

MINOR: Multivariate Time Series Iterative Cleaning Algorithm

Aoqian Zhang, Yinru Sun, Pengxiang Hao, Yifeng Gong, Boyang Li, Jing Geng
Beijing Institute of Technology, Beijing, China

{aoqian.zhang, yinru.sun, pengxiang.hao, yifeng.gong, liboyang, janegeng}@bit.edu.cn

Abstract—Errors are common in time series data, such as sensor measurements. Existing methods tend to focus on single errors in univariate data, but do not provide satisfactory results for consecutive errors, especially in the more general multivariate data. Modeling each dimension separately in one run can lead to bias, as the correlation between dimensions and the effects of existing errors are not taken into account. We also note that current methods suffer from the problem of over-repair, i.e. clean observations may be modified after data cleaning. In this paper, we propose **MINOR**, an iterative cleaning algorithm for multivariate time series with limited labels. We formalize the repair problem and propose a bidirectional validation that uses a local speed constraint and the past and future information to solve the over-repair problem. We also present a unidirectional validation that supports online computations. We analyze the properties of our proposals and compare them with SOTA methods in terms of effectiveness, efficiency, and the impact of applications such as classification. Experiments on real datasets show that **MINOR** can have higher repair accuracy in various situations and improve time efficiency over the proposed incremental techniques. Interestingly, it can be effective even when there are few labels around 1%.

Index Terms—multivariate time series, data cleaning, consecutive errors, speed constraint

I. INTRODUCTION

Outliers are common in time series and can be divided into two categories: errors and anomalies [1], based on the aim of the analyst. We focus on errors, i.e., the unwanted data that result from sensor damage, measurement inaccuracies, data transmission or system malfunctions [2], [3]. Identifying and correcting these erroneous values is crucial for ensuring the accuracy and reliability of time series analysis [4].

A. Motivation

Various models [5]–[7] are widely used in time series forecasting, anomaly detection and can also be adapted for data repair. Unfortunately, existing errors in the observations lead to a biased and less accurate model. Recently, an iterative minimum-change-aware algorithm [8] has been proposed to repair the time series step by step, assuming that high confidence repairs in the former iterations could help the repair in the latter steps, and exhibits good performance over consecutive errors. It requires the help of plenty of labels (about 10%-20%), which may not be applicable for large datasets, and still suffers from the problem of over-repair, i.e., some correct data points are also changed. On the other hand, it focuses on univariate time series and cannot take into account

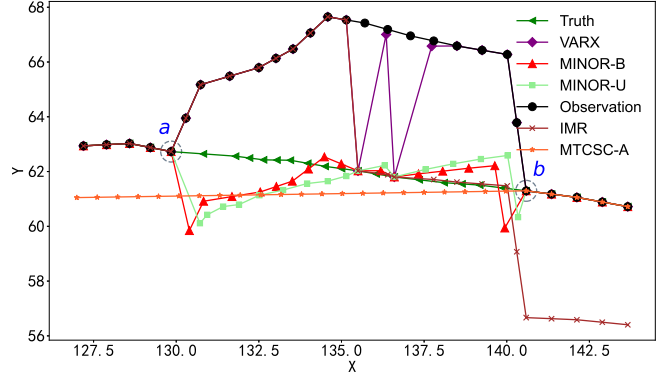


Fig. 1. Example of observations and different repairs

the correlation between dimensions when handling the more general multivariate cases.

Constraint-based methods [9]–[11] consider the speed and acceleration constraints on value changes across different timestamps and [12] also takes the row-wise constraint like conditional regression rules [13] into account. They perform well in repairing time series, especially for single errors, if the constraints are captured accurately. Nevertheless, due to the evolution of time series, it is always difficult to extract perfect rules, and the existing adaptive methods still rely on some important preset parameters.

The statistical-based methods [14], [15] construct a probability distribution model of the speed changes between neighboring points over the entire data or in the streaming scenario to repair small errors. [16] redefines the cleaning problem by minimizing the perplexity of time series. Although single errors are successfully repaired, the above methods are still proposed for univariate time series and their performance on consecutive errors is not satisfactory. The more recent learning-based methods [17], [18] pay more attention to the anomalies, but not to the errors, i.e. the observations that deviate so strongly from other observations. These methods always require a large amount of clean data for training and are highly dependent on fine-tuning.

Example 1: Figure 1 presents an example of GPS data (in black). Obviously, errors occur in the period from point *a* to *b*, where the observations are shifted from the truth. To repair the errors, the truth of some observations are labeled.

Existing speed constraint-based cleaning (MTCSC) [9]

could not effectively detect and repair such continuous errors because speed constraints restrict the amount of value change and the continuous error can also make the incorrect distribution to be large enough to behave like a normal one in a given time period. The univariate iterative method IMR [8] shows a better result using labels, but still encounters the serious problem of over-repair (see repair after point b).

Finally, our proposal MINOR with modeling the entire dimensions and bidirectional repair using past and future information achieves repairs that are closest to the truth.

B. Solution

Intuited by the idea of using master data in data repairing [8], [19], we employ a limit number of labeled truth of dirty observations to advance the repair. To overcome the above problems of the existing methods, we model the time series over all dimensions together and learn the parameters iteratively instead of considering each dimension separately. As for the *over-repair* problem that originally clean value may be modified, we utilize the global and local speed constraint to validate the repair candidate generated from the model and hence reduce the number of false positives. We also introduce bidirectional repair to take advantage of all information from the past and the future. Based on the above techniques, we propose MINOR to clean the multivariate time series with consecutive errors.

However, there still exists some challenges. (1) Convergence is crucial for iterative methods, as it ensures that the method gradually approaches the exact solution of the problem with the desired accuracy through repeated iterations [20]. It may be difficult to prove the convergence of the proposal in the general case. If not, time efficiency becomes another problem. Will it converge or can it give satisfactory results if we terminate it early? (2) Time series is generated continuously, which means we may not get enough future information for bidirectional repair. Will the method work with only historical data or only few numbers of future data (p is small enough) and support online computation? (3) Like [9], it is still difficult to explain why it makes sense to consider the speed constraint across all dimensions. Moreover, is the local speed constraint effective when the data label is rather rare and what if such a local constraint is far away from the target data points?

C. Contribution

The proposed MINOR (Multivariate time series Iterative cleanNing algORithm) focuses on the consecutive errors and our major contributions in this paper are summarized as:

- 1) We formalize the repair problem over multivariate time series with labels in Section II-A. The adaptation of existing time series modeling techniques (such as VAR and VARX) is introduced for data repair.
- 2) We devise an iterative repair algorithm MINOR in Section III-A. According to the minimum change principle [21], errors are reduced gradually, leading to more accurate parameter estimation and better repair over consecutive errors.

3) We present unidirectional and bidirectional validation to judge the validity of candidates in Section III-D, using the global/local speed constraint to avoid the problem of over-repair. In addition to the forward direction, each point can also be derived backwards in bidirectional validation.

4) We design efficient speedup techniques for all major steps in each repair iteration in Section IV. Instead of performing operations over all n points from the scratch, incremental computation between different iterations could reduce the complexity of parameter estimation from $O(n)$ to $O(1)$, of candidate generation from $O(n)$ to $O(m)$.

5) We analyze the experimental results on real-world datasets with real and synthetic errors and discuss the superiority and limitations of our proposed methods in Section V. Remarkably, unlike existing methods, MINOR could achieve superior results with few labels (around 1%).

Table I lists the notations frequently used in this paper.

TABLE I
NOTATIONS

Symbol	Description
\mathbf{x}	time series
$\mathbf{x}_i, \mathbf{x}[i]$	i -th data point in \mathbf{x}
\mathbf{y}	truth-labeled/repared sequence of \mathbf{x}
$\mathbf{y}^{(k)}$	sequence \mathbf{y} in the k -th iteration
\mathbf{z}	distance between \mathbf{x} and labeled/repared \mathbf{y}
t_i	timestamp of i -th data point in \mathbf{x}
s_g	speed constraint derived by 3σ rule.
D	dimension of time series \mathbf{x}
Φ	parameter of VAR(p)/VARX(p) with order p
τ	predefined threshold of convergence
\mathbf{Z}, \mathbf{V}	input matrix for computation

II. PRELIMINARIES

In this section, we first introduce time series and other related terms in our work. Then, we define the repair problem and adapt the existing time series modeling technique that can be used for repairing multivariate data, i.e., VAR with only observations and VARX considering labeled data.

A. Problem Statement

Definition 1 (Time Series and Labels): A time series is a sequence of data points indexed in time order. Consider a time series of n observations, $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$. The i -th observation (data point) $\mathbf{x}_i \in \mathbb{R}^D$ consists of D dimensions $\{x_{i,1}, \dots, x_{i,D}\}$ and is observed at time t_i . Let \mathbf{y} denote the labeled or repaired sequence of \mathbf{x} , each \mathbf{y}_i is either the labeled truth or the repaired value of data point \mathbf{x}_i .

Definition 2 (Distance): The distance between two data points is the Euclidean distance between them, denoted by $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{\ell=1}^D (x_{i,\ell} - x_{j,\ell})^2}$.

Following [9] and Definition 2, the speed constraint is defined as follows:

Definition 3 (Speed Constraint): A speed constraint s with window size w is the maximum speed s_{\max} over the time series \mathbf{x} . We say that a time series \mathbf{x} satisfies the speed constraint s , denoted by $\mathbf{x} \models s$, if for any $\mathbf{x}_i, \mathbf{x}_j$ in a window,

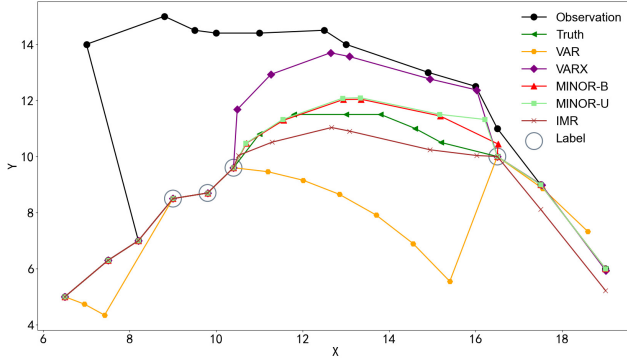


Fig. 2. Example of observations and repairs

i.e., $0 < t_j - t_i \leq w$, it has $\frac{d(x_i, x_j)}{t_j - t_i} \leq s$, where window w denotes a period of time.

Problem 1: Given a time series x and a partially labeled subset y of x , the repairing problem is to find the repairs y_i of x_i that are not labeled in y .

Example 2 (Observation x , partially labeled y , and fully repaired y): Consider a multivariate time series ($D = 2$), $x = \{(6.5, 5), (7.5, 6.3), (8.2, 7), (7, 14), (8.8, 15), (9.5, 14.5), (10, 14.4), (11, 14.4), (12.5, 14.5), (13, 14), (14.9, 13), (16, 12.5), (16.5, 11), (17.5, 9), (19, 6)\}$ with 15 data points of observations as illustrated in Figure 2, where consecutive errors occur on x_4 to x_{13} . Suppose that four points x_4, x_5, x_6, x_{13} are labeled with truth which is marked by gray circle in Figure 2. In the repaired y , x_7, \dots, x_{12} are changed from $\{(10, 14.4), (11, 14.4), (12.5, 14.5), (13, 14), (14.9, 13), (16, 12.5)\}$ to $\{(11, 10.8), (11.8, 11.5), (13, 11.5), (13.8, 11.5), (14.6, 11), (15.2, 10.5)\}$, respectively. The labeled y_4, \dots, y_6 and y_{13} will not be modified in the repair result.

B. VAR Model

Time series models can be adapted to data repairing. For instance, the VAR (Vector Autoregression) model can be considered as:

$$x'_t = \sum_{i=1}^p x_{t-i} \Phi_i + \epsilon_t \quad (1)$$

where x'_t is the prediction of x_t , p is the order, Φ_i is the parameter of the model. For the multivariate time series whose dimension is D , x'_t and x_t are both column vectors of $D \times 1$, while Φ_i is a $D \times D$ matrix. ϵ_t is white noise (usually Gaussian white noise).

If x'_t significantly differs from the observation x_t , having $d(x'_t, x_t) > \tau$ where τ is a predefined threshold, this predication is accepted $x'_t = x_t$, a.k.a. a repair. The intuition behind is that a farther distance indicates the higher probability of being an error. The threshold τ can be determined by observing the statistical distribution of distances between x'_t and x_t .

The VAR-based repairing procedure is thus: (1) replace x_t by y_t if it is labeled, (2) learn parameter Φ_i of VAR(p) from x , and (3) fill all unlabeled y_t by VAR(p) over x , having

$$y_t = \begin{cases} x'_t & \text{if } y_t \text{ is unlabeled and } d(x'_t, x_t) > \tau \\ x_t & \text{otherwise} \end{cases} \quad (2)$$

Example 3 (Repair by VAR, Example 2 continued): Consider again $x = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$ in Figure 2. For simplicity, we use VAR(1) with order $p = 1$, i.e., $x'_t = x_{t-1} \Phi_1$. Let $\tau = 0.5$. By ordinary least square, we have $\Phi_1 = \begin{pmatrix} 1.039 & -0.176 \\ 0.004 & 1.175 \end{pmatrix}$, $x'_2 = x_1 \Phi_1 = (6.9, 4.7)$. Since $d(x'_2, x_2) = 1.7 > \tau$, x'_2 is accepted as new x_2 . Labeled points such as x_4 will not change in each iteration. Similarly, $x'_7 = x_6 \Phi_1 = (11.2, 9.5)$. Finally, we obtain the final repaired result $y = \{(6.5, 5), (6.9, 4.7), \dots, (18.6, 7.3)\}$ with RMS error 2.33 (see section V for definition).

C. VARX Model

In order to utilize the labeled y , we consider the VARX (vector autoregressive model with exogenous inputs) model:

$$y'_t = \sum_{i=1}^p (y_{t-i} - x_{t-i}) \Phi_i + \epsilon_t \quad (3)$$

where y'_t is the possible repair of x_t , and others are the same to the VAR model. As shown in Equation (3), not only the preceding observations x_{t-i} will affect the determination of y'_t , but also the previously labeled/repared y_{t-i} .

The VARX-based repairing procedure is thus: (1) learn parameter Φ_i of VARX(p) from x and partially labeled y , and (2) fill all unlabeled y_t by VARX(p), similar to Equation (2) by replacing x'_t with y'_t .

Following [8], we consider the VARX model as our base model in iteration, as it can utilize the difference between the observed errors and the labeled truths.

Example 4 (Repair by VARX, Example 2 continued): Consider again $x = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$ in Figure 2. For simplicity, we use VARX(1) with order $p = 1$, i.e., $y'_t = x_t + (y_{t-1} - x_{t-1}) \Phi_1$. Let $\tau = 0.5$. Similar to VAR in Example 3, we estimate the parameter Φ by ordinary least square, having $\Phi_1 = \begin{pmatrix} 0.408 & -2.97 \\ -0.024 & 0.009 \end{pmatrix}$. Again, the labeled y_4, \dots, y_6 is not modified. For the 7th point, we have $y'_7 = x_7 + (y_6 - x_6) \Phi_1 = (10.5, 11.7)$. Since $d(y'_7, y_7) = 2.7 > \tau$, we accept y'_7 as new y_7 . Finally, the repair result by VARX is $y = \{(6.5, 5), (7.5, 6.3), \dots, (19, 5.9)\}$ with RMS error 1.16 which is lower than that of VAR in Example 3.

III. REPAIR ALGORITHM

In contrast to the existing models that over-smooth the data in one pass, we propose to repair the data iteratively, since high confident repairs in the former iterations can support the repairs in the latter steps. Specifically, we (1) recover the most obvious error in each round, i.e., the one with the largest repair distance to the observations, to also reduce the number of iterations; (2) propose a unidirectional repair to take into

account historical data and a bidirectional repair to also utilize the information from the near future; (3) use global/local speed constraints to validate the repair candidate and solve the over-repair problem.

A. Iterative Repairing

Let $\mathbf{y}^{(k)}$ denote the sequence \mathbf{y} in the k -th iteration, where $\mathbf{y}^{(0)}$ is the partially labeled time series in the input. Since $\mathbf{y}^{(0)}$ is incomplete (generally only around 1% as shown in Section V), we assign $\mathbf{y}_t^{(0)} = \mathbf{x}_t$ if $\mathbf{y}_t^{(0)}$ is not labeled for initialization. Recall that the labeled values should not be repaired, i.e., $\mathbf{y}_t^{(k)} = \mathbf{y}_t^{(0)}$ if $\mathbf{y}_t^{(0)}$ is labeled.

The proposed MINOR(p), shown in Algorithm 1, is to iteratively repair the given time series, whose inputs are the observations \mathbf{x} and partially labeled $\mathbf{y}^{(0)}$. It outputs $\mathbf{y}^{(k)}$ with all the labeled $\mathbf{y}_t^{(0)}$ unchanged and unlabeled $\mathbf{y}_t^{(0)}$ repaired.

The main steps include:

(S1) Parameter estimation, in Line 2, learns the parameter of VARX(p) (or reverse VARX(p)) model in the k -th iteration, denoted by $\Phi^{(k)}$ (or $\Phi^{(k)r}$), from \mathbf{x} and the current $\mathbf{y}^{(k)}$.

(S2) Candidate generation, in Line 3, computes the possible repairs $\hat{\mathbf{y}}^{(k)}$, according to VARX(p) (or reverse VARX(p)) w.r.t. \mathbf{x} , $\mathbf{y}^{(k)}$ and $\Phi^{(k)}$ (or $\Phi^{(k)r}$).

(S3) Repair validation, in Line 4, determines one of the repairs to accept, $\mathbf{y}_t^{(k+1)} = \hat{\mathbf{y}}_t^{(k)}$, which has the minimum repair distance and is validated w.r.t. the speed constraint.

It is noted that we conduct two types of repairing and Algorithm 1 exhibits the general case. As shown in Line 5, the procedure repeats, until the repair converges, i.e., having

$$d(\mathbf{y}_j^{(k)}, \mathbf{y}_j^{(k+1)}) \leq \tau, j = 1, \dots, n. \quad (4)$$

where τ a threshold of convergence, or a maximum number of iterations is reached. Setting *max-num-iterations* is a remedy to avoid waiting for convergence in practice (see Section V-B4 for discussion and evaluation).

Algorithm 1: MINOR(p)

Input: time series \mathbf{x} and partially labeled $\mathbf{y}^{(0)}$

Output: $\mathbf{y}^{(k)}$ with all the labeled $\mathbf{y}_i^{(0)}$ unchanged and unlabeled $\mathbf{y}_j^{(0)}$ repaired

```

1 for  $k \leftarrow 0$  to max-num-iterations do
2    $\Phi^{(k)} \leftarrow \text{Estimate}(\mathbf{x}, \mathbf{y}^{(k)});$ 
3    $\hat{\mathbf{y}}^{(k)} \leftarrow \text{Candidate}(\mathbf{x}, \mathbf{y}^{(k)}, \Phi^{(k)});$ 
4    $\mathbf{y}^{(k+1)} \leftarrow \text{Validate}(\mathbf{x}, \mathbf{y}^{(k)}, \hat{\mathbf{y}}^{(k)});$ 
5   if Converge( $\mathbf{y}^{(k)}, \mathbf{y}^{(k+1)}$ ) then
6     break;
7    $k \leftarrow k + 1;$ 
8 return  $\mathbf{y}^{(k)}$ 
```

Example 5 (Algorithm overview, Example 2 continued): Consider again $\mathbf{x} = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$ in Figure 2. According to four labeled data points, we assign $\mathbf{y}^{(0)} = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$, where the unlabeled points are initialized by $\mathbf{y}_t^{(0)} = \mathbf{x}_t$.

In each iteration: MINOR(1) S1. learns the parameter, e.g., $\Phi_1^{(0)} = \begin{pmatrix} 0.408 & -2.977 \\ -0.024 & 0.009 \end{pmatrix}$ for $p = 1$; S2.

generates candidates for repairing, such as $\hat{\mathbf{y}}^{(0)} = \{-, -, -, +, +, +, (10.5, 11.7), -, -, -, -, -, +, (17.5, 9), -\}$, where ‘+’ is labeled point, and ‘-’ denotes no candidates; and S3. validates the candidate, such as $\hat{\mathbf{y}}_7^{(0)}, \hat{\mathbf{y}}_{14}^{(0)}$ do not match the speed constraint. By computing the validity of each of them, $\hat{\mathbf{y}}_{14}^{(0)}$ is selected as the repair result in the first iteration.

The final output of MINOR – B is $\mathbf{y}^{(51)} = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$ with RMS error 0.47 after 51 iterations. MINOR – U outputs $\mathbf{y}^{(44)} = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$, with RMS error 0.48 after 44 iterations.

B. Parameter Estimation

The parameter estimation step S1 (in Line 2 in Algorithm 1) estimates the parameter $\Phi^{(k)}$ for MINOR(p), given $\mathbf{x}, \mathbf{y}^{(k)}$. For simplicity, for a data point \mathbf{x}_j , we define the following $1 \times D$ vector

$$\begin{aligned} \mathbf{z}_j^{(k)} &= \mathbf{y}_j^{(k)} - \mathbf{x}_j, j = 1, \dots, n. \\ \{\mathbf{z}_{j,1}^{(k)}, \dots, \mathbf{z}_{j,D}^{(k)}\} &= \{\mathbf{y}_{j,1}^{(k)} - \mathbf{x}_{j,1}, \dots, \mathbf{y}_{j,D}^{(k)} - \mathbf{x}_{j,D}\} \end{aligned} \quad (5)$$

Existing methods such as Ordinary Least Square [22] or Yule-Walker Equations [23] can be directly employed. For instance, by Ordinary Least Square, we have

$$\Phi^{(k)} = ((\mathbf{Z}^{(k)})' \mathbf{Z}^{(k)})^{-1} (\mathbf{Z}^{(k)})' \mathbf{V}^{(k)} \quad (6)$$

where

$$\begin{aligned} \mathbf{V}^{(k)} &= \begin{pmatrix} \mathbf{z}_{p+1}^{(k)} \\ \mathbf{z}_{p+2}^{(k)} \\ \vdots \\ \mathbf{z}_n^{(k)} \end{pmatrix}_{n-p \times D}, \quad \Phi^{(k)} = \begin{pmatrix} \Phi_1^{(k)} \\ \Phi_2^{(k)} \\ \vdots \\ \Phi_p^{(k)} \end{pmatrix}_{p \times D \times D}, \\ \mathbf{Z}^{(k)} &= \begin{pmatrix} \mathbf{z}_p^{(k)} & \mathbf{z}_{p-1}^{(k)} & \dots & \mathbf{z}_1^{(k)} \\ \mathbf{z}_{p+1}^{(k)} & \mathbf{z}_p^{(k)} & \dots & \mathbf{z}_2^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z}_{n-1}^{(k)} & \mathbf{z}_{n-2}^{(k)} & \dots & \mathbf{z}_{n-p}^{(k)} \end{pmatrix}_{n-p \times p \times D}. \end{aligned}$$

It is noted that, the reversed parameter $\Phi^{(k)r}$ will be also estimated following the similar way for the ease of bidirectional repair.

Example 6 (Parameter estimation on $\mathbf{y}^{(0)}$, Example 5 continued): Consider again $\mathbf{x} = \{(6.5, 5), (7.5, 6.3), \dots, (19, 6)\}$ in Figure 2. Given order $p = 1$, we have

$$\mathbf{V}^{(0)} = \begin{pmatrix} 0 & 0 & 2 & 1 & 0.9 & \dots & 0 \\ 0 & 0 & -5.5 & -6.3 & -4.9 & \dots & 0 \end{pmatrix},$$

$$\mathbf{Z}^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 2 & 1 & \dots & 0 \\ 0 & 0 & 0 & -5.5 & -6.3 & \dots & 0 \end{pmatrix}$$

Referring to Equation 6, the parameter is estimated as

$$\Phi^{(0)} = \begin{pmatrix} 0.408 & -2.977 \\ -0.024 & -0.009 \end{pmatrix}$$

C. Candidate Generation

The repair candidate generation step S2 (in Line 3 in Algorithm 1) employs VARX(p) to infer the candidate repair with the estimated parameter $\Phi^{(k)}$.

1) *Candidates for Unidirectional Repair*: For unidirectional repair, we only need to generate forward candidates, specifically, for each point t , $\hat{\mathbf{y}}_t^{(k)}$ is given by

$$\hat{\mathbf{y}}_t^{(k)} = \sum_{i=1}^p (\mathbf{z}_{t-i}^{(k)} \Phi_i^{(k)} + \mathbf{x}_t) \quad (7)$$

according to $\mathbf{z}_{t-1}^{(k)}, \dots, \mathbf{z}_{t-p}^{(k)}$. We note that only candidates with $d(\hat{\mathbf{y}}_t^{(k)}, \mathbf{y}_t^{(k)}) > \tau$ need to be considered referring to the convergence condition in Equation 4 and thus be added to the candidate set $\hat{\mathbf{y}}_t^{(k)}$.

2) *Candidates for Bidirectional Repair*: As for bidirectional situation, VARX(p) will be also employed on the time series $\mathbf{y}^{(k)}$ reversely (from larger timestamp to smaller), namely, for each point t , the reverse candidate $\hat{\mathbf{y}}_t^{(k)r}$ will be given by

$$\hat{\mathbf{y}}_t^{(k)r} = \sum_{i=1}^p (\mathbf{z}_{t+i}^{(k)} \Phi_i^{(k)r} + \mathbf{x}_t) \quad (8)$$

according to $\mathbf{z}_{t+1}^{(k)}, \dots, \mathbf{z}_{t+p}^{(k)}$ and the parameters $\Phi^{(k)r}$ for the reverse repair. Likewise, we only consider candidates satisfying $d(\hat{\mathbf{y}}_t^{(k)r}, \mathbf{y}_t^{(k)}) > \tau$ and add them to the reverse candidate set $\hat{\mathbf{y}}_t^{(k)}$. Hence, there will be two different candidates for each point t .

Example 7 (Repair candidate $\hat{\mathbf{y}}^{(0)}$, Example 6 continued): Consider the parameter $\Phi^{(0)} = \begin{pmatrix} 0.408 & -2.977 \\ -0.024 & -0.009 \end{pmatrix}$, estimated in Example 6. Let threshold $\tau = 0.5$. When we use unidirectional repair, we generate candidates as $\hat{\mathbf{y}}_7^{(0)} = \mathbf{z}_6^{(0)} \Phi^{(0)} + \mathbf{x}_7 = (10.5, 11.7)$ referring to Equation 7, with $d(\hat{\mathbf{y}}_7^{(0)}, \mathbf{y}_7^{(0)}) = 2.7 > 0.5$. So on and so forth, we have the repair candidates are $\hat{\mathbf{y}}^{(0)} = \{-, -, -, +, +, +, (10.5, 11.7), -, -, -, -, -, +, (17.5, 9), -\}$, where ‘+’ corresponds to the labeled points and ‘-’ denotes no candidate.

Similarly, for bidirectional repairing, we have $\Phi^{(0)r} = \begin{pmatrix} -1.302 & 4.504 \\ -0.481 & 1.719 \end{pmatrix}$, thus $\hat{\mathbf{y}}_{12}^{(0)r} = \mathbf{z}_{13}^{(k)} \Phi^{(0)r} + \mathbf{x}_{12} = (16.5, 10.8)$ referring to Equation 8, and $d(\hat{\mathbf{y}}_{12}^{(0)r}, \mathbf{y}_{12}^{(0)}) = 1.7 > 0.5$. So on and so forth, we have the repair candidates are $\hat{\mathbf{y}}^{(0)r} = \{-, -, -, +, +, +, -, -, -, -, -, (16.5, 10.8), +, -, -\}$. It shows that the repair candidates from different direction can be quite different.

D. Repair Validation

The repair validation step S3 (in Line 4 in Algorithm 1) validates a repair to be accepted, i.e., assigning $\mathbf{y}_t^{(k+1)} = \hat{\mathbf{y}}_t^{(k)}$ to the above-mentioned generated repair candidate. According to the minimum change principle, the repair that differs least from its observation is preferred. Moreover, inspired by

[9], [10], the speed constraint is often useful in real-world scenarios and its effectiveness has been proven in various cases. Therefore, we introduce the *local speed constraints* s_t/s_t^l , which are determined by two nearest consecutive labels before/after the point t , to check whether a repair candidate can be accepted to solve the *over repair* problem. Remarkably, only one data point is repaired in each iteration, which is more efficient than the NP-hard problem of minimizing the total changes while considering the integrity constraints [21].

1) *Validation for Unidirectional Repair*: We first introduce some key variables as follows:

$$s_t = \frac{d(\mathbf{y}_{t_{f1}}^{(0)}, \mathbf{y}_{t_{f2}}^{(0)})}{|t_{f1} - t_{f2}|}, \quad s_t^l = \frac{d(\mathbf{y}_{t_{l1}}^{(0)}, \mathbf{y}_{t_{l2}}^{(0)})}{|t_{l1} - t_{l2}|}. \quad (9)$$

$$v_t = \frac{d(\hat{\mathbf{y}}_t, \mathbf{y}_{t_{f1}}^{(0)})}{|t - t_{f1}|}, \quad v_t^l = \frac{d(\hat{\mathbf{y}}_t, \mathbf{y}_{t_{l1}}^{(0)})}{|t - t_{l1}|}. \quad (10)$$

$$v_{t,o} = \frac{d(\mathbf{x}_t, \mathbf{y}_{t_{f1}}^{(0)})}{|t - t_{f1}|}, \quad v_{t,o}^l = \frac{d(\mathbf{x}_t, \mathbf{y}_{t_{l1}}^{(0)})}{|t - t_{l1}|}. \quad (11)$$

$$\text{valid}(\hat{\mathbf{y}}_t) = 1 - \frac{v_t - s_t}{v_{t,o} - s_t}. \quad (12)$$

Equation (9) shows how to compute the local speed constraints of point t , where $\mathbf{y}_{t_{f1}}^{(0)}$ and $\mathbf{y}_{t_{f2}}^{(0)}$ stand for the first and second former data point (i.e., $t_{f2} < t_{f1} < t$) with labels. Similarly, $\mathbf{y}_{t_{l1}}^{(0)}$ and $\mathbf{y}_{t_{l2}}^{(0)}$ are the first and second latter data points ($t_{l2} > t_{l1} > t$) with labels. Candidate speeds v_t/v_t^l in Equation (10) are the speeds between the candidate value of point t and the adjacent data point with labels before/after it, while $v_{t,o}/v_{t,o}^l$ shown in Equation (11) stand for the observation speeds between the original value of point t and the adjacent data point with labels before/after it, respectively.

The validity of a candidate value, computed by Equation (12), indicates the extent to which it represents a satisfactory repair. The reason for this is that both the candidate and the observation violate the local speed constraint, but if the degree of violation of the candidate is less than that of the observation, it can still be a possible repair. The greater the validity, the higher the probability that the candidate value is a valid repair.

Algorithm 2 introduces the validation process for unidirectional repairs. After sorting all candidates in ascending order based on the distance between the candidate value and its observation in Line 1, we create a dictionary in Line 2 where we store all validity scores. For each candidate c_t , Lines 3-6 try to find out whether it can be directly accepted in the current iteration. If yes, then the corresponding point t is assigned the candidate value and returned as a repair sequence. Otherwise, the function Assign also computes its validity and saves it in the dictionary. If no candidate is accepted directly, the candidate with the maximum validity is selected in Line 7. If the selected validity is greater than 0, this candidate is also accepted, as shown in Lines 8-9. Otherwise, no data is changed, which means that the proposed MINOR is terminated. Finally, the repaired sequence $\mathbf{y}^{(k+1)}$ is returned.

Algorithm 2: Validate – $U(\mathbf{x}, \mathbf{y}^{(0)}, \mathbf{y}^{(k)}, \hat{\mathbf{y}}^{(k)}, s_g)$

Input: time series \mathbf{x} , partially labeled $\mathbf{y}^{(0)}$, repaired sequence $\mathbf{y}^{(k)}$, candidates $\hat{\mathbf{y}}^{(k)}$, constraint s_g
Output: the repaired sequence $\mathbf{y}^{(k+1)}$

- 1 $Y^{(k)} \leftarrow \text{Sort } \hat{\mathbf{y}}^{(k)} \text{ in ascending order by } d(\hat{\mathbf{y}}_t^{(k)}, \mathbf{x}_t);$
- 2 $valid \leftarrow$ a new dict with *validity* of $Y^{(k)}$;
- 3 **foreach** $c_t \in Y^{(k)}$ **do**
- 4 **if** Assign($c_t, \mathbf{x}_t, \mathbf{y}^{(0)}, s_g, valid$) **then**
- 5 $\mathbf{y}_t^{(k)} \leftarrow c_t;$
- 6 **return** $\mathbf{y}^{(k)}$
- 7 $c_t \leftarrow \arg \max_{\hat{\mathbf{y}}_t} valid(\hat{\mathbf{y}}_t);$
- 8 **if** $valid(c_t) > 0$ **then**
- 9 $\mathbf{y}_t^{(k)} \leftarrow c_t;$
- 10 **return** $\mathbf{y}^{(k)}$

Algorithm 3: Assign($\hat{\mathbf{y}}_t, \mathbf{x}_t, \mathbf{y}^{(0)}, s_g, valid$)

Input: candidate $\hat{\mathbf{y}}_t$, observation \mathbf{x}_t , partially labeled $\mathbf{y}^{(0)}$, constraint s_g , validity dict $valid$
Output: whether $\hat{\mathbf{y}}_t$ is accepted

- 1 Compute $s_t, v_t, v_{t,o}$ by Equations 9-11;
- 2 **if** $v_t \leq \min(s_t, s_g)$ **then**
- 3 **return** True
- 4 **if** $v_{t,o} > s_t$ **then**
- 5 Compute $valid(\hat{\mathbf{y}}_t)$ by Equation 12;
- 6 **return** False

Algorithm 3 (function Assign) returns the result as to whether the specified candidate value can be a valid repair. First, we compute all associated key variables in Line 1. Then the candidate speed v_t is compared with the local speed constraint s_t and the global speed constraint s_g . If the candidate speed satisfies these speed constraints, the Boolean value True is returned in Line 3, which means that it can be a valid repair. Otherwise, we proceed to compare the observation speed and the local speed constraint. If the observation also violates the constraint, the validity of the given candidate is computed in Lines 4-5. However, if the observation satisfies the constraint but the candidate does not, then the validity of this candidate is left at 0, which means that it cannot be a possible repair, since accepting the candidate would worsen the error.

2) *Validation for Bidirectional Repair:* Bidirectional repair, as the name implies, involves the simultaneous repair of the time series in both forward and backward directions. On the one hand, we can generate two different candidate values for each point t . On the other hand, each candidate value is constrained by bidirectional speed constraints, i.e. the former s_t and the latter s_t^l .

Validity Computation: For a specified candidate, there exist a former validity and a latter validity as follows:

$$valid^f(\hat{\mathbf{y}}_t) = \begin{cases} 1, & v_t \leq s_t \\ 0, & v_t > s_t, v_{t,o} = s_t \\ \max(-1, 1 + \frac{v_t - s_t}{v_{t,o} - s_t}), & v_t > s_t, v_{t,o} < s_t \\ \min(1, 1 - \frac{v_t - s_t}{v_{t,o} - s_t}), & v_t > s_t, v_{t,o} > s_t \end{cases}, \quad (13)$$

$$valid^l(\hat{\mathbf{y}}_t) = \begin{cases} 1, & v_t^l \leq s_t^l \\ 0, & v_t^l > s_t^l, v_{t,o}^l = s_t^l \\ \max(-1, 1 + \frac{v_t^l - s_t^l}{v_{t,o}^l - s_t^l}), & v_t^l > s_t^l, v_{t,o}^l < s_t^l \\ \min(1, 1 - \frac{v_t^l - s_t^l}{v_{t,o}^l - s_t^l}), & v_t^l > s_t^l, v_{t,o}^l > s_t^l \end{cases} \quad (14)$$

$$\Delta_t^f = |t - t_{f1}|, \quad \Delta_t^l = |t - t_{l1}| \quad (15)$$

$$valid(\hat{\mathbf{y}}_t) = \frac{valid^f(\hat{\mathbf{y}}_t) \cdot \exp(-\Delta_t^f) + valid^l(\hat{\mathbf{y}}_t) \cdot \exp(-\Delta_t^l)}{\exp(-\Delta_t^f) + \exp(-\Delta_t^l)} \quad (16)$$

W.l.o.g, we explain how the Equation (13) comes about. The former validity of a given candidate $valid^f(\hat{\mathbf{y}}_t)$ is computed under 4 different cases, w.r.t. the relationship between the former local speed constraint s_t and the former observation speed v_t :

Case 1 it satisfies s_t , set to 1;

Case 2 it violates s_t but $v_{t,o} = s_t$, set to 0;

Case 3 it violates s_t but $v_{t,o}$ satisfies s_t , set to a negative value in $(-1, 0)$ based on the degree of violation;

Case 4 both violate s_t , set to a positive value in $(0, 1)$ based on the degree of violation.

Considering that the confidence in the speed constraint is low because the labels are further away on one side, we cut off the validity in Case 3 and Case 4. This avoids the scenario that (1) a candidate satisfies the constraint of the closer label (whose confidence is higher) but is rejected because it does not satisfy the low confidence constraint; (2) a candidate satisfies the low confidence constraint but does not fulfill the high confidence constraint and is nevertheless accepted.

Similarly, the latter validity can be computed by Equation (14). Then, we consider the time interval between the point t and the former/latter neighboring point with the label as the confidence coefficient of the former/latter validity, represented in Equation (15). Finally, the union validity $valid(\hat{\mathbf{y}}_t)$ can be computed by Equation (16), as a weighted average of the former and latter validity.

Check by Monotonicity: For a given candidate, we will first check its monotonicity.

Definition 4 (Monotonicity): Monotonicity is the property that a point does not move back and forth after several repairs and causes a dead loop. In the k -th iteration, if point t is not repaired before or the most recent repair is in the h -th iteration, which makes $(\hat{\mathbf{y}}_t - \mathbf{y}_t^{(k)}) \cdot (\mathbf{y}_t^{(k)} - \mathbf{y}_t^{(h)}) \geq 0$, then the candidate $\hat{\mathbf{y}}_t$ meets monotonicity.

Figure 3 illustrates the repair trajectories of three data points. For simplicity, we assume that each point has four

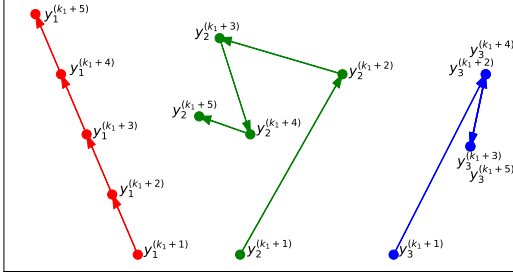


Fig. 3. Example of monotonicity

consecutive repairs. The arrows between the data points represent the directions of the displacement vectors during the repair. The first point (in red) satisfies monotonicity, since the angles between the displacement vectors of the successive repairs are acute (dot product no less than 0). Conversely, the repair of the second point (in green) results in an obtuse angle between the displacement vectors (dot product less than 0), which violates monotonicity. If non-monotonic points are not removed, bidirectional repair may lead to a situation similar to the third point (in blue), where the point returns to a value close to its previous state after several iterations, leading to redundant cycles or even deadlocks.

In bidirectional repair, MINOR – B generates two candidates for each point t and we find in practice that these two candidates may be close to each other. If points that violate monotonicity are not pruned, it may be that $y_t^{(k+2)} \approx y_t^{(k)}$ applies, i.e. it loops every two iterations, making the method difficult to terminate. Therefore, we introduce the Algorithm 4 to remove all points that violate monotonicity according to the Definition 4. The given candidate is discarded if the dot product is less than 0 (Line 4).

Algorithm 4: Check($y^{(k)}$, $Y^{(k)}$)

Input: repaired sequence $y^{(k)}$, candidates $Y^{(k)}$

Output: $Y^{(k)}$ after invalid candidates removed

```

1 foreach  $c_t \in Y^{(k)}$  do
2   for  $h \leftarrow k-1$  to 0 do
3     if  $y_t^{(h)} \neq y_t^{(k)}$  then
4       if  $(c_t - y_t^{(k)}) \cdot (y_t^{(k)} - y_t^{(h)}) < 0$  then
5          $Y^{(k)} \leftarrow Y^{(k)} - \{c_t\}$ ;
6       break;
7 return  $Y^{(k)}$ 

```

Bidirectional Validation: Algorithm 5 provides the process for bidirectional repair. After pruning by Check, all candidates are sorted in ascending order as in Algorithm 2. Lines 4-19 attempt to find out whether a candidate c_t can be accepted directly or whether a validity is assigned. As shown

Algorithm 5: Validate – B(x , $y^{(0)}$, $y^{(k)}$, $\hat{y}^{(k)}$, $\hat{y}^{(k)r}$, s_g)

Input: time series x , partially labeled $y^{(0)}$, repaired sequence $y^{(k)}$, candidates $\hat{y}^{(k)}$, $\hat{y}^{(k)r}$, constraint s_g

Output: the repaired sequence $y^{(k+1)}$

```

1  $Y^{(k)} \leftarrow \text{Check}(y^{(k)}, \hat{y}^{(k)} \cup \hat{y}^{(k)r})$ ;
2  $Y^{(k)} \leftarrow \text{Sort } Y^{(k)}$  in ascending order by  $d(\hat{y}_t^{(k)}, x_t)$ ;
3  $valid \leftarrow$  a new dict with validity of  $Y^{(k)}$ ;
4 foreach  $c_t \in Y^{(k)}$  do
5   if  $c_t \in \hat{y}^{(k)}$  and  $y_{t+1}^{(k)}$  is not repaired/labeled then
6     if Assign( $c_t, x_t, y^{(0)}, s_g, valid$ ) then
7        $y_t^{(k)} \leftarrow c_t$ ;
8       return  $y^{(k)}$ 
9   continue;
10  if  $c_t \in \hat{y}^{(k)r}$  and  $y_{t-1}^{(k)}$  is not repaired/labeled then
11    if Assign( $c_t, x_t, y^{(0)r}, s_g, valid$ ) then
12       $y_t^{(k)} \leftarrow c_t$ ;
13      return  $y^{(k)}$ 
14    continue;
15  Compute  $s_t, s_t^l, v_t, v_t^l, v_{t,o}, v_{t,o}^l$  by Equations 9-11;
16  if  $\max(v_t, v_t^l) \leq \min(s_g, s_t, s_t^l)$  then
17     $y_t^{(k)} \leftarrow c_t$ ;
18    return  $y^{(k)}$ 
19  Compute  $valid(c_t)$  by Equations 13-16;
20  $c_t \leftarrow \arg \max_{\hat{y}_t} valid(\hat{y}_t)$ ;
21 if  $valid(c_t) > 0$  then
22    $y_t^{(k)} \leftarrow c_t$ ;
23 return  $y^{(k)}$ 

```

in Line 5 and Line 10, the bidirectional speed constraints do not always apply. The reason is that in the early stage, most of the points have not been repaired yet and the generated candidates are usually closer to the label points in their own repair direction (forward or backward only). Therefore, we relax the constraints to speed up the process until the neighbor point is repaired (Lines 15-19). Lines 5-9 consider the forward direction and Lines 10-14 consider the backward direction. Lines 20-23 are identical to the Algorithm 2.

Example 8 (Validate repair $\hat{y}_t^{(6)}$, Example 7 continued): Consider the candidates in the 6-th iteration of MINOR – B(1), $\hat{y}^{(6)} = \{-, -, -, +, +, +, (10.5, 11.7), -, (12.5, 14.4), (13.0, 13.8), (15.0, 12.7), (16.1, 11.9), +, (17.5, 9.0), -\}$, $\hat{y}^{(6)r} = \{-, -, (8.2, 6.8), +, +, +, (10.0, 14.3), -, -, -, (15.1, 12.2), (16.5, 10.7), +, -\}$, first, we check the monotonicity of candidates by Algorithm 4, remaining $\hat{y}_{14}^{(6)}$, $\hat{y}_7^{(6)r}$, $\hat{y}_3^{(6)r}$, $\hat{y}_{11}^{(6)}$, $\hat{y}_7^{(6)}$, and sorted in the ascending order by $d(\hat{y}_t^{(k)}, x_t)$. Let's take $\hat{y}_7^{(6)}$ for example, since $y_6^{(6)}$ is labeled and $y_8^{(6)}$ has been repaired in previous iteration, so we will consider the the validity bidirectionally. By Equations 9-11 $s_7 = 1.08$, $s_7^l = 2.36$, $s_g = 7.1$, $v_7 = 2.06$, $v_7^l = 1.04$, $\min(v_7, v_7^l) > \max(s_7, s_7^l, s_g)$, so we can't accept it immediately, and compute

$v_{7,o} = 4.82, v_{7,o}^l = 1.31, \text{valid}^f(\hat{\mathbf{y}}_7^{(6)}) = 0.74, \text{valid}^l(\hat{\mathbf{y}}_7^{(6)}) = 1, \Delta_t^f = 1, \Delta_t^l = 6, \text{valid}(\hat{\mathbf{y}}_7^{(6)}) = 0.74$. Finally, the validity of each candidate is 0, 0.03, 0, 1.0, 0.74, and $\hat{\mathbf{y}}_{11}^{(6)r}$ is accepted in this iteration.

IV. EFFICIENT COMPUTATION

The three main steps in Algorithm 1 suffer from the high time cost of matrix computation, so different techniques are presented for each of them to enable efficient computation in our iterative repair scenario.

A. Pruning in Estimation

The matrices $\mathbf{Z}^{(k)}, \mathbf{V}^{(k)}$ are involved to estimate the parameter $\Phi^{(k)}$ in the k -th iteration, whose element $\mathbf{z}_j^{(k)}$ denotes the difference between the labeled or repaired value and the observation. We then show two techniques to speed up the parameter estimation.

1) *Matrix Pruning*: Similar to [8], most of the elements in $\mathbf{Z}^{(k)}, \mathbf{V}^{(k)}$ are equal to 0. The following conclusion still holds that the same parameter $\Phi^{(k)}$ can still be computed by Equation (6) after removing the rows in $\mathbf{Z}^{(k)}$ whose values are equal to 0 and the corresponding rows in $\mathbf{V}^{(k)}$.

2) *Incremental Computation*: In each iteration in Algorithm 1, the parameter $\Phi^{(k)}$ is estimated by Equation (6) w.r.t. $\mathbf{Z}^{(k)}$ and $\mathbf{V}^{(k)}$ over all $(nD + npD)$ points. We show that, similar to [8], $\Phi^{(k)}$ can be computed incrementally by considering only the changed values instead of the entire $\mathbf{Z}^{(k)}$ and $\mathbf{V}^{(k)}$. Specifically, all the $(p^2 D^2 + pD)$ values can be recursively updated. And hence, the time complexity of parameter estimation in each iteration is reduced from $O(n)$ to $O(1)$ time.

B. Incremental Candidate Generation

In the multivariate case, the matrix calculation in candidate generation is much slower than the numerical operations in the univariate case, and parameter estimation is not even the most time-consuming part. In fact, Equation (7) is an intermediate step of the following matrix multiplication:

$$\hat{\mathbf{C}}^{(k)} = \mathbf{X} + \mathbf{Z}^{(k)} \Phi^{(k)} \quad (17)$$

where

$$\hat{\mathbf{C}}^{(k)} = \begin{pmatrix} \hat{\mathbf{y}}_{p+1}^{(k)} \\ \hat{\mathbf{y}}_{p+2}^{(k)} \\ \vdots \\ \hat{\mathbf{y}}_n^{(k)} \end{pmatrix}, \mathbf{X} = \begin{pmatrix} \mathbf{x}_{p+1} & \cdots & \mathbf{x}_2 \\ \mathbf{x}_{p+2} & \cdots & \mathbf{x}_3 \\ \vdots & \vdots & \ddots \\ \mathbf{x}_{n-1} & \cdots & \mathbf{x}_{n-p+1} \end{pmatrix}_{n-p \times D}$$

To generate all candidates, $\mathbf{Z}^{(k)}$ must be updated with a time complexity of $O(npD)$, and then the computation is performed according to Equation (17), which has a time complexity of $O(npD^2)$. However, \mathbf{X} remains unchanged throughout the repair process, $\Phi^{(k)}, \mathbf{Z}^{(k)}$ can be computed incrementally from $\mathbf{Z}^{(k-1)}$.

Let r be the changed point, i.e. $\mathbf{y}_r^{(k)} \neq \mathbf{y}_r^{(k-1)}$. Then, for $1 \leq i \leq n-p, 1 \leq j \leq p$, we have

$$\mathbf{z}_{ij}^{(k)} = \begin{cases} \mathbf{z}_{ij}^{(k-1)}, & r \neq p+i-j \\ \mathbf{y}_r^{(k)} - \mathbf{x}_r, & r = p+i-j \end{cases} \quad (18)$$

According to Equation (18), only up to p numbers need to be updated. Similarly, it is unnecessary to compute all candidates according to Equation (17), because the labeled points are rare (around 1% to 5%). Then only few of $\mathbf{z}_i^{(k)}$ in $\mathbf{Z}^{(k)}$ is non-zero at the beginning. Assume that m points are labeled or repaired, the time complexity of candidate generation in each iteration is reduced from $O(npD^2)$ to $O(mp^2D^2)$ time, i.e., $O(n)$ to $O(m)$, where $mp \ll n$.

Example 9 (Incremental Candidate Generation, Example 8 continued): In example 8, $\hat{\mathbf{y}}_{11}^{(6)r} = (15.1, 12.2)$ is accepted, so in the 7-th iteration, $p = 1, r = 11, \mathbf{z}_{11,1}^{(7)} = \mathbf{y}_{11}^{(7)} - \mathbf{x}_{11} = (0.2, -0.8)$, and other $\mathbf{z}^{(7)}$ will remain unchanged, namely $\mathbf{z}_{i,j}^{(7)} = \mathbf{z}_{i,j}^{(6)}, 1 + i - j \neq r$.

C. Memorization in Validation

In Algorithm 3 and Algorithm 5, local speed constraints s_t/s_t^l and observation speeds $v_{t,o}/v_{t,o}^l$ are required to compute the validity of the point t . As the labeled sequence $\mathbf{y}^{(0)}$ and the observations \mathbf{x} do not change during the repair, these variables can only be computed once and then memorized. This avoids the repeated computation of Euclidean distances, which can be very time-consuming when repairing high-dimensional data.

V. EXPERIMENT

In this section, we evaluate our proposed MINOR by comparing it with 6 SOTA approaches under various different settings and scenarios. Moreover, we highlight the key findings and discuss the limitations. The code and data of this work are available online¹.

A. Settings

We run experiments on a Windows 10 server with a 2.80GHz Intel Core CPU and 32GB RAM. Deep-learning methods are employed with GPU without comparing the efficiency.

1) *Datasets*: We perform an experimental analysis with three multivariate time series datasets. (1) The GPS dataset is collected from a person carrying a smartphone while walking (about 10%) and taking a bus. There are 678 dirty points out of 1,731 in the dataset, identified by comparing the map and the trajectory. Their most probable values on the map are labeled as ground truth. (2) The Intel Lab Data (ILD) [24] dataset contains information collected from 54 sensors at Intel Berkeley Research Lab at 31-second intervals. We select temperature and humidity in our test because they are highly correlated. After simple pre-processing to eliminate missing values, the Sensor ID 31 data yielded 48,018 data points. Continuous two-dimensional Gaussian noise is injected to simulate typical sensor errors: Shift and Innovational (see Section V-C4 for details). (3) The ECG dataset [25] contains features extracted from ECG signals with 32 dimensions. Despite the presence of labeled outliers, the true values and timestamps are missing. We assume that the dataset is acquired at a fixed frequency and remove the labeled outliers, giving us

¹<https://github.com/zaqthss/minor>

TABLE II
SUMMARY OF DATASETS

	Size	#Dim	Error	#Series
ILD [24]	48k	2	Clean after pre-process	1
ECG [25]	12k	32	Clean after pre-process	1
GPS	1.7k	2	Embedded	1
Car [26]	577	1	Clean	120
Lightning2 [26]	637	1	Clean	121
DPTW [26]	80	1	Clean	600
IEPGRTrain [26]	601	1	Clean	62

TABLE III
SUMMARY OF COMPARED METHODS

Algorithm	Dimension	Process	Type
MINOR – U	multivariate	batch	smooth + constraint
MINOR – B	multivariate	batch	smooth + constraint
MINOR – O	multivariate	online	smooth + constraint
IMR [8]	univariate	online	smooth
Akane [16]	univariate	batch	statistical
VARX [28]	multivariate	batch	smooth
MTCSC [9]	multivariate	batch	constraint
TranAD [17]	multivariate	online	deep learning
USAD [18]	multivariate	online	deep learning

a total of 93,948 data points, and we use the first 12,000 for the ease of computation. Similar to the ILD dataset, a continuous high-dimensional Gaussian noise is introduced.

2) Metrics:

- RMSE [27] is employed to evaluate the repair.
- Time cost. The time required to complete the cleaning process, excluding error injection, data loading and RMSE calculation.

3) *Baselines*: We compare our proposals with 6 baselines, which are listed in Table III. IMR is an iterative repairing algorithm that cleans univariate time series data step by step according to the minimum change principle. Akane exploits the inherent cyclic patterns in time series through an analogous comparison with fixed combinations in text data, introduces the concept of perplexity, and aims to minimize perplexity for data cleaning. VARX is a typical model used in multivariate time series data modeling based on smoothing techniques. MTCSC is able to adaptively capture speed constraints and considers the data distribution to clean data in a streaming manner. Given the relative scarcity of deep learning-based algorithms for cleaning time series, we introduce two SOTA anomaly detection approaches TranAD and USAD, and consider their predictions/reconstructions as repair candidates.

B. Comparison on Real Errors

The experiments on real errors over GPS data consider various settings, including (1) order p , (2) convergence threshold τ , (3) max-num-iterations, and (4) labeling rate. Similar results are also observed in ILD and ECG and omitted.

1) *Case Study*: Figure 4 shows a sample section of the GPS data. Since it contains both walking and bus segments, there are significant speed differences within the dataset. Therefore,

TABLE IV
GPS DATA WITH REAL ERRORS

	RMSE	repair time(ms)	iteration numbers
Dirty	0.8456	-	-
MINOR – B	0.3474	204.01	2854
-w/o CIC	0.3474	3012.82	2854
-w/o IC	0.3474	4211.43	2854
-Uni	0.4478	233.58	11390
MINOR – U	<u>0.4170</u>	35.21	2081
-w/o CIC	<u>0.4170</u>	299.77	2081
-w/o IC	<u>0.4170</u>	534.11	2081
VARX	0.7112	0.17	-
IMR	0.7504	27.23	4566
MTCSC – C	0.7240	1.1	-
MTCSC – A	0.6344	11.2	-
Akane	2.9409	152	-
TranAD	162.18	-	-
USAD	161.97	-	-

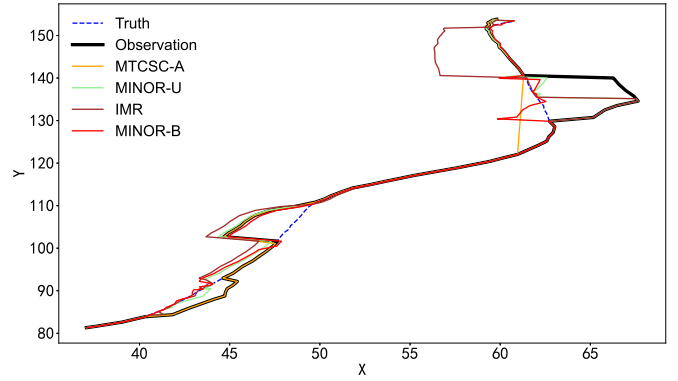


Fig. 4. Case Study of GPS

we also compare MTCSC – A, which can adaptively adjust the speed constraints. Table IV shows the repair results of the different methods, where we compare MINOR – B and MINOR – U with different variants: -w/o CIC denotes the method without incremental candidate generation, and -w/o IC denotes the method without incremental computations. The comparison shows that our proposed incremental computation strategy reduces the time cost to 5% of the original, which proves its necessity. We also compared MINOR – B with its univariate version, MINOR – B – Uni, which cleans each dimension separately. The results show that considering the entire dimensions as a whole has an effect.

2) *Varying Order p* : As shown in Figure 5(a), VARX performs a bit better with increasing p , as more historical values are included in the prediction of each value. Interestingly, MINOR – B performs much better than others when the order $p = 1$, but worse when p is larger. This is because Lines 5 and 10 in Algorithm 5 are developed specifically for $p = 1$ and would otherwise fail. However, the use of the simplest model MINOR – B(1) is sufficient and is recommended for all cases. As shown in Figure 5(b), MINOR – B causes considerable time costs due to bidirectional validation. Therefore,

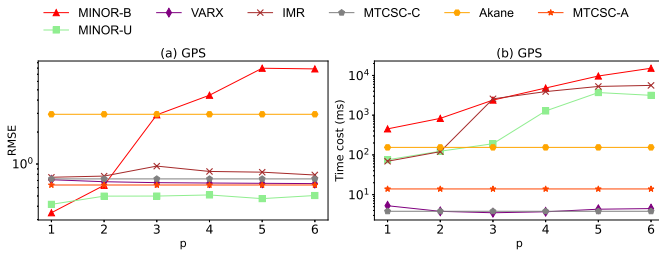


Fig. 5. Varying order p on GPS

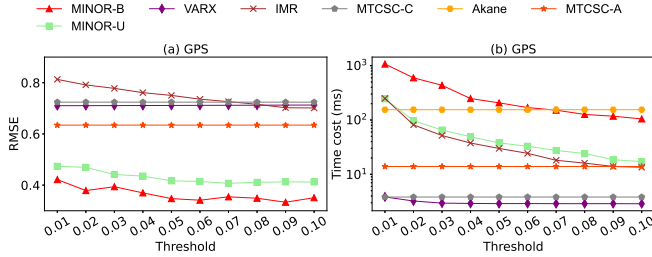


Fig. 6. Varying threshold on GPS

we introduce MINOR – U, which achieves decent results (the best when $p > 1$) with much less time cost.

3) *Varying Convergence Threshold τ* : Figure 6(a) shows that our MINOR – B and MINOR – U perform quite stable when threshold increases, with MINOR – B performing the best. In theory, the larger the threshold, the faster our methods converge, so as IMR. As expected, it is observed in Figure 6(b) that with increasing threshold values, the time costs of MINOR – B and MINOR – U gradually decrease.

4) *Specifying Maximum Number of Iterations*: Figure 7(b) shows that all iterative methods converge over 10k iterations. However, Figure 7(a) shows that both IMR and MINOR – U have better repair accuracy initially as the number of iterations increases, but then deteriorate. This problem is particularly severe for IMR, indicating that this algorithm performs significant over-repairs, i.e. it repairs correct data into incorrect states. In contrast, our algorithm MINOR – B shows robust performance with a stable decrease in RMSE, indicating that it effectively mitigates the problem of over-repair.

5) *Varying Labeling Rate*: Figure 8(a) demonstrates that the repair performance of MINOR – B and MINOR – U is greatly improved with the increase of labeling rate. It is noted that even with 3% labels (1% for ILD and ECG), our proposals can still outperform others. The reason why the performance of the other methods also improves slightly with the labeling rate increases is that the labeled points are considered correctly repaired in the RMSE calculation. Figure 8(b) shows that labeling rate has little influence over running time.

C. Comparison on Synthetic Errors

The experiments on synthetic errors focus on varying error rates, error lengths, data sizes and error patterns, respectively. Moreover, the support for online computing is also tested.

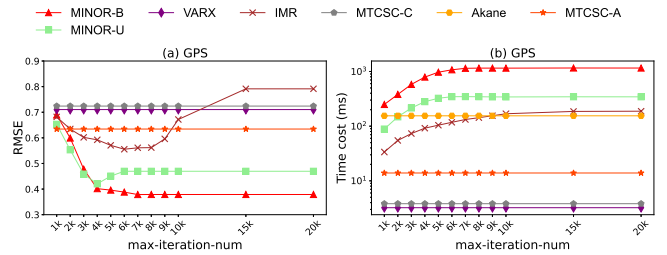


Fig. 7. Varying maximum iteration number on GPS

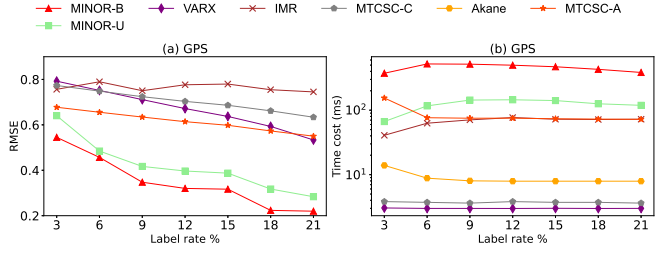


Fig. 8. Varying label rate on GPS

1) *Varying Error Rate $e\%$* : Figures 9(a)(c) show that MINOR – B achieves the best accuracy and both are robust against error rates. It is noted that MTCSC has better robustness to error rates, especially for the high-dimensional data ECG with 32 dimensions. Figures 9(b)(d) show that our proposals incur higher time costs as the error rate increases, since the number of iterations increases. If the error rate in Figure 9(d) is 7.5% or more, the time cost of MINOR – U remains stable as its preset max-num-iterations is reached.

2) *Varying Error Length*: In Figures 10(a)(c), our proposals show improved repair accuracy with increasing error length, with MINOR – B performing best and MINOR – U second best, except in cases with small error length on ILD. This emphasizes their strength in dealing with large consecutive errors. And due to the labeling strategy, we only label one point when the error length is smaller than 10, which leads to poorer performance. It can be also shown in Figures 10(b)(d) that we cost little time as we converge very quickly.

3) *Varying Data Size n* : Figures 11(a)(c) show that our proposals also have good scalability in terms of data volume, achieve high accuracy (best and second best). MINOR – B runs faster than MINOR – U in Figure 11(d) because it can be terminated faster.

4) *Varying Error Pattern*: We consider two different error patterns, Shift and Innovational. Shift means that a data set is collectively shifted, while the latter means that errors manifest as gradual increases or decreases over time [29]. As shown in Figure 9(a) and Figure 12(a), MINOR – B consistently demonstrates the best repair performance, followed closely by MINOR – U, on both of the error patterns. The time cost in terms of error rates are also the same, shown in Figure 12(b).

5) *Evaluation on Online Computing*: Time series are generated continuously in some cases. Although MINOR – B

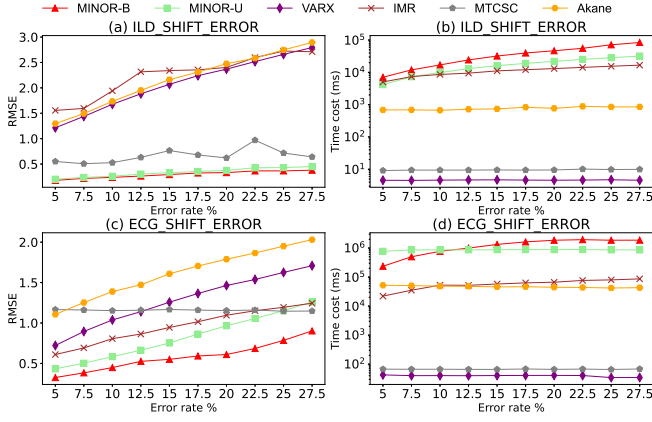


Fig. 9. Varying error rate on ILD(a-b) and ECG(c-d) with SHIFT error

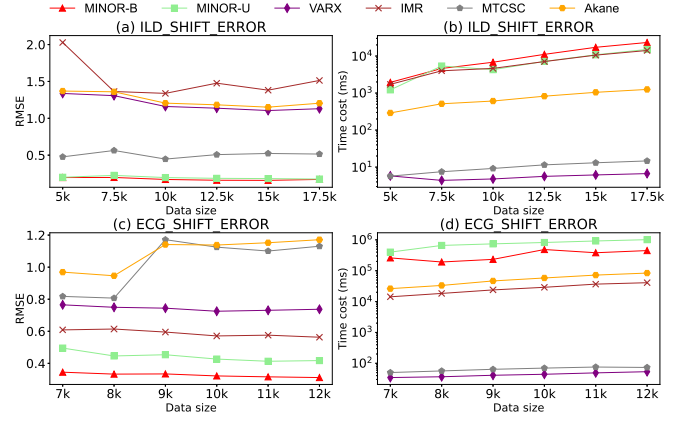


Fig. 11. Varying data size on ILD(a-b) and ECG(c-d) with SHIFT error

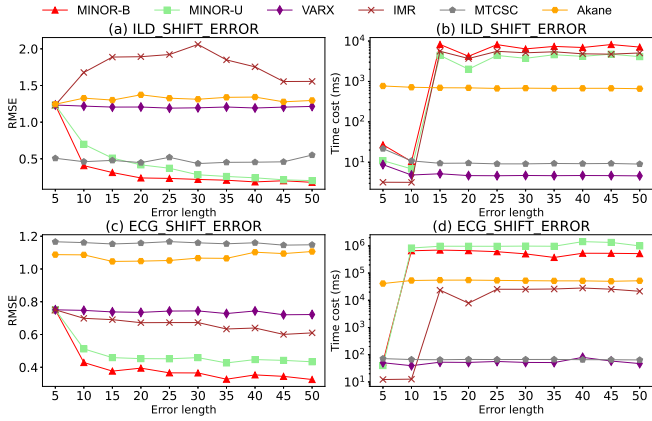


Fig. 10. Varying error length on ILD(a-b) and ECG(c-d) with SHIFT error

performs better than MINOR – U, the former cannot be used online because it requires the information in the future. Following [8], we present MINOR – O based on MINOR – U, where all historical data are treated as one labeled segment, the parameter is estimated incrementally, the repair candidate is generated for each incoming point, and the candidate is accepted if it satisfies the local speed constraint or is more reasonable than the observation, without iteration. Here, we omit the global speed constraint since the streaming data may evolve over time.

Figure 13 shows the repair results on ILD. We repeat each test 10 times (as in other experiments) and report the average to obtain reliable results. However, we still see some fluctuations in Figure 13(b) due to the fact that the errors are not uniform. Since originally clean data points are not labeled, IMR and VARX cause over-repair and reduce data quality. VARX has a low throughput because it has to relearn the parameter each time. Our MINOR – O still has the best accuracy with a higher efficiency, which shows its capability in online situations.

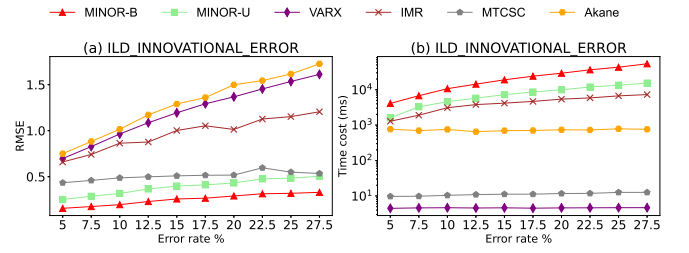


Fig. 12. Varying error rate on ILD with INNOVATIONAL error

D. Applications

We perform classification and clustering tasks to investigate the impact of different cleaning methods on applications. Specifically, we evaluate the performance of the original data as “Clean”, introduce 10% random shift errors as “Dirty”, and repair the data using different methods. All four datasets are naturally divided into training and test datasets, into each of which we inject synthetic errors. The clustering task only uses the training sets. For classification, we use KNN [30] classifier, select the best K via a grid search and take the F1-score [2] as metric. For clustering, we use K – means [31] and take the RI [32] as metric. As expected, Figure 14 shows that our MINOR – B performs better than others and is closer to the results of “Clean”. After cleaning, the distance between the time series is calculated more correctly.

E. Highlights and Limitations

We summarize the experimental highlights as follows: 1) Our proposal shows superior performance in cleaning time series with consecutive errors (above 45%) compared to existing methods, with accuracy remaining stable even at high error rates (above 20%), high dimensions (above 30) and different patterns; 2) Considering the entire dimensions as a whole has an effect; 3) Incremental computation can greatly improve efficiency (only 5% time after); 4) MINOR – B(1) is sufficient in most of the cases and MINOR – U is capable of online computing; 5) It suggests that we can set a relative larger

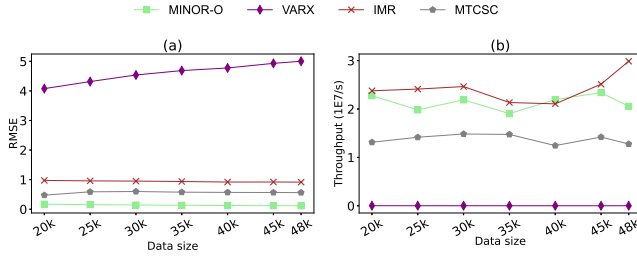


Fig. 13. Online computing on ILD

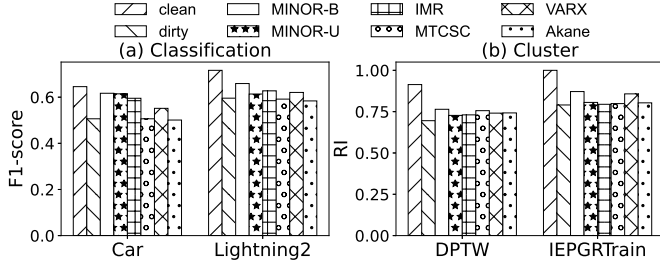


Fig. 14. Classification and Clustering over clean, dirty and repaired data

threshold or a smaller max-num-iterations to speed up the methods without losing the repair accuracy; 6) The proposed MINOR – B can effectively solve the over-repair problem. 7) Our method achieves good results with very few labeled points (1% - 3%), which shows its usefulness in practice; 8) Data cleaning promotes the accuracy of data science applications, and ours behave best.

On the other hand, our work still has some limitations. 1) There is no proof of convergence in the general case; 2) The time efficiency can still be improved; 3) We need some (even few) manually labeled truths and cannot initiate the repair process independently by labeling a small subset of data.

VI. RELATED WORK

A. Traditional Cleaning

a) *Constraint-based Cleaning*: SCREEN [10] proposes an online cleaning technique that takes speed constraints into account but is limited to processing univariate data. SpeedAcc [11], extends its focus to the consideration of acceleration constraints in univariate time series analysis. RCSWS [33] provides a solution for cleaning GPS data using distance constraints and statistical analysis in the context of sliding windows. Clean4MTS [12] proposes a two-step method for cleaning high-dimensional time series data by validating data quality in rows and columns and shows good results. However, it relies heavily on rules, and inaccurate extracted rules may somewhat affect the accuracy. MTCSC [9] can adaptively extract the speed constraint and leverages the minimum fix principle to improve the repair accuracy, but some key parameters such as window size still need to be carefully preset.

b) *Smoothing-based Cleaning*: Moving average [34] is widely used to smooth time series data and make predictions. The exponentially weighted moving average (EWMA) [35]

uses exponentially decreasing weights over time. IMR [8] repairs time series data iteratively, but is not able to use the correlation between dimensions in multivariate time series data to support the repair process. However, these smoothing-based methods often change a significant portion of the dataset, which could change the original data distributions.

c) *Statistical-based Cleaning*: LsGreedy [14] models velocity changes between neighboring points to correct small errors using velocity constraints. [36] contains an incremental clustering method where historical data is first clustered and the average values of the clusters are used for repairs. STPM [37] extracts detailed data patterns from historical datasets to improve the data cleaning process. However, the statistically based cleaning techniques can occasionally lead to excessive cleaning and misinterpret typical anomalies as noise or outliers. The recent work Akane [16] introduces a comprehensive four-phase algorithmic framework for improving adjustment accuracy, but it is only applicable to univariate data.

B. Machine Learning-based Cleaning

HoloClean [38] uses weakly supervised learning to build a probabilistic cleaning model for relational data. TranAD [17] integrates transformer-based encoder-decoder networks with adversarial training for prediction-driven anomaly detection. USAD [18] combines an encoder-decoder architecture with adversarial training and uses the constructed value as an anomaly score.

CONCLUSION

In this paper, we study the data cleaning problem over consecutive errors in multivariate time series with a limited number of labels. Existing methods work well on single errors in univariate data, but they are unable to learn an accurate model to handle the consecutive errors in multivariate data. Moreover, due to the inability of the model, some works suffer from the problem of over-repair, where the original clean data may still be altered after repair. To solve the above problems, we propose MINOR, which is based on the minimum change principle and estimates the parameter iteratively over all dimensions together.

We formalize the repair problem and introduce the iterative framework, which consists of three main steps: parameter estimation, candidate generation, and repair validation. To prevent repairing the correct data, we provide a bidirectional repair method MINOR-B that uses both the historical and future data. Local speed constraints can be extracted and help to decide whether the derived candidate can be accepted. To support online computation, we present a unidirectional validation method MINOR-U. We also provide efficient computational techniques to reduce the runtime. Experiments on different datasets with different error rates, error lengths, data sizes and error patterns show that our proposals work well on consecutive errors even with few labels around 1%. The convergence guarantee in the general case is complex and is left for future work.

REFERENCES

- [1] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, “A review on outlier/anomaly detection in time series data,” *ACM Comput. Surv.*, vol. 54, no. 3, pp. 56:1–56:33, 2022.
- [2] C. C. Aggarwal, “Outlier analysis second edition,” 2016.
- [3] A. Karkouch, H. Mousannif, H. A. Moatassime, and T. Noël, “Data quality in internet of things: A state-of-the-art survey,” *J. Neww. Comput. Appl.*, vol. 73, pp. 57–81, 2016.
- [4] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.
- [5] B. K. Nelson, “Time series analysis using autoregressive integrated moving average (arima) models,” *Academic emergency medicine*, vol. 5, no. 7, pp. 739–744, 1998.
- [6] J. H. Stock and M. W. Watson, “Vector autoregressions,” *Journal of Economic perspectives*, vol. 15, no. 4, pp. 101–115, 2001.
- [7] L. Bauwens, S. Laurent, and J. V. Rombouts, “Multivariate garch models: a survey,” *Journal of applied econometrics*, vol. 21, no. 1, pp. 79–109, 2006.
- [8] A. Zhang, S. Song, J. Wang, and P. S. Yu, “Time series data cleaning: From anomaly detection to anomaly repairing,” *Proc. VLDB Endow.*, vol. 10, no. 10, pp. 1046–1057, 2017.
- [9] A. Zhang, Z. Wu, Y. Gong, Y. Yuan, and G. Wang, “Multivariate time series cleaning under speed constraints,” 2024. [Online]. Available: <https://arxiv.org/abs/2411.01214>
- [10] S. Song, A. Zhang, J. Wang, and P. S. Yu, “SCREEN: stream data cleaning under speed constraints,” in *SIGMOD Conference*. ACM, 2015, pp. 827–841.
- [11] 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.
- [12] X. Ding, G. Li, H. Wang, C. Wang, and Y. Song, “Time series data cleaning under expressive constraints on both rows and columns,” in *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 2024, pp. 3682–3695.
- [13] R. Kang, S. Song, and C. Wang, “Conditional regression rules,” in *ICDE*. IEEE, 2022, pp. 2481–2493.
- [14] A. Zhang, S. Song, and J. Wang, “Sequential data cleaning: A statistical approach,” in *SIGMOD Conference*. ACM, 2016, pp. 909–924.
- [15] H. Wang, A. Zhang, S. Song, and J. Wang, “Streaming data cleaning based on speed change,” *VLDB J.*, vol. 33, no. 1, pp. 1–24, 2024.
- [16] X. Han, H. Xiong, Z. He, P. Wang, C. Wang, and X. S. Wang, “Akane: Perplexity-guided time series data cleaning,” *Proc. ACM Manag. Data*, vol. 2, no. 3, p. 121, 2024.
- [17] S. Tuli, G. Casale, and N. R. Jennings, “Tranad: Deep transformer networks for anomaly detection in multivariate time series data,” *Proc. VLDB Endow.*, vol. 15, no. 6, pp. 1201–1214, 2022.
- [18] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, “USAD: unsupervised anomaly detection on multivariate time series,” in *KDD*. ACM, 2020, pp. 3395–3404.
- [19] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, “Towards certain fixes with editing rules and master data,” *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 173–184, 2010.
- [20] L. A. Hageman, *Applied iterative methods*. Elsevier, 2014.
- [21] 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*. ACM, 2005, pp. 143–154.
- [22] C. R. Rao, *Linear statistical inference and its applications*. John Wiley & Sons, 2009, vol. 22.
- [23] B. Cheng, “Yule–walker equations,” *Wiley StatsRef: Statistics Reference Online*, 2014. [Online]. Available: <http://dx.doi.org/10.1002/9781118445112.stat05549>
- [24] S. Madden, “Intel berkeley research lab data,” <https://db.csail.mit.edu/labdata/labdata.html>, 2003, accessed: 2024-04-10.
- [25] S. Yoon, J.-G. Lee, and B. S. Lee, “Ultrafast local outlier detection from a data stream with stationary region skipping,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1181–1191.
- [26] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, “The ucr time series classification archive,” July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.
- [27] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, “Adaptive cleaning for RFID data streams,” in *VLDB*. ACM, 2006, pp. 163–174.
- [28] W. B. Nicholson, D. S. Matteson, and J. Bien, “Varx-l: Structured regularization for large vector autoregressions with exogenous variables,” *International Journal of Forecasting*, vol. 33, no. 3, pp. 627–651, 2017.
- [29] R. S. Tsay, “Outliers, level shifts, and variance changes in time series,” *Journal of forecasting*, vol. 7, no. 1, pp. 1–20, 1988.
- [30] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [31] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.
- [32] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.
- [33] C. Fang, F. Wang, B. Yao, and J. Xu, “Gpsclean: A framework for cleaning and repairing GPS data,” *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 3, pp. 40:1–40:22, 2022.
- [34] D. R. Brillinger, *Time series - data analysis and theory*, ser. Classics in applied mathematics. SIAM, 2001, vol. 36.
- [35] E. S. Gardner, “Exponential smoothing: The state of the art—part ii,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 637–666, 2006. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0169207006000392>
- [36] D. J. Hill and B. S. Minsker, “Anomaly detection in streaming environmental sensor data: A data-driven modeling approach,” *Environ. Model. Softw.*, vol. 25, no. 9, pp. 1014–1022, 2010.
- [37] M. Milani, Z. Zheng, and F. Chiang, “Currentclean: Spatio-temporal cleaning of stale data,” in *ICDE*. IEEE, 2019, pp. 172–183.
- [38] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, “Holoclean: Holistic data repairs with probabilistic inference,” *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1190–1201, 2017.