Time Series Data Cleaning under Expressive Constraints on both Rows and Columns

Xiaoou Ding*, Genglong Li†, Hongzhi Wang*, Chen Wang‡, Yichen Song§
*†School of Computer Science and Technology, Harbin Institute of Technology.
‡National Engineering Laboratory for Big Data Software, EIRI, Tsinghua University.
§Harbin Institute of Technology.
Email: *{dingxiaoou,wangzh}@hit.edu.cn, †1190400205@stu.hit.edu.cn, ‡wang_chen@tsinghua.edu.cn
§22S003013@stu.hit.edu.cn

***Abstract*—Devices with thousands of sensors generate billions of data points, and the generated time series data are suffering data quality problems. Traditional constraint-based techniques contribute much to data cleaning applications, however, cleaning methods which support expressive constraints on time series data are still insufficient. Since time series data has more noteworthy characteristics, existing cleaning approaches are challenged to provide good repair especially for continuous errors and correlated errors.**

Motivated by this, we propose a novel data cleaning method for time series according to expressive constraints supporting arithmetic operations between attributes and time-context constraints. In violation detection phase, we propose specialized violation degree quantification functions and design violation cell discovery algorithm to profile and identify errors hidden in time series. In data repairing phase, we formalize the cleaning task as one constrained optimization problem, and design repair objective considering both modification cost and conformance degree of constraints. We efficiently reduce the search space in repairing with evaluation of time-context constraints, and propose a bidirectional repairing algorithm. We provide theoretically analysis for the repairing method. Experiments results on 3 real IoT datasets with 5 metrics show that the proposed method outperforms 7 state-of-art cleaning techniques specialized for time series. We improve repairing effectiveness by 60% and reduce 70% time costs with involving constraints on temporal order into cleaning process.

***Index Terms*—data cleaning, constraint-based data repairing, time series data quality management**

I. Introduction

With the development of data acquisition technology and the wide application of intelligent sensors, time series data is being accumulated at an unprecedented rate in many fields such as industry, energy, healthcare, etc. Time series data are defined as sequences of values that are *continuously* measured and recorded at a specific time or fixed period [1], [2]. In time series data management applications including IoT field, devices with thousands of sensors generate billions of data [3], and high-quality data is challenged to be either obtained from machine sensors or maintained in databases. Since time series data management is one of the core issues for Edge-Cloud big data techniques, data quality (DQ) problems for time series have drawn more attention from both academia and industry [1], [2]. An important manifestation of this is that the data may significantly deviate from normal conditions, resulting in data errors. Low-quality time-series data not only leads to significant time and labor costs in the data preparation phase but also has a significant impact on the reliability of subsequent application activities such as process management, decision-making support.

However, compared to traditional relational data, time series has much more *noteworthy* characteristics, including temporal orders, strong correlation among attributes, and uncertainty in data acquisition process, etc. And data involves more complex quality issues [4]. Time series data is strongly correlated among sequences as well as over contextual time periods. On one hand, the complex relationships between multiple sequences makes it more challenged to either capture and repair data errors. One the other hand, errors in time series trend to have cumulative effects, other than occur in discrete time points. In such cases, utilizing existing cleaning algorithms to identify and repair errors in time series may result in inaccuracies. We present a motivated example for an IoT dataset to illustrate the problems of existing cleaning methods on time series data below.

Example 1: Figure 2 depicts data from a wind power plant, which describes the operating condition of induced draft fans. Here lists 4 attributes CT102, CT103, CT104 and CT111 recording the oil temperatures at different measurement points for the system. From the correlation matrix in Figure 2 (a), there is a strong correlation between these four attributes. Errors may occur at different time periods in attributes, as shown in Figure (b), which appears on multiple consecutive timestamps. We show the cleaning result of the second continuous error in CT111 in Figure 2 (c). We apply two state-of-art cleaning methods for stream data under speed constraints [5] and speed+acceleration constraints from [6], and displayed repair results using red and yellow lines, respectively. They both are seriously affected by the wrong data value at the beginning time point, and the cleaning results have a large gap with the ground truth. When attempting to repair errors with dependencies between attributes, we involve statistical constraints or CRRs to describe the relationship between the four attributes and get the result outlined in the purple. Since the repair process takes into account the regression function fitting with data from multiple attributes, the repair result fluctuated for many times. When considering both inter-

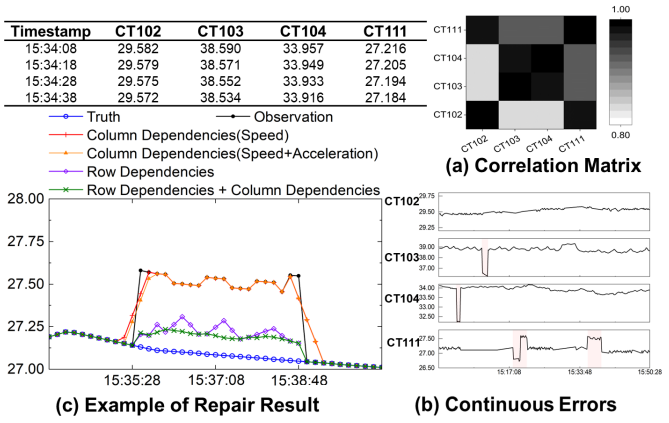


Fig. 1. Motivated example demonstration

attribute constraints and time context information, we achieve a better repair. ■

To summarize, the limitations of cleaning algorithms mainly for the reason that they only consider one aspect of correlation information, and fail to profile data from both rows and columns, which may lead to less sensitivity in identifying errors. In addition, time series data cleaning tasks highly require the applied constraints to support arithmetic operations instead of simple relationships e.g., $=, <, >$, and to support the relaxation of the exactly "equals" relationship in quality validation. Although expressive data quality constraints such as SC [7] , CC [8] , CRR [9] have been proposed in recent years, much attention has been paid in constraint discovery, while the cleaning methodologies that target complex constraints are insufficient.

When we attempt to clean multivariate time series taking into an overall consideration the complex relationship among data, it is obvious to be a challenged task! Difficulties come from many aspects, mainly including: (1) it demands well-designed cleaning algorithms considering the rich expressions of data quality constraints, which means answering the 3W2H question for specialized time series data cleaning. Considering the existence of numerical data, the satisfaction of rules by data is not binary, typically manifested as the degree of deviation (or violation) from a certain rule, and this will inevitably lead to (2) a huge search space for repairing multi-dimensional sequences in the continuous real domain. (3) it is of great necessity to involve robust mechanisms for errors with long durations, to prevent the repair result deviating significantly from the ground truth, which would possibly happen in time series database. In addition, (4) the cost functions which guides the rule-based cleaning for time series also needs to be re-examine. The traditional minimal repair principle [10] may not always lead to the most expected repair in numerical data, and worse still the non-binary violations are not easy to be evaluated in the repair process.

Contributions. Motivated by this, we studied an interesting problem about multivariate time series data cleaning, and proposed novel cleaning algorithms with the combination of expressive constraints from both attributes (*i.e.,* rows) and time orders (*i.e.,* columns). We summarize our contribution as follows:

(1) Problem formalization. We formalize time series cleaning problems with expressive constraints defined on both rows and columns, and propose a two-step method to clean multivariate time series both reliably and efficiently.

(2) Violation detection. We propose a monotonically increasing convex function to evaluate violation degrees of errors with expressive constraints, and solve the violation cell discovery problem to achieve accurately error detection for time series.

(3) Data repair. We formalize the cleaning task as one constrained optimization problem, and proposed a novel bidirectional-repairing algorithm to make a wise and effective decision on the specific value modification among violations, applying both complex arithmetic dependencies among sequences and temporal contextual correlations. We reduce the repair search space obviously with evaluation of time-context constraints.

(4) Experimental Evaluation. We perform sufficient experiments on 3 real-world IoT datasets with 5 metrics, and compared 7 state-of-art cleaning techniques specialized for time series data. The proposed method captures and repairs errors more completely and accurately. The accuracy of the proposed method is significantly higher than that of existing methods, with an average increase of 2.5 times. We improve metric *MNAD* by 30% and *RRA* by 40% compared to the cleaning method using only row-constraints. We achieve the reduction of cleaning time cost by 70% and at the same time improving precision by 60%+, with involving constraints on temporal order into cleaning process.

Our codes, dataset samples, and extended paper version are available at Github[1].

Organization. The rest of this paper is organized as follows. Section II introduces the overview of the proposed problem. Section III discusses the violation detection phase, and Section IV introduces the repairing method with essential theoretical argumentation and examples. We report the experimental results in Section V, introduce related works in Section VI, and draw our conclusion in Section VII.

II. Problem Overview

A. Preliminaries

$I = \{S_1, ..., S_M\}$ contains aligned M-dimensional (a.k.a M-attribute) time series. For each sequence $S = \langle s_1, ..., s_N \rangle$, $s_n = (x_n, t_n), (n \in 1, 2, ..., N)$ denotes a cell in S, where x_n is a real-valued number with a time point t_n. Below we use $S[t_n]$ as the value of S at time t_n for short. $W = (i, j)$ denotes a time window containing several consecutive time points from t_i to t_j. A subsequence $S[W] = \langle s_i, ..., s_j \rangle$ is the projection of S on W. Let S_Ω denote several selected sequences from I, i.e., $\Omega \subseteq \{1, 2, ..., M\}$ $(|\Omega| \geqslant 1)$. All the cells with the same t in S_Ω form a 1d-vector $S_\Omega[t]$. The domains of the vector $S_\Omega[t_n]$ are specified as $Dom^\Omega = \prod_{\forall m \in \Omega} \mathbf{Dom}_m$. The

[1]https://github.com/Aries99C/Clean4MTS

projection of sequences I_Ω on W forms a $(j-i+1) \times |\Omega|$ matrix, denoted as $S_\Omega[W]$.

***Definition** 1:* **Expressive DQ constraint among attributes** constrain the relationships between sequences on the same timestamp t or in a given time interval, denoted as $\varphi\,(g, \Omega, \rho)$, where $g : Dom^\Omega \longmapsto \mathbb{R}$ is the predefined projection in the form of a function supporting complex arithmetic operations (including ML models) and ρ denotes the bias. Given data vector $S_\Omega[t_n]$ and a tolerate threshold τ, it has $S_\Omega[t_n] \models \varphi$, iff it satisfies

$$|g\,(S_\Omega\,[t_n])\,| \leqslant \tau,$$

Otherwise, *i.e.,* $|g\,(S_\Omega\,[t_n])\,| > \tau$, $S_\Omega[t_n]$ is regarded to violate φ, denoted as $S_\Omega[t_n] \not\models \varphi$.

Note that the expressive DQ constraint φ not only support traditional constraints *e.g.,* denial constraints (DC) [11], but also is compatible with the state-of-art complex DQ constraints, including conformance constraints (CC) [8], statistical constraints [7], and condition regression rule (CRR) [9], which are able capture linear arithmetic dependencies among multiple attributes and support values within an acceptable range (or deviation) rather than an exact value. We mainly use CRR in our cleaning solution, for it is applicable in the context of multidimensional time series, and we briefly introduce CRR in Definition 2 referring to [9]. For example, $\varphi_{\text{CRR}}, \mathbb{C}$: $(f, \Omega_{\mathsf{Date} \cup \mathsf{Latitude}}, \rho = 0.5$ denotes that the linear regression $|\mathsf{Latitude} - f(\mathsf{Date})| \leq 0.5$ applies to condition $\mathbb{C}$ e.g, tuples with $\mathsf{Date} \geq$ *2006-9-12*.

***Definition** 2:* **φ-constraint instance: Condition regression rule (CRR).** One CRR denotes to be φ_{CRR}: $(f, \Omega_{X \cup y}, \rho)$, where $f : X \longmapsto y$ is a regression function from a vector formed by the sequences S_{Ω_X} to the value in the sequence S_{Ω_y} in the same row. $I \models \varphi_{\text{CRR}}$ iff $\forall t \in T$, $|S_{\Omega_y}\,[t] - f\,(S_{\Omega_X}\,[t])\,| \leqslant \rho$. Otherwise, $I \not\models \varphi_{\text{CRR}}$.

We then formalize the DQ constraints for temporal orders in Definition 3.

***Definition** 3:* **DQ constraint for temporal orders**, denoted as $\psi(h, S, \omega, lb, ub)$, requires that all ω consecutive points in S in chronological order are supposed to satisfy

$$lb \leqslant h(S[t_i], ..., S[t_{i+\omega}]) \leqslant ub,$$

where h is the predefined function expressing the temporal dependencies, such as speed constraints [5] of the value change. lb and ub are the tolerate lower bound and upper bound of h, respectively. Subsequence $S[W]$ is regarded to satisfy ψ iff the h measurement of each subsequence with the length of ω in $S[W]$ (assume that $|S[W]| > |\omega|$) locates in $[lb, ub]$. Otherwise, $S[W] \not\models \psi$.

Specifically, we involve state-of-art constraints *i.e.,* speed constraints (SC) [5] and acceleration constraints [6] in our solution. We briefly introduce SC in Definition 4.

***Definition** 4:* **ψ-constraint instance: Speed constraint (SC).** One SC defined on sequence S is formalized as $\psi_s\left(\frac{\Delta x}{\Delta t}, S, 1, s_{\min}, s_{\max}\right)$, which restricts the speed of value change between two adjacent timestamps in $[s_{\min}, s_{\max}]$. That is, $\forall t_j - t_i \leq w_m$, $s^m_{\min} \leqslant \frac{S_m[t_j] - S_m[t_i]}{t_j - t_i} \leqslant s^m_{\max}$. Such SC could be put as $\psi = [s_{\min}, s_{\max}]$ for short.

Such SC could be put as $s = [s_{\min}, s_{\max}]$ for short, and $[-0.6, 1]$ denotes the increase amount of oil temperature should not be larger than $1°C$ while the decrease amount is not larger that $0.6°C$.

B. Problem Statement

We propose the studied cleaning problem in Problem 1.

PROBLEM *1:* Given a dirty time series data instance I, and a set of data quality constraints $\Sigma = \Sigma_\varphi \cup \Sigma_\psi$ specified for data I including both constraints on attributes and time orders, the expressive-constraint-based cleaning of I has two steps: *i*) *violation detection*, where errors in I are identified by violation detection algorithms *w.r.t* Σ, and *ii*) *data repair*, where I is modified to I' such that I' does not contain any violations of Σ, *i.e.,* $I' \models \Sigma$.

We stress that the problem description is in the similar form with the traditional rule-based cleaning problem formalized in [12], however, both the detection and the repair steps require more novel and robust mechanism specialized for time series data as mentioned in Section I. For the violation detection phase, since that it is not enough to only determine whether I satisfy an expressive constraint or not, especially for the errors, detection techniques are expected to capture errors sensitively and completely as possible. We propose the violation detection algorithm *ErrDetect*, which achieves well-profiling of violations with quantifying function of violation degree *w.r.t* a DQ constraint and an efficient error location function.

For the repair phase, it cannot be ignored that not all of the cells in a violation detected are actually error data. The critical things to achieve good repair are well-profiling the violations and taking advantage of the semantic information of the quality constraints to determine the modification of a subset of cells. As most existing techniques clean data according to the "minimum repair principle" and the guarantee of the elimination of violations, we propose repairing algorithm and take into account relationships in both rows and columns, which also efficiently reduces the candidate repairing space with ψ-type constraints on temporal orders. During the process, we specialize the "minimum repair principle" on time series, and comprehensively achieve minimizing the degree of violations.

III. VIOLATION DETECTION

A. Detection algorithm overview

Since that expressive φ-constraints allow the expressed regression relationships for time series data for some deviation *w.r.t* ρ, so essentially, it also describes the degree to which the data satisfies the constraint, rather than a simple binary satisfaction. In order to more accurately capture different patterns of errors in time series (such as slow changes or sudden changes), we propose violation degree quantification function in Definition 5 to well profile the feature of errors, and thus achieve effective detection. Below we use Ω to

Algorithm 1: VIOCELLDISCOVER

Input: Data I, constraint φ expressing complex association between sequences
Output: Violation set $\mathcal{VC}$

foreach $t \in T$ **do**
  Update Pro with t and $vd(S_\Omega[t])$;
$ProSegs \leftarrow$ Divide Pro by pruning all-negative subsequences longer than Δ_w ;
foreach $Pro \in ProSegs$ **do**
  $\mathcal{VC}_l \leftarrow$ Violation cell determination according to Pro;
  Update $\mathcal{VC}$ with $\mathcal{VC}_l$;
return $\mathcal{W}$

replace $\Omega_{X\cup y}$ to denote the set of indexes of sequence in I for simplicity.

***Definition** 5:* **Violation degree quantification function** $vd(\cdot)$. Given data I and one expressive DQ constraint *e.g.*, $\varphi_{\text{CRR}}(f, \Omega, \rho)$, its violation degree at time t is quantified as

$$vd(S_\Omega[t]) = \begin{cases} 1 - e^{-(|S_{\Omega_y}[t] - f(S_{\Omega_X}[t])| - \rho)/\rho}, & S_\Omega[t] \not\models \varphi_{\text{CRR}} \\ \frac{|S_{\Omega_y}[t] - f(S_{\Omega_X}[t])| - \rho}{\rho}, & S_\Omega[t] \models \varphi_{\text{CRR}}. \end{cases}$$

And we introduce the properties of $vd(S_\Omega[t])$ function in Proposition 1.

***Proposition** 1: Function* $vd(S_\Omega[t])$ w.r.t *one* φ_{CRR} *has critical properties including:* i) $S_\Omega[t_n] \models \varphi_{\text{CRR}} \Leftrightarrow -1 \leqslant vd(S_\Omega[t_n]) \leqslant 0$, *while* $S_\Omega[t_n] \not\models \varphi_{\text{CRR}} \Leftrightarrow 0 < vd(S_\Omega[t_n]) < 1$. *While* $0 < vd(S_\Omega[t_n]) < 1$ *is regarded as positive violation degree, and a larger* $vd(\cdot)$ *value represents the observed data deviate from the expected relationship among them; and* ii) *to take* $|S_{\Omega_y}[t] - f(S_{\Omega_X}[t])|$ *as independent variable of the function,* $vd(S_\Omega[t])$, $vd(S_\Omega[t])$ *a monotonically increasing convex function.* ■

With the designed violation detection mechanism discussed below, such properties contribute to capturing errors spanning a period of time more completely while reducing the involvement of clean data into the violation set.

***Definition** 6:* **Violation cell discovery problem**. Given expressive φ-constraint, *e.g.,* φ_{CRR}, let $VC = (W, \varphi_{\text{CRR}})$ be a violation cell, where $W = (s, e)$ is a time window starting form t_s and ending in t_e. The violation detection on I is solved by discovering all such violation cells $\mathcal{V}$ from I, which satisfy the following requirement: *i*) **Mean-positive**. The mean of violation degree at each time point of window W is positive, *i.e.,* $\forall n \in [s, e]$, *Mean*$(\{vd(S_\Omega[t_n]\}) > 0$. *ii*) **Obvious-boundary**. Each cell has obvious boundaries *w.r.t* violation degree functions, *i.e.,* $vd(S_\Omega[t_s]) > 0$ and $vd(S_\Omega[t_e]) > 0$, while $vd(S_\Omega[t_{s-1}]) \leqslant 0$ and $vd(S_\Omega[t_{e+1}]) \leqslant 0$. *iii*) **Independent**. Each tuple in I could only be identified to be contain in one violation cell, i.e., $\forall VC_i, VC_j \in \mathcal{V}$, $VC_i.W \cap VC_j.W = \emptyset$. *iiii*) **Maximal**, To achieve concise and effectiveness, each VC satisfying the above requirement has the maximal length. That is, given violation cell $VC.W = (s, e)$, $\nexists$ such a violation cell $VC.W = (s', e')$ that $s' < s$ and $e' > e$.

We first present the whole algorithm of VCs discovery in Algorithm 1, along with the example shown in Fig. 2 which mainly contains three steps.

Step 1: Violation degree Quantification. We first evaluate the satisfaction of I *w.r.t* the given φ_{CRR}, and obtain a violation degree sequence $S_{vd} = \{vd(S_\Omega[t_1], ..., vd(S_\Omega[t_M])\}$, which has the same length with I. S_{vd} consists of elements ranging from -1 to 1, alternating between all-positive and all-negative subsequences (see Section III-B).

Step 2: Sequence profiling. As we are more interested in violations rather than normal data, we are able to prune all-negative subsequences using a length threshold Δ_w to improve cleaning efficiency. Thus, we obtain several sequence fragments, where some are all-positive sequences, while some are composed of alternating positive and (short) negative sequences (see Section III-C).

Step 3: Violation cell determination. Aiming to determine whether the segments are real error data or not, we introduce the *gain* of violation degree for each positive- or negative-subsequence in one segment. For the start time point of each segment, we compute whether the currently accumulated gain is greater than 0. If the violation gain of the entire segment is ultimately greater than 0, it is regarded as a real violation cell. We compute and decide each violation cell according to Definition 6 with the solution to the longest positive subsequence sum problem (see Section III-D).

B. Segmentation on violation degree sequence

After obtaining the violation degree sequence *w.r.t* φ: $S_{vd} = \{vd(S_\Omega[t_1], ..., vd(S_\Omega[t_M])\}$, it could be found that S_{vd} contains alternate all-positive and all-negative subsequences as formalized in Definition 7. To focus on the candidate errors rather than data in normal values, we prune all-negative subsequences from S_{vd} with a length threshold Δ_w, and do not consider process them any more. That is, if data satisfy φ for sufficient time duration over Δ_w, it is considered to be clean. In this case, we focus on potential violations and avoid processing the clean data.

***Definition** 7:* (**All-positive & negative subsequence**). A subsequence $S_{vd}[s, e]$ starting from t_s and ending in t_e is regarded as all-positive (resp. all-negative) subsequence if the violation degree of all the data points in $[t_s, t_e]$ is larger (resp. smaller) than 0, denoted by $S^+_{vd}[s, e]$ (resp. $S^-_{vd}[s, e]$).

There remains several discrete segments in S_{vd} after pruning. There are two kind of patterns for the segment, *i.e.,* the segment only contains positive violation degree values (see Segment 1 in Fig. 2), denoted by $S_{vd}[s, e] = S^+_{vd}[s, e]$, and the segment consists of alternate $S^+_{vd}[s', e']$s and small-length $S^-_{vd}[s'', e'']$s, *e.g.,* Segment 2,3,4,5 in Fig. 2. We note that the later segment patterns are common in real IoT data, for complex errors are likely not be identified as violations at all timestamps, which may fall within the bias range in a few time points *w.r.t* the given constraint. We profile and determine real violations for these segments in the follow-up procedure.

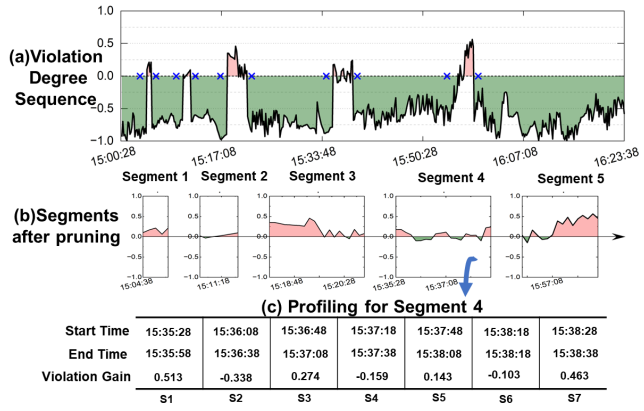


Fig. 2. Demonstration of violation detection procedure

C. Segments profiling

Since both all-positive and all-negative subsequences may exist in one segment, we propose violation gain in Definition 8 to effectively evaluate each segment, which also assist to reduce repeated calculations in the detection phase.

***Definition** 8:* (**Violation gain**). Given an all-positive subsequence $S^+_{vd}[s,e]$ (or $S^-_{vd}[s,e]$), the violation gain of such subsequence describes the sum of violation degree values, denoted as

$$\textsc{Gain}[t_s,t_e]=\sum_{\forall t\in[t_s,t_e]} vd\left(S_\Omega\left[t\right]\right).$$

Since we prune all-negative subsequences with length longer than Δ_w, each segment remained to start with an all-positive subsequence and also end with an all-positive subsequence. There are $2\gamma-1$ such all-positive or all- negative subsequences in total (where $\gamma\in\mathbb{N}^+$) in a segment, denoted as $S^+_{vd}[s^1,e^1], S^-_{vd}[s^2,e^2], S^+_{vd}[s^3,e^3],...,S^-_{vd}[s^{2\gamma-2},e^{2\gamma-2}]$, $S^+_{vd}[s^{2\gamma-1},e^{2\gamma-1}]$. Accordingly, we evaluate each positive and negative subsequence in one segment with a triple $(s,e,\textsc{Gain}[s,e])$, and thus we are able to profile S_{vd} with maintaining a *Pro* set $\{(s^1,e^1,\textsc{Gain}[s^1,e^1]),...,(s^{2\gamma-1},e^{2\gamma-1},\textsc{Gain}[s^{2\gamma-1},e^{2\gamma-1}])\}$, where γ is the total number of S^+_{vd}s. The profile set ranking odd are gained from the locally longest segments with all positive degrees and the others are derived from the ones with all positive degrees. Obviously, such results could be obtained by streaming process.

D. Violation cell determination

We finally identify real errors from the candidate violations according to the *Pro* set.

Given $Pro=\{(t^1_s,t^1_e,\textsc{Gain}[t^1_s,\ t^1_e]),...,\ (t^{2\gamma-1}_s,t^{2\gamma-1}_e,\textsc{Gain}[t^{2\gamma-1}_s,t^{2\gamma-1}_e])\}$, as shown in Fig. 2(c), this problem requires to find μ timestamps in $\mathcal{T}^l_s$, $\mathcal{T}^l_e$ of Pro ranking odd respectively (with positive gain), *i.e.*, $t^{k_1}_s,...t^{k_\mu}_s$ and corresponding $t^{k_1}_e,...t^{k_\mu}_e$ sort by time, where $k_\mu\leqslant 2\gamma-1$ and $\forall i\in[1,\mu]$ k_i is odd. And they constitute μ different time windows, *i.e.*, (s^α,e^α) ($\alpha\in\{k_1,...,k_\mu\}$) which should meet all of the following conditions: (1) μ should be minimal; (2) Any start and end time point of the locally longest segments with all positive degrees will be covered by these windows and the gain of the segments corresponding to each time window (s^α,e^α) should be positive, *i.e.*, $\textsc{Gain}[t^\alpha_s,t^\alpha_e]>0$.

For the solution of the above problem, we use Pro to construct an unweighted digraph $\mathcal{G}_{gain}\left(V,E\right)$. We initialize the each vertex v_k in the graph $\mathcal{G}_{gain}\left(V,E\right)$ with all triples in Pro with positive gain, *i.e.*, $\left(t^{2k-1}_s,t^{2k-1}_e,\textsc{Gain}[t^{2k-1}_s,t^{2k-1}_e]\right)$, where $k\in\{1,..,\gamma\}$. For ease of calculation, we set a termination vertex $v_{\gamma+1}$. We have set the following connection rules for directed edges between vertexes in $\mathcal{G}_{gain}\left(V,E\right)$. If a directed edge $e_{i,j}$ $(1\leqslant i<j\leqslant\gamma+1)$ from v_i to v_j, the gain of the segmented, *i.e.*, $Gain\left[t^{2i-1}_s,t^{2j-3}_e\right]$, specified by the time window $\left(s^{2i-1},e^{2j-3}\right)$ should exceed 0. The above problem is thus transformed to the following problem. Given the constructed unweighted digraph $\mathcal{G}_{gain}\left(V,E\right)$, find the path from v_1 to $v_{\gamma+1}$. According to the rules for constructing edges defined above, as $\textsc{Gain}\left[t^{2i-1}_s,t^{2i-1}_e\right]>0$, there is at least one edge from v_i to v_{i+1} ($\forall i\in\{1,...,\gamma\}$). Thus there is at least one path from v_1 to $v_{\gamma+1}$.

VIOCELLDISCOVER searches for the objective path while building directed edges in the graph $\mathcal{G}_{gain}\left(V,E\right)$ with a vertex queue VQ and an array TA initialized to 0 with length $\gamma+1$ to record the index of the previous vertex has been traversed. The algorithm execution process is as follows:

(1) It starts at the vertex v_1, and traverse all the following vertexes $v_2,...,v_{\gamma+1}$. Assume that the vertex traversed currently is v_j, if $\textsc{Gain}\left[t^1_s,t^{2j-3}_e\right]$ is positive, then put v_j in VQ and set $TA\left[j\right]=1$.

(2)Execute a loop. If the VQ pops up with $v_{\gamma+1}$, the loop is terminated. Otherwise, assuming that the vertex is v_i, traverse all the following vertex $v_{i+1},...,v_{\gamma+1}$. When v_j is scanned, if $TA\left[j\right]$ equals 0 and $\textsc{Gain}\left[t^1_s,t^{2j-3}_e\right]$ is positive, then put v_j in VQ and set $TA\left[j\right]=i$.

(3) Find the path through TA, all directed edges $e_{i,j}$ in it correspond to a time window (s^{2i-1},s^{2j-3}) for localizing a violation. All the windows constitute a set $\mathcal{VC}.W$.

IV. Data Repairing

A. Repairing algorithm overview

We next perform repairing on the uncovered violations from the above step. Existing rule-based data cleaning approaches generally follow two goals: 1) to modify data to eliminate violations of a given constraint, and 2) to modify less data as possible (known as the minimum repair principle), as people are more interested in repairs where both the number of the repaired cells and the value changes of cells are minimized. We follow the above goals in time series cleaning, in addition, it is necessary to consider updating data to fit the complex constraint expression better. Otherwise, the repair result would theoretically fall on the boundary required by φ. When ρ is set large, such repairs fail to be accurate. Therefore, we propose to reduce the violation degree to ensure the quality of the repair result. Accordingly, we propose a novel repair cost for time series below.

***Definition** 9:* Let I' be one possible repair of I, the **repair cost** between I and I' under a given φ at time t is

$$cost(I, I') = \lambda_1 dist_1\left(S_\Omega\left[t\right], S'_\Omega\left[t\right]\right) + \lambda_2 dist_0\left(S_\Omega\left[t\right], S'_\Omega\left[t\right]\right)$$

where l_1-distance describe the average value change in a violation cell, denoted by $dist_1 = \sum_{\forall m \in \Omega} \frac{|S_m[t] - S'_m[t]|}{range(S'_m[t])}$, and l_0-distance describes the number of the changed cells, which can be denoted by a binary function $dist_0\left(S_m\left[t\right], S'_m\left[t\right]\right) = 0, \text{when} S_m\left[t\right] = S'_m\left[t\right]$, and $dist_0\left(S_m\left[t\right], S'_m\left[t\right]\right) = 1$, when $S_m\left[t\right] \neq S'_m\left[t\right]$. $range\left(S'_m\left[t\right]\right)$ is the value range including the maximum value modification of $S'_m\left[t\right]$ of every candidate repair result in the search process, in which $|S_m\left[t\right], S'_m\left[t\right]|$ is normalized. λ_1 and λ_2 are two constants to achieve the weight combination of the l_1-distance and l_0-distance.

Note that λ_1 and λ_2 are key factors that assist to maintain balance between the repair cost and the degree of conformance to the quality constraint in the repair phase. The weights had better be significantly greater than 1 intuitively, especially for λ_1 considering l_1-distance is continuous while l_0-distance is discrete. λ_2 restricts the number of cells to modify in a "row". The influence of choice of λ_1 and λ_2 on the performance of our proposed methods will be analyzed further with the experimental result in Section V-B1. Instead of directly use the range of domain $\mathbf{Dom}_m$, we calculated it after pre-pruning by means of ψ-constraints expressing temporal dependency, as discussed in Section IV-C below.

***Proposition** 2: The domain of $cost(I, I')$ is $[0, \lambda_1 + \lambda_2]$.*

Proposition 2 guarantees violations to be resolved after repairing, as further explained in Section IV-D. With the specially-designed repair cost, we propose the repair objective function for the studied problem with repair cost combined with the corresponding violation degree. Thus, we are able to formalize the repair problem as a constrained optimization problem in Definition 10.

***Definition** 10:* **The constrained optimization repairing problem**. Given data I, the detected set of violation cells *w.r.t* a φ-constraint, and ψ-constraints on sequences $S \in S_\Omega$, the clean data I' w.r.t φ and ψ is obtained by solving the problem

$$\min_{S'_\Omega[t]} \ cost(I, I') + vd\left(S'_\Omega\left[t\right]\right)$$
$$s.t. \begin{cases} vd\left(S'_\Omega\left[t\right]\right) \leqslant 0 \\ \forall m \in \Omega, S_m[t] \models \psi. \end{cases}$$

We highlight the search space on multi-dimensional time series data is quite large with the naive solution which searches the repair result in the value ranges of the corresponding constraints. Result may also be unsatisfactory without considering the temporal dependencies, as shown in Example 1. To solve this, we introduce temporal constraints in the repair process. However, the introduction would add to the complexity of the feasible region required by constraints on both rows and columns. That is, it costs $(e-s+1)|\Omega|$ time to directly figure out the global optimum with strict restriction to consider all the cells in the detected time window $VC.W = [s, e]$. We propose a bidirectional repair method in Algorithm 2, which requires the modified data to strictly satisfy expressive constraints on rows, and effectively reduces the repair search space by using the temporal correlation constraint without increasing the number of cells to be modified at the same time. In this case, the introduced ψ-constraints bring two advantages: one is to reduce the larger search space, and the other is to ensure that data conforms to the temporal relationship, so that the repair results are more in line with expectations.

Algorithm 2: DATAREPAIR

Input: the dirty I with the detected set $\mathcal{VC}$ of violation cells, the set Σ_ψ of ψ-constraints
Output: I after repair

1 **foreach** $VC = (W = [s, e], \varphi) \in \mathcal{VC}$ **do**
2 **while** $t_s < t_e$ **do**
3 $SS\left(t_s, \Omega\right) \leftarrow$ PREPRUNE $\left(\Sigma_\psi, S_\Omega, t_s, VC\right)$;
4 $SS\left(t_e, \Omega\right) \leftarrow$ PREPRUNE $\left(\Sigma_\psi, S_\Omega, t_e, VC\right)$;
5 $cost\left(I, I', t_s\right) \leftarrow$ GETCOST $\left(SS\left(t_s, \Omega\right), S_\Omega\left[t_s\right]\right)$;
6 $cost\left(I, I', t_e\right) \leftarrow$ GETCOST $\left(SS\left(t_e, \Omega\right), S_\Omega\left[t_e\right]\right)$;
7 $S'_\Omega\left[t_s\right] \leftarrow$ EVOREPAIR $\left(cost\left(I, I', t_s\right), SS\left(t_s, \Omega\right), vd\left(\cdot\right)\right)$;
8 $S'_\Omega\left[t_e\right] \leftarrow$ EVOREPAIR $\left(cost\left(I, I', t_e\right), SS\left(t_e, \Omega\right), vd\left(\cdot\right)\right)$;
9 Update I with $S'_\Omega\left[t_s\right]$ and $S'_\Omega\left[t_e\right]$;
10 $t_s \leftarrow t_{s+1}$, and $t_e \leftarrow t_{e-1}$;
11 **if** $t_s = t_e$ **then**
12 Execute the steps on line 3,5,7,9;
13 **return** I'

We clean each violation cell VC obtained from Algorithm 1, and begin with the start time t_s and the end time t_e of VC. The repair is performed from both sides to the middle in VC. Each iteration for the current (s, t) pair consist of three steps, namely search space pre-pruning with ψ-constraints, repair cost generation, and evolutionary-based repairing.

Step 1: Search space pre-pruning with ψ-constraints. ψ-constraints describing data dependencies on temporal orders are used to to prune the meaningless candidate search space, so as to effectively narrow the candidate repair range. Such constraints e.g., speed constraints could be discovered from clean data in preprocessing. We design function PREPRUNE *w.r.t* ψ-constraints and the feasible solution that eliminates the violation on φ-constraints is guaranteed to be found in the pruned search space (see Section IV-B).

Step 2: Repair cost generation. Note again the repair cost consists of the average normalized l_1-distance term with continuous values and the normalized l_0-distance term with discrete values before and after modification. The design of $range(S[t])$ in l_1-distance is important which guides the modification of the data. Thus, we propose function GETCOST to obtain the current repair cost with the pruned search space (see Section IV-C).

Step 3: Evolutionary-based repairing. With the repair cost, we perform the solution of the constrained optimization problem for the current s and t. Considering the excellent performance of evolutionary algorithms in dealing with optimization

problems with complex objective functions and constraints [13], we design a customized evolutionary repairing algorithm EvoRepair, where the population initialization and fitness function are well-designed to ensure to achieve the theoretical optimal solution and to guarantee the convergence of repairing. Specifically, regardless of the number of iterations, the individual with the best fitness function in the population will certainly be able to eliminate the violation of the φ-constraints (see Section IV-D).

At the last step of each iteration, Algorithm 2 updates I with the repair version $S'_\Omega[t_s]$ and $S'_\Omega[t_e]$, and then lets t_{s+1} and t_{e-1} becomes the new start and end point. The repair process will finish until all timestamps in this cells have been visited.

B. Preprune with ψ-constraints

The detection phase exactly locates the time intervals involving errors, and identifies sequence set S_Ω *w.r.t* the violated constraint. However, to determine which cells in $S_\Omega[t]$ are actually dirty needs further computation. Without considering temporal dependencies in data, the search space of naive solution of the optimization problem in Definition 10 is quite huge, for it searches in all the sequences in S_Ω involved the violated φ. Besides the costly computation, the naive solution is also easy to over-repair the clean data. Thus, we make use of ψ-constraints to express temporal dependencies to prune the corresponding search space, and the expected repairs are regarded to lie in search space.

***Definition** 11:* **Search Space**. Given a $VC = (W, \varphi)$, let the value range on $S_m[t]$ (for $\forall m \in \Omega$) be $SS(t, m)$. the repair search space at t represents the value ranges of all cells in t, denoted as

$$SS(t, \Omega) = \prod_{\forall m \in \Omega} SS(t, m).$$

We apply speed constraint $\psi = (s_{\min}, s_{\max})$ [5] as the instance which describes the changing speed between adjacent timestamps or within a time duration. Before Preprune function, we initialize both $SS(t_s, m)$ and $SS(t_e, m)$ ($\forall m \in \Omega$) to v_m, and then prune $SS(t_s, m)$ and $SS(t_e, m)$ from the speed constraint ψ with $S_m[t_{s-1}]$ and $S_m[t_{e+1}]$ outside $VC.W$ respectively, which consists reliable accurate temporal information.

According to the property of speed constraint [5], given an accurate speed constraint ψ, if there is a cell expected with the ground truth, ψ limits the value range of another cell in the same sequence, whose timestamp t_j is within the range of $[t_j - w_m, t_j + w_m]$. If $j < i$, the value range of $S_m[t_i]$ derived is $[S_m[t_j] + s^m_{\min}(t_i - t_j), S_m[t_j] + s^m_{\max}(t_i - t_j)]$. Otherwise, it has $[S_m[t_j] + s^m_{\max}(t_i - t_j), S_m[t_j] + s^m_{\min}(t_i - t_j)]$. If $S_m[t_j]$ is *clean*, and t_j and t_i are two consecutive time points in T, ψ and $S_m[t_i]$ can specify one of the most competent ranges to ensure temporal dependencies expressed by the speed constraint. Also, we assume that in each speed constraint s_m, $s^m_{\min} \leqslant 0 \leqslant s^m_{\max}$, which is considered necessary in [14].

During Preprune function, in the cases that $t_{e+1} - t_s > w_m$ and $t_e - t_{s-1} > w_m$, $SS(t, m)$ is easily derived. The lower bounds of $SS(t_s, m)$ and $SS(t_e, m)$ are respectively

$$\min(v^m_{\min}, S_m[t_{s-1}] + s^m_{\min}(t_s - t_{s-1})),$$
$$\min(v^m_{\min}, S_m[t_{e+1}] + s^m_{\max}(t_e - t_{e+1})).$$

while the upper bounds are

$$\max(v^m_{\max}, S_m[t_{s-1}] + s^m_{\max}(t_s - t_{s-1})),$$
$$\max(v^m_{\max}, S_m[t_{e+1}] + s^m_{\min}(t_e - t_{e+1})).$$

It gets more complicated for the cases that $t_{e+1} - t_s \leqslant w_m$ or $t_e - t_{s-1} \leqslant w_m$. That is because $S_m[t_{e+1}]$ and ψ influence $SS(t_s, m)$ as $t_{e+1} - t_s \leqslant w_m$. To avoid the extremely rare conflicts, Preprune prioritizes these restriction. More exactly, take $SS(t_s, m)$ for example, it must be with in the value range derived by the union of v_m and that determined by ψ and $S_m[t_{s-1}]$. Thus, $SS(t_s, m)$ is initialized as $[\min(v^m_{\min}, S_m[t_{s-1}] + s^m_{\min}(t_s - t_{s-1})), \max(v^m_{\max}, S_m[t_{s-1}] + s^m_{\max}(t_s - t_{s-1}))]$. Based on this, only when the value range derived by ψ and $S_m[t_{e+1}]$ intersects current $SS(t_s, m)$, then it is updated to the union of them. Otherwise, $SS(t_s, m)$ remain not changed. Further, to guarantee the repair result to resolve the violation of φ, when necessary, Preprune does the minimal extension to make them just cover $S_m[t_{s-1}]$ or $S_m[t_{e+1}]$ respectively. at the last step of determining $SS(t_s, m)$ or $SS(t_e, m)$. We note that such cases rarely happen in practice.

We provide the analysis results of the effectiveness issues of function Preprune in the whole repair process below.

***Proposition** 3: With reliable speed constraints for trustworthy historical data, $SS(t_s, m)$ and $SS(t_e, m)$ are not empty and the repair result S'_Ω satisfying φ is guaranteed to be found within $SS(t_s, \Omega)$ and $SS(t_e, \Omega)$. Once the repair resolves the violation, it could eliminate the violation in the next iteration.*

***Proposition** 4: With Preprune on ψ-constraints, given $S_m[t_n]$, it is at least reduced to $\frac{(s^m_{\max} - s^m_{\min})}{v^m_{\max} - v^m_{\min}}\Delta t$ times, where $s^m_{\min}$ and $s^m_{\max}$ are the bounds of the speed constraint ψ, and Δt is a time interval between t_n and the closes timestamp outside the current violation cell.*

As quite a few violation cells need to be repaired, the reduction result in Proposition 4 is considerable.

We also realize that the accuracy of quality constraints expressing temporal dependencies affects the effectiveness of the repair. We apply state-of-art discovery methods in literature [5], [6] to guarantee the quality of ψ-constraints.

C. Determine repair cost

The heuristic information guiding data repair becomes important to find more satisfactory results. In the l_1-distance term of $cost(\cdot)$, $range(S'_m[t_n])$ on each cell $S_\Omega[t_n]$ weighs and balances the cost of modification. Thus it has a significant impact on the repair result. The cost function is designed to meet the following requirements: (1) It helps realize normalization, or it reflects the range of candidate repair result; and (2) It reflects the expectation of the repair

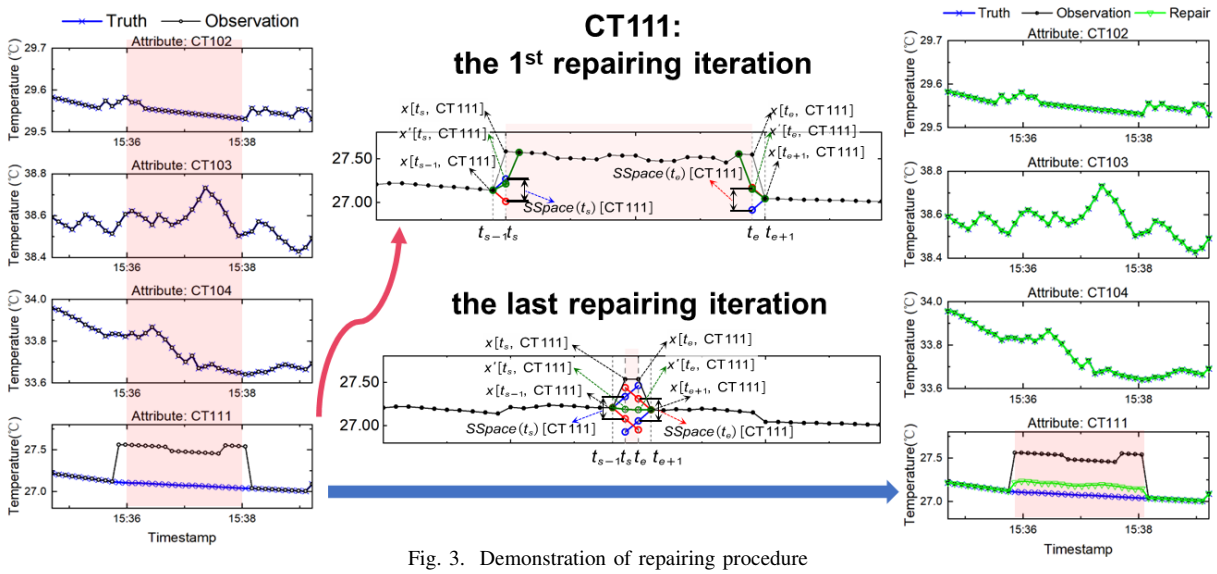


Fig. 3. Demonstration of repairing procedure

result. Specifically, for $S_\Omega[t_n]$, if the original value of $S_m[t_n]$ is beyond the search space $SS(t_n\, m)$ returned by PREPRUNE, *range* $(S'_m[t_n])$ could be relatively larger to allow for greater modifications. Considering what are mentioned above, we define each *range* $(S'_m[t])$ for $\forall m \in \Omega$ with the result of PREPRUNE as:

$$range(S'_m[t]) = \max(ub(SS(t,m)), S_m[t]) - \min(lb(SS(t,m)), S_m[t])$$

where $ub(SS(t,m))$ and $lb(SS(t,m))$ denote the upper and lower bound of $SS(t,m)$ respectively.

***Proposition** 5: The calculation for function range$(S'_m[t_n])$ for $\forall m \in \Omega$ could satisfy the above-mentioned requirements. As if to search the repair result $S'_m[t_n]$ in $SS(t_n, m)$, $|S_m[t_n] - S'_m[t_n]| <$ range$(S'_m[t_n])$ is obtained, and range$(S'_m[t_n])$ becomes larger if the original value $S_m[t_n]$ lies outside $SS(t_n, m)$, range$(S'_m[t_n])$ is guaranteed to be larger than the range of $SS(t_n, m)$.*

D. Evolutionary-based repairing

After deriving the search space considering temporal dependencies and pruning plenty of unnecessary candidate results as well, we apply evolutionary algorithm EVOREPAIR, with specially customized fitness score function in Definition 12 to search for the expected repair result.

***Definition** 12:* **The customized fitness function**. For each candidate repair result, denoted by $S'_\Omega[t]$ corresponding to the original data $S_\Omega[t]$ to repair, the fitness score is:

$$fitscore(S'_\Omega[t_n]) = \begin{cases} cost(I, I') + vd(S'_\Omega[t]), & S'_\Omega[t] \models \varphi \\ cost(I, I') + vd(S'_\Omega[t]) + \lambda_1 + \lambda_2, & S'_\Omega[t] \not\models \varphi \end{cases}$$

The problem of finding the candidate with lowest fitness score in the Definition 12 is the same as that of solving the constrained optimization problem for repair in the Definition 10. Thus, we involve evolutionary algorithms to solve the repair problem. We evaluate the repair effectiveness in Proposition 6.

***Proposition** 6: Given two candidates in the population denoted by $S'_\Omega[t_n]$ and $S''_\Omega[t_n]$, $S'_\Omega[t_n] \models \varphi$ while $S'_\Omega[t_n] \not\models \varphi$, the inequality relationship that $fitscore(S'_\Omega[t_n]) < fscore(S''_\Omega[t_n])$ holds true. It means that the fitness score of each candidate which still violates the quality constraints is much higher than all of the candidates with the elimination of violation.*

Besides the customized fitness score, we also consider the method of initializing suitable population at the beginning. We put $S_\Omega[t_{s-1}]$ or $S_\Omega[t_{e+1}]$ into the initial population respectively, as $S_\Omega[t_{s-1}]$ is within $SS(t_s, \Omega)$, and $S_\Omega[t_{e+1}]$ is within $SS(t_e, \Omega)$. In the first iteration, EVOREPAIR with elite selection strategy ensures that the repair result at least resolve the violation of given quality constraints, no matter whatever stopping condition is set. Therefore, in the following iterations, Algorithm 2 at least finds the repair result with no violation of given quality constraints with Proposition 3. For the remaining initial population, EVOREPAIR generates them by changing only on cell of $S_\Omega[t_s]$ or $S_\Omega[t_e]$ to another value within $SS(t_s, \Omega)$, and $SS(t_e, \Omega)$ respectively. Specially, if $S_m[t_s]$ or $S_m[t_e]$ lies beyond the corresponding $SS(t_s, m)$, and $SS(t_e, m)$, respectively, more candidates with the change in sequence S_m will be initialized.

***Example** 2:* Fig. 3 shows the repair process of the violation cell in including CT102, CT103, CT104 and CT111. The proposed DATAREPAIR algorithm considers both row and column dependencies and make a wise decision on the value modification for all the cells in it. With the speed constraints CT102: $\psi = (-0.0024, 0.0024)$; CT103: $\psi = (-0.016, 0.016)$; CT104: $\psi = (-0.006, 0.006)$; CT111: $\psi = (-0.012, 0.012)$, in the first iteration, it has $S_\Omega[t_s] = [29.57, 38.52, 33.80, 27.58]$, $S_\Omega[t_e] = [29.53, 38.55, 33.65, 27.54]$. We prune the search space for $S_\Omega[t_{s-1}]$ and $S_\Omega[t_{e+1}]$, where $SS(t_s, \Omega) = \{[29.53, 29.58], [38.41, 38.75], [33.78, 33.91], [27.01, 27.26]\}$.

Also, $SS(t_e, \Omega) = \{[29.50, 29.55], [38.33, 38.67], [33.58, 33.70], [26.91, 27.17]\}$. We derive the range according to Section IV-C on t_s, and obtain 0.0496,0.3356,0.1232 and 0.5666 for CT102, CT103, CT104 and CT111, respectively. DataRepair modifies $S_{11}[t_s]$ from 27.58 to 27.21 without changing other data. Similarly, DataRepair only modifies $S_{11}[t_e]$ from 27.54 to 27.15. Then the time window to deal with shrinks, and current t_s and t_e are removed from it. After visiting all time points in the window, the repair finishes.

V. Experimental Evaluation

A. Experimental setting.

Experimental dataset. We present our experimental results on three real datasets collected from the IoT domain including *OilTemp*, *Power* and *Energy*. (1) *OilTemp* dataset records the temperature of lubricating oil in the different components of a wind turbine lubrication system. It includes several aligned and highly correlated time series. (2) *Power* dataset contains multidimensional time series reflecting the real operation process of a wind power generation system. (3) *Energy* is a time series dataset capturing diverse measurements in a low energy building every 10 minutes from 2016/1/1 to 2016/5/27.

Metrics. For the original data I, let I_{repair} and I_{truth} be the repaired data and the ground truth, respectively. With regard to a cell $S_m^{\mathsf{repair}}[t_n]$, $S_m^{\mathsf{truth}}[t_n]$ is its repaired version and $S_m^{\mathsf{truth}}[t_n]$ denotes its ground truth. We apply the following types of metrics to evaluate the effectiveness of data cleaning methods.

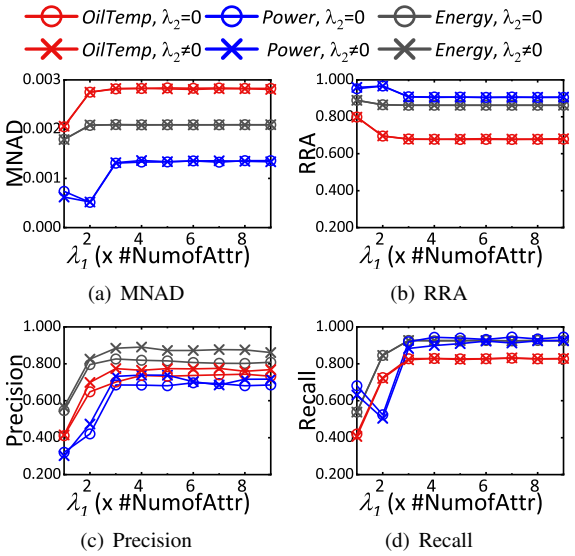
(1). Mean Normalized Absolute Distance (*MNAD*). [15] is used to evaluate the closeness of repair I_{repair} to the truth I_{truth} which denoted as $\Delta(I_{\mathsf{repair}}, I_{\mathsf{truth}})$. The lower the value is, the more accurate the repair is, calculated by $\sum_{m=1}^{M}\sum_{n=1}^{N}\frac{|Norm_m(S_m^{\mathsf{repair}}[t_n]-S_m^{\mathsf{truth}}[t_n])|}{M\cdot N}$, where $Norm_m$ is the normalization function applied to S_m.

(2). Relative Repair Accuracy (*RRA*). [5] normalizes the distance MNAD measures and takes the error in I into consideration. RRA verifies the effectiveness of cleaning methods by computing $1-\frac{\Delta(I_{\mathsf{repair}},I_{\mathsf{truth}})}{\Delta(I,I_{\mathsf{truth}})+\Delta(I,I_{\mathsf{repair}})}$.

(3). *Precision* (P) & *Recall* (R). $\mathsf{P} = \frac{\#\mathsf{correctRepairCells}}{\#\mathsf{RepairCells}}$ measures the ratio of the number of cells which are closer to the corresponding ground truth to the number of cells which are changed after cleaning. $\mathsf{R} = \frac{\#\mathsf{correctRepairCells}}{\#\mathsf{ErrCells}}$ measures the ratio of the number of cells where the method makes a "good" repair to the number of cells where true error takes place.

Comparison Methods. We implement all algorithms introduced in this paper, and the whole cleaning process is named Clean4MTS. Besides our method, we also implement the following baseline time series cleaning methods including constraint-based, labeling-based and smoothing-based strategies.

- Speed [5] is one state-of-art constraint-based time series cleaning method under the direction of SC. It repairs every sequence independently considering the range of the speed of value change over time.



(a) MNAD (b) RRA (c) Precision (d) Recall

Fig. 4. Effectiveness evaluation with varying λ_1 and λ_2 in the repair cost

- Speed + Acc [6] is the extended version of Speed, which is guided by both speed and acceleration constraints in one sequence.
- Vari [16] is a constraint-based method and consider only temporal dependencies on every sequence in the form of a corresponding *variance* constraint.
- IMR [17]: one of state-of-art methods, which performs a iterative minimum repairing method processing every sequence with a small number of labeled data.
- EWMA [18]: a symbolic smoothing-based method.

Specifically, we use Speed(L) and Speed + Acc(L) to represent the local methods with median principle, and use Speed(G) & Speed + Acc(G) to represent those global methods by means of linear programming. Besides, we also implement Clean4MTS+, an extended version of Clean4MTS, in which both speed and acceleration constraints are used in the repair phase.

The speed constraints, acceleration constraints and variance constraints used in these methods are pre-defined by observing the corresponding statistical distributions. The conditional regression rules are configured via grid searches of the model and its parameter selection in advance. All of the constraints used are discovered on the trustworthy datasets. In IMR, the label rates set are 20%.

B. Influences of Parameters and Design Choices

All of the datasets *OilTemp*, *Power* and *Energy* we use in the Section V-B and V-C contains 10k data points and the error rates of them are all 10%. More specifically, the *MNAD* between the input multi-dimension time series with errors in *OilTemp*, *Power* and *Energy*, and its ground truth is respectively 0.00578, 0.00787 and 0.00853. Most of the errors are with time of duration.

1) Varying the Weight of Repair Cost λ_1 and λ_2: λ_1 and λ_2 are two key factors that help achieve the balance between

TABLE I
ABLATION STUDY ON 3 REAL-WORLD MULTI-DIMENSIONAL TIME SERIES DATASETS

	OilTemp					***Power***					***Energy***				
	MNAD	*RRA*	*Prec*	*Rec*	*Time*	*MNAD*	*RRA*	*Prec*	*Rec*	*Time*	*MNAD*	*RRA*	*Prec*	*Rec*	*Time*
Clean4MTS	0.00283	0.758	0.763	0.889	10.24	0.00133	0.907	0.723	0.874	7.01	0.00203	0.865	0.895	0.921	5.34
Clean4MTSNR	0.00403	0.476	0.577	0.606	9.74	0.00452	0.598	0.522	0.701	6.99	0.00536	0.557	0.646	0.525	5.46
Clean4MTSNT	0.00457	0.400	0.337	0.822	25.63	0.00134	0.909	0.443	0.874	23.39	0.00499	0.616	0.469	0.898	16.57
Clean4MTSNV	0.00322	0.614	0.773	0.569	11.50	0.00156	0.819	0.820	0.530	6.73	0.00229	0.803	0.786	0.627	5.74

TABLE II
EFFECTIVENESS AND EFFICIENCY EVALUATION WITH STATE OF ART APPROACHES

	OilTemp					***Power***					***Energy***				
	MNAD	*RRA*	*Prec*	*Rec*	*Time*	*MNAD*	*RRA*	*Prec*	*Rec*	*Time*	*MNAD*	*RRA*	*Prec*	*Rec*	*Time*
Clean4MTS	0.00283	0.758	0.763	0.889	10.14	0.00133	0.907	0.723	0.874	7.01	0.00203	0.865	0.895	0.921	5.34
Clean4MTS+	0.00282	0.680	0.756	0.886	14.34	0.00131	0.908	0.636	0.902	10.47	0.00206	0.865	0.747	0.922	7.09
Speed(L) [5]	0.00548	0.612	0.206	0.434	0.65	0.00484	0.449	0.319	0.741	0.65	0.00833	0.546	0.181	0.913	0.54
Speed(G) [5]	0.00331	0.598	0.381	0.700	12.43	0.00348	0.721	0.307	0.705	13.08	0.00405	0.754	0.274	0.983	8.18
Speed + Acc(L) [6]	0.02509	0.039	0.167	0.390	2.24	0.00950	0.102	0.212	0.579	1.92	0.01713	0.290	0.180	0.908	1.75
Speed + Acc(G) [6]	0.00354	0.576	0.207	0.705	15.21	0.00332	0.714	0.381	0.700	18.96	0.00408	0.753	0.257	0.981	11.60
Vari [16]	0.00338	0.597	0.394	0.859	4.21	0.00599	0.535	0.260	0.967	5.61	0.00469	0.659	0.262	0.955	3.73
IMR [17]	0.00582	0.402	0.214	0.983	86.45	0.00401	0.371	0.204	0.987	96.20	0.00544	0.532	0.339	0.990	66.98
EWMA [18]	0.01475	0.127	0.012	0.997	0.02	0.01078	0.160	0.019	0.988	0.02	0.01641	0.237	0.015	0.983	0.02

the repair cost and the degree of conformance to the given conditional regression rule in the repair phase of Clean4MTS. From the Fig. 4, we notice that λ_1 plays a key role in the effective performance of Clean4MTS. When λ_1 is too low, although *MNAD* from repair to truth is relatively better, *RRA* and *P* scores are much worse. It shows that if the repair cost is not attached enough attention, Clean4MTS is likely to modify more clean data and less truly error data, thereby leading to the poorer repair performance. This suggests that the minimum repairing principle is an extraordinarily important component in the repair phase. Conversely, if λ_1 is too high, the performance on *MNAD* and *RRA* is worse while *P* is a little better. This demonstrates that the naive minimum repairing principle may prevent us from making enough modification of true errors. The influences of λ_2 are not so apparent compared with λ_1. Briefly speaking, λ_2 restricts the number of cells Clean4MTS modifies in a row. Intuitively, if we have $\lambda_2 = \frac{\#PossibleRepair}{\#\ AllowRepair}$, it means that among #*PossibleRepair* cells in a row, the number of modification is at most #*AllowRepair*. We compare the results of whether to set λ_2 to 0. And it demonstrates that λ_2 can improve the *P* of Clean4MTS to some degree. Appropriate λ_1 and λ_2 can help Clean4MTS make a more correct determination on which cells to repair.

2) Justifying Design Choices of Clean4MTS: Clean4MTS identifies violation in the unit of time windows, and uses constraints to express temporal dependencies to prune the search search space, and perform a bi-directional iterative repair. When determining cells to repair, Clean4MTS not only considers minimum changes, but also the violation degree of the give constraints. We perform an ablation study with another three algorithms to anatomize the role of each element in the proposed Clean4MTS, as results shown in Table I.

- Clean4MTSNR shares the same data repairing algorithm as Clean4MTS, while capturing violation in the unit of single data points in detection phase.
- Clean4MTSNT shares the same violation detection algorithm with Clean4MTS, but it does not introduce speed constraint and repairs the data points as violation independently.
- Clean4MTSNV shares the same violation detection algorithm and almost the same data repairing algorithm as Clean4MTS. However, it does not consider the degree of compliance with the given quality constraint.

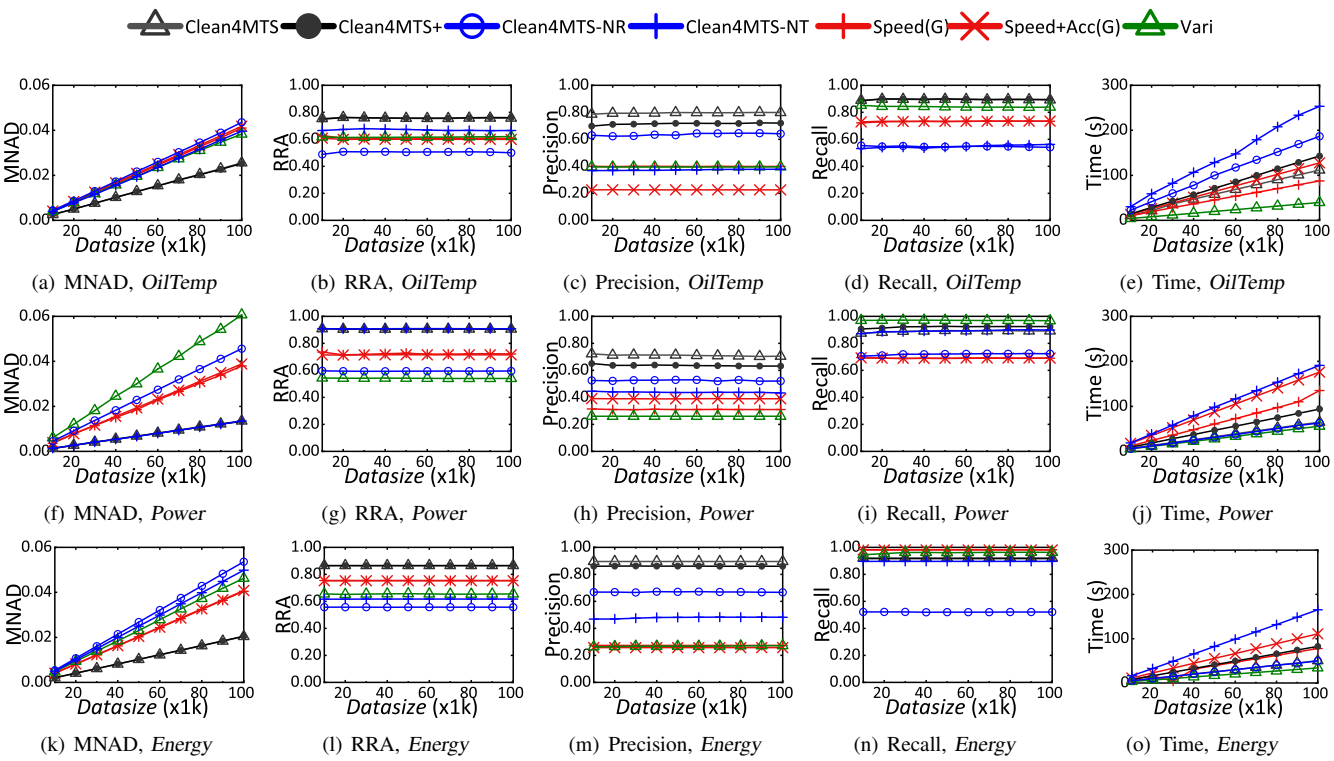
Comparing the result of Clean4MTSNR and Clean4MTS, it is noticed that Clean4MTSR takes a little more time, but has much greater performance on all the given effectiveness metrics. The reason is that the proposed detection algorithm is more suitable for capturing the continuous errors in the multi-dimensional time series data accurately and completely. Then, in the repair phase, the temporal dependency information considered is more reliable due to the quality of contextual data, and Clean4MTS is able to make a more accurate decision.

We study the difference between the result of Clean4MTSNT and that of Clean4MTS as well. Though they have similar R, all other metrics with regard to Clean4MTSNT including *MNAD*, *RRA*, *P* and *Time* gets apparently worse. That demonstrates the negative impact of not pruning unnecessary search space by introducing constraints of temporal dependencies. Our proposed data repairing algorithm makes Clean4MTS more effective and efficient.

Further more, we evaluate the effect of consider both repair cost and the violation degree of the given conditional regression rule comprehensively in the repair phase by contrasting the result of Clean4MTSNV and that of Clean4MTS. We observe that it sacrifices the value of Precision, but makes a significant improvement to Recall, and reduces *MNAD* and increases the value of RRA as well. At the appropriate time, introducing the effect of quantifying the degree of conformance to the given expressive constraints might make the repair result closer to the truth.

C. Comparison with state of art approaches

In this part, we make a preliminary comparison among different time series cleaning methods applied in the multidimensional context, with the same settings in Section V-B. With respect to the model setting of Clean4MTS and Clean4MTS+,



(a) MNAD, *OilTemp* (b) RRA, *OilTemp* (c) Precision, *OilTemp* (d) Recall, *OilTemp* (e) Time, *OilTemp*

(f) MNAD, *Power* (g) RRA, *Power* (h) Precision, *Power* (i) Recall, *Power* (j) Time, *Power*

(k) MNAD, *Energy* (l) RRA, *Energy* (m) Precision, *Energy* (n) Recall, *Energy* (o) Time, *Energy*

Fig. 5. Effectiveness and efficiency evaluation with varying data size

λ_1 is times the number of possible modified attributes and λ_2 is half of it. Table II presents the results.

First of all, we observe that most of the existing methods get relatively satisfactory Recall. Notwithstanding, their Precision are very low, which means that they change lots of clean data and cause information loss easily. Conversely, Clean4MTS and Clean4MTS+ maintain a high level of *P*, while their performance on *R* is fairly good. This phenomenon implies that the cleaning method we propose can more accurately identify the error in multidimensional time series data at the cell level. Furthermore, Clean4MTS and Clean4MTS+ outperform on the effectiveness metrics including *MNAD* and *RRA*. This suggests that the method proposed make a wiser decision on the value modification of the repaired cells and so that the data is much closer to the ground truth.

It is undeniable that methods including EWMA, Speed(L) and Speed + Acc(L) are time saving, as they support streaming computation. However, they behave badly in this experimental setting with plenty of continuous errors. The values of *MNAD* after cleaning are even worse. That is because the smoothing-based methods modifies almost all the data, most of which is clean indeed, and Speed(L) and Speed + Acc(L) are not applicable to repairing errors with duration, as they are seriously affected by the unreliable contextual data when determining which cells to change and their value modification.

The adverse impact of untrustworthy contextual data on methods in the supervision of speed constraints (or speed + acceleration constraints) get some alleviation when they pursue the global optimum. Table II reports that Speed(G) and Speed + Acc(G) have significantly better performance on *MNAD*, *RRA* and *P*. However, Clean4MTS and Clean4MTS+ are more rewarding compared to Speed(G) and Speed + Acc(G), due to the fact that they takes similar time, but Clean4MTS and Clean4MTS+ gains more satisfactory *MNAD*, *RRA* and *P*.

We also notice that making an additional introduction of acceleration constraints on the basis of speed constraints does not improve the effectiveness of methods and even worsens it. That is especially true of the local method, like Speed + Acc(L), for the reason that methods guided by acceleration constraints need more contextual data to consider temporal dependencies. Therefore, their performance is more vulnerable to the data quality of the context when repairing.

With regard to IMR, its performance heavily depends on the manual label, and has strict requirement of the labels' number, quality and position. What's more, it costs much more time while having lower value of effective metrics compared to the proposed method.

D. Effectiveness and efficiency evaluation with data size

We study the effective and efficiency of our proposed method and existing methods with varying data sizes of given dataset. The error rate of the data to clean is 10 %. We also study the influence of identifying violation in the unit of time windows in the detection phase, and the bi-directional iterative algorithm in the repair phase with different data sizes by implementing Clean4MTSNR and Clean4MTSNT. Fig. 5 demonstrates the results over given datasets. Besides the discoveries summarized in the last section, we notice that

the time cost between Clean4MTSNR and Clean4MTS is not very apparent. It implies that detecting in the unit of time windows can be also efficient in most cases where errors are relatively sparse in the time series, while capturing the errors more accurately and completely. And considering temporal dependencies in the repair phase can actually save time and contribute to more effective repair.

VI. Related Work

We summarize a few works related to our studied problems from data cleaning field.

Rule-based relational data cleaning. Relational data cleaning techniques mainly include rule-based [19], [20], statistical-based [21], human-in-the-loop [22]–[24], and learning-based [25] methods. Rule-based cleaning utilizes domain-specific knowledge and fully leverages the relationships in the data to improve data quality [26]. There are two types of repairs in this process: one is to fully trust the given constraints and only consider repairing the data; the other is to not fully trust the constraints, and to modify both the data and the (imperfect) constraints. The research on the former has been extensively conducted, which contains local cleaning (*e.g.,* [10]) and holistic cleaning. [27] proposed to detect conflicts in data representation based on hypergraph model, and developed cleaning tool Holoclean. As using outdated or incorrect constraints can lead to unreliable data cleaning results and erroneously modify otherwise correct data, Chiang and Miller were among the first to propose the problem of modifying both data and rules [28], utilizing the minimum description length principle and a unified repair model (URM) to measure the costs of repairing. A continuous data cleaning model [29] was proposed to use classifiers to determine whether repairs should be executed on the data, constraints, or both. Song et al. solved the repairing problem of finding denial constraints (Dc) [11], [30] that minimizes the cost of dataset modification, within a given range of variable threshold values for the constraints, in order to avoid both over-fitting and under-fitting. Further, evaluate-clean loop [31] was addressed to involve error explanation and tracing to improve the repair strategies. [12] provided an overview of the end-to-end data cleaning process, and Fatemeh et al. [32] pointed out that in emerging data management systems, data cleaning techniques still face challenges such as holistic integrity constraints discovery, heterogeneity-aware data cleaning and validation of cleaning effects.

Numerical data cleaning. As is realized that quality issues are serious in numerical data, *e.g.,* time series data in IoT field, more attention has been put to specialized numerical data cleaning techniques. Survey papers [2], [33], [34] summarized the research progress of time series data cleaning, mainly including smoothing-based, statistical-based, and constraint-based repair methods. Tutorials *e.g.,* [1] conducted data quality assessments on IoT data from the perspectives of data validity, integrity, and consistency. [35] formalized four types of data quality rules for temporal numerical data from the perspectives of "row (entity)" and "column (attribute)". Techniques have been developed to support the complex data quality requirements on numerical data, *e.g.,* arithmetic operations among attributes, and the relaxation of the exact "equal" relationships. [36] extended the relational data dependencies and proposed sequential dependencies (Sds) to describe the semantics of numerical data. [6] solved the time series data cleaning problem based on speed constraints on single sequence. [16] introduced variance constraints in time series cleaning. [37] detected and labeled kinds of errors with a non-parametric concept of neighborhood, considering single errors, collective errors, and changing points. In recent years, [7] proposed statistical constraints to capture relationships among numerical attributes which are hard to be captured by traditional dependencies to improve data quality in training models. To support arithmetic operations (primarily linear data dependencies) and allow for small ranges of relaxation [9] provided conditional regression rules (Crrs) and captured regression models from time series data. [8] proposed conformance constraints (Ccs) and evaluated them in trusted machine learning and data drift.

In our previous work, we developed a data cleaning system *Cleanits* [38] for time series and achieved efficient cleaning utilizing the relationship discovery algorithms [39]. We have proposed multi-constraints error detection and diagnosis method for industrial numerical sequential data [40], [41], and developed detection algorithms library *i.e., Cleanit-MEDetect* for identifying multiple errors in multivariate time series [42]. Although it has been explored, breakthroughs are still expected to address the data cleaning challenges considering the unique characteristics of time series data, and our work in this paper complement existing data cleaning techniques considering to apply expressive constraints from both rows and columns for time series.

VII. Conclusions

In this paper, we proposed a novel method for cleaning time series data under expressive constraints both among attributes and along time context. We introduced violation degree functions to profile errors and proposed violation cell discovery algorithm to identify real errors in data. We formalized the cleaning problem with well-designed cost functions, and pruned the repair search space by evaluation with temporal context constraints. Experimental results on real IoT data with 5 metrics show that the proposed method outperform state-of-art time series cleaning method. The advantages of our method illustrates the necessity of considering data relationship on both columns and rows in cleaning tasks. Future work includes 1) deep theoretical research of data cleaning with multiple types of constraints describing the dependence on columns and rows, and 2) the continuous data cleaning procedure development with well-designed modification selection algorithms.

References

[1] S. Song and A. Zhang, "Iot data quality," in *The 29th ACM International Conference on Information and Knowledge Management CIKM*, 2020, pp. 3517–3518.

[2] A. Karkouch, H. Mousannif, H. A. Moatassime, and T. Noël, "Data quality in internet of things: A state-of-the-art survey," *J. Netw. Comput. Appl.*, vol. 73, pp. 57–81, 2016.

[3] [Online]. Available: http://iotdb.incubator.apache.org/

[4] C. Wang, J. Qiao, X. Huang, S. Song, H. Hou, T. Jiang, L. Rui, J. Wang, and J. Sun, "Apache iotdb: A time series database for iot applications," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 195:1–195:27, 2023.

[5] S. Song, A. Zhang, J. Wang, and P. S. Yu, "SCREEN: stream data cleaning under speed constraints," in *SIGMOD*, pp. 827–841.

[6] S. Song, F. Gao, A. Zhang, J. Wang, and P. S. Yu, "Stream data cleaning under speed and acceleration constraints," *ACM Trans. Database Syst.*, vol. 46, no. 3, pp. 10:1–10:44, 2021.

[7] J. N. Yan, O. Schulte, M. Zhang, J. Wang, and R. Cheng, "SCODED: statistical constraint oriented data error detection," in *Proceedings of the International Conference on Management of Data, SIGMOD*. ACM, 2020, pp. 845–860.

[8] A. Fariha, A. Tiwari, A. Radhakrishna, S. Gulwani, and A. Meliou, "Conformance constraint discovery: Measuring trust in data-driven systems," in *SIGMOD*. ACM, 2021, pp. 499–512.

[9] R. Kang, S. Song, and C. Wang, "Conditional regression rules," in *38th IEEE International Conference on Data Engineering, ICDE*. IEEE, 2022, pp. 2481–2493.

[10] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi, "A cost-based model and effective heuristic for repairing constraints by value modification," in *SIGMOD*. ACM, 2005, pp. 143–154.

[11] X. Chu, I. F. Ilyas, and P. Papotti, "Discovering denial constraints," *Proc. VLDB Endow.*, vol. 6, no. 13, pp. 1498–1509, 2013.

[12] I. F. Ilyas and X. Chu, *Data Cleaning*. ACM, 2019.

[13] M. Schleich, Z. Geng, Y. Zhang, and D. Suciu, "Geco: Quality counterfactual explanations in real time," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1681–1693, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1681-schleich.pdf

[14] Y. Su, Y. Gong, and S. Song, "Time series data validity," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 85:1–85:26, 2023. [Online]. Available: https://doi.org/10.1145/3588939

[15] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *SIGMOD*, 2014, pp. 1187–1198.

[16] W. Yin, T. Yue, H. Wang, Y. Huang, and Y. Li, "Time series cleaning under variance constraints," in *Database Systems for Advanced Applications - DASFAA Workshops*, 2018, pp. 108–113.

[17] A. Zhang, S. Song, J. Wang, and P. S. Yu, "Time series data cleaning: From anomaly detection to anomaly repairing," *PVLDB*, vol. 10, no. 10, pp. 1046–1057, 2017.

[18] R. Fried and A. C. George, "Exponential and holt-winters smoothing," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 488–490. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_244

[19] W. Fan and F. Geerts, *Foundations of Data Quality Management*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[20] E. Livshits, B. Kimelfeld, and S. Roy, "Computing optimal repairs for functional dependencies," *ACM Trans. Database Syst.*, vol. 45, no. 1, pp. 4:1–4:46, 2020.

[21] A. I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W. Ku, and X. Xie, "Learning-based cleansing for indoor RFID data," in *SIGMOD*, 2016, pp. 925–936.

[22] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, "Guided data repair," *Proc. VLDB Endow.*, vol. 4, no. 5, pp. 279–289, 2011.

[23] J. Fan and G. Li, "Human-in-the-loop rule learning for data integration," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 104–115, 2018.

[24] E. K. Rezig, M. Ouzzani, A. K. Elmagarmid, W. G. Aref, and M. Stonebraker, "Towards an end-to-end human-centric data cleaning framework," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD*, 2019, pp. 1:1–1:7.

[25] M. Mahdavi and Z. Abedjan, "Semi-supervised data cleaning with raha and baran," in *11th Conference on Innovative Data Systems Research, CIDR 2021,*, 2021.

[26] Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: a survey," *VLDB J.*, vol. 24, no. 4, pp. 557–581, 2015.

[27] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Holoclean: Holistic data repairs with probabilistic inference," *PVLDB*, vol. 10, no. 11, pp. 1190–1201, 2017.

[28] F. Chiang and R. J. Miller, "A unified model for data and constraint repair," in *Proceedings of the 27th International Conference on Data Engineering, ICDE*, S. Abiteboul, K. Böhm, C. Koch, and K. Tan, Eds. IEEE Computer Society, 2011, pp. 446–457.

[29] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller, "Continuous data cleaning," in *IEEE 30th International Conference on Data Engineering, ICDE*, 2014, pp. 244–255.

[30] E. H. M. Pena, E. C. de Almeida, and F. Naumann, "Discovery of approximate (and exact) denial constraints," *Proc. VLDB Endow.*, vol. 13, no. 3, pp. 266–278, 2019.

[31] I. F. Ilyas, "Effective data cleaning with continuous evaluation," *IEEE Data Eng. Bull.*, vol. 39, no. 2, pp. 38–46, 2016.

[32] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: Challenges and opportunities," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1986–1989, 2019.

[33] X. Wang and C. Wang, "Time series data cleaning: A survey," *IEEE Access*, vol. 8, pp. 1866–1881, 2020.

[34] X. Ding, H. Wang, G. Li, H. Li, Y. Li, and Y. Liu, "Iot data cleaning techniques: A survey," *Intelligent and Converged Networks*, vol. 3, no. 4, p. 15, 2022.

[35] T. Dasu, R. Duan, and D. Srivastava, "Data quality for temporal streams," *IEEE Data Eng. Bull.*, vol. 39, no. 2, pp. 78–92, 2016.

[36] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava, "Sequential dependencies," *PVLDB*, vol. 2, no. 1, pp. 574–585, 2009.

[37] K. Le and P. Papotti, "User-driven error detection for time series with events," in *36th IEEE International Conference on Data Engineering, ICDE*, 2020, pp. 745–757.

[38] X. Ding, H. Wang, J. Su, Z. Li, J. Li, and H. Gao, "Cleanits: A data cleaning system for industrial time series," *PVLDB*, vol. 12, no. 12, pp. 1786–1789, 2019.

[39] X. Ding, Y. Li, C. Wang, H. Wang, and H. Li, "Time series data quality rules discovery with both row and columndependencies," *Journal of Software (Chinese)*, vol. 34, no. 3, pp. 1065–1086, 2023.

[40] Z. Liang, H. Wang, X. Ding, and T. Mu, "Industrial time series determinative anomaly detection based on constraint hypergraph," *Knowl. Based Syst.*, vol. 233, p. 107548, 2021.

[41] Z. Li, X. Ding, and H. Wang, "An effective constraint-based anomaly detection approach on multivariate time series," in *APWeb-WAIM*, vol. 12318, 2020, pp. 61–69.

[42] X. Ding, Y. Song, H. Wang, D. Yang, and Y. Liu, "Cleanits-medetect: Multiple errors detection for time series data in cleanits," in *28th International Conference, DASFAA*, vol. 13946, 2023, pp. 674–678.