints, data points x_3 and x_4, with timestamp distance $5 - 3 \leq 2$ in a window, are identified as violations to $s_{\min} = -0.5$, since the speed is $\frac{10-13}{5-3} = -1.5 < -0.5$. Similarly, x_4 and x_5 with speed $\frac{15-10}{7-5} = 2.5 > 0.5$ are violations to $s_{\max} = 0.5$.

To remedy the violations (denoted by red lines), a repair on x_4 can be performed, i.e., $x'_4 = 14$ (the white data point). As illustrated in Figure 2(a), the repaired sequence satisfies both the maximum and minimum speed constraints. The repair distance is $\Delta(x, x') = |10 - 14| = 4$.

Note that if the window size is too small such as $w = 1$, the violations between x_3 and x_4 (as well as x_4 and x_5) could not be detected, since their timestamp distance is $2 > 1$. On the other hand, if the window size is too large, say $w = 10$, then all the pairs of data points in x have to be compared. Although the same repair x' is obtained, the computation overhead is obviously higher (and unnecessary). Nevertheless, we propose to determine an adaptive window size (in Section 5) for balancing accuracy and efficiency.

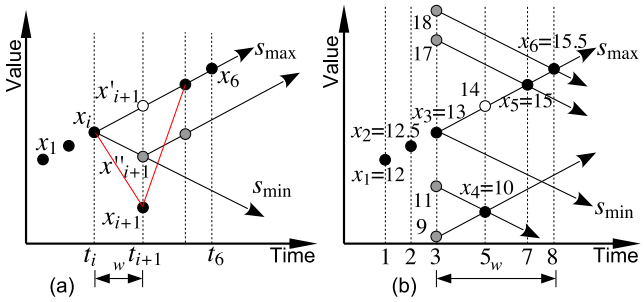

Figure 2: Possible repairs under speed constraints

2.2 Global Optimum

The cleaning problem is to find a repaired sequence that satisfies the speed constraints and minimally differs from the original sequence, called *global optimum*.

Problem 1. *Given a finite sequence x of n data points and a speed constraint s, the* global optimal repair *problem is to find a repair x' such that $x' \vDash s$ and $\Delta(x, x')$ is minimized.*

A broad class of repair problems have been found to be NP-hard, for instance, repairing under functional dependencies for categorized data [14], or repairing under denial constraints that supports numeric data [17]. It is not the case for repairing under speed constraints.

We write the global optimal repair problem as

$$\min \quad \sum_{i=1}^{n} |x_i - x'_i| \tag{2}$$

$$\text{s.t.} \quad \frac{x'_j - x'_i}{t_j - t_i} \leq s_{\max}, \qquad t_i < t_j \leq t_i + w, \quad 1 \leq i \leq n, 1 \leq j \leq n \tag{3}$$

$$\frac{x'_j - x'_i}{t_j - t_i} \geq s_{\min}, \qquad t_i < t_j \leq t_i + w, \quad 1 \leq i \leq n, 1 \leq j \leq n \tag{4}$$

where $x'_i, 1 \leq i \leq n$, are variables in problem solving.

The correctness of result x' in the aforesaid problem is obvious. Formula (2) is exactly the repair distance in formula (1) to minimize. The speed constraints are ensured in formulas (3) and (4), by considering all the t_j in the window starting from t_i, for each data point i in the sequence.

By transforming the problem to a linear programming (LP) problem, existing solvers can directly be employed. (See Appendix A for transformation details.)

3. INTEGRAL CLEANING

The global optimum considers the entire sequence as a whole, and does not support online cleaning over streaming data. To support integral repair w.r.t. the current short period in a stream, we study the local optimum, which concerns only the constraints locally in a window. By sliding windows in the sequence, the result of local optimum x_{local} guarantees to satisfy the speed constraints in the entire sequence, i.e., also a feasible solution to the constraints in formulas (3) and (4) of global optimum. Since the global optimum returns a minimum distance repair x_{global} that satisfies all the constraints, we always have $\Delta(x, x_{\text{local}}) \geq \Delta(x, x_{\text{global}})$. (For the upper bound, unfortunately, we don't

find any constant factor.) Referring to the minimum change principle [1] that a repair with lower repair distance is more likely to be the truth, the repair accuracy of local optimum may not be as high as that of global optimum. Although we don't have a theoretical upper bound of local repair distance compared to the global one, in practice, the repair distances of local and global repairs are very close (as shown in Figures 10(b) and 11(b)). Compared to the global optimum, the local optimal approach can show significant improvement in time costs (about 2 order of magnitude improvement in Figure 11(a)) but without losing much repair accuracy (only a small decrease of 0.01 in Figure 11(d)).

3.1 Local Optimum

We say a data point x_k *locally satisfies* the speed constraint s, denoted by $x_k \vDash s$, if for any x_i in the window starting form x_k, i.e., $t_k < t_i \leq t_k + w$, it has $s_{\min} \leq \frac{x_i - x_k}{t_i - t_k} \leq s_{\max}$.

Problem 2. *Given a data point x_k in a sequence x and a speed constraint s, the* local optimal repair *problem is to find a repair x' such that $x'_k \vDash s$ and $\Delta(x, x')$ is minimized.*

Similar to the global optimum, we write the local optimal repair problem as

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} |x_i - x'_i| \qquad (5) \\ \text{s.t.} \quad & \frac{x'_k - x'_i}{t_k - t_i} \leq s_{\max}, \quad t_k < t_i \leq t_k + w, 1 \leq i \leq n \\ & \frac{x'_k - x'_i}{t_k - t_i} \geq s_{\min}, \quad t_k < t_i \leq t_k + w, 1 \leq i \leq n \end{aligned}$$

where $x'_i, 1 \leq i \leq n$ are variables in problem solving.

The local optimal repair in formula (5) modifies only the data points i with $t_k \leq t_i \leq t_k + w$ in the window of the current x_k, i.e., much fewer variables. The constraints (in the window) are not sacrificed.

Example 3 (Local optimum)**.** *Consider again the sequence $x = \{12, 12.5, 13, 10, 15, 15.5\}$ in Example 2 and the speed constraints $s_{\max} = 0.5$ and $s_{\min} = -0.5$ with window size $w = 5$, as illustrated in Figure 2(b).*

Let $k = 3$ be the currently considered data point. Referring to formula (5), the constraint predicates for the local optimum on $k = 3$ are:

$$\begin{aligned} &\frac{x'_4 - x'_3}{5-3} \leq 0.5, \quad \frac{x'_5 - x'_3}{7-3} \leq 0.5, \quad \frac{x'_6 - x'_3}{8-3} \leq 0.5, \\ &\frac{x'_4 - x'_3}{5-3} \geq -0.5, \quad \frac{x'_5 - x'_3}{7-3} \geq -0.5, \quad \frac{x'_6 - x'_3}{8-3} \geq -0.5. \end{aligned}$$

The local optimal solution with the minimum distance is $x'_3 = 13, x'_4 = 14, x'_5 = 15, x'_6 = 15.5$. That is, $x'_3 = x_3 = 13$ is not necessary to be modified w.r.t. the local optimum on $k = 3$.

3.2 The Median Principle

Intuitively, a solution with the minimum distance (i.e., as close as possible to each point) probably lies in the *middle* of the data points. We propose to efficiently search the local optimum in the scope of such middle data points, namely the **Median Principle** (in Proposition 3). Following this median principle, we devise a linear time algorithm for computing the local optimal repair, instead of $O(n^{3.5}L)$ by LP.

Before presenting the median principle, let us first show that computing the local optimum w.r.t. x_k is indeed equivalent to determine an optimal repair x'_k, where the solution of other x'_i (in formula (5)) can be naturally derived.

3.2.1 Reformulating the Local Optimum Problem

We transform the local optimal repair problem in formula (5) to a new form w.r.t. only one variable x'_k. The idea is to illustrate that there always exists an optimal solution x', whose x'_i can be derived from x'_k.

Proposition 1. *Let x^* be a local optimal solution w.r.t. x_k. The following x' is also local optimal, with $x'_k = x^*_k$ and*

$$x'_i = \begin{cases} x'_k + s_{\max}(t_i - t_k) & \text{if } \frac{x'_k - x_i}{t_k - t_i} > s_{\max} \\ x'_k + s_{\min}(t_i - t_k) & \text{if } \frac{x'_k - x_i}{t_k - t_i} < s_{\min} \\ x_i & \text{otherwise} \end{cases} \qquad (6)$$

where $t_k < t_i \leq t_k + w, 1 \leq i \leq n$.

Formula (6) constructs an optimal solution x' upon x^*_k, where either no change or border change w.r.t. $s_{\max}$ and $s_{\min}$ needs to be made. By border changes, we mean $\frac{x'_k - x'_i}{t_k - t_i} = s_{\max}$ or $\frac{x'_k - x'_i}{t_k - t_i} = s_{\min}$. Intuitively, as illustrated in Figure 3, all the values in the range of $[x'_k + s_{\min}(t_i - t_k), x'_k + s_{\max}(t_i - t_k)]$ are valid repair candidate for x'_i. If the speed exceeds $s_{\max}$, a repair on the "border" drawn by $s_{\max}$ is obviously the closest to x_i, i.e., with the minimum repair distance.

We denote $g(x_i, x'_k) = |x_i - x'_i| =$

$$\begin{cases} x'_k - x_i - s_{\max}(t_k - t_i) & \text{if } \frac{x'_k - x_i}{t_k - t_i} > s_{\max} \\ x_i - x'_k - s_{\min}(t_k - t_i) & \text{if } \frac{x'_k - x_i}{t_k - t_i} < s_{\min} \\ 0 & \text{otherwise} \end{cases}$$

for $t_k < t_i \leq t_k + w, 1 \leq i \leq n$. The local optimal repair problem in formula (5) can be rewritten as

$$\min_{x'_k} \sum_{i=1}^{n} g(x_i, x'_k), \qquad (7)$$

where x'_k is the only variable in problem solving.

3.2.2 Median Solution in a Finite Set of Candidates

According to Proposition 1, the local optimal repair problem is equivalent to find a x'_k that can minimize formula (7). To this end, we first capture a finite set of candidates for x'_k, where the optimal solution can always be found. Let

$$\begin{aligned} X_k^{\min} &= \{x_i + s_{\min}(t_k - t_i) \mid t_k < t_i \leq t_k + w, 1 \leq i \leq n\}, \\ X_k^{\max} &= \{x_i + s_{\max}(t_k - t_i) \mid t_k < t_i \leq t_k + w, 1 \leq i \leq n\}. \end{aligned}$$

Intuitively, as shown in Figure 4, each candidate in $X_k^{\max}$ corresponds to a possible x'_k such that x_i serves as a border repair w.r.t. x'_k (as presented in Figure 3). Referring to the aforesaid discussion on minimum distances of border repairs, it is not surprising to have the following conclusion.

Lemma 2. *We can always find a local optimal solution x^* w.r.t. x_k such that $x^*_k \in X_k^{\min} \cup X_k^{\max} \cup \{x_k\}$.*

Let $m = |\{i \mid t_k < t_i \leq t_k + w, 1 \leq i \leq n\}|$ be the number of data points in the window starting from k. It is easy to see $2m + 1$ candidates in $X_k^{\min} \cup X_k^{\max} \cup \{x_k\}$.

For any x'_k, the construction of solution x' is indeed to "shrink" data points in violations to the border. **Intuitively,**

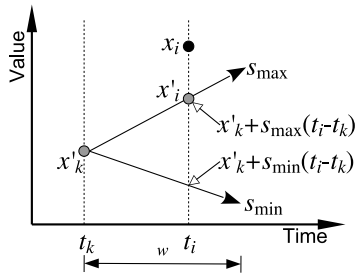

Figure 3: Build solution from x'_k

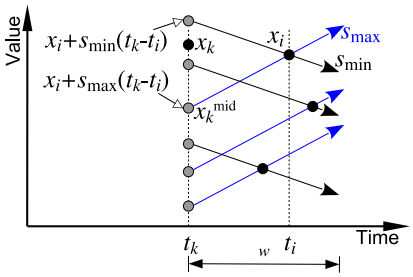

Figure 4: Capture candidates for x'_k

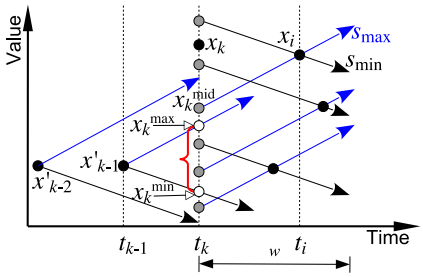

Figure 5: Compute x'_k in integral repair

a candidate in the middle of all data points x_i probably has less shrink distances.

Let x_k^{mid} denote the median of all candidates,

$$x_k^{\text{mid}} = \text{median}(X_k^{\min} \cup X_k^{\max} \cup \{x_k\}). \tag{8}$$

The following shows that the median x_k^{mid} is exactly the optimal solution to the problem in formula (7), and can be used to build the local optimal solution by Proposition 1.

Proposition 3 (The Median Principle)**.** *A solution x' with x'_i determined by formula (6) and $x'_k = x_k^{\text{mid}}$ is local optimal.*

Example 4 (Candidates and local optimum, Example 3 continued)**.** *Consider data points 4, 5 and 6, in Figure 2(b), whose timestamps are within t_3+w, w.r.t. the current $k=3$. Each data point suggests two candidates w.r.t. $s_{\min}$ and $s_{\max}$ for $X_3^{\min}$ and $X_3^{\max}$, respectively. For instance, $x_4 = 10$ contributes $10-0.5(5-3)=9$ in $X_3^{\max}$ and $10+0.5(5-3)=11$ in $X_3^{\min}$. Finally, the candidate sets are*

$$X_3^{\min} = \{11, 17, 18\}, \qquad X_3^{\max} = \{9, 13, 13\}.$$

According to formula (8), we have $x_3^{\text{mid}} = 13$.

Referring to Proposition 3, by $x'_3 = x_3^{\text{mid}}$ and formula (6), we build a solution $x'_3 = 13, x'_4 = 14, x'_5 = 15, x'_6 = 15.5$. It is exactly the local optimal solution in Example 3.

3.3 Streaming Computation

The integral cleaning algorithm is to iteratively determine the local optimal x'_k, for $k = 1, 2, \ldots$. Let us first assume that data points come in-order, i.e., $t_j < t_i$ for any $j < i$. (The handling of out-of-order arrival will be introduced in the next section.)

3.3.1 Candidate Range

Consider x_k, where $x'_1, \ldots, x'_{k-1}$ have been determined in the previous steps. Referring to the speed constraints, each fixed $x'_j, t_k - w \le t_j < t_k, 1 \le j < k$, indicates a range of candidates for x'_k, i.e., $[x'_j + s_{\min}(t_k - t_j), x'_j + s_{\max}(t_k - t_j)]$.

The following proposition states that considering the last x'_{k-1} is sufficient to determine the range of possible repairs for x'_k. The rationale is that for any $1 \le j < i < k$, x'_i should be in the range specified by x'_j as well. In other words, the candidate range of x'_k specified by x'_i is subsumed in the range by x'_j.

Proposition 4. *For any $1 \le j < i < k, t_k - w \le t_j < t_i < t_k$, we have $x'_j + s_{\min}(t_k - t_j) \le x'_i + s_{\min}(t_k - t_i)$, and $x'_i + s_{\max}(t_k - t_i) \le x'_j + s_{\max}(t_k - t_j)$.*

For instance, as illustrated in Figure 5, the candidate range of x'_k specified by x'_{k-2} subsumes that by x'_{k-1}. Consequently, we can obtain a tight range of candidates for x'_k by x'_{k-1}, i.e., $[x_k^{\min}, x_k^{\max}]$ as presented in Figure 5, where

$$\begin{aligned} x_k^{\min} &= x'_{k-1} + s_{\min}(t_k - t_{k-1}), \\ x_k^{\max} &= x'_{k-1} + s_{\max}(t_k - t_{k-1}). \end{aligned} \tag{9}$$

The repair problem thus becomes to finding the local optimum x'_k in the range of $[x_k^{\min}, x_k^{\max}]$.

3.3.2 Optimal Solution in Candidate Range

Formula (8) gives a repair candidate x_k^{mid} suggested by x_i after x_k ($t_k < t_i$), while formula (9) indicates a candidate range $[x_k^{\min}, x_k^{\max}]$ specified by x_{k-1} before x_k.

If the suggested local optimal solution x_k^{mid} in formula (8) drops into the range of $[x_k^{\min}, x_k^{\max}]$ in formula (9), the optimal solution is directly obtained, i.e., $x'_k = x_k^{\text{mid}}$. Otherwise, we need to re-calculate the local optimum w.r.t. the range $[x_k^{\min}, x_k^{\max}]$.

Fortunately, we have the following monotonicity of the function in formula (7).

Proposition 5. *For any $u_1, u_2, v_1, v_2 \in X_k^{\min} \cup X_k^{\max} \cup \{x_k\}$ such that $u_1 \le u_2 \le x_k^{\text{mid}} \le v_1 \le v_2$, we have*

$$\sum_{i=1}^{n} g(x_i, u_1) \ge \sum_{i=1}^{n} g(x_i, u_2) \ge \sum_{i=1}^{n} g(x_i, x_k^{\text{mid}}),$$

$$\sum_{i=1}^{n} g(x_i, x_k^{\text{mid}}) \le \sum_{i=1}^{n} g(x_i, v_1) \le \sum_{i=1}^{n} g(x_i, v_2).$$

That is, for any candidate $u < x_k^{\max} < x_k^{\text{mid}}$, it always has $\sum_{i=1}^{n} g(x_i, u) \ge \sum_{i=1}^{n} g(x_i, x_k^{\max})$. $x_k^{\max}$ is thus the optimal solution in the range of $[x_k^{\min}, x_k^{\max}]$. Similar conclusion can also be made for $v > x_k^{\min} > x_k^{\text{mid}}$.

Consequently, according to Proposition 5, the local optimal solution is directed computed by

$$x'_k = \begin{cases} x_k^{\max} & \text{if } x_k^{\max} < x_k^{\text{mid}} \\ x_k^{\min} & \text{if } x_k^{\min} > x_k^{\text{mid}} \\ x_k^{\text{mid}} & \text{otherwise} \end{cases} \tag{10}$$

Algorithm 1 presents the integral repair of a sequence x w.r.t. local optimum under the speed constraint s. For each data point k in the sequence, $k = 1, 2, \ldots, n$, Lines 3 and 4 computes the candidate range in formula (9). By considering all the succeeding data points i in the window of k, Line 10 calculates x_k^{mid} in formula (8). Finally, x'_k is obtained following the computation in formula (10).

It is easy to see that the number of distinct data points in a window is at most w. The median in the window can be trivially found in $O(w)$, i.e., the average complexity of quickselect [9]. Considering all the n data points in the sequence, Algorithm 1 runs in $O(nw)$ time. For a fixed w, it is

Algorithm 1: Local(x, s)

Data: an ordered sequence x and speed constraints s
Result: a repair x' of x w.r.t. local optimum

```
1  for k ← 1 to n do
2      X_k^min ← ∅; X_k^max ← ∅;
3      x_k^min ← x'_{k-1} + s_min(t_k − t_{k-1}), or −∞ for k = 1;
4      x_k^max ← x'_{k-1} + s_max(t_k − t_{k-1}), or +∞ for k = 1;
5      for i ← k + 1 to n do                    // compute x_k^mid
6          if t_i > t_k + w then
7              break;
8          X_k^min ← X_k^min ∪ {x_i + s_min(t_k − t_i)};
9          X_k^max ← X_k^max ∪ {x_i + s_max(t_k − t_i)};
10     x_k^mid ← median(X_k^min ∪ X_k^max ∪ {x_k});
11     if x_k^max < x_k^mid then                // compute x'_k
12         x'_k ← x_k^max;
13     else if x_k^min > x_k^mid then
14         x'_k ← x_k^min;
15     else
16         x'_k ← x_k^mid;
17 return x'
```

a linear time, constant space algorithm. In practice, to minimize the changes, we may heuristically skip the repairing on those points x_k that satisfy the speed constraints with its neighbors, i.e., $x_k^{\min} \le x_k \le x_k^{\max}$ and $x_{k+1}^{\min} \le x_{k+1} \le x_{k+1}^{\max}$.

Example 5 (Example 4 continued)**.** *Consider the next data point $k = 4$ in the current sequence $\{12, 12.5, 13, 10, 15, 15.5\}$, in Figure 2(b). According to formula (9), a candidate range $[x_k^{\min}, x_k^{\max}] = [12, 14]$ is given by $x'_3 = 13$ (which is determined in the previous step in Example 4).*

Following the same line of Example 4, we compute $X_4^{\min} = \{16, 17\}$ and $X_4^{\max} = \{14, 14\}$ by points 5 and 6 that are in the window of point 4. It follows $x_4^{\mathrm{mid}} = 14$, which is in the candidate range $[12, 14]$. According to formula (10), the local optimal repair on $k = 4$ is $x'_4 = 14$.

The integral repair moves on to the next $k = 5$ and terminates when reaching the end of the sequence. A repaired sequence $\{12, 12.5, 13, 14, 15, 15.5\}$ is finally returned.

4. OUT-OF-ORDER CLEANING

When a delayed data point comes, the straightforward approach is to insert the data point into the right position of the sequence according to timestamps, and recompute all the results for data points near (and after) the inserted position. However, complete re-computation is not necessary. Instead, we can efficiently update the repairs over the previously computed results. To further reduce the computation, we propose a heuristic strategy that updates the results only when the previous results are known to be invalid for sure.

4.1 Updating Local Optimum

Consider an out-of-order arrival x_k, $t_k < t_{k-1}$. We reorder the sequence by timestamps, i.e., removing x_k and inserting it as a new x_ℓ where $x_\ell = x_k, t_{\ell-1} < t_\ell < t_{\ell+1}, \ell < k$.

The updates introduced by x_ℓ include two aspects: (1) for $x_j, j < \ell$, where x_ℓ suggests candidates for determine x_j^{mid}; and (2) for $x_i, i > \ell$, whose candidate range $[x_i^{\min}, x_i^{\max}]$ is influenced (directly or indirectly) by x'_ℓ.

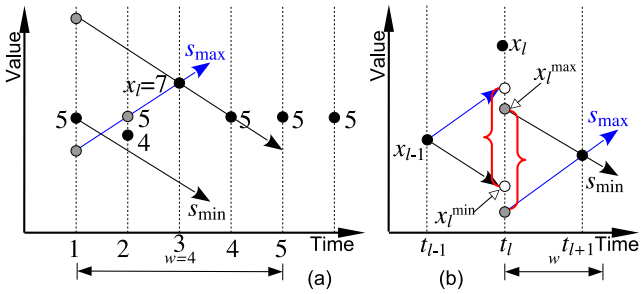

Figure 6: Update x'_ℓ and compute x'_ℓ in heuristic

First, for any x_j with $t_\ell - w \le t_j < t_\ell, 1 \le j < \ell$, we update its median by adding the candidates w.r.t. the new x_ℓ in $X_j^{\min}$ and $X_j^{\max}$ (in Lines 6 and 7 in Algorithm 2). It is worth noting that x_j^{mid} may remain unchanged after introducing two new candidates by x_ℓ. If one of the candidate is greater than x_j^{mid} and the other is less than x_j^{mid}, obviously, x_j^{mid} is still the median in the new set $X_j^{\min} \cup X_j^{\max} \cup \{x_j\}$.

Proposition 6. *For any x_j with $t_\ell - w \le t_j < t_\ell$, if $s_{\min} \le \frac{x_j^{\mathrm{mid}} - x_\ell}{t_j - t_\ell} \le s_{\max}$ for the new arrived x_ℓ, x_j^{mid} is not changed.*

Once the medians are updated, we recalculate the repairs w.r.t. the new medians. It further leads to the updates on succeeding ones (including $i > \ell$, with candidate range $[x_i^{\min}, x_i^{\max}]$ determined by previous repairs).

Algorithm 2 presents the update of x' when a delayed x_ℓ is newly observed. The algorithm is called before Line 2 of Algorithm 1, when an out-of-order data point comes and is placed in the position ℓ after reordering the sequence.

Algorithm 2: Update(x', s, ℓ, k)

Data: a repair x' with the first k data points ordered by timestamps and a newly observed x_ℓ, $\ell < k$
Result: an updated repair x' of x w.r.t. local optimum

```
1  compute x_ℓ^mid by formula (8);
2  for i ← ℓ − 1 to 1 do
3      if t_i < t_ℓ − w then
4          break;
5      if (x_j^mid − x_ℓ)/(t_j − t_ℓ) < s_min or (x_j^mid − x_ℓ)/(t_j − t_ℓ) > s_max then
6          X_j^min ← X_j^min ∪ {x_ℓ + s_min(t_j − t_ℓ)};
7          X_j^max ← X_j^max ∪ {x_ℓ + s_max(t_j − t_ℓ)};
8          update x_j^mid by formula (8);
9  for j ← i + 1 to k do
10     update x_i^min, x_i^max by formula (9);
11     update x'_i by formula (10);
12     if x'_i is unchanged, i > ℓ then
13         break;
14 return x'
```

Example 6 (Out-of-order update)**.** *Let $x = \{5, 4, 5, 5, 5\}$ be the currently processed sequence, with timestamps $t = \{1, 2, 4, 5, 6\}$, as illustrated in Figure 6(a). The speed constraints are $s_{\max} = 2$ and $s_{\min} = -2$ with window size $w = 4$.*

Suppose that the next data point is $x_6 = 7$ with timestamp $t_6 = 3 < t_5 = 6$, i.e., an out-of-order arrival. We first reorder the sequence by timestamps, $x = \{5, 4, 7, 5, 5, 5\}$ and $t = \{1, 2, 3, 4, 5, 6\}$, where the new data point locates in $\ell =$

3. *Referring to the window size $w = 4$, data points 1 and 2 are influenced by the new point.*

For x_1, suppose that $x_1^{\text{mid}} = 5$ before the new point ℓ is inserted. We compute $\frac{x_1^{\text{mid}} - x_\ell}{t_1 - t_\ell} = \frac{5-7}{1-3} = 1$, which is in the range $[s_{\min}, s_{\max}]$. According to Proposition 6, $x_1^{\text{mid}} = 5$ will not be changed after inserting ℓ. For x_2, we have $x_2^{\text{mid}} = 4$ before inserting ℓ and $\frac{4-7}{2-3} = 3 > s_{\max}$. That is, x_2^{mid} should be updated after inserting ℓ, having $x_2^{mid} = 5$. And $x_3^{\text{mid}} = 7$ for $\ell = 3$ can also be computed by formula (8). Since x_2^{mid} is updated and x_3^{mid} is newly introduced, the repair should be refreshed following formula (10), i.e., $x' = \{5, 5, 7, 5, 5, 5\}$.

4.2 Heuristic

Although exact result w.r.t. local optimum is guaranteed, the update (possibly on all the data points near and after x_ℓ) is costly. If a data point is delayed at the very beginning of the sequence, almost the entire sequence may be updated.

Heuristically, we could choose to update only when the existing repairs are found to be invalid for sure. That is, after inserting the data point on position ℓ, the existing x' violates speed constraints (no matter what the value x'_ℓ is).

Such invalid scenario occurs, because of the disagreement on the possible values for x'_ℓ. The existing repair $x'_{\ell-1}$ specifies a range of possible repairs for x'_ℓ, as presented in formula (9). Symmetrically, $x'_{\ell+1}$ as an existing repair also indicates another range for x'_ℓ. Contradiction between two ranges may occur (if $x'_{\ell+1}$ is not in the window of $x'_{\ell-1}$).

We capture the minimum and maximum candidates for x'_ℓ that are specified by $x'_{\ell-1}$ and $x'_{\ell+1}$ together,

$$\begin{aligned} x_\ell^{\min} &= \max(x'_{\ell-1} + s_{\min}(t_\ell - t_{\ell-1}), x'_{\ell+1} + s_{\max}(t_\ell - t_{\ell+1})), \\ x_\ell^{\max} &= \min(x'_{\ell-1} + s_{\max}(t_\ell - t_{\ell-1}), x'_{\ell+1} + s_{\min}(t_\ell - t_{\ell+1})), \end{aligned} \tag{11}$$

where $t_{\ell+1} - t_{\ell-1} \leq 2 \cdot w$. Here, $x_\ell^{\min}$ takes the maximum of bounds (in formula (9)) determined by $\ell - 1$ and $\ell + 1$, as illustrated in Figure 6(b), and similarly $x_\ell^{\max}$ takes the lower one of the bounds given by $\ell - 1$ and $\ell + 1$. In other words, a x'_ℓ within the range $[x'_{\ell-1}, x'_{\ell+1}]$ will satisfy the speed constraints w.r.t. both $\ell - 1$ and $\ell + 1$.

Obviously, if $x_\ell^{\min} > x_\ell^{\max}$, contradiction occurs. That is, the current repair x' is invalid, and the update by Algorithm 2 should be performed.

On the other hand, if valid candidates exist, i.e., $x_\ell^{\min} \leq x_\ell^{\max}$, we can heuristically select a repair x'_ℓ in $[x_\ell^{\min}, x_\ell^{\max}]$ without updating the others. As shown in Lines 3 and 4 in Algorithm 3, x_ℓ is taken as x_ℓ^{mid} in heuristic. According to formula (10), x'_ℓ leaves unchanged if x_ℓ is in $[x_\ell^{\min}, x_\ell^{\max}]$. Otherwise, the border $x_\ell^{\min}$ or $x_\ell^{\max}$, which is closer to x_ℓ, is assigned as the repair x'_ℓ.

A natural concern is how often the full Update in Line 7 in Algorithm 3 occurs. We show in the following conclusion that Update would never be performed if the time interval $t_{\ell+1} - t_{\ell-1}$ is within w.

Proposition 7. *For any $x_\ell^{\min} > x_\ell^{\max}$ computed by formula (11), it always has $w < t_{\ell+1} - t_{\ell-1} \leq 2 \cdot w$.*

In other words, the Update may occur only when $t_{\ell-1}$ and $t_{\ell+1}$ are far away ($> w$). In practice, such large "breaks" appear rarely especially in continuous monitoring streams. By greatly avoiding Update, the Heuristic approach can significantly reduce the repair time costs (in the experiments).

Algorithm 3: Heuristic(x', s, ℓ, k)

Data: a repair x' with the first k dat points ordered by timestamps and a newly observed x_ℓ, $\ell < k$, and speed constraints s

Result: an updated repair x' of x w.r.t. local optimum

1 compute $x_\ell^{\min}, x_\ell^{\max}$ by formula (11);
2 **if** $x_\ell^{\min} \leq x_\ell^{\max}$ **then**
3 $\quad x_\ell^{\text{mid}} \leftarrow x_\ell$;
4 $\quad$ compute x'_ℓ by formula (10);
5 $\quad$ **return** x'
6 **else**
7 $\quad$ **return** Update(x', s, ℓ, k)

Example 7 (Heuristic, Example 6 continued)**.** *Consider again the reordered sequence, $x = \{5, 4, 7, 5, 5, 5\}$ with timestamps $t = \{1, 2, 3, 4, 5, 6\}$, where $\ell = 3$ is the newly inserted data point, as illustrated in Figure 6(a).*

Referring to formula (9), data point $\ell-1$ gives a candidate range $[2, 6]$ for x'_ℓ w.r.t. the speed constraints $s_{\max} = 2$ and $s_{\min} = -2$. Symmetrically, another candidate range $[3, 7]$ for x'_ℓ is also determined by $\ell+1$. By taking the intersection of two candidate ranges, we have $[x_\ell^{\min}, x_\ell^{\max}] = [3, 6]$ as defined in formula (11). Since $x_\ell^{\min} \leq x_\ell^{\max}$, we heuristically determine the repair $x'_3 = 6$ without update the other repair results, having $x' = \{5, 4, 6, 5, 5, 5\}$.

5. ADAPTIVE WINDOWS

The trade-off in setting window size w for speed constraints is: small windows fail to capture the minimum repair (e.g., if w=1 in Figure 2(b) of Example 2), while large windows obviously increase the computation overhead (as more constrained data pairs need to be specified in formulas (3) for global optimum or (5) for local optimum). It is non-trivial to predefine an appropriate window size. Even worse, the arrival rate (the number of data points in a period) may vary. Rather than a fixed window size, we propose to automatically determine the adaptive window size w online.

To tackle the aforesaid trade-off in choosing window sizes, we introduce a statistical sampling-based method. The idea is to model data points as *random samples* of approaching the speed constraints. The adaption of windows, extending or shrinking, is thus performed based on the closeness to the extreme speeds ($s_{\max}$ or $s_{\min}$) of the samples, grounded in statistical sampling theory.

Sampling Model. Consider a window starting from i with length w, $W_i = \{j \mid 0 < t_j - t_i \leq w\}$. Assume that the (repaired) data points in the window satisfy speed constraint s.

Intuitively, the speed constraint (say $s_{\max}$) takes strong effect, if a x_j in the window approaches $x_i + s_{\max}(t_j - t_i)$, i.e., the speed $\frac{x_j - x_i}{t_j - t_i}$ approaches the constraint $s_{\max}$. On the other hand, the speed constraints are useless, if the speed $\frac{x_j - x_i}{t_j - t_i}$ is far away from $s_{\min}$ or $s_{\max}$. Let

$$p_{i,j} = \frac{\frac{x_i - x_j}{t_i - t_j} - s_{\min}}{s_{\max} - s_{\min}} \tag{12}$$

denote the degree of x_j approaching the (maximum) speed constraints, having $0 \leq p_{i,j} \leq 1$. When the speed $\frac{x_j - x_i}{t_j - t_i} = s_{\max}$, we have $p_{i,j} = 1$. And $p_{i,j} = 0$ corresponds to the

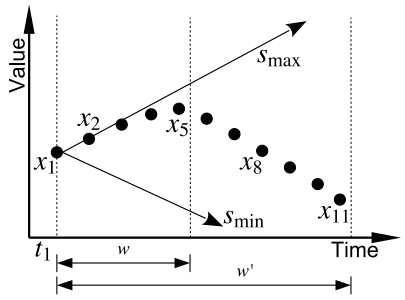

Figure 7: Adjust w based on closeness to speed constraints

speed equal to $s_{\min}$. In contrast, if the speed is far away from the speed constraints, e.g., in the middle of $\frac{1}{2}(s_{\max} + s_{\min})$ with $p_{i,j} = 0.5$, the speed constraint is useless.

Assume that each data point in the window has the same probability p of approaching the maximum speed constraint w.r.t. x_i. The probability p can be estimated by the average degree of data points approaching the maximum speed constraints, $p_i = \frac{\sum_{j \in W_i} p_{i,j}}{|W_i|}$.

We view each data point as a random sample of approaching the maximum speed constraint. The total number of data points that reach $s_{\max}$, denoted by S, is thus a random variable that follows *binomial distribution*, i.e., $S \sim B(|W_i|, p_i)$. Referring to the standard probability theory, the expectation and variance can be expressed as $\mathrm{E}(S) = |W_i| \cdot p_i$ and $\mathrm{Var}[S] = |W_i| \cdot p_i \cdot (1 - p_i)$.

Adaptive Window. Next, we introduce how to use the aforesaid binomial sampling model to adjust the window sizes. Intuitively, the closer the speeds of data points to the constraints $s_{\min}$ and $s_{\max}$, the more necessary the constraints are to guard the correctness of data points. That is, the window could be enlarged to involve more data points. On the other hand, if the observed speeds are far from the border $s_{\min}$ and $s_{\max}$, the constraints are useless. The windows may shrink for computation efficiency.

In normal approximation [2], a reasonable approximation to $B(n, p)$ is given by the normal distribution $\mathcal{N}(np, np(1-p))$. This approximation generally improves as n increases and is better when p is not near to 0 or 1. Rules are designed to decide whether n is large enough, and p is far enough from the extremes of zero or one. A commonly used rule states that the normal approximation is appropriate only if everything within 3 standard deviations of its mean is within the range of possible values, i.e., $np \pm 3\sqrt{np(1-p)} \in [0, n]$.

In this sense, we can use the evaluation of normal approximation (whether the samples are not enough) to infer whether the window size is not large enough, or (whether the probability is far enough from 0 or 1) to infer whether the window size is too large such that speed constraints are useless (with data points far from the speed constraints $s_{\min}$ and $s_{\max}$). Referring to the rule for normal approximation, if $\mathrm{E}(S) \pm 3\sqrt{\mathrm{Var}[S]} \in [0, |W_i|]$, the samples are large enough and the probability p_i is far enough from the extremes of zero or one, i.e., far away from the speed constraints (which are useless). Thereby, we reduce the window size, e.g., according to w' in Figure 7 in the following Example 8. On the other hand, if $\mathrm{E}(S) \pm 3\sqrt{\mathrm{Var}[S]} \notin [0, |W_i|]$, more samples are needed by increasing the window size (e.g., as suggested by

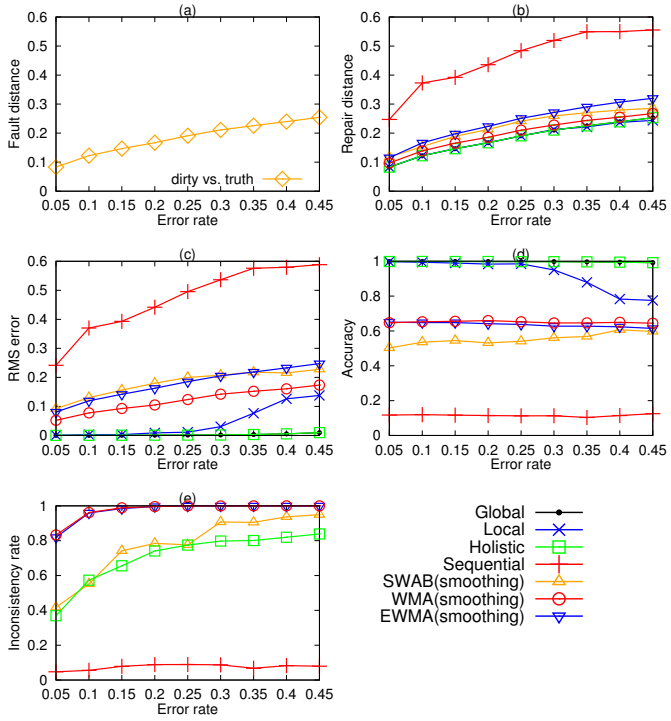

Figure 8: Varying error rates (FUEL)

w in Figure 7). (Please refer to Appendix C for more details on normal approximation and adjusting window sizes.)

Example 8. *Consider two windows* $W_1 = \{2, 3, 4, 5\}$ *with size* w *and* $W_1' = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ *with size* w' *in Figure 7. By formula (12), we compute* $p_{1,2} = 1$*, since* x_2 *exactly reaches the maximum speed w.r.t.* x_1*. Similar* $p_{1,j}$ *can be computed for* $j = 3, \ldots, 11$*, e.g.,* $p_{1,8} = 0.5$ *which is neither close to* $s_{\max}$ *nor* $s_{\min}$*.*

For W_1*, the probability* $p_1 = 0.91$ *is computed by the average of* $p_{1,2}, \ldots, p_{1,5}$*. The* $\mathrm{E}(S) \pm 3\sqrt{\mathrm{Var}[S]}$ *values corresponding to the 4 samples in* W_1 *are* $4 \cdot 0.91 \pm 3\sqrt{4 \cdot 0.91 \cdot 0.09}$*, i.e., 1.92 or 5.35. The later one does not belong to [0,4]. The window* W_1 *suggests to extend the window size.*

For W_1' *with 10 samples, the computed probability is* $p_1' = 0.52$*. The* $\mathrm{E}(S) \pm 3\sqrt{\mathrm{Var}[S]}$ *values are 0.46 and 9.93, in the range of [0,10]. That is, the window* W_1' *could shrink.*

Finally, for each data point i, it makes a decision on enlarging or shrinking the window sizes (for the succeeding windows). Such decision making may also be conducted periodically, e.g., in every 10 minutes.

6. EXPERIMENT

In the experimental evaluation, we employ 3 real datasets, FUEL, STOCK and GPS. (Please refer to Appendix D.1 for dataset preparation.) The evaluation criteria includes: 1) the fault distance between dirty and truth data, 2) the repair distance between dirty input and repair result, 3) the RMS error [10] between the repair result and truth data, 4) the repair accuracy relative to the dirty data, 5) the rate of inconsistent data retained in the repair result. (See Appendix D.2 for the relationships among the measures.)

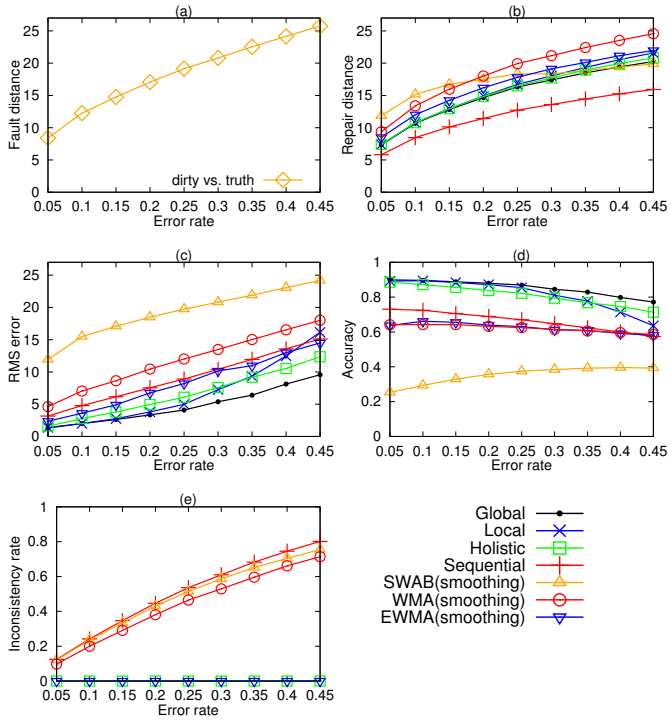

Figure 9: Varying error rates (STOCK)

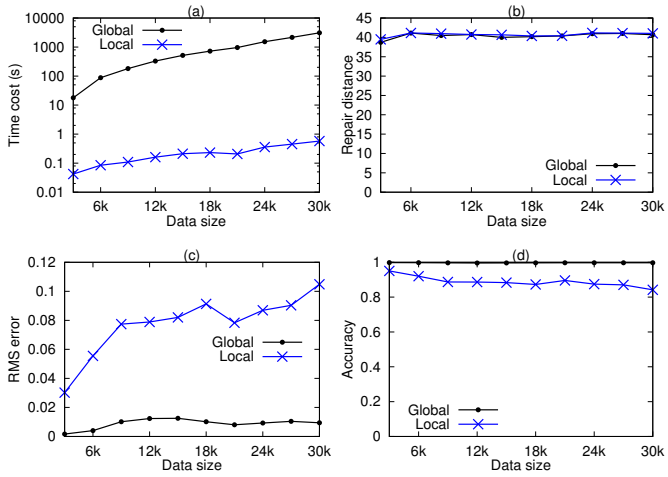

Figure 10: Global vs. Local (FUEL)

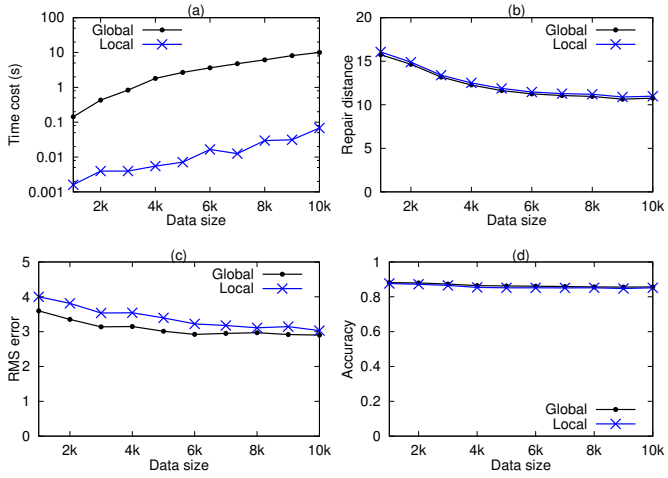

Figure 11: Global vs. Local (STOCK)

Table 1: GPS data with manually labeled ground truth

Method	Repair distance	RMS error	Accuracy	Inconsistency
Global	0.1198	0.0658	0.6881	–
Local	0.1311	0.0682	0.6886	–
Holistic	0.1197	0.0653	0.6842	0.0403
Sequential	0.9589	0.9646	0.0001	0.1285
SWAB	0.6368	0.6372	0.0649	0.0639
WMA	1.5700	1.5627	0.0184	0.1492
EWMA	1.7054	1.6993	0.0166	0.0556

6.1 Comparison with Existing Approaches

In the first experiment, we compare our proposed SCREEN with Global and Local optimum to the following approaches in two categories: 1) smoothing-based, SWAB [13], WMA and EWMA [7], and 2) constraint-based, Holistic repair [4] and repair with Sequential Dependency [8].

Figures 8 and 9 consider various error rates (denoting the amount of injected errors) with data sizes (the number of data points/the length of the sequence) 3k and 2k, in FUEL and STOCK, respectively. Figures 10 and 11 study the scalability by varying data sizes (with error rate 0.3 and 0.2). The full results of all compared methods are in Figures 18 and 19. Table 1 presents the results over the GPS dataset with manually labeled ground truth.

First, it is not surprising that RMS error of all the smoothing methods is high, while their accuracy is lower, compared with our proposed Global or Local (e.g., in Table 1). As illustrated in Figure 8(b), the repair distances of SWAB, WMA and EWMA smoothing are very similar. These smoothing methods show similar RMS error and accuracy as well in Figures 8(c)(d). It is notable that the inconsistency rates of smoothing are very high in Figure 8(e). That is, a large number of data points are still in violation w.r.t. speed constraints. The corresponding accuracy of smoothing is thus significantly lower than our proposal in Figure 8(d).

Holistic repair, the approximation of our Global as introduced in Appendix B, show similar accuracy as Global. However, its time cost is close to the exact approach Global as well. Even worse, as shown in Figures 8(e), the inconsistency rate of Holistic is high. An inconsistency rate 0.4 denotes that about 40% data points are still involved in violation to speed constraints, after Holistic repair. For STOCK with window size $w = 1$, there are fewer constraint predicates declared between data points. Therefore, inconsistencies remained after Holistic repair are fewer as well, i.e., lower inconsistency rate in Figures 9(e).

Sequential dependencies (SDs) consider the constraints on value difference (e.g., ≤ 5) of two consecutive data points. When the time interval of any two consecutive data points is the same, for instance, in every trading day or in every 5 minutes, SDs denote exactly the speed constraints. There-

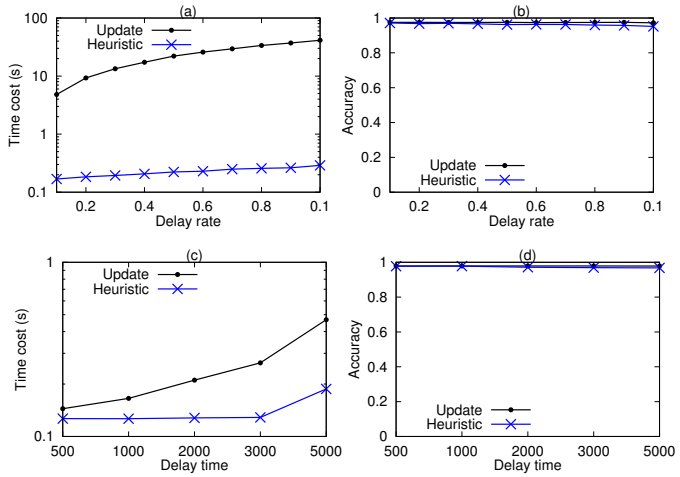

Figure 12: Evaluation on out-of-order (FUEL)

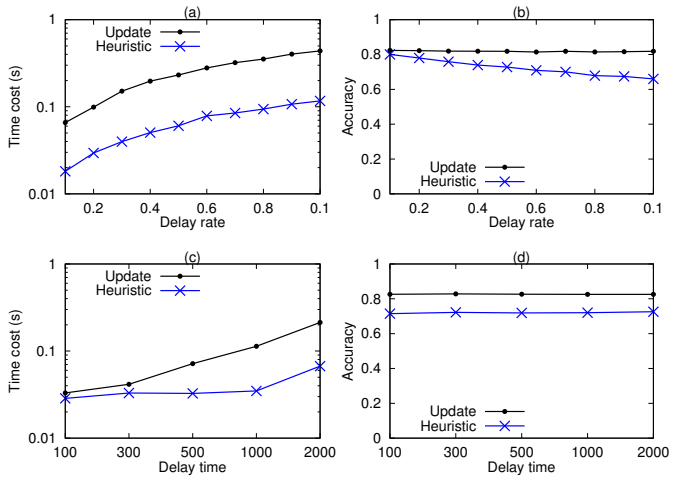

Figure 13: Evaluation on out-of-order (STOCK)

fore, in STOCK, the accuracy of Sequential is relatively high, while its RMS error is low (compared with SWAB). However, if the data point arrival is dynamic without a fixed time interval, such as FUEL, SDs can no longer denote the speed semantics. Consequently, the performance of Sequential is low in Figures 8(c)(d), even worse than SWAB.

The repair distance of Global is always lower than that of Local, in Figures 10(b) and 11(b). The corresponding RMS error of Global is lower in Figures 10(c) and 11(c) and the repair accuracy of Global is higher in Figures 10(d) and 11(d). Nevertheless, the Local method have very similar repair distance and accuracy to Global, especially compared with the other baseline approaches in Figures 18 and 19. The results verify our analysis (e.g., $\Delta(x, x_{\text{local}}) \geq \Delta(x, x_{\text{global}})$ at the beginning of Section 3), and demonstrate the time and accuracy performance of Local optimum compared to Global in practice.

6.2 Evaluation over Streaming Data

Next, we evaluate the online cleaning over streaming data. Since none of the existing methods support out-of-order arrivals, we mainly compare the proposed Update technique in Algorithm 2 and Heuristic in Algorithm 3 for handling de-

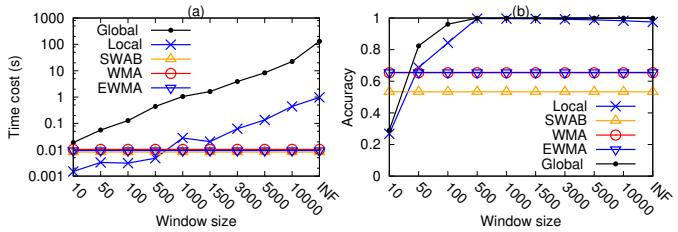

Figure 14: Evaluation on various window sizes

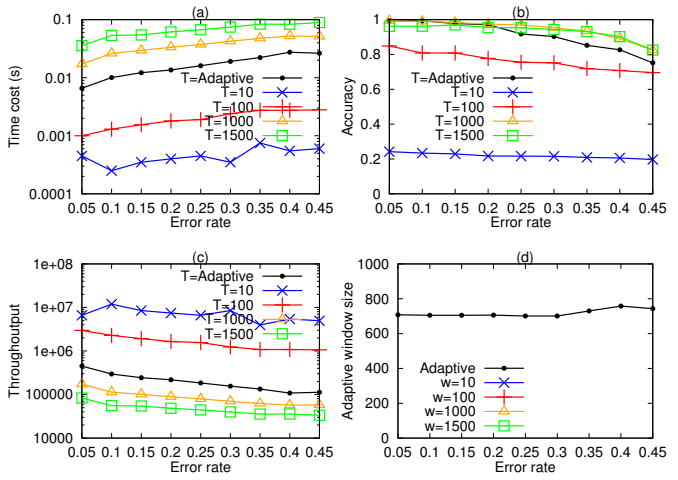

Figure 15: Evaluation on adaptive window sizes

layed data points. Figures 12 and 13 report the out-of-order evaluation, by varying delay rate (e.g., 0.1 denotes that 10% of data points are delayed in arrival) and delay time (referring to the average distance between the actual timestamp and the arrival time). Settings of delay time for Figures 12(a)(b) and 13(a)(b) are 5000 and 1000, respectively, and the delay rate in Figures 12(c)(d) and 13(c)(d) is 0.05.

Generally, the larger the delay rate or delay time is, the higher the time costs are. For Heuristic, since the full update occurs only when the previously repaired results are found to be wrong for sure, its time cost is significantly lower. As shown in Figures 12 and 13, Heuristic method has comparable accuracy with Update, while the former one can show 1-2 orders of magnitude improvement in time costs.

6.3 Evaluation on Adaptive Windows

This experiment evaluates the cleaning performance under various window sizes of speed constraints. Since the window size is fixed to 1 for STOCK, we focus on the FUEL dataset (with data size 3k, and error rate 0.05 for Figure 14).

As shown in Figure 14, the Local method requires a bit larger window size (e.g., 100) to achieve the same accuracy of Global (at 50). For a window larger than 500, both Global and Local have considerably high accuracy. The time cost of an infinite window (INF, throughout the lifetime of the data stream) is extremely high, while a smaller window size (e.g., 500) that has much lower time cost can achieve an accuracy as high as INF. The results verify that, in a real setting, it is not necessary to consider extremely large windows, since a small window size is sufficient to achieve the same accuracy with significantly lower time cost. On the other hand, if the window size is too small, e.g., $w = 10$ in Figure 14, many

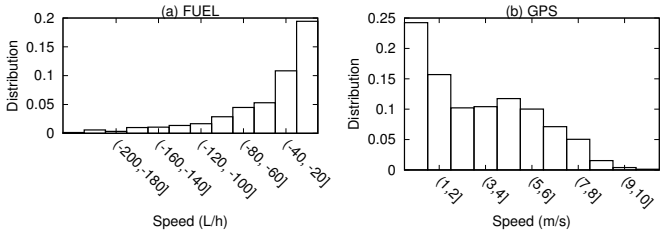

Figure 16: Statistical distribution on speeds

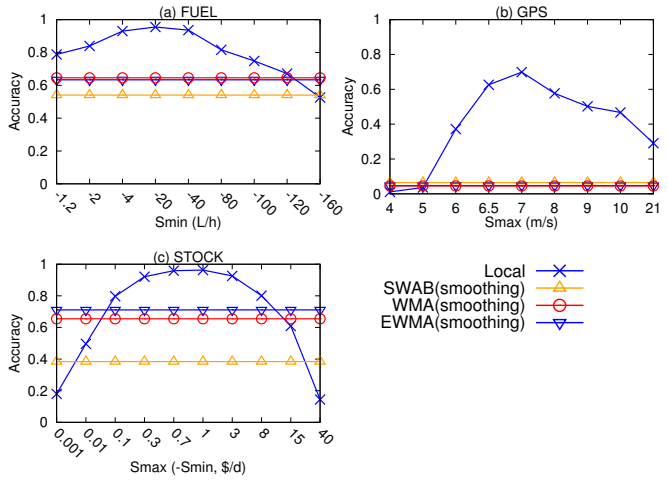

Figure 17: Evaluation on various speed constraints

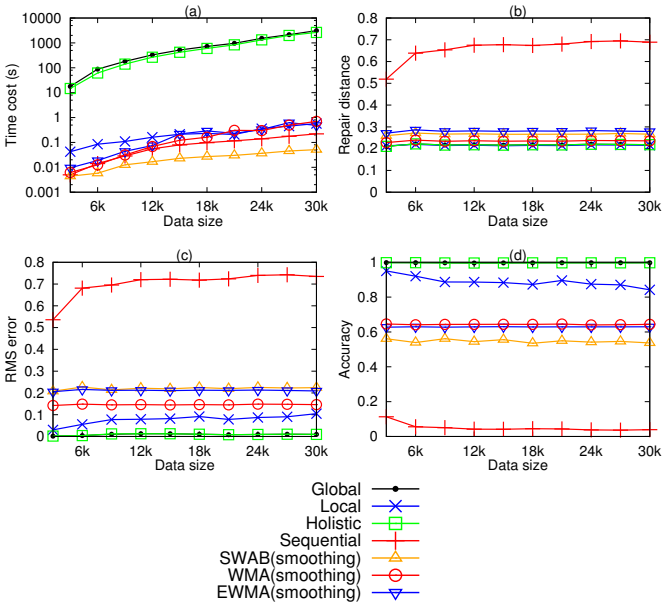

Figure 18: Scalability (FUEL)

data points will be out of control by the speed constraints, and fewer repairs are performed. The repair accuracy is thus lower (even lower than the smoothing baselines).

Nevertheless, as illustrated in Figure 15, our adaptive window technique can determine an appropriate window size (around 700), which shows relatively low time cost and high throughput without losing much accuracy. These results verify our trade-off analysis between cleaning accuracy and efficiency in Section 5. In other words, our proposed adaptive windows can balance the effectiveness and efficiency.

6.4 Capturing Speed Constraints

We note that in most scenarios, the speed constraint is natural, e.g., the fuel consumption of a crane should not be negative ($s_{\max} = 0$ for FUEL), while some others could be derived. According to our consultation with experts of the equipment, the fuel consumption does not exceed 40 liters per hour ($s_{\min} = -40$). For STOCK, the speed constraints are naturally derived by the business semantics. The price limit in the market declares that the increase or decrease of daily price should not exceed $l \cdot r$ where l is the price of the last trading day and $r = 10\%$ is a percentage. We consider the highest price $h = 60$ in the dataset, having $l \leq h$ for any day in the period. The maximum speed $s_{\max}$ (with $w = 1$) in the period is thus $h \cdot r = 6$, while $s_{\min} = -h \cdot r = -6$. The GPS dataset is collected by a person carrying a smartphone and walking around at campus. We require 7 meters per second as the maximum walking speed of the person.

Nevertheless, for a particular domain where speed knowledge is not available, the speed constraints can be extracted from data. We consider the statistical distribution of speeds by sampling data pairs over FUEL and GPS in Figure 16. As mentioned, the FUEL meter change is non-increasing ($s_{\max} = 0$) during consumption, while the walking speed for GPS is non-negative ($s_{\min} = 0$). Thereby, we mainly observe the speeds for determining $s_{\min}$ for FUEL and $s_{\max}$ for GPS. Referring to statistics, it is to determine a confidence interval that acts as a good estimate of the unknown population parameter $s_{\min}$ ($s_{\max}$). In applied practice, confidence intervals are typically stated at the 95% confidence level [11]. In other words, 95% of the speeds are regarded as accurate (within $s_{\min}$ or $s_{\max}$). It suggests $s_{\max} = 7$ for GPS and $s_{\min} = -80$ for FUEL (although the expert's suggestion is -40 in the experiments). As illustrated in Figure 17, given such speed constraints (either -80 or -40 for FUEL), the accuracies are much higher than the smoothing baselines.

Generally, if the speed constraints are set too loose, e.g., $s_{\min} = -160$ in Figure 17(a) or $s_{\max} = 40$ in Figure 17(c), almost everything will pass the examination of speed constraints without repairing and thus the repair accuracy (of Local) is low. On the other hand, if the speed constraints are too tight, say $s_{\max} = 0.001$ in Figure 17(c), most values would be regarded as violations to such tight constraints. With over-repairing, the corresponding repair accuracy is low too. Nevertheless, the accuracy of the constraint-based method (Local) is higher than that of the constraint-oblivious baselines, in a wide range of speed constraints, e.g., either the expert suggested -40 or the aforesaid statistical suggestion -80 in Figure 17(a). For GPS data in Figure 17(b), there is also a wide range, from 6 to 21, where the constraint-aware Local clearly outperforms the constraint-oblivious baselines. Indeed, in common sense, it is irrational for a person walking with a speed greater than 21 meters per second.

6.5 Summary of Experiments

We summarize the experimental results as follows: 1) The accuracy is significantly improved by our proposal (Global and Local) compared to the existing SWAB, WMA and EWMA smoothing; 2) Without considering the precise speed constraints, the existing SD-based repair has much lower ac-

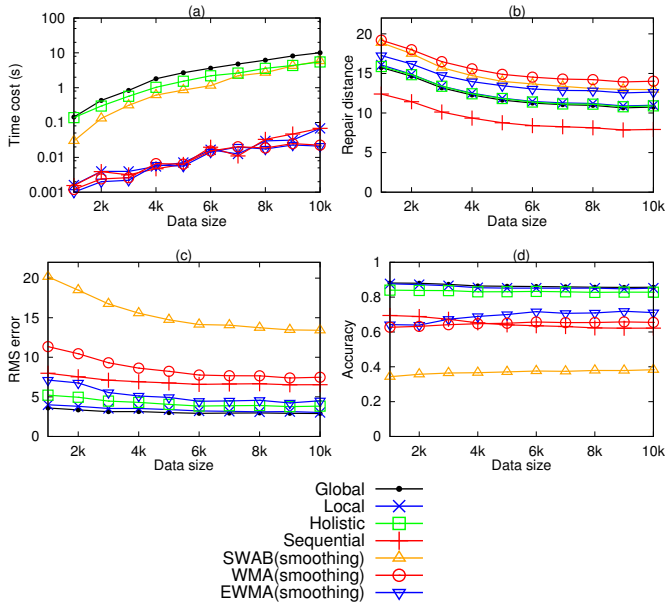

Figure 19: Scalability (STOCK)

curacy than our Global and Local; 3) The proposed Local approach with online cleaning supports shows orders of magnitude improvement in time costs compared to the Holistic method; 4) The heuristic update strategy significantly reduces the time costs in handling out-of-order arrival; 5) The adaptive window technique can automatically suggests an appropriate window size (around 700), which shows relatively low time costs without losing much accuracy.

7. RELATED WORK

Smoothing-based Data Cleaning. The SWAB smoothing [13] is a linear interpolation/regression-based, online smoothing method of stream data. With a sliding window, SWAB uses linear interpolation or linear regression to find the approximating line of a time series. Besides, the moving average [3] is also commonly used to smooth time series data and make forecasts. A simple moving average (SMA) is the unweighted mean of the last k data. This average is used for forecasting the next value of the time series. Whereas in the simple moving average the past observations are weighted equally, a weighted moving average (WMA) multiplies factors to give different weights to data at different positions in the sample window, e.g., using the inverse value of time interval as the weight. Moreover, the exponentially weighted moving average (EWMA) [7] assigns exponentially decreasing weights over time. It is obvious to see that all these smoothing methods will modify a large number of data points. Therefore, as the example illustrated in Figure 1, the major issue of smoothing is the serious damage of the originally correct data points. One of our major contributions in this paper is the employment of speed constraints to supervise the more accurate cleaning. Following the minimum change principle in constraint-based repairing, the original precise values are maximally preserved. Consequently, the accuracy of our proposed method is much higher than that of smoothing, as observed in our experimental evaluation.

Constraint-based Data Repairing. To the best knowledge of our knowledge, Holistic cleaning [4] is the only existing constraint-based technique that can support speed constraints (expressed by denial constraints). Since the approach is proposed for repairing the general (tableau) data, it cannot support the online/integral cleaning over sliding windows in streaming data. In this sense, one of our contributions in this study is the SCREEN with local optimum, which supports not only online cleaning but also out-of-order data arrival. Consequently, as illustrated in the experiments, our proposal can show up to 4 orders of magnitude improvement in time costs compared with Holistic cleaning.

Moreover, Sequential Dependency (SD) [8] cannot express precisely the speed constraints. SDs concern the difference of two consecutive data points in a sequence. As discussed, data streams often deliver data points in various time intervals. Given different timestamp distances, the value difference of two consecutive points does not exactly denote the speed semantics. Owing to such imprecise constraint knowledge, as presented in the experiments, the accuracy of SD (Sequential) based repair could be much lower compared to our speed constraint-based proposal. Our another contribution is the employment of the more accurate speed rather than the simple value distance in repairing streaming data.

Besides our studied speed constraints, Fischer et al. [5] propose a nice notation, Stream Schema, for representing structural and semantic constraints on data streams. The Stream Schema concerns general constraints with various semantics such as orderings between attribute values, while our study focuses only on the specific speed constraints over numeric values. As a promising future direction, it is interesting to extend the stream data cleaning w.r.t. the more general Stream Schema constraints.

8. CONCLUSIONS

In this study, we first indicate the inappropriateness of the smoothing-based stream data cleaning. It could not repair the dirty data such as large spikes, and even worse may seriously damage the originally accurate values. Following the same line of employing integrity constraints for relational data cleaning, in this paper, we propose SCREEN, the first constraint-based stream data cleaning approach. The repairing of imprecise data is guided by the innovative constraints on speed. The speed constraint semantics could be easily captured, such as daily price limit in financial markets, or fuel consumption limit of devices. With speed constraints, SCREEN supports online streaming cleaning in linear time, out-of-order arrival of data points, and high throughput via adaptive window sizes. In particular, the novel *Median Principle* can fast identify the local optimum, following the intuition that a solution with the minimum distance (i.e., as close as possible to each point) probably lies in the middle of the data points. Experiments on real datasets demonstrate that our SCREEN can show significantly higher repair accuracy than the smoothing-based approach, and up to 4 orders of magnitude improvement in time performance compared to the state-of-the-art data cleaning methods.

Acknowledgement

This work is supported in part by China NSFC under Grants 61325008, 61202008 and 61370055; US NSF through grants CNS-1115234, and OISE-1129076.

9. REFERENCES

[1] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD Conference*, pages 143–154, 2005.

[2] G. E. P. Box. Statistics for experimenters: an introduction to design, data analysis, and model building. 1978.

[3] D. R. Brillinger. *Time series: data analysis and theory*, volume 36. Siam, 2001.

[4] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *ICDE*, pages 458–469, 2013.

[5] P. M. Fischer, K. S. Esmaili, and R. J. Miller. Stream schema: providing and exploiting static metadata for data stream processing. In *EDBT*, pages 207–218, 2010.

[6] Full Version. http://ise.thss.tsinghua.edu.cn/sxsong/doc/screen.pdf.

[7] E. S. Gardner Jr. Exponential smoothing: The state of the art–part ii. *International Journal of Forecasting*, 22(4):637–666, 2006.

[8] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.

[9] C. A. R. Hoare. Quicksort. *Comput. J.*, 5(1):10–15, 1962.

[10] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive cleaning for rfid data streams. In *VLDB*, pages 163–174, 2006.

[11] H. Z. Jerrold. Biostatistical analysis. *Biostatistical analysis*, 1999.

[12] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC*, pages 302–311, 1984.

[13] E. J. Keogh, S. Chu, D. M. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

[14] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, pages 53–62, 2009.

[15] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(2):97–108, 2012.

[16] M. Liu, M. Li, D. Golovnya, E. A. Rundensteiner, and K. T. Claypool. Sequence pattern query processing over out-of-order event streams. In *ICDE*, pages 784–795, 2009.

[17] A. Lopatenko and L. Bravo. Efficient approximation algorithms for repairing inconsistent databases. In *ICDE*, pages 216–225, 2007.

APPENDIX

A. TRANSFORMATION TO LP

We transform the global optimal repair problem in formula (2) to a standard linear programming (LP) problem, so that existing solvers can directly be employed.

Table 2: Notations

Symbol	Description
x	sequence
$x[i]$ or x_i	value of i-th data point in x
t	timestamp
s	speed constraint
w	window size of speed constraint
n	length of a finite sequence
x'	repair of sequence x
$x_i^{\min}, x_i^{\max}$	the lower, upper bound of a valid repair x'_i

Let $u_i = \frac{|x'_i - x_i| + (x'_i - x_i)}{2}$ and $v_i = \frac{|x'_i - x_i| - (x'_i - x_i)}{2}$. We have $|x'_i - x_i| = u_i + v_i$ and $x'_i - x_i = u_i - v_i$. It follows

$$\min \quad \sum_{i=1}^{n} u_i + v_i$$

$$\text{s.t.} \quad \frac{u_j - v_j + v_i - u_i - x_i + x_j}{t_j - t_i} \le s_{\max}, \quad t_i < t_j \le t_i + w, \quad 1 \le i \le n, 1 \le j \le n$$

$$\frac{u_j - v_j + v_i - u_i - x_i + x_j}{t_j - t_i} \ge s_{\min}, \quad t_i < t_j \le t_i + w, \quad 1 \le i \le n, 1 \le j \le n$$

$$u_i \ge 0, v_i \ge 0, \quad 1 \le i \le n$$

where u_i, v_i are the variables in problem solving.

Example 9 (Global optimum, Example 2 continued)**.** *Consider again the sequence x and the speed constraints $s_{\max} = 0.5$ and $s_{\min} = -0.5$ with window size $w = 2$ in Example 2.*

According to formulas (3) and (4), the constraint predicates declared w.r.t. $s_{\max}$ and $s_{\min}$ are:

$$\frac{x'_2 - x'_1}{2-1} \le 0.5, \frac{x'_4 - x'_3}{5-3} \le 0.5, \frac{x'_2 - x'_1}{2-1} \ge -0.5, \frac{x'_4 - x'_3}{5-3} \ge -0.5,$$
$$\frac{x'_3 - x'_1}{3-1} \le 0.5, \frac{x'_5 - x'_4}{7-5} \le 0.5, \frac{x'_3 - x'_1}{3-1} \ge -0.5, \frac{x'_5 - x'_4}{7-5} \ge -0.5,$$
$$\frac{x'_3 - x'_2}{3-2} \le 0.5, \frac{x'_6 - x'_5}{8-7} \le 0.5, \frac{x'_3 - x'_2}{3-2} \ge -0.5, \frac{x'_6 - x'_5}{8-7} \ge -0.5.$$

The corresponding transformation is as follows,

$$\frac{u_2 - v_2 + v_1 - u_1 - 12 + 12.5}{2-1} \le 0.5 \quad \dots$$
$$\frac{u_3 - v_3 + v_2 - u_2 - 12.5 + 13}{3-2} \le 0.5 \quad \dots$$
$$\frac{u_3 - v_3 + v_1 - u_1 - 12 + 13}{3-1} \le 0.5 \quad \dots$$

where $u_1 = \frac{|x'_1 - x_1| + (x'_1 - x_1)}{2}$, $v_1 = \frac{|x'_1 - x_1| - (x'_1 - x_1)}{2}, \dots$.

By solving the problem with these constraint predicate (using LP solvers), the global optimal solution is exactly the repair x' in Example 2, with $x'_4 = 14$ and the minimum repair distance 4.

Referring to Karmarkar's algorithm [12], it is sufficient to conclude that the global optimal repair problem is polynomial time solvable.

Corollary 8. *The global optimal repair can be computed in $O(n^{3.5}L)$ time, where n is the size of sequence, and L is the number of bits of input.*

B. GLOBAL OPTIMUM APPROXIMATION

As presented in Corollary 8, solving the global optimal repair problem in formula (2) by existing LP solvers is still costly, owing to the large number of speed constraint predicates in formulas (3) and (4). Rather than considering all the constraint relationships over the entire data set and eliminating all the violations at one time, existing approximate repair approaches [4, 14] often greedily repair the data (pairs) involved in violations, in multiple rounds.

In light of the greedy strategy in data repairing [4], we consider only the pairs x_i, x_j that violates the speed constraints, i.e., either $\frac{x_j - x_i}{t_j - t_i} > s_{\max}$ or $\frac{x_j - x_i}{t_j - t_i} < s_{\min}$.

$$\min \quad \sum_{i=1}^{n} |x_i - x_i'| \tag{13}$$

$$\text{s.t.} \quad \frac{x_j' - x_i'}{t_j - t_i} \le s_{\max}, \qquad \frac{x_j - x_i}{t_j - t_i} > s_{\max}, t_i < t_j \le t_i + w, \quad 1 \le i \le n, 1 \le j \le n \tag{14}$$

$$\frac{x_j' - x_i'}{t_j - t_i} \ge s_{\min}, \qquad \frac{x_j - x_i}{t_j - t_i} < s_{\min}, t_i < t_j \le t_i + w, \quad 1 \le i \le n, 1 \le j \le n \tag{15}$$

Its solution x' eliminates the violation between the aforesaid data points i and j. Since less constraint predicates are specified, the problem solving could be more efficient.

However, new violations may be introduced between the modified x_i' and x_k' in x', where x_i, x_k are not in violation before repairing (i.e., the constraint between x_i and x_k is not previously considered in formulas (14) and (15)). It needs to iteratively repair the violations introduced in the previous round of problem solving. As presented in [4], the iteration terminates till no new x_i' is modified in the last round.

The major issue of this approximation is the soundness of repairing w.r.t. speed constraints, i.e., not all the violations are guaranteed to be eliminated. The reason is that the greedy repair could be trapped in local optima, and the number of violations cannot be further reduced. In other words, as also observed in the experiments in Section 6, a large number of data are still in violation to the speed constraints in the returned approximate repair results.

Example 10 (Approximation, Example 9 continued)**.** *Rather than enumerating all the constraint predicates in Example 9 for global optimum, the approximation considers only the pairs of data points in violations, i.e., (x_3, x_4) and (x_4, x_5) as indicated in Example 2 (red lines in Figure 2(a)).*

According to formulas (14) and (15), the constraint predicates for approximation method are

$$\frac{x_4' - x_3'}{5 - 3} \ge -0.5, \frac{x_5' - x_4'}{7 - 5} \le 0.5,$$

which is exactly a subset of the constraint predicates in Example 9 for global optimum.

Let $x' = \{12, 12.5, 13, 12, 13, 15.5\}$ be the solution w.r.t. the aforesaid constraint predicates. The modified data points are $x_4' = 12$ and $x_5' = 13$ denoted by gray points in Figure 2(a). Note that the modification $x_5' = 13$ introduces a new violation between data points 5 and 6.

Therefore, another round of problem solving is evoked, with the following constraint predicate:

$$\frac{x_6' - x_5'}{8 - 7} \le 0.5.$$

Let the result be $x'' = \{12, 12.5, 13, 12, 15, 15.5\}$, where the modified data point is $x_5'' = 15$. That is, the value is changed back to the original x_5, and the computation is trapped in local optima. According to [4], since data point 5 has been repaired in the previous step and no new data point is modified in this iteration, the program terminates and leaves the violation unsolved between $x_4'' = 12$ and $x_5'' = 15$.

C. NORMAL APPROXIMATION FOR ADAPTIVE WINDOWS

Consider two window sizes w and w' in Figure 7 in Example 8. We show the Binomial probability mass function for $B(n, p)$ and normal probability density function approximation for $\mathcal{N}(np, np(1-p))$ in Figure 20. For w with $n = 4$ data points and $p = 0.91$, we plot $B(4, 0.91)$ and $\mathcal{N}(3.64, 0.3276)$ in Figure 20(a). For w' with $n = 10$ and $p = 0.52$, Figure 20(b) illustrates $B(10, 0.52)$ and $\mathcal{N}(5.2, 2.496)$.

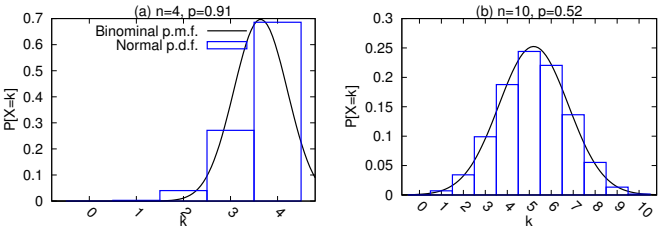

Figure 20: Distribution approximation

In normal approximation [2], if $np \pm 3\sqrt{np(1-p)} \in [0, n]$, then n is large enough, and p is far enough from the extremes of zero or one. That is, the normal approximation is appropriate for binomial distribution, e.g., in Figure 20(b). It is the case for w' as presented in Example 8. As shown in Figure 7, a large number of data points within w' are far from the speed constraints $s_{\min}$ and $s_{\max}$. In other words, the number of data points n is large enough, and p is far enough from the extremes of 0 or 1 referring to the modeling of $p_{i,j}$ in formula (12). Since speed constraints are useless in a large window, according to the intuition of adapting windows, the window may shrink for computation efficiency.

On the other hand, for w with $np \pm 3\sqrt{np(1-p)} \notin [0, n]$, the rule for evaluating normal approximation indicates that n is *not* large enough, and p is *not* far enough from the extremes of zero or one (i.e., the normal approximation is inappropriate as illustrated in Figure 20(a)). By enlarging w, more data points may be involved under the guard of speed constraints to ensure their correctness.

D. MORE DETAILS IN EXPERIMENTS

All programs are implemented in Java. Experiments were performed on a PC with 3.4 GHz CPU and 16 GB RAM.

D.1 Real Dataset Preparation

The STOCK[2] dataset records the daily prices of a stock from 1984-09 to 2010-02, with 12826 data points in total. Since the STOCK data is originally clean, following the same line of precisely evaluating the repair effectiveness [1], errors are injected by randomly replacing the values of some data

[2]http://finance.yahoo.com/q/hp?s=AIP.L+Historical+Prices

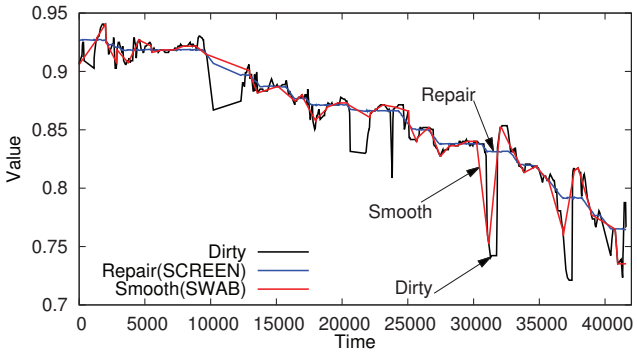

Figure 21: Example of real data FUEL

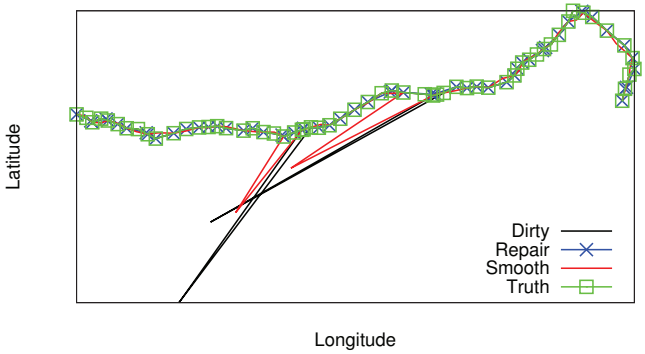

Figure 22: Example of real data GPS

points. For example, an error rate 0.1 denotes that 10% data point values are replaced. For each replaced data point, it takes a random value between the minimum and maximum values in the dataset.

The FUEL dataset gathers readings of fuel gauge of a crane, in every 20-40 seconds when the device is working. There are total 29750 data points collected in a period of 32 days. The entire sequence is divided into several segments by refueling, so that the speed constraint on fuel consumption does not apply across two segments separated by refueling. For the originally dirty FUEL, since the ground truth of each data point is unknown, we first filter out the dirty data points by speed constraints. The remaining data points that are clean w.r.t. speed constraints perform the aforesaid error injection (as done in STOCK).

In order to evaluate over a real dataset with *true errors* (instead of synthetically injected errors), a real GPS dataset is collected by a person carrying a smartphone and walking around at campus. Since we know exactly the path of walking, a number of 368 dirty points are manually identified (among total 3441 points in the trajectory). True locations of dirty points are also manually labeled, as ground truth.

Figures 21 and 22 present two segments of FUEL and GPS data, respectively, with the original dirty points, the truth points and the repairs. Similar to the STOCK example in Figure 1, large spikes (where a data point suddenly jumps far away from the path and the following one comes back soon) also appear in Figures 21 and 22. As shown, the smooth (SWAB) method is not able to correct such mistakes. Our proposed SCREEN repair can successfully address all the spikes, while preserve the original data largely. Consequently, as the results reported in Table 1, the accuracy of

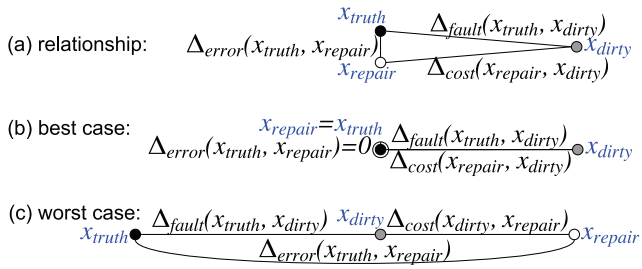

Figure 23: Accuracy measure

our constraint-based Global and Local repairing is significantly higher than that of smoothing methods.

D.2 Evaluation Criteria

Let x_{truth} be the ground truth of clean sequence, x_{dirty} be the error sequence with faults embedded, and x_{repair} be the repaired sequence. We denote $\Delta_{dist}(x_{repair}, x_{dirty})$ the distance paid in repairing, i.e., *the minimum change metric* in formula (2) in the optimization problem formulation.

The rationale of the minimum change metric [1] is built upon the discipline that people or systems always try to minimize mistakes in practice, i.e., minimizing the fault distance $\Delta_{fault}(x_{truth}, x_{dirty})$ between the ground truth and the dirty data. Following this discipline, it finds a repair that is closest to the input, i.e., minimizing the repair distance $\Delta_{dist}(x_{repair}, x_{dirty})$, as an *estimate* of the truth.

To study the relationship between $\Delta_{fault}(x_{truth}, x_{dirty})$ and $\Delta_{dist}(x_{repair}, x_{dirty})$, Figures 8(a) and 9(a) observe the fault distance $\Delta_{fault}(x_{truth}, x_{dirty})$ between the ground truth and the dirty sequence. As illustrated, e.g., in Figures 8(a) and (b), the fault distance is roughly *proportional* to the repair distance (for those approaches with accurate repairs such as Global and Local). The results verify the rationale of using the minimum distance repair to estimate the ground truth.

However, it is obvious to see that x_{repair} may not always be an *accurate estimate* of x_{truth}, even though both of them are minimized w.r.t. their distances to the input x_{dirty}. In Figure 23(c), although $\Delta_{dist}(x_{repair}, x_{dirty})$ is proportional to $\Delta_{fault}(x_{truth}, x_{dirty})$, x_{repair} may be far away from x_{truth}.

The *repair accuracy measures* evaluate how close the minimum distance x_{repair} estimates x_{truth}, e.g., by root-mean-square error (RMS) [10], which denotes the error distance $\Delta_{error}(x_{truth}, x_{repair})$ between the ground truth x_{truth} and the repaired sequence x_{repair}. The lower the RMS error $\Delta_{error}(x_{truth}, x_{repair})$ is, the closer (more accurate) the repair is to the ground truth.

It is notable that RMS error is not normalized (e.g., into a range of [0,1] for easy interpretation), and considers only the x_{truth} and x_{repair} but not their distances to the dirty data x_{dirty}. To this end, we further observe the normalized accuracy relative to x_{dirty} by

$$1 - \frac{\Delta_{error}(x_{truth}, x_{repair})}{\Delta_{dist}(x_{repair}, x_{dirty}) + \Delta_{fault}(x_{truth}, x_{dirty})},$$

According to triangle inequality on distances, in the worst case, we have $\Delta_{error}(x_{truth}, x_{repair}) = \Delta_{dist}(x_{repair}, x_{dirty}) + \Delta_{fault}(x_{truth}, x_{dirty})$ with $accuracy$=0. For the best repair results, $\Delta_{error}(x_{truth}, x_{repair}) = 0$, we have $accuracy$=1.