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

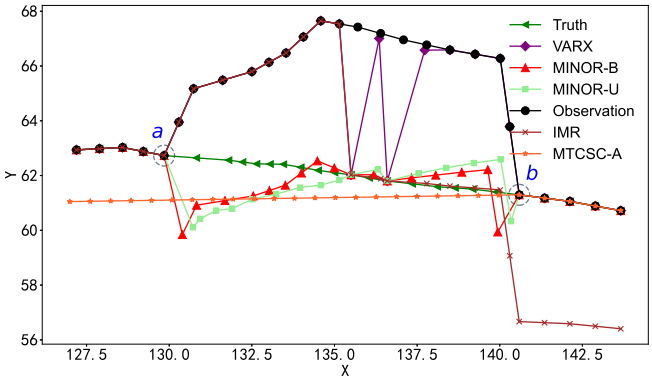

Fig. 1. Example of observations and different repairs

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 is 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 limited number of labeled truth of dirty observations to advance the repair. The truth value can be obtained via manual annotation or reliable sensors with a relatively long sensing period [20]. 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 exist 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 [21]. 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 a 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 cleaNing 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 [22], 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
$\boldsymbol{x}$	time series
$\boldsymbol{x}_i$, $\boldsymbol{x}[i]$	i-th data point in $\boldsymbol{x}$
$\boldsymbol{y}$	truth-labeled/repaired sequence of $\boldsymbol{x}$
$\boldsymbol{y}^{(k)}$	sequence $\boldsymbol{y}$ in the k-th iteration
$\boldsymbol{z}$	distance between $\boldsymbol{x}$ and labeled/repaired $\boldsymbol{y}$
t_i	timestamp of i-th data point in $\boldsymbol{x}$
s_g	speed constraint derived by 3σ rule.
D	dimension of time series $\boldsymbol{x}$
$\boldsymbol{\Phi}$	parameter of VAR(p)/VARX(p) with order p
τ	predefined threshold of convergence
$\boldsymbol{Z}$, $\boldsymbol{V}$	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 techniques that can be used for repairing multivariate data, i.e., VAR with only observations and VAR 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, $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The i-th observation (data point) $\boldsymbol{x}_i \in \mathbb{R}^D$ consists of D dimensions $\{x_{i,1}, \ldots, x_{i,D}\}$ and is observed at time t_i. Let $\boldsymbol{y}$ denote the labeled or repaired sequence of $\boldsymbol{x}$, each $\boldsymbol{y}_i$ is either the labeled truth or the repaired value of data point $\boldsymbol{x}_i$.*

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

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

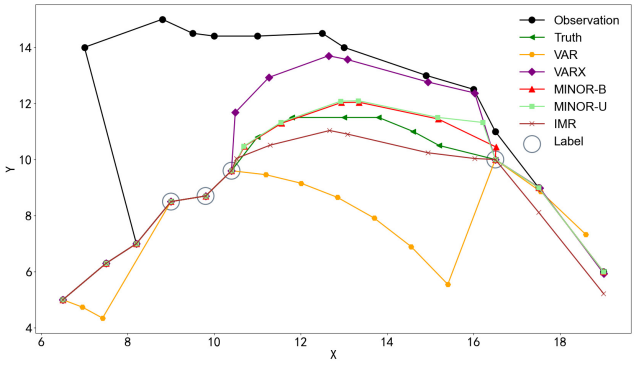

Fig. 2. Example of observations and repairs

Definition 3 (Speed Constraint)**.** *A* speed constraint s *with window size* w *is the maximum speed* $s_{\max}$ *over the time series* $\boldsymbol{x}$*. We say that a time series* $\boldsymbol{x}$ satisfies *the speed constraint* s*, denoted by* $\boldsymbol{x} \vDash s$*, if for any* $\boldsymbol{x}_i, \boldsymbol{x}_j$ *in a window, i.e.,* $0 < t_j - t_i \leq w$*, it has* $\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)}{t_j - t_i} \leq s$*, where window* w *denotes a period of time.*

Problem 1. *Given a time series* $\boldsymbol{x}$ *and a partially labeled subset* $\boldsymbol{y}$ *of* $\boldsymbol{x}$*, the repairing problem is to find the repairs* $\boldsymbol{y}_i$ *of* $\boldsymbol{x}_i$ *that are not labeled in* $\boldsymbol{y}$*.*

Example 2 (Observation $\boldsymbol{x}$, partially labeled $\boldsymbol{y}$, and fully repaired $\boldsymbol{y}$)**.** *Consider a multivariate time series* ($D = 2$)*,* $\boldsymbol{x} = \{(6.5, 5), (7.5, 6.3), (8.2, 7), (7, 14), (8.8, 15), (9.5, 14.5), (10, 14.4), (11, 14.4), (12.5, 14.5), (13, 14), (14.9, 13), (16, 12.5), (16.5, 11), (17.5, 9), (19, 6)\}$ *with 15 data points of observations as illustrated in Figure 2, where consecutive errors occur on* $\boldsymbol{x}_4$ *to* $\boldsymbol{x}_{13}$*. Suppose that four points* $\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_{13}$ *are labeled with truth which is marked by a gray circle in Figure 2. In the repaired* $\boldsymbol{y}$*,* $\boldsymbol{x}_7, \ldots, \boldsymbol{x}_{12}$ *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* $\boldsymbol{y}_4, \ldots, \boldsymbol{y}_6$ *and* $\boldsymbol{y}_{13}$ *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:

$$\boldsymbol{x}'_t = \sum_{i=1}^{p} \boldsymbol{x}_{t-i} \boldsymbol{\Phi}_i + \boldsymbol{\epsilon}_t \tag{1}$$

where $\boldsymbol{x}'_t$ is the prediction of $\boldsymbol{x}_t$, p is the order, $\boldsymbol{\Phi}_i$ is the parameter of the model. For the multivariate time series whose dimension is D, $\boldsymbol{x}'_t$ and $\boldsymbol{x}_t$ are both row vectors of $1 \times D$, while $\boldsymbol{\Phi}_i$ is a $D \times D$ matrix. $\boldsymbol{\epsilon}_t$ is white noise (usually Gaussian white noise).

If $\boldsymbol{x}'_t$ significantly differs from the observation $\boldsymbol{x}_t$, having $d(\boldsymbol{x}'_t, \boldsymbol{x}_t) > \tau$ where τ is a predefined threshold, this predication is accepted $\boldsymbol{x}'_t = \boldsymbol{x}_t$, 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 $\boldsymbol{x}'_t$ and $\boldsymbol{x}_t$.

The VAR-based repairing procedure is thus: (1) replace $\boldsymbol{x}_t$ by $\boldsymbol{y}_t$ if it is labeled, (2) learn parameter $\boldsymbol{\Phi}_i$ of $\mathsf{VAR}(p)$ from $\boldsymbol{x}$, and (3) fill all unlabeled $\boldsymbol{y}_t$ by $\mathsf{VAR}(p)$ over $\boldsymbol{x}$, having

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

Example 3 (Repair by VAR, Example 2 continued)**.** *Consider again* $\boldsymbol{x} = \{(6.5, 5), (7.5, 6.3), \ldots, (19, 6)\}$ *in Figure 2. For simplicity, we use* $\mathsf{VAR}(1)$ *with order* $p = 1$*, i.e.,* $\boldsymbol{x}'_t = \boldsymbol{x}_{t-1} \boldsymbol{\Phi}_1$*. Let* $\tau = 0.5$*. By ordinary least square, we have* $\boldsymbol{\Phi}_1 = \begin{pmatrix} 1.039 & -0.176 \\ 0.004 & 1.175 \end{pmatrix}$*,* $\boldsymbol{x}'_2 = \boldsymbol{x}_1 \boldsymbol{\Phi}_1 = (6.9, 4.7)$*. Since* $d(\boldsymbol{x}'_2, \boldsymbol{x}_2) = 1.7 > \tau$*,* $\boldsymbol{x}'_2$ *is accepted as new* $\boldsymbol{x}_2$*. Labeled points such as* $\boldsymbol{x}_4$ *will not change in each iteration. Similarly,* $\boldsymbol{x}'_7 = \boldsymbol{x}_6 \boldsymbol{\Phi}_1 = (11.2, 9.5)$*. Finally, we obtain the final repaired result* $\boldsymbol{y} = \{(6.5, 5), (6.9, 4.7), \ldots, (18.6, 7.3)\}$ *with RMS error 2.33 (see section V for definition).*

C. VARX Model

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

$$\boldsymbol{y}'_t = \sum_{i=1}^{p} (\boldsymbol{y}_{t-i} - \boldsymbol{x}_{t-i}) \boldsymbol{\Phi}_i + \boldsymbol{\epsilon}_t \tag{3}$$

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

The VARX-based repairing procedure is thus: (1) learn parameter $\boldsymbol{\Phi}_i$ of $\mathsf{VARX}(p)$ from $\boldsymbol{x}$ and partially labeled $\boldsymbol{y}$, and (2) fill all unlabeled $\boldsymbol{y}_t$ by $\mathsf{VARX}(p)$, similar to Equation (2) by replacing $\boldsymbol{x}'_t$ with $\boldsymbol{y}'_t$.

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* $\boldsymbol{x} = \{(6.5, 5), (7.5, 6.3), \ldots, (19, 6)\}$ *in Figure 2. For simplicity, we use* $\mathsf{VARX}(1)$ *with order* $p = 1$*, i.e.,* $\boldsymbol{y}'_t = \boldsymbol{x}_t + (\boldsymbol{y}_{t-1} - \boldsymbol{x}_{t-1}) \boldsymbol{\Phi}_1$*. Let* $\tau = 0.5$*. Similar to VAR in Example 3, we estimate the parameter* $\boldsymbol{\Phi}$ *by ordinary least square, having* $\boldsymbol{\Phi}_1 = \begin{pmatrix} 0.408 & -2.97 \\ -0.024 & 0.009 \end{pmatrix}$*. Again, the labeled* $\boldsymbol{y}_4, \ldots, \boldsymbol{y}_6$ *is not modified. For the 7th point, we have* $\boldsymbol{y}'_7 = \boldsymbol{x}_7 + (\boldsymbol{y}_6 - \boldsymbol{x}_6) \boldsymbol{\Phi}_1 = (10.5, 11.7)$*. Since* $d(\boldsymbol{y}'_7, \boldsymbol{y}_7) = 2.7 > \tau$*, we accept* $\boldsymbol{y}'_7$ *as new* $\boldsymbol{y}_7$*. Finally, the repair result by VARX is* $\boldsymbol{y} = \{(6.5, 5), (7.5, 6.3), \ldots, (19, 5.9)\}$ *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 $\boldsymbol{y}^{(k)}$ denote the sequence $\boldsymbol{y}$ in the k-th iteration, where $\boldsymbol{y}^{(0)}$ is the partially labeled time series in the input. Since $\boldsymbol{y}^{(0)}$ is incomplete (generally only around 1% as shown in Section V), we assign $\boldsymbol{y}_t^{(0)} = \boldsymbol{x}_t$ if $\boldsymbol{y}_t^{(0)}$ is not labeled for initialization. Recall that the labeled values should not be repaired, i.e., $\boldsymbol{y}_t^{(k)} = \boldsymbol{y}_t^{(0)}$ if $\boldsymbol{y}_t^{(0)}$ is labeled.

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

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

(S3) Repair validation, in Line 4, determines one of the repairs to accept, $\boldsymbol{y}_t^{(k+1)} = \hat{\boldsymbol{y}}_t^{(k)}$, 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

$$d(\boldsymbol{y}_j^{(k)}, \boldsymbol{y}_j^{(k+1)}) \leq \tau, j = 1, \ldots, n. \tag{4}$$

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

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

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

In each iteration: MINOR(1) *S1. learns the parameter, e.g.,* $\boldsymbol{\Phi}_1^{(0)} = \begin{pmatrix} 0.408 & -2.977 \\ -0.024 & 0.009 \end{pmatrix}$ *for* $p = 1$*; S2. generates candidates for repairing, such as* $\hat{\boldsymbol{y}}^{(0)} = \{-, -, -, +, +, +, (10.5, 11.7), -, -, -, -, -, +, (17.5, 9), -\}$*, where '+' is labeled point, and '–' denotes no candidates; and S3. validates the candidate, such as* $\hat{\boldsymbol{y}}_7^{(0)}$*,* $\hat{\boldsymbol{y}}_{14}^{(0)}$ *do not match the speed constraint. By computing the validity of each of them,* $\hat{\boldsymbol{y}}_{14}^{(0)}$ *is selected as the repair result in the first iteration.*

The final output of MINOR – B *is* $\boldsymbol{y}^{(51)} = \{(6.5, 5), (7.5, 6.3), \ldots, (19, 6)\}$ *with RMS error* 0.47 *after 51 iterations.* MINOR – U *outputs* $\boldsymbol{y}^{(44)} = \{(6.5, 5), (7.5, 6.3), \ldots, (19, 6)\}$*, 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 $\boldsymbol{\Phi}^{(k)}$ for MINOR(p), given $\boldsymbol{x}, \boldsymbol{y}^{(k)}$. For simplicity, for a data point $\boldsymbol{x}_j$, we define the following $1 \times D$ vector

$$\begin{aligned} \boldsymbol{z}_j^{(k)} &= \boldsymbol{y}_j^{(k)} - \boldsymbol{x}_j, j = 1, \ldots, n. \\ \{z_{j,1}^{(k)}, \ldots, z_{j,D}^{(k)}\} &= \{y_{j,1}^{(k)} - x_{j,1}, \ldots, y_{j,D}^{(k)} - x_{j,D}\} \end{aligned} \tag{5}$$

Existing methods such as Ordinary Least Square [23] or Yule-Walker Equations [24] can be directly employed. For instance, by Ordinary Least Square, we have

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

where

$$\boldsymbol{V}^{(k)} = \begin{pmatrix} \boldsymbol{z}_{p+1}^{(k)} \\ \boldsymbol{z}_{p+2}^{(k)} \\ \vdots \\ \boldsymbol{z}_n^{(k)} \end{pmatrix}_{n-p \times D}, \boldsymbol{\Phi}^{(k)} = \begin{pmatrix} \boldsymbol{\Phi}_1^{(k)} \\ \boldsymbol{\Phi}_2^{(k)} \\ \vdots \\ \boldsymbol{\Phi}_p^{(k)} \end{pmatrix}_{p*D \times D},$$

$$\boldsymbol{Z}^{(k)} = \begin{pmatrix} \boldsymbol{z}_p^{(k)} & \boldsymbol{z}_{p-1}^{(k)} & \ldots & \boldsymbol{z}_1^{(k)} \\ \boldsymbol{z}_{p+1}^{(k)} & \boldsymbol{z}_p^{(k)} & \ldots & \boldsymbol{z}_2^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{z}_{n-1}^{(k)} & \boldsymbol{z}_{n-2}^{(k)} & \ldots & \boldsymbol{z}_{n-p}^{(k)} \end{pmatrix}_{n-p \times p*D}.$$

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

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

$$\boldsymbol{V}^{(0)} = \begin{pmatrix} 0 & 0 & 2 & 1 & 0.9 & \ldots & 0 \\ 0 & 0 & -5.5 & -6.3 & -4.9 & \ldots & 0 \end{pmatrix}'$$

$$\boldsymbol{Z}^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 2 & 1 & \ldots & 0 \\ 0 & 0 & 0 & -5.5 & -6.3 & \ldots & 0 \end{pmatrix}'$$

Referring to Equation 6, the parameter is estimated as $\mathbf{\Phi}^{(0)} = \begin{pmatrix} 0.408 & -2.977 \\ -0.024 & -0.009 \end{pmatrix}$

C. Candidate Generation

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

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

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

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

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

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

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

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

Similarly, for bidirectional repairing, we have $\mathbf{\Phi}^{(0)^r} = \begin{pmatrix} -1.302 & 4.504 \\ -0.481 & 1.719 \end{pmatrix}$, *thus* $\hat{\boldsymbol{y}}_{12}^{(0)^r} = \boldsymbol{z}_{13}^{(k)} \mathbf{\Phi}^{(\mathbf{0})^r} + \boldsymbol{x}_{12} = (16.5, 10.8)$ *referring to Equation 8, and* $d(\hat{\boldsymbol{y}}_{12}^{(0)^r}, \boldsymbol{y}_{12}^{(0)}) = 1.7 > 0.5$. *So on and so forth, we have the repair candidates are* $\hat{\boldsymbol{y}}^{(0)^r} = \{-, -, -, +, +, +, -, -, -, -, -, (16.5, 10.8), +, -, -\}$. *It shows that the repair candidates from different directions 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 $\boldsymbol{y}_t^{(k+1)} = \hat{\boldsymbol{y}}_t^{(k)}$ 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 the 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, each iteration repairs only one data point, outperforming the NP-hard problem of minimizing total changes under integrity constraints [22] through problem decomposition. This achieves lower overall costs despite evaluating multiple candidates per iteration.

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

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

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

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

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

Equation (9) shows how to compute the local speed constraints of point t, where $\boldsymbol{y}_{t_{f1}}^{(0)}$ and $\boldsymbol{y}_{t_{f2}}^{(0)}$ stand for the first and second former data point (i.e., $t_{f2} < t_{f1} < t$) with labels. Similarly, $\boldsymbol{y}_{t_{l1}}^{(0)}$ and $\boldsymbol{y}_{t_{l2}}^{(0)}$ 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 $\boldsymbol{c}_t$, 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

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

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

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

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 $\boldsymbol{y}^{(k+1)}$ is returned.

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

Input: candidate $\hat{\boldsymbol{y}}_t$, observation $\boldsymbol{x}_t$, partially labeled $\boldsymbol{y}^{(0)}$, constraint s_g, validity dict $valid$
Output: whether $\hat{\boldsymbol{y}}_t$ 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 $valid(\hat{\boldsymbol{y}}_t)$ 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. As the sequence of speed changes remains stationary for a period of time [15], shorter intervals between the labels and candidates enhance the confidence of s_t. Consequently, to improve repair accuracy, in theory, boundary points of segments where errors occur should be labeled for error localization, and labels should be evenly distributed within segments to minimize unlabeled points' maximum distance to their nearest label. However, the above cases are not always applicable. In practice, 1% to 3% labels on dirty points with random positions within segments are sufficient to get good repair performance.

Let m be the number of labeled/repaired points, p the order, and n the length of the time series. Initially, m approximates the label count, which is around 1% of n. It gradually increases during iterations but remains significantly smaller than n. In Algorithm 2, sorting in Line 1 takes $O(mpD + mp\log(mp))$ while the remaining steps take $O(mpD)$, so the overall time complexity is $O(mpD + mp\log(mp))$.

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:

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

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

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

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

W.l.o.g, we explain how the Equation (13) comes about. The former validity of a given candidate $valid^f(\hat{\boldsymbol{y}}_t)$ 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

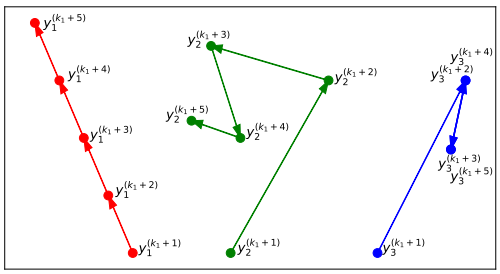

Fig. 3. Example of monotonicity

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 $valid(\hat{\boldsymbol{y}}_t)$ can be computed by Equation (16), as a weighted average of the former and latter validity , where validity derived from a closer label has greater weight due to its higher confidence.

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 $(\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t^{(k)}) \cdot (\boldsymbol{y}_t^{(k)} - \boldsymbol{y}_t^{(h)}) \geq 0$ (where $\cdot$ denotes the Euclidean dot product) , then the candidate $\hat{\boldsymbol{y}}_t$ meets monotonicity.*

Figure 3 illustrates the repair trajectories of three data points. For simplicity, we assume that each point has four 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 $\boldsymbol{y}_t^{(k+2)} \approx \boldsymbol{y}_t^{(k)}$ applies, i.e. it loops every two iterations, making the method difficult to terminate. Therefore, we introduce the Algorithm 4 with a time complexity of $O(mpD)$, where m denotes the number of labeled or repaired points, and p denotes the order, 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$(\boldsymbol{y}^{(k)}, Y^{(k)})$

Input: repaired sequence $\boldsymbol{y}^{(k)}$, candidates $Y^{(k)}$
Output: $Y^{(k)}$ after invalid candidates removed

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

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

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

1 $Y^{(k)} \leftarrow$ Check$(\boldsymbol{y}^{(k)}, \hat{\boldsymbol{y}}^{(k)} \cup \hat{\boldsymbol{y}}^{(k)^r})$;
2 $Y^{(k)} \leftarrow$ Sort $Y^{(k)}$ in ascending order by $d(\hat{\boldsymbol{y}}_t^{(k)}, \boldsymbol{x}_t)$;
3 $valid \leftarrow$ a new dict with *validity* of $Y^{(k)}$;
4 **foreach** $\boldsymbol{c}_t \in Y^{(k)}$ **do**
5 **if** $\boldsymbol{c}_t \in \hat{\boldsymbol{y}}^{(k)}$ ***and*** $\boldsymbol{y}_{t+1}^{(k)}$ *is not repaired/labeled* **then**
6 **if** Assign$(\boldsymbol{c}_t, \boldsymbol{x}_t, \boldsymbol{y}^{(0)}, s_g, valid)$ **then**
7 $\boldsymbol{y}_t^{(k)} \leftarrow \boldsymbol{c}_t$;
8 **return** $\boldsymbol{y}^{(k)}$
9 **continue**;
10 **if** $\boldsymbol{c}_t \in \hat{\boldsymbol{y}}^{(k)^r}$ ***and*** $\boldsymbol{y}_{t-1}^{(k)}$ *is not repaired/labeled* **then**
11 **if** Assign$(\boldsymbol{c}_t, \boldsymbol{x}_t^r, \boldsymbol{y}^{(0)^r}, s_g, valid)$ **then**
12 $\boldsymbol{y}_t^{(k)} \leftarrow \boldsymbol{c}_t$;
13 **return** $\boldsymbol{y}^{(k)}$
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 $\boldsymbol{y}^{(k)} \leftarrow \boldsymbol{c}_t$;
18 **return** $\boldsymbol{y}^{(k)}$
19 Compute $valid(\boldsymbol{c}_t)$ by Equations 13-16;
20 $\boldsymbol{c}_t \leftarrow \arg\max_{\hat{\boldsymbol{y}}_t} valid(\hat{\boldsymbol{y}}_t)$;
21 **if** $valid(\boldsymbol{c}_t) > 0$ **then**
22 $\boldsymbol{y}_t^{(k)} \leftarrow \boldsymbol{c}_t$;
23 **return** $\boldsymbol{y}^{(k)}$

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 $\boldsymbol{c}_t$ can be accepted directly or whether a validity is assigned. As shown 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. Algorithm 5 has the same time complexity as Algorithm 2.

Example 8 (Validate repair $\hat{\boldsymbol{y}}_t^{(6)}$, Example 7 continued)**.** *Consider the candidates in the 6-th iteration of* MINOR − B(1), $\hat{\boldsymbol{y}}^{(6)} = \{-,-,-,+,+,+,(10.5,11.7),-,(12.5,14.4),(13.0,13.8),(15.0,12.7),(16.1,11.9),+,(17.5,9.0),-\}$, $\hat{\boldsymbol{y}}^{(6)^r} = \{-,-,(8.2,6.8),+,+,+,(10.0,14.3),-,-,-,(15.1,12.2),(16.5,10.7),+,-\}$, *first, we check the monotonicity of candidates by Algorithm 4, remaining* $\hat{\boldsymbol{y}}_{14}^{(6)}$, $\hat{\boldsymbol{y}}_7^{(6)^r}$, $\hat{\boldsymbol{y}}_3^{(6)^r}$, $\hat{\boldsymbol{y}}_{11}^{(6)^r}$, $\hat{\boldsymbol{y}}_7^{(6)}$, *and sorted in the ascending order by* $d(\hat{\boldsymbol{y}}_t^{(k)}, \boldsymbol{x}_t)$. *Let's take* $\hat{\boldsymbol{y}}_7^{(6)}$ *for example, since* $\boldsymbol{y}_6^{(6)}$ *is labeled and* $\boldsymbol{y}_8^{(6)}$ *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* $v_{7,o} = 4.82, v_{7,o}^l = 1.31, valid^f(\hat{\boldsymbol{y}}_7^{(6)}) = 0.74, valid^l(\hat{\boldsymbol{y}}_7^{(6)}) = 1, \Delta_t^f = 1, \Delta_t^l = 6, valid(\hat{\boldsymbol{y}}_7^{(6)}) = 0.74$. *Finally, the validity of each candidate is* 0, 0.03, 0, 1.0, 0.74, *and* $\hat{\boldsymbol{y}}_{11}^{(6)^r}$ *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 $\boldsymbol{Z}^{(k)}$, $\boldsymbol{V}^{(k)}$ are involved to estimate the parameter $\boldsymbol{\Phi}^{(k)}$ in the k-th iteration, whose element $\boldsymbol{z}_j^{(k)}$ 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 $\boldsymbol{Z}^{(k)}$, $\boldsymbol{V}^{(k)}$ are equal to 0. The following conclusion still holds that the same parameter $\boldsymbol{\Phi}^{(k)}$ can still be computed by Equation (6) after removing the rows in $\boldsymbol{Z}^{(k)}$ whose values are equal to 0 and the corresponding rows in $\boldsymbol{V}^{(k)}$.

2) Incremental Computation: In each iteration in Algorithm 1, the parameter $\boldsymbol{\Phi}^{(k)}$ is estimated by Equation (6) w.r.t. $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{V}^{(k)}$ over all $(nD + npD)$ points. We show that, similar to [8], $\boldsymbol{\Phi}^{(k)}$ can be computed incrementally by considering only the changed values instead of the entire $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{V}^{(k)}$. Specifically, all the $(p^2D^2 + pD)$ 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 in the following matrix multiplication:

$$\widehat{\boldsymbol{C}}^{(k)} = \boldsymbol{X} + \boldsymbol{Z}^{(k)}\boldsymbol{\Phi}^{(k)} \tag{17}$$

where

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

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

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

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

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 $\boldsymbol{z}_t^{(k)}$ in $\boldsymbol{Z}^{(k)}$ 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,* $\hat{\boldsymbol{y}}_{11}^{(6)^r} = (15.1, 12.2)$ *is accepted, so in the 7-th iteration,* $p = 1, r = 11, \boldsymbol{z}_{11,1}^{(7)} = \boldsymbol{y}_{11}^{(7)} - \boldsymbol{x}_{11} = (0.2, -0.8)$, *and other* $\boldsymbol{z}^{(7)}$ *will remain unchanged, namely* $\boldsymbol{z}_{i,j}^{(7)} = \boldsymbol{z}_{i,j}^{(6)}, 1 + i - j \neq r$.

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 $\boldsymbol{y}^{(0)}$ and the observations $\boldsymbol{x}$ 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.

Finally, Algorithm 1 comprises three steps with complexities of $O(1)$, $O(mp^2D^2)$, and $O(mpD + mp\log(mp))$, so

TABLE II
SUMMARY OF DATASETS

	Size	#Dim	Error	#Series
ILD [25]	48k	2	Clean after pre-process	1
ECG [26]	12k	32	Clean after pre-process	1
GPS	1.7k	2	Embedded	1
Car [27]	577	1	Clean	120
Lightning2 [27]	637	1	Clean	121
DPTW [27]	80	1	Clean	600
EOGVS [27]	1250	1	Clean	362

its overall complexity is $O(k(mp^2D^2 + mp\log(mp)))$ with k iterations.

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 online[1].

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) [25] 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 [26] 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 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 [28] is employed to evaluate the repair.
- Time cost. The time required to complete the cleaning process, excluding error injection, data loading and RMSE calculation.

[1]https://github.com/zaqthss/minor

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 [29]	multivariate	batch	smooth
MTCSC [9]	multivariate	batch	constraint
TranAD [17]	multivariate	online	deep learning
USAD [18]	multivariate	online	deep learning

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
-w/o SC	0.6589	10917.20	100000
-w/o iter.	0.7111	2.11	-
-Uni	0.4478	233.58	11390
MINOR – U	0.4170	35.21	2081
-w/o CIC	0.4170	299.77	2081
-w/o IC	0.4170	534.11	2081
-w/o SC	0.8303	618.64	7873
-w/o iter.	0.7111	1.50	-
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	-	-

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.

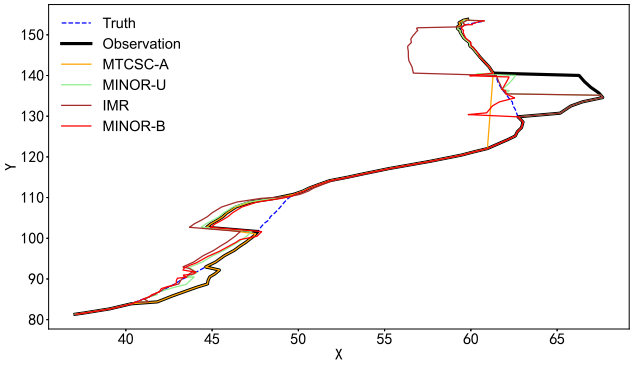

Fig. 4. Case Study of GPS

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, 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, -w/o IC denotes the method without incremental computations, -w/o SC denotes the method without speed constraints, i.e., without Algorithm 2 and 5, -w/o iter. denotes the method without iterative repairing, i.e., similar to VARX, the entire time series is repaired in a single pass. The comparison shows that our proposed incremental computation strategy reduces the time cost to 5% of the original, which proves its necessity, and demonstrates that our method significantly outperforms isolated iterative repairing and standalone speed constraints-based methods. 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, we introduce MINOR − U, which achieves decent results (the best when $p > 1$) with much less time cost.

3) Varying Convergence Threshold τ: Figure 5(c) shows that our MINOR − B and MINOR − U perform quite stably 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 5(d) that with increasing threshold values, the time costs of MINOR − B and MINOR − U gradually decrease.

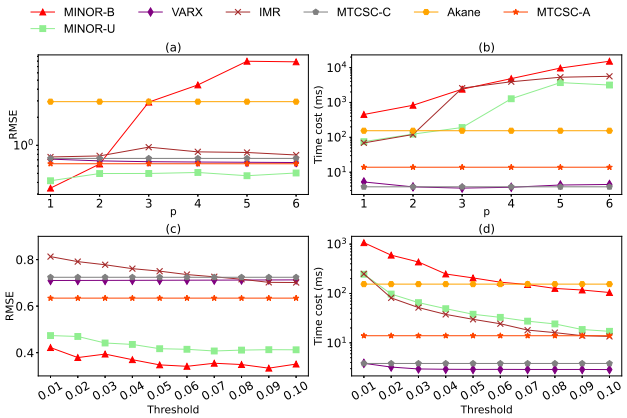

Fig. 5. Varying order p (a-b) and threshold (c-d) on GPS

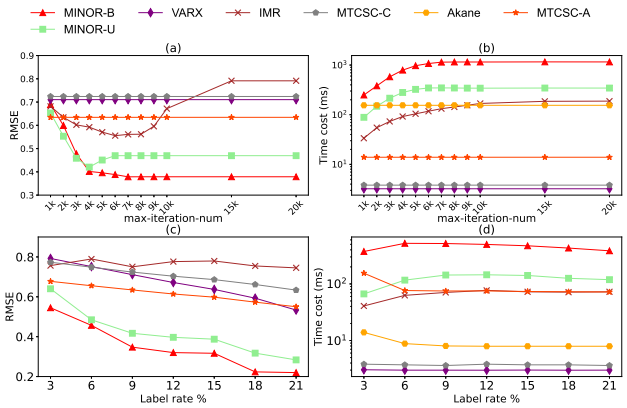

Fig. 6. Varying maximum iteration number (a-b) and label rate (c-d) on GPS

4) Specifying Maximum Number of Iterations: Figure 6(b) shows that all iterative methods converge over 10k iterations. However, Figure 6(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 6(c) 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 6(d) shows that labeling rate has little influence over running time.

6) The effect of the interval between labeled and repaired points: Figure 7(a) shows how the repair performance on dirty data points varies with the interval between them and their nearest label. Specifically, all dirty data points with an interval

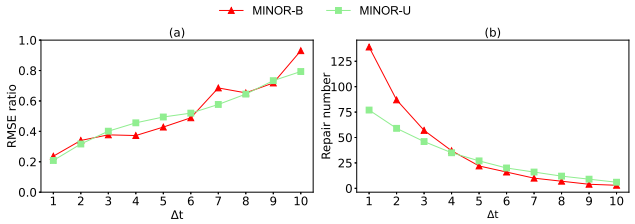

Fig. 7. The effect of the interval between labeled and repaired points on GPS

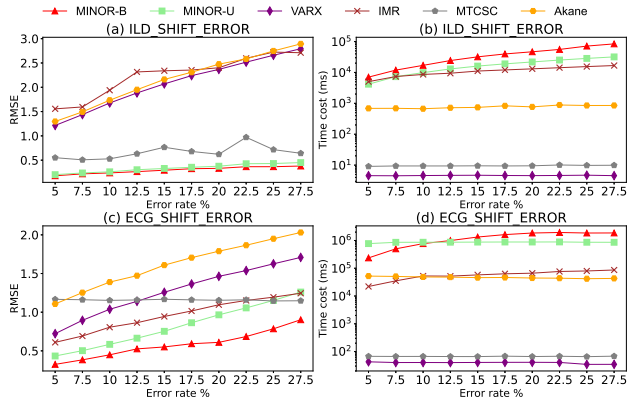

Fig. 8. Varying error rate on ILD(a-b) and ECG(c-d) with SHIFT error

Δt from their nearest label are selected, where MINOR – U only considers the nearest previous label while MINOR – B considers the nearest label, whether preceding or following the dirty data point. Then the ratio of the RMSE of these data points after repair to that before repair is calculated and denoted as "RMSE ratio". It increases with Δt, indicating deteriorating repair performance, i.e., smaller Δt leads to better repair. Figure 7(b) shows that compared to MINOR – U, there are more dirty points repaired with smaller Δt in MINOR – B, explaining better results in previous experiments.

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.

1) Varying Error Rate e%: Figures 8(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 8(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 8(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 9(a)(c), our proposals show improved repair accuracy with increasing error length, with MINOR – B performing best and MINOR – U second

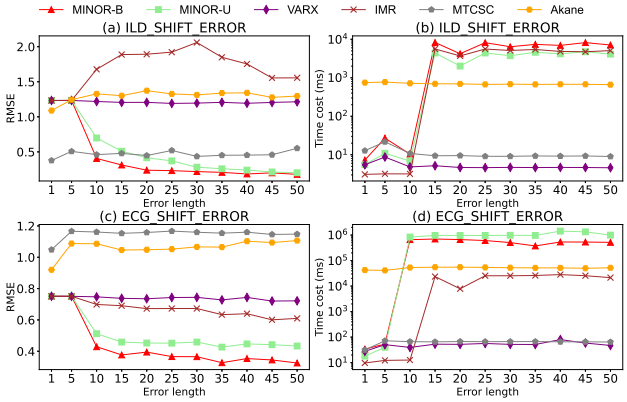

Fig. 9. Varying error length on ILD(a-b) and ECG(c-d) with SHIFT error

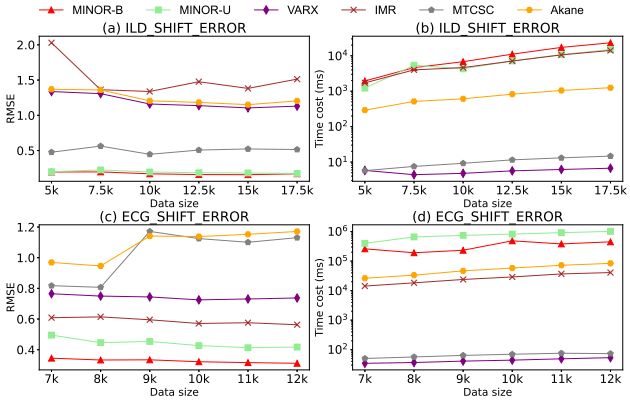

Fig. 10. Varying data size on ILD(a-b) and ECG(c-d) with SHIFT error

best, except in cases with small error length (e.g., an error length of 1, which represents spike errors) 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 also be shown in Figures 9(b)(d) that we cost little time as we converge very quickly.

3) Varying Data Size n: Figures 10(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 10(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 [30]. As shown in Figure 8(a) and Figure 11(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 is also the same, shown in Figure 11(b).

5) Evaluation on Online Computing: Time series are generated continuously in some cases. Although MINOR – B performs better than MINOR – U, the former cannot be used

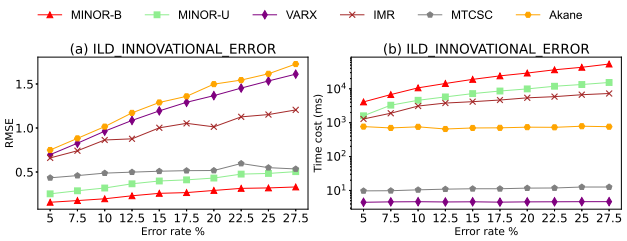

Fig. 11. Varying error rate on ILD with INNOVATIONAL error

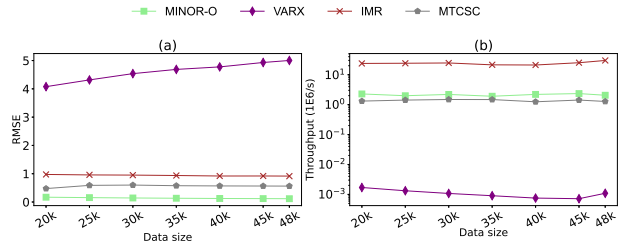

Fig. 12. Online computing on ILD

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 12 shows the repair results on ILD. We inject errors using the same way as in Experiment V-C3, 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 12(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 higher efficiency, which shows its capability in online situations.

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 [31] classifier, select the best K via a grid search and take the F1-score [2] as metric. For clustering, we use K – means [32] and take the RI [33] as a metric. As expected, Figure 13 shows that our MINOR – B performs better than others and is closer to the

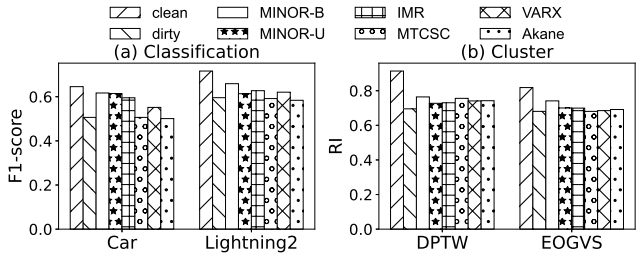

Fig. 13. Classification and Clustering over clean, dirty and repaired data

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) With the definition of monotonicity enabling the bidirectional repairing method, 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 relatively larger 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) With local speed constraints from labels, 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 [34] 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 [35] is widely used to smooth time series data and make predictions. The exponentially weighted moving average (EWMA) [36] 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. [37] contains an incremental clustering method where historical data is first clustered and the average values of the clusters are used for repairs. STPM [38] 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 [39] 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. Netw. 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," *Proc. ACM Manag. Data*, vol. 2, no. 6, pp. 245:1–245:26, 2024.

[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] Y. Zheng, F. Liu, and H. Hsieh, "U-air: when urban air quality inference meets big data," in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. ACM, 2013, pp. 1436–1444.

[21] L. A. Hageman, *Applied iterative methods*. Elsevier, 2014.

[22] 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.

[23] C. R. Rao, *Linear statistical inference and its applications*. John Wiley & Sons, 2009, vol. 22.

[24] B. Cheng, "Yule–walker equations," *Wiley StatsRef: Statistics Reference Online*, 2014. [Online]. Available: http://dx.doi.org/10.1002/9781118445112.stat05549

[25] S. Madden, "Intel berkeley research lab data," https://db.csail.mit.edu/labdata/labdata.html, 2003, accessed: 2024-04-10.

[26] 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.

[27] 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/.

[28] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*. ACM, 2006, pp. 163–174.

[29] 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.

[30] R. S. Tsay, "Outliers, level shifts, and variance changes in time series," *Journal of forecasting*, vol. 7, no. 1, pp. 1–20, 1988.

[31] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[32] 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.

[33] 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.

[34] 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.

[35] D. R. Brillinger, *Time series - data analysis and theory*, ser. Classics in applied mathematics. SIAM, 2001, vol. 36.

[36] 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

[37] 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.

[38] M. Milani, Z. Zheng, and F. Chiang, "Currentclean: Spatio-temporal cleaning of stale data," in *ICDE*. IEEE, 2019, pp. 172–183.

[39] 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.

[40] J. Li, Z.-L. Li, H. Wu, and N. You, "Trend, seasonality, and abrupt change detection method for land surface temperature time-series analysis: Evaluation and improvement," *Remote Sensing of Environment*, vol. 280, p. 113222, 2022.

MINOR: Multivariate Time Series Iterative Cleaning Algorithm (Paper ID: 1027)

To Meta Reviewer

Thank you very much for your comments. We have addressed the reviewers' concerns and revision requests in the revised manuscript. The added contents according to comments are highlighted in red color. Please find our response as follows.

C1: *The authors claim that repairing a single data point in each iteration is more efficient. However, this approach might increase the number of repair candidates considered in each iteration, potentially affecting overall efficiency. A more detailed explanation of this claim is necessary to clarify its rationale.*

Reply: Our claim stems from the differing computational complexities between global and localized repair strategies. The localized iterative approach uses a simpler, more efficient stepwise decision-making process, enabling faster repairing than methods requiring full dataset processing. See R1.O1 for more detailed replies. □

C2: *The interval between labeled points and those being repaired appears to influence the repair validation process, particularly in computing speed constraints and validity scores. The authors should discuss the impact of this interval in greater detail with experiments.*

Reply: As suggested, we add an experiment in Section V-B6. For both MINOR − U and MINOR − B, smaller interval between the labeled points and those being repaired leads to more accurate validity assessment, thus yielding better repair results. See R1.O2 for more detailed replies. □

C3: *In the definition of monotonicity, the formulation* $(\hat{y}_t - y_t^{(k)})(y_t^{(k)} - y_t^{(h)}) \geq 0$ *is ambiguous because* y_t *here represents a point in multi-variable time series.*

Reply: We check the formulation carefully and add the clarification at the corresponding equation in Definition 4 of Section III. See R1.O3 for more detailed replies. □

C4: *The time complexity of the MINOR algorithm is expected.*

Reply: As suggested, we add the time complexity in Section III and Section IV. See R1.O4 for more detailed replies. □

C5: *The solution is tested on shift and innovational errors between all error types. It is unclear how well MINOR would perform on abrupt errors, spikes, or other error types. Experiments on these are expected.*

Reply: As suggested, we include the case of an error length of 1 to show MINOR's performance on spike errors in Section V-C2. See R2.O1 for more detailed replies. □

C6: *The paper appears to combine existing methods like speed constraints and iterative learning, all from the existing methods, with limited novelty in how these are applied to the problem of time series repair. It needs a careful and fine-grained ablation study.*

Reply: Yes, MINOR combines speed constraints and iterative cleaning. Its technical novelty lies in three key innovations addressing prior work limitations:

(1) Monotonicity definitions and Algorithm 4 prevent deadlock in bidirectional repairing, enabling its application. (2) Our method derives local speed constraints from labels without preset parameters, outperforming MTCSC in robustness. (3) Our method only labels data points in segments where errors occur rather than points across the whole datasets, thereby reducing labeling rate from 10%-20% to 1%-3%.

We add an ablation study to evaluate methods without speed constraints or iterative repairing. The results are in Table IV, with detailed analysis in Section V-B1.

See R2.O2 for more detailed replies. □

C7: *The effectiveness of MINOR heavily relies on parameters such as the speed constraint and convergence threshold. These parameters are often dataset-specific and require fine-tuning. The authors should discuss how to come up with such parameters for various datasets and the reason behind.*

Reply: (1) Global speed constraint is computed by 3σ rule, and local speed constraints are derived from labeled points without manual pre-setting parameters. (2) Parameter $p = 1$ is recommended for its optimal repair performance and good efficiency. (3) The convergence threshold τ is set by grid search due to its performance sensitivity and weak correlation with dataset characteristics. (4) The maximum-iteration-num is also grid-searched to balance repair quality and computational cost, especially for high-dimensional data. Future work will investigate systematic methods for determining τ and maximum-iteration-num for broader applicability. See R3.O1 for more detailed replies. □

C8: *The proposed method relies on a subset of labeled points in the time series. However, it is not clear how to come up with such a subset, and what points need to be labeled to maximize the accuracy of the proposed technique.*

Reply: (1) The truth value can be obtained via manual annotation or reliable sensors with a relatively long sensing period. (2) Labeling error segment start/end points with even label distribution can improve repair performance. However, random labeling, used in experiments, still shows promising results. See R3.O2 for more detailed replies. □

C9: *The bidirectional validation requires future information, limiting its applicability in online and real-time scenarios. While unidirectional validation offers a workaround, it compromises accuracy.*

Reply: (1) Bidirectional repair can be adapted for online/real-time scenarios by buffering data until obtaining a label. However, its full online/real-time implementation needs further study. (2) Although unidirectional repair compromises accuracy, the experiment in Section V-C5 shows that MINOR − O's performance is comparable to MINOR − U in RMSE. See R3.O3 for more detailed replies. □

To Reviewer 1

Thank you very much for reading our paper carefully. Below is our response to your comments.

O1: *O1: The authors claim that repairing a single data point in each iteration is more efficient. However, this approach might increase the number of repair candidates considered in each iteration, potentially affecting overall efficiency. A more detailed explanation of this claim is necessary to clarify its rationale.*

Reply: Our claim regarding efficiency stems from fundamental differences in complexity between global and localized repair strategies. Directly addressing the global optimization problem that involves all data points while simultaneously considering integrity constraints requires minimizing the overall changes according to the minimum change principle. This typically requires searching through an exponential solution space, resulting in prohibitively high time costs.

In contrast, repairing a single data point decomposes this global NP-hard problem into multiple localized subproblems with reduced complexity, thereby avoiding the need for the global resolution across the entire dataset. Although each iteration may involve evaluating multiple candidates, each evaluation operation has substantially lower time cost. In short, the localized iterative repairing strategy essentially employs a simpler yet more efficient decision-making mechanism at each step, making its repairing process more efficient than the method requiring processing the entire dataset. And in Section IV, we present that the time complexity of Algorithm 1 is $O(k(mp^2D^2 + mp\log(mp)))$ with k finite iterations, which is theoretically still less than the global NP-hard problem.

We add "Remarkably, each iteration repairs only one data point, outperforming the NP-hard problem of minimizing total changes under integrity constraints [22] through problem decomposition. This achieves lower overall costs despite evaluating multiple candidates per iteration." in Section III-D. □

O2: *O2: The interval between labeled points and those being repaired appears to influence the repair validation process, particularly in computing speed constraints and validity scores. The authors should discuss the impact of this interval in greater detail.*

Reply: As suggested, we add an experiment in Section V-B6. Our findings indicate that the shorter the interval between the labeled point and the point being repaired, the more accurate the assessment of the candidate's validity will be, and hence the repair result will be closer to the truth. Below, we discuss MINOR − U and MINOR − B separately.

(1) For MINOR − U: In Algorithm 3, validity computation at timestamp t begins by calculating local speed constraint s_t, candidate speed v_t, and observation speed $v_{t,o}$ by Equations 9-11. Typically, since the global speed constraint s_g derived by 3σ rule is quite loose, s_t is usually less than s_g. According to Line 2 in Algorithm 3, s_t is then used to calculate the validity of candidates. Because the sequence of speed changes remains stationary for a period of time [15], smaller interval between the labeled points and those being repaired enhances s_t's guidance, leading to more accurate validity assessment, thus yielding better repair results.

We add "As the sequence of speed changes remains stationary for a period of time [15], shorter intervals between the labels and candidates enhance the confidence of s_t." in the description of Algorithm 3 in Section III-D1.

(2) For MINOR − B: As discussed prior to Example 8 in Section III, in the early stage of the iterations, most of the points have not been repaired yet and the generated candidates are usually closer to the labeled points in their own repair directions (forward or backward only). Due to the validity derived from these labeled points having relatively high confidence, as shown in Lines 5 and 10 of Algorithm 5, the validity of candidates is calculated using Algorithm 3, similarly to MINOR − U. Later in the repair process, many candidates fail to meet the conditions in Lines 5 and 10 of Algorithm 5. In such cases, local speed constraints from both sides are considered for better repair performance. Likewise, local speed constraints derived from closer labeled points have greater weights in Equations 16 due to their higher confidence. Furthermore, experiments in Section V indicate that this method generally enables MINOR-B to achieve lower RMSE than other methods. We add "where validity derived from a closer label has greater weight due to its higher confidence." in Section III-D2.

As suggested, we add an experiment in Section V-B6 to investigate the effect the interval between labeled and repaired points on the repair performance on dirty data points. Specifically, all dirty data points with an interval Δt from their nearest label are selected, where MINOR − U only considers the nearest previous label while MINOR − B considers the nearest label, whether preceding or following the dirty data point, then the ratio of the RMSE of these data points after repair to that before repair is calculated and denoted as "RMSE ratio". Figure 7 shows that the ratio increases with Δt, indicating deteriorating repair performance, i.e., smaller Δt leads to better repair. Figure 7(b) shows that compared to MINOR − U, there are more dirty points repaired with smaller Δt in MINOR − B, leading to better repair performance shown in previous experiments.

Unfortunately, in the current version, we have not been able to provide a rigorous proof of this. We plan to investigate this further in our future work. □

O3: *O3: In the definition of monotonicity, the formulation $(\hat{y}_t - y_t^{(k)})(y_t^{(k)} - y_t^{(h)}) \geq 0$ is ambiguous because y_t here represents a point in multi-variable time series.*

Reply: As suggested, we carefully checked the formulation in this section to ensure eliminating the ambiguity, and add "where $\cdot$ denotes the Euclidean dot product." at the corresponding equation in Definition 4 of Section III to ensure the formulation is no longer ambiguous. □

O4: *O4: The time complexity of the MINOR algorithm is not provided. Including this information would enhance the*

readers' understanding of the algorithm's computational requirements.

REPLY: As suggested, we add "Let m be the number of labeled/repaired points, p the order, and n the length of the time series. Initially, m approximates the label count, which is around 1% of n. It gradually increases during iterations but remains significantly smaller than n. In Algorithm 2, sorting in Line 1 takes $O(mpD + mp\log(mp))$ while the remaining steps take $O(mpD)$, so the overall time complexity is $O(mpD + mp\log(mp))$." in Section III-D1, append "with a time complexity of $O(mpD)$, where m denotes the number of labeled or repaired points, and p denotes the order" to the second-to-last paragraph before Example 8 in Section III-D2, add "Algorithm 5 has the same time complexity as Algorithm 2." in the last paragraph before Example 8, and finally append "Finally, Algorithm 1 comprises three steps with complexities of $O(1)$, $O(mp^2D^2)$, and $O(mpD + mp\log(mp))$, so its overall complexity is $O(k(mp^2D^2 + mp\log(mp)))$ with k iterations." at the end of Section IV to explain the overall time complexity of MINOR. □

TO REVIEWER 2

Thank you very much for reading our paper carefully and your comments. Below is our response to your comments.

O1: *O1: The solution is tested on shift and innovational errors between all error types. It is unclear how well MINOR would perform on abrupt errors, spikes, or other error types.*

REPLY: As suggested, we include the case of an error length of 1 to show MINOR's performance on spike errors in Section V-C2. The result is shown in Figure 9, and we add "e.g., an error length of 1, which represents spike errors" in the analysis.

Abrupt errors can be categorized into sudden changes in trend and abrupt variations in seasonality [40]. The former is the same as the shift errors, for which the performance of MINOR has been discussed in this study. The latter presents as spike errors, and MINOR's performance in handling such errors has been validated through the experiment described above. □

O2: *O2: The paper appears to combine existing methods like speed constraints and iterative learning, all from the existing methods, with limited novelty in how these are applied to the problem of time series repair. For example, speed constraints are well-established in prior work, and while the bidirectional validation and iterative framework add some refinements, they lack a strong technical distinction from existing techniques.*

REPLY: Yes, MINOR combines speed constraints with an iterative cleaning algorithm. However, its technical novelty lies in three key innovations that address limitations of prior work:

(1) Monotonicity constraints for bidirectional repairing. In Section III, we introduced the definition of monotonicity, a property ensuring repair progress consistently toward convergence, as Definition 4. As discussed in the first paragraph after Definition 4, without removing non-monotonic points, bidirectional repair may result in a situation similar to the third point (in blue) in Figure 3, where the point returns to a value close to its previous state after several iterations, leading to redundant cycles or even deadlocks. This issue almost inevitably occurred without employing Algorithm 4 in our past experiments. By introducing the definition of monotonicity and Algorithm 4, we successfully avoid deadlocks, and results in Section V also demonstrate that the bidirectional repair method MINOR − B outperforms the unidirectional one MINOR − U.

(2) Local speed constraints derived from labels without setting parameters in advance. Unlike MTCSC [9] requires setting speed constraint in advance, our method derives local speed constraints by labels without any preset parameters. Specifically, Equation 9 in Section III-D1 calculates local speed constraints s_t for every point from adjacent labeled points, eliminating dependency on preset global speed constraints. Besides, Algorithms 2 and 5 evaluates candidates via Equations 9-16, prioritizing repairs that minimize constraint violations. Results in Table IV show MINOR − B and MINOR − U outperforms MTCSC − A under such real-world errors, demonstrating superior robustness.

(3) Fewer labels are required. Existing method IMR [8] often requires labels throughout the entire dataset to avoid over-repair problem, typically involving 10% to 20% labels of the data points. This limits the method's applicability. In contrast, our method only requires labels in segments where errors occur, requiring only 1% to 3% labels. And through the local speed constraints derived from labels, our method can handle consecutive errors while avoiding over-repair problem by such few labels, as verified in Figure 1.

The three aforementioned points have been mentioned in the Conclusion section previously. However, to address the reader's focus, we add "With the definition of monotonicity enabling the bidirectional repairing method" in Section V-E point 4 corresponding to the aforementioned item (1), and add "With local speed constraints from labels" in point 7 corresponding to items (2) and (3).

We also conduct an ablation study by evaluating variants of our method with speed constraints removed or iterative repairing disabled. Results are recorded in Table IV and detailed analysis is provided in Section V-B. The result demonstrates that our method significantly outperforms isolated iterative repairing and standalone speed constraints-based method. □

TO REVIEWER 3

Thank you very much for reading our paper carefully and your comments. Below is our response to your comments.

O1: *O1: The effectiveness of MINOR heavily relies on parameters such as the speed constraint and convergence threshold. These parameters are often dataset-specific and require fine-tuning. The authors should discuss how to come up with such parameters for various datasets.*

REPLY: (1) Speed constraints: As discussed in response to R2.O2.(2), the global speed constraint is calculated by 3σ rule, and the local speed constraints derive from labeled points. Specifically, MINOR calculates local speed constraints s_t for every point from adjacent labeled point by Equations 9 in Section III-D1 without setting any parameters in advance.

(2) Order p: According to Figure 5, determining p is relatively simple. Both $\mathsf{MINOR-U}$ and $\mathsf{MINOR-B}$ show good repair performance when $p = 1$; however, $\mathsf{MINOR-U}$ with greater p has higher time cost, while $\mathsf{MINOR-B}$ is unable to perform proper repairs when $p > 1$ due to method shortcomings. So the simplest model of $p = 1$ is recommended for all cases as we have discussed in Section V-B2.

(3) Threshold τ: This parameter is hard to determine due to its dataset-sensitive nature. As shown in Figure 5, a τ that is too small results in a sharp increase in the repair cost and a decline in repair performance, while a τ that is too large leads to poor repair performance. Unfortunately, we have not found any association between τ and certain characteristics of the dataset. Therefore, we consider it as a hyper-parameter and set it in advance by grid search for each dataset.

(4) Max-iteration-num: It is similar to τ. The parameter aims to balance the repair performance and time cost. A proper parameter can achieve good performance with acceptable time. However, Iteration count for convergence is complexly related to dimensions, data volume and τ, and unfortunately, no direct formula exists. Thus, we treat it as a hyperparameter and use grid search for each dataset.

In the future work, we will investigate how to better come up with parameters like τ and max-iteration-num for various datasets to enhance the practicality of our method. □

O2: *O2: The proposed method relies on a subset of labeled points in the time series. However, it is not clear how to come up with such a subset, and what points need to be labeled to maximize the accuracy of the proposed technique.*

REPLY: (1) As suggested, we add "The truth value can be obtained via manual annotation or reliable sensors with a relatively long sensing period [20]." in the first paragraph in Section I-B. Since very few labeled points are required for MINOR (ranging from 1% to 3%), the cost of obtaining these labels remains within an acceptable range.

(2) To improve repair performance for consecutive errors, in theory, we can do the following two things: firstly, the two boundary points of a segment where errors occur should be labeled(i.e., data points at the start and end timestamps). This allows MINOR to precisely localize the error scope, focusing repairs within the segment and preserving clean data, as shown in Figure 2 of Section II-A. Secondly, within the error segment, distributing additional labels evenly minimizes the maximum distance between any unlabeled point and its nearest labeled point. As discussed in Section III-D1 and R1.O2, shorter intervals between labeled and repaired points enhance the confidence of validity derived from local speed constraints. And regularly spaced labels enable the method to leverage tighter temporal correlations for evaluating repairs, leading to more accurate results. However, it should be noted that such configuration described above was not used in our experiments. Specifically, labels are randomly positioned within the segments where errors occur. And we repeat each test 10 times and report the average. Consistent with results in Section V, this less controlled configuration still leads to promising performance, thereby demonstrating our method's practical utility.

We add "Consequently, to improve repair accuracy, in theory, boundary points of segments where errors occur should be labeled for error localization, and labels should be evenly distributed within segments to minimize unlabeled points' maximum distance to their nearest label. However, the above cases are not always applicable. In practice, 1% to 3% labels on dirty points with random positions within segments are sufficient to get good repair performance." in Section III-D1 corresponding to the content above. □

O3: *O3: The bidirectional validation requires future information, limiting its applicability in online and real-time scenarios. While unidirectional validation offers a workaround, it compromises accuracy.*

REPLY: (1) First, we apologize for the errors in Figure 12(b). Owing to a coding error, the throughput results for $\mathsf{MINOR-O}$, MTCSC, and VARX in experiment were reported as ten times higher than the actual values. However, it should be noted that this did not affect the comparative throughput rankings of the four methods, and the conclusions of this experiment were not affected either. The figures have been corrected now.

(2) Yes, the bidirectional repair method requires future information. However, we propose a concept that facilitates its application in online and real-time scenarios. In these scenarios, rather than repairing a data point immediately upon receiving it, the data point is temporarily stored in a buffer sequence. When obtaining a labeled point (as described in Section I-B, wherein such labeled points are obtained using reliable sensors with a relatively long sensing period), the previously repaired data points are concerned as one labeled segment and concatenated with the entire buffer sequence. Through this process, the bidirectional repair method can be applied in online and real-time scenarios. However, the specific implementation requires further exploration in future research.

(3) Yes, although the unidirectional repair algorithm provides a solution, it compromises repair accuracy. However, in our experiments on error length and online computing on the ILD dataset, the same errors were injected at identical error rates and in a consistent manner. We add "We inject errors using the same way as in Experiment V-C3" in Section V-C5 to emphasize this. By comparing Figures 10(a) and 12(a), it is evident that the repair performance of $\mathsf{MINOR-O}$ did not deteriorate significantly relative to that of $\mathsf{MINOR-U}$. And as indicated by the throughput, the time cost remains acceptable, demonstrating the practicality of this method. □

The above is our response. In addition, we apologize for an error in Figure 13(b). Due to a coding error, the results in Figure 13(b) could not be reproduced. We have made corrections to address this issue and replaced Figure 13(b) after verifying its reproducibility. However, it should be noted that the original conclusions remain unaffected.