Cleaning Time Series under Seasonal and Trend Constraints

ZIJIE CHEN, Tsinghua University, China
AOQIAN ZHANG, Beijing Institute of Technology, China and Tangshan Research Institute, Beijing Institute of Technology, China
SHAOXU SONG*, Tsinghua University, China

Time series data are often found to be dirty, e.g., with anomalies or sensor failures. Such dirty data obviously hinder the downstream analysis tasks such as forecasting, clustering or classification. Simply discarding the potentially dirty data points is not an option, making the time series incomplete and incompatible to machine learning models. While many time series data cleaning techniques have been developed in the last decade, e.g., with the help of constraints on value fluctuation, the seasonal features are surprisingly ignored. In this paper, we propose to clean time series by first capturing seasonal and trend constraints, and then enforcing them for cleaning. Unfortunately, directly applying existing seasonal-trend decomposition methods is found imprecise (itself affected by errors) and incomplete (not computed at the beginning or end of the series). Moreover, unlike efficient cleaning with simple value fluctuation constraints, the time series cleaning problem with seasonal and trend constraints is proved to be NP-complete. In this sense, we first improve seasonal and trend filter with tolerance to errors and extension on two directions. Then, an efficient heuristic is designed to iteratively repair the time series and refine the seasonal and trend constraints. The approach has now become a built-in function in a product system Apache IoTDB. Experiments on real-world datasets demonstrate the superiority of our proposal in cleaning seasonal time series and improving downstream applications.

CCS Concepts: • **Information systems** → **Data cleaning**.

Additional Key Words and Phrases: Time Series Data, Seasonal Trend Decomposition, Data Cleaning

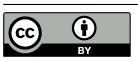
ACM Reference Format:
Zijie Chen, Aoqian Zhang, and Shaoxu Song. 2025. Cleaning Time Series under Seasonal and Trend Constraints. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 308 (December 2025), 26 pages. https://doi.org/10.1145/3769773

1 INTRODUCTION

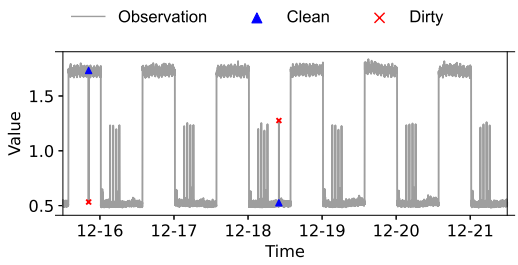
Seasonal features are commonly observed in time series. For instance, the load on electrical grids and the power consumption of a city in the daily cycle, as shown in Figure 1. Seasonal-trend decomposition (STD), which decomposes the data into the seasonal, trend and other components, is widely used in seasonal data analysis [15, 35]. The seasonal and trend components represent the regular patterns in the data, capturing periodic variations and long-term trends, respectively.

*Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

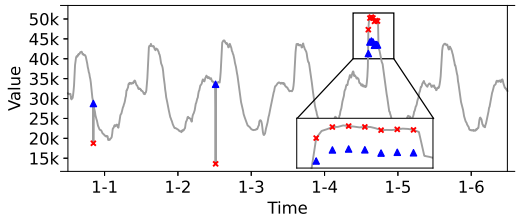
Authors' addresses: Zijie Chen, Tsinghua University, Beijing, China, chenzj23@mails.tsinghua.edu.cn; Aoqian Zhang, Beijing Institute of Technology, Beijing, China and Tangshan Research Institute, Beijing Institute of Technology, TangShan, China, aoqian.zhang@bit.edu.cn; Shaoxu Song, Tsinghua University, Beijing, China, sxsong@tsinghua.edu.cn.



ACM 2836-6573/2025/12-ART308
https://doi.org/10.1145/3769773



(a) Load on electrical grids



(b) Power consumption

Fig. 1. Example of dirty seasonal time series and the corresponding clean series.

1.1 Motivation

However, time series data often contain errors due to sensor or instrument failures [25, 38, 42]. Not surprisingly, these errors may break the seasonality of the time series, and thus obstruct the downstream data science applications performing, such as time series forecasting, classification and clustering as shown in Section 7.2.2.

Example 1. *As shown in Figure 1a, the load on electrical grids in a factory presents a periodic sudden increase at the start of working hours and a periodic sudden decrease in the end of working time. Routine inspections also produce some spikes in electrical load every night. However, the grid data occasionally decreases at noon, around time 500 denoted by red ×, due to sensor errors rather than the decrease in electricity supply. Such errors violate seasonal features of grid data in the real world. They prevent the discovery of factories with similar electricity consumption patterns.*

Likewise, in Figure 1b, the power consumption of Tetouan city [22] increases gently in the morning, then decreases slightly in the afternoon. At night, it increases to the peak value and then falls to the initial value before the next morning. However, at time 600 in Figure 1b, a shift error [39] occurs owing to granularity mismatch or unit error. There also exists some single point errors in power data. Obviously, these errors break the seasonal features of power data, and impede the prediction of power consumption in the city.

1.2 Intuition

To address the aforesaid single and consecutive errors in time series, a number of cleaning methods have been developed in the past decade, using many constraints like speed [25] and statistical features [38]. However, due to the large deviation between periodic sudden increases and their neighboring values, such smoothing-based methods will *mistakenly* treat these increases, regular in each day, as dirty data to be repaired, e.g., $x_{980}, x_{1040}, x_{1100}, \cdots$ in Figure 1a. Therefore, it is

necessary to leverage seasonal features to distinguish between different periods to detect dirty data. Surprisingly, none of the existing repair methods consider such seasonal features.

In this paper, inspired by seasonal-trend decomposition [4, 11, 18], we present a novel constraint-based method designed for cleaning seasonal time series data. Seasonal-Trend decomposition, which decomposes the data into the seasonal, trend and other components, is widely used in seasonal data analysis [15, 35]. The seasonal and trend components represent the regular patterns in the data, capturing periodic variations and long-term trends, respectively. Inspired by the seasonal and trend components denoting the underlying patterns, we consider points that deviate significantly from these components as dirty data. Therefore, we define the seasonal and trend constraints to detect dirty data. As a remedy, we utilize the seasonal and trend components of the corresponding points with white noise to generate a repair. The white noise captures short-term random fluctuations in the data.

1.3 Challenge

Unfortunately, existing decomposition methods such as STL [4], OnlineSTL [18] and OneShotSTL [11] are found to be both *imprecise* and *incomplete* to capture seasonal and trend constraints for cleaning time series.

1.3.1 Imprecise Seasonal Component. The seasonal filters [4, 11, 18] are sensitive to errors. For example, the errors in Figure 1b may cause the above decomposition methods to produce imprecise seasonal components (see Figure 4 below). Then seasonal components at the same times on other days may deviate from their true values due to the influence of these imprecise components. This may also affect the aforementioned repair method. In this sense, the existing decomposition methods must be adapted to the repair of dirty data.

1.3.2 Incomplete Trend Component. Moreover, the seasonal-trend decomposition methods [4, 11, 18] use a trend filter to extract the trend component. These methods use a sliding window to estimate the trend feature referring to the neighbor points on either side. However, there are no sufficient observations to compute at the beginning and end of the series, which leads to incomplete trend components near the endpoints. For example, in Figure 1a, the period of grid data is 1440 as a day with data collection in every minute. For the dirty point occurring at time 500, i.e., in the first half of the period, there are no enough points for the sliding window to compute trend component. Hence, we need to extend the trend components at the beginning and end of time series.

1.4 Contribution

To the best of our knowledge, this is the first study on cleaning time series having seasonal features. Our main contributions in this paper are summarized as follows.

(1) We formalize the problem of *cleaning time series under seasonal and trend constraint* in Section 3. It is to find a repaired series in which each value satisfies the seasonal and trend constraints, and the modification is minimized. Unfortunately, Theorem 1 proves that the corresponding decision problem is NP-complete.
(2) We address imprecise seasonal components in Section 4.1, avoiding the sensitivity to violations in existing methods. As shown in Figure 4, OnlineSTL [18] mistakenly detects x_{434} and x_{578} as errors. Proposition 1 guarantees that our median-based filters tolerate specific errors and accurately detect violations in Figure 5.
(3) We address incomplete trend components in Section 4.2, handling boundary violations ignored by prior methods. As shown in Figure 4, OnlineSTL fails to detect x_{50} while our method correctly

Table 1. Frequently used notations

Symbol	Description
x	time series data
x_i	i-th data point of x
t_i	trend component of x_i
s_i	seasonal component of x_i
d_i	de-trend component of x_i
n	time series length
m	period length
w	trend filter window size
η	seasonal and trend constraint threshold
V	set of detected errors
x_i^*	clean data of x_i
x_i'	repair data of x_i

identifies it in Figure 5. Proposition 2 ensures the extension is robust to violations when errors are extreme.
(4) We enforce seasonal and trend constraints for cleaning in Sections 5. To our knowledge, this is the first iterative cleaning method to use trend and seasonal constraints for time series cleaning. First, for data points that violate the constraint, we generate repair values. Proposition 3 shows that the proposed seasonal repair is optimal for extreme dirty data. Furthermore, we design an iterative decomposition-based cleaning algorithm. Note that the decomposition result of the repaired series may differ from that of the original series. Consequently, some other data points may be detected as violations. Therefore, performing detection and repair iteratively ensures that all points satisfy the seasonal and trend constraints. This process is proven to converge when errors are rare in Proposition 4.
(5) We deploy the proposed method in a time series database Apache IoTDB [14, 30] in Section 6. It is now available for using as a built-in function of this product system. The repair function can be called directly as a SQL statement [6].
(6) We report an extensive experimental evaluation on real-world data in Section 7. The results demonstrate that our method achieves the best performance in cleaning *seasonal time series* compared to the existing methods. As a consequence, the downstream applications are improved after cleaning, such as forecasting, clustering and classification.

Table 1 lists the frequently used notations in this paper. The proofs of some theoretical results are provided in the supplementary.

2 RELATED WORK

We discuss the related studies on (1) seasonal time series analysis, (2) seasonal-trend decomposition and (3) time series cleaning.

2.1 Seaconal Time Series Analysis

Many time series analysis methods account for seasonality. We contrast our proposed approach with seasonal prediction and anomaly detection as examples. (1) Prediction and cleaning are orthogonal tasks: seasonal prediction methods, such as SARIMA [24], predict future unobserved values in a time series, while our cleaning method repairs/modifies existing/observed erroneous

values. Cleaning can improve downstream prediction, i.e., complementary as evaluated in Section 7.2.2. Moreover, seasonal prediction models can detect violations by thresholding the distance between predicted and observed values. A method using SARIMA for detection and linear interpolation [26] for repair (SARIMA_Linear) is evaluated in Section 7.2 and Section 7.3. Due to inaccuracies in both steps, SARIMA_Linear yields limited repair quality. (2) Seasonal anomaly detection methods, such as matrix profile [36], identify anomalous points, while our cleaning methods involves both error detection and repairing to enhance data quality. Notably, anomaly detection can be combined with interpolation for cleaning. Thus, Section 7.2 and Section 7.3 evaluate a hybrid method using matrix profile to detect and linear interpolation to repair. Since Matrix Profile cannot adapt to anomalies of varying scales under a fixed subsequence length, the repair quality is limited.

2.2 Seasonal-Trend Decomposition

As seasonal features are commonly observed, seasonal-trend decomposition is important to facilitate anomaly detection and forecasting. Seasonal-trend decomposition is divided into batch methods and online methods. (1) Batch methods decompose the entire data. It can deal with long cycles, etc., but the computation is complicated, and the decomposition process is very time-consuming. The latest methods such as RobustSTL [32] have further improved the decomposition performance, at the expense of efficiency. (2) Online methods have high decomposition efficiency, and can be applied to real-time processing scenarios. However, its performance is poor and cannot handle violations. Currently, there are only two online methods, OnlineSTL [18] and OneShotSTL [11]. OnlineSTL uses a tri-cube filter to extract trend component and exponential smoothing to extract cycle component. OneShotSTL uses an l1-norm filter to extract trends and cycle components. Both methods cannot handle extreme violations.

2.3 Time Series Cleaning

Time series cleaning methods offer various options for cleaning data, which can be categorized into three types referring to [31]. (1) Smoothness-based methods often smooth out highly irregular fluctuations in short intervals. The Exponentially Weighted Moving Average (EWMA) [13] is a typical method widely used in time series analysis. It refers to an average of data with exponentially decreasing weights. However, the original series may suffer from overly modifying with these methods, leading to low accuracy. (2) Constraint-based methods formalize rules for some evidence patterns to guide the repair. SCREEN [25] is a typical method that constrains the speed of value changes. Unfortunately, the method cannot handle consecutive violations. (3) Statistical-based methods learn models from the statistical features of the data to generate a repair. LsGreedy [38] is a typical method that maximizes the probability distribution of speed changes. (4) Model-based methods leverage the results of time series prediction models for data repair. IMR [39] is a typical method that utilizes ARX models to predict and repair while iteratively estimating the model parameters until convergence. To the best of our knowledge, all these methods ignore seasonal features.

3 PROBLEM STATEMENT

In this section, we first present seasonal-trend decomposition (STD) and its resulting components. Then, we introduce the seasonal and trend constraint on the decomposed components. Finally, we define the problem of cleaning time series under seasonal and trend constraints.

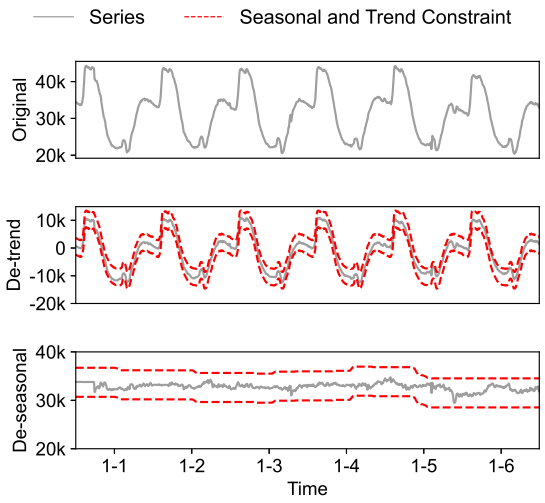


Fig. 2. Seasonal and trend constraint on clean data.

3.1 Seasonal and Trend Constraints

Consider a time series of n observations, $x = \{x_1, \ldots, x_n\}$, where x_i is the value of the i-th data point. To extract features from x, we consider the additive decomposition model [2],

$$x_i = t_i + s_i + r_i, \tag{1}$$

where t_i is the trend component of x_i reflecting the long-term progression, s_i is the seasonal component of x_i reflecting seasonality with a period m, which can be estimated by period detection methods [9, 28, 33], and r_i is the residual component of x_i reflecting irregular influences.

Inspired by the seasonal and trend components in decomposition, in this section we introduce seasonal and trend constraints to enforce the stationarity of seasonal and trend features in time series.

All existing seasonal trend decompositions employ a combination of trend filters, seasonality filters, denoising filters [32], differencing filters, or optimization filters [7, 34]. For simplicity, we assume that STD only uses one trend filter and seasonality filter for decomposition. The trend filter, denoted as $\mathcal{T}_i$, is a function used to extract the trend component t_i. It typically takes data within a window of length w centered at x_i as input, and outputs its corresponding t_i in polynomial time, i.e.,

$$t_i = \mathcal{T}_i\left(\left[x_{i+j};\ \lfloor -\frac{w-1}{2} \rfloor \leq j \leq \lfloor \frac{w-1}{2} \rfloor\right] \in \mathcal{R}^w\right). \tag{2}$$

Since the seasonal component s cannot be extracted on data with trend, we first subtract the trend component from the original series, i.e., $x - t$. It leads to the detrended series $d = d_1, \ldots, d_n$ by subtracting the trend component from the original series, i.e., $d_i = x_i - t_i$. The detrend process removes the trend component directly from the original series, eliminating trend features to facilitate the extraction of seasonal features. The seasonality filter, denoted as $\mathcal{S}_i$, is a function used to extract the seasonal component s_i, can then take data with the same phase as d_i as input, and

output its corresponding s_i in polynomial time, i.e.

$$s_i = \mathcal{S}_i\left(\left[d_j;\ 1 \le j \le n \wedge (j-i) \mid m\right] \in \mathcal{R}^{\lfloor \frac{n-i \mod m}{m} \rfloor}\right). \tag{3}$$

for $i = 1, \ldots, m$.

Inspired by time series stationarity constraints, which means the statistical properties of a time series do not change over time, we introduce a seasonal and trend constraint to limit the distance between the original series and its seasonal and trend components. Intuitively, this constraint guarantees that the underlying seasonal and trend patterns remain stable over time. The threshold for this constraint is denoted by η.

Definition 1 (Seasonal and Trend Constraint)**.** *Given time series data x_i and seasonal and trend components $s_i \in s$, $t_i \in t$ decomposed by x_i, the distance between x_i and its components $s_i + t_i$ should be within the threshold η_1 and η_2, i.e.,*

$$\eta_1 \le s_i + t_i - x_i \le \eta_2.$$

The constraint can be interpreted from two perspectives. First, $s - \eta_2 \le x - t \le s + \eta_1$ constrains the de-trend series to remain within a distance of η_1 and η_2 from the seasonal component, thus ensuring the stability of the seasonal pattern. Second, $t - \eta_2 \le x - s \le t + \eta_1$ constrains the de-seasonal series to remain within a distance of η from the trend component, thus ensuring the stability of the trend pattern.

Example 2 (Seasonal and trend constraint)**.** *Consider the time series in Figure 2, a real segment from the Power dataset [22], with 864 observations collected in ten-minute intervals over 6 days. We use time-based x-axis labels (e.g., "1-1", "1-2") to represent month and day. This highlights daily periodicity of the data, i.e., the period is 144. De-trend refers to the original series with the trend component removed, i.e., $x - t$. Similarly, de-seasonal denotes the original series with the seasonal component removed, i.e., $x - s$. In Figure 2, the second and third subplots illustrate the two interpretations of the seasonal and trend constraints, i.e. $s - \eta \le x - t \le s + \eta$ and $t - \eta \le x - s \le t + \eta$, respectively. As shown, all data points in the clean series lie within the range defined by seasonal and trend constraint (the red dashed lines). Therefore, the clean series satisfy the constraints.*

3.2 Optimization Problem

In practice, the ground truths for errors are unknown; otherwise, there is no need to clean. Hence, it is infeasible to optimize based on the distance between error and truth. Consequently, existing constraint-based methods [3, 25, 40] propose to minimize the data modification such that the specified constraints are satisfied, rather than considering how far it might be from the true (unknown) signal. Following the same principle, our optimization problem also aims to minimize the modification between the dirty input x and repaired output x'. Let us first define the repair cost,

Definition 2. *The repair cost is evaluated by the number of data points modified from the original x to the cleaned x',*

$$\Delta(x, x') = card(\{x_i \in x\ ;\ x_i \ne x'_i\}).$$

We perform cleaning by modifying the data to satisfy the seasonal and trend constraints, which incurs a repair cost. Moreover, a higher repair cost reflects a larger overhead. Therefore, we propose to find a repair that (i) satisfies the constraints and (ii) minimizes the repair cost while decomposing.

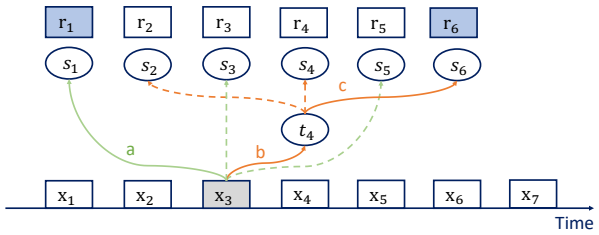


Fig. 3. Transformation for the example set $Y_3 = \{u_1, u_6\}$.

Problem 1. *Given a time series x with n observations, trend filter $\mathcal{T}$, seasonal filter $\mathcal{S}$, and constraint threshold η, the cleaning problem is to find cleaned series x' such that $\Delta(x, x')$ is minimized, and each x'_i satisfies the seasonal and trend constraint.*

We can formalize the cleaning problem as

$$
\begin{aligned}
\min \quad & \Delta(x, x') = card(\{x_i \in x \,;\; x_i \neq x'_i\}) \\
\text{s.t.} \quad & t'_i = \mathcal{T}_i\left(\left[x_{i+j};\ \lfloor -\frac{m-1}{2} \rfloor \leq j \leq \lfloor \frac{m-1}{2} \rfloor\right] \in \mathcal{R}^m\right) \\
& d'_i = x'_i - t'_i \\
& s'_i = \mathcal{S}_i\left(\left[d'_j;\ 1 \leq j \leq n \wedge (j-i) \mid m\right]\right) \\
& \eta_1 \leq s_i + t_i - x_i \leq \eta_2 \\
& i = 1, 2, \ldots, n
\end{aligned}
$$

where x'_i are variables in problem solving, t'_i, s'_i and d'_i are the trend, seasonal and de-trend components of x'_i.

The problem of minimizing repair cost under seasonal and trend constraints is generally hard. We investigate the decision version of the original decomposition problem, and illustrates its hardness as follows.We give only a sketch of the proof and a few details. The full proof is available in [27].

Theorem 1. *Given time series $x = x_1, x_2, \cdots, x_n$, trend filters $\mathcal{T}_i$, seasonality filters $\mathcal{S}_i$, constraint threshold η and constant τ, the problem is NP-complete to determine whether there is a cleaned time series $x' = x'_1, x'_2, \cdots, x'_n$ having (1) seasonal and trend constraint $\eta_1 \leq s_i + t_i - x_i \leq \eta_2$ satisfied for each $x'_i \in x'$, (2) the repair cost $\Delta(x, x') \leq \tau$.*

Proof sketch. Given that $\mathcal{T}_i$ and $\mathcal{S}_i$ are executable within polynomial time, we can also verify whether the solution is correct or not in polynomial time, hence the decomposition problem is in NP. We prove the NP-complete of the decomposition problem by a reduction from the set cover problem, one of Karp's 21 NP-Complete problems [16]. Figure 3 shows the transformation of a set cover problem instance into a valid repair problem instance. □

4 CAPTURING SEASONAL AND TREND CONSTRAINTS

In this section, we first determine the trend filter $\mathcal{T}$ and seasonal filter $\mathcal{S}$ proposed in Section 3.1 to capture seasonal and trend constraint in Section 4.1. Moreover, considering the incomplete trend components, we propose a component extension method in Section 4.2.

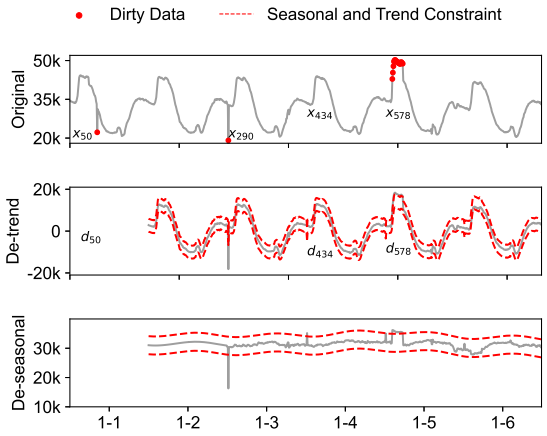


Fig. 4. Seasonal and trend constraint by OnlineSTL on dirty data. The dirty data point x_{50} is not identified due to incomplete trend component, while clean data points x_{434} and x_{578} are mistakenly identified as violations to the imprecise seasonal constraint.

4.1 Median Filter

As introduced in Section 3.1, the cleaning method uses trend filter and seasonal filter. However, existing decomposition methods, such as OnlineSTL, and their proposed seasonal and trend filters often suffer from two main limitations: (1) intolerance to violations, where the resulting components are distorted due to violations, and (2) the use of sliding windows in trend filters leads to incomplete trend components. As a result, these methods are inadequate for capturing constraints for cleaning. In this section, we present a simple yet robust method for computing constraints to ensure error tolerance in the cleaning process.

Based on these insights, we propose a trend constraint that limits the distance between the trend of x_i and the median value within a window of length w centered at x_i. In particular, the trend filter is shown as

$$t = \text{median}\left(\left[x_{i+j};\ \lfloor -\frac{w-1}{2} \rfloor \le j \le \lfloor \frac{w-1}{2} \rfloor\right]\right) \tag{4}$$

for $i = 1 - \lfloor -\frac{w-1}{2} \rfloor, \ldots, n - \lfloor \frac{w-1}{2} \rfloor$, implementing $\mathcal{T}$ in Formula 2. Given the seasonality, we set the window size equal to the period, i.e., $w = m$.

Before capturing the seasonal constraint, we first avoid the influence of the trend feature in the original series by subtracting, i.e., $d_i = x_i - t_i$ for $i = 1, \ldots, m$. We again use the median as $\mathcal{S}_i$ for $1 \le i \le n$ to avoid the influence of errors. Note that the trend component is absent in the first and last half-periods, these data are excluded from the calculation

$$s_i = \text{median}(\{d_j;\ 1 \le j \le n \wedge (j - i) \mid m\}) \tag{5}$$

for $i = 1, \ldots, m$, which implements $\mathcal{S}$ in Formula 3.

We now show that our violation-tolerant constraint can completely eliminate the impact of violations under certain cases.

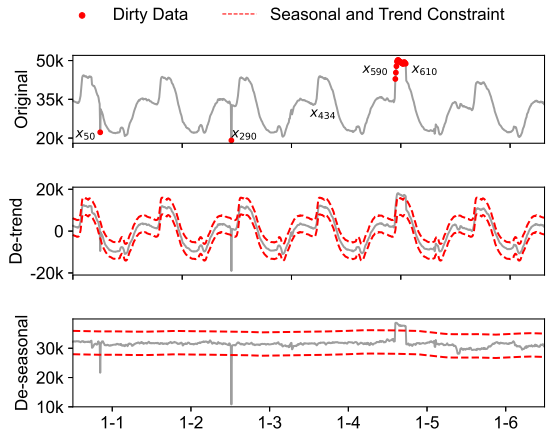


Fig. 5. Seasonal and trend constraint by our proposal, where all dirty data, x_{50}, x_{290}, x_{590} to x_{610} are successfully identified.

Proposition 1. *Let* $x_i^* \geq \max(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ *and* $x_j^* \leq \min(x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n)$ *be the maximum and minimum values in a clean series. For any violations* $x_i > x_i^*$ *and* $x_j < x_j^*$ *occurring on these two values, the trend and seasonal constraint by Formulas 4 and 5 on* $[x_1, \cdots, x_i, \cdots, x_j, \cdots, x_n]$ *with violations is exactly the same as that on the clean series* $[x_1, \cdots, x_i^*, \cdots, x_j^*, \cdots, x_n]$.

The proof of Proposition 1 is provided in the supplementary.

4.2 Component Extension

Recall that when defining trend constraint, in Formulas 4, we use a window centered at x_i to limit the smoothness of time series. However, we do not have enough observations to compute the moving median on both sides of the series, resulting in incomplete trend components near the endpoints. To solve this problem, we propose an extension method that incorporates the repair process.

We first shift the indices of the original series to the back by one period m, i.e., the original series is now $x_m, x_{m+1}, \ldots, x_{m+n}$. Since the incomplete trend is caused by the lack of data at the end, we can combine the results of the local seasonal-trend decomposition to give the prediction value of the original series, i.e., $x_i = t_{m+\lfloor \frac{m-1}{2} \rfloor} + s_{i+m}$ for $i = 1, \ldots, m$, $x_i = t_{n+m-\lfloor \frac{m-1}{2} \rfloor} + s_{i-m}$ for $i = n+m+1, \ldots, n+2m$. Finally, we can compute the extension of t as follows. At this point, we shift the indices back.

$$t_i = \text{median}(\{x_j; 1-\lfloor -\frac{m-1}{2} \rfloor \leq j \leq i+\lfloor \frac{m-1}{2} \rfloor\} \cup \{t_{1+\lfloor \frac{m-1}{2} \rfloor}+s_j; m+i-\lfloor \frac{m-1}{2} \rfloor \leq j < m+1-\lfloor -\frac{m-1}{2} \rfloor\}) \tag{6}$$

for $i = 1, \ldots, -\lfloor -\frac{m-1}{2} \rfloor$.

$$t_i = \text{median}(\{x_j; i+\lfloor -\frac{m-1}{2} \rfloor \leq j \leq n-\lfloor \frac{m-1}{2} \rfloor\} \cup \{t_{n-\lfloor \frac{m-1}{2} \rfloor}+s_j; n-m-\lfloor \frac{m-1}{2} \rfloor < j \leq i-m+\lfloor \frac{m-1}{2} \rfloor\}) \tag{7}$$

for $i = n+1-\lfloor \frac{m-1}{2} \rfloor, \ldots, n$.

Note that the trend extension is also error tolerant. Similar to Proposition 1, we have the same extension results with/without errors under certain cases.

Proposition 2. *Let* $W_k = \{x_l;\ ((l-k) \mid m) \wedge (l \neq k)\} \cup \{x_l;\ 1 \leq l \leq m\}$ *for* $1 \leq k \leq m$ *in a clean series,* $x_i^* \geq \max(W_k)$ *and* $x_j^* \leq \min(W_k)$ *be the maximum and minimum values in* W_k*. For any errors* $x_i > x_i^*$, $x_j < x_j^*$ *occurring on these two values, the trend extension by Formula 6 on* $[x_1, \cdots, x_i, \cdots, x_j, \cdots, x_n]$ *with errors is exactly the same as that on the clean series* $[x_1, \cdots, x_i^*, \cdots, x_j^*, \cdots, x_n]$.

The proof of Proposition 2 is provided in the supplementary.

Example 3 (End-to-end comparison of OnlineSTL and violation-tolerant filter, Example 2 continued)**.** *Consider the Power dataset again. Figures 4 and 5 illustrate the seasonal and trend constraint captured by OnlineSTL and our proposed violation-tolerant filter seperately. OnlineSTL is a representative of online decomposition methods. To study how the errors affect the result of OnlineSTL, we inject single point errors in* x_{50} *and* x_{290} *and continuous errors in twenty points* $x_{590}, \ldots, x_{609}$*. We analyze the data point* x_{434} *and* x_{50}*. For* x_{434}*, OnlineSTL is obviously affected by errors in de-trend component* d_{434}*. Note also that* x_{50} *has no trend component, as shown in Figure 4, since* d_{50} *is in the first period of the series. In real-world scenarios, such errors frequently occur. For example, GPS data often contains errors during startup, searching for satellite signals. Existing decomposition methods yield incomplete trend components, then such errors cannot be detected and repaired. Such errors degrade downstream task performance. For example, in power consumption forecasting, continuous errors in* $x_{590}, \ldots, x_{609}$ *may cause the subsequent prediction* x_{610} *to deviate.*

In contrast, our method ensures that the decomposition of x_{434} *is not affected by surrounding errors, preventing it from being mistakenly detected as a violation. Therefore, the decomposition in Figure 5 is more robust to violations than that in Figure 4. Based on the trend component obtained, we also conduct an expansion. This allows the detection of errors occurring in the first and last half cycles. For example, the error* x_{50} *has no residual component in Figure 4. Expanding the trend component resulted in a de-trend component* $x_{50} - t_{50}$ *of -7452.68. Clearly, cleaning with extended components improves downstream tasks such as prediction, where boundary points are less affected by errors.*

5 CLEANING UNDER SEASONAL AND TREND CONSTRAINTS

In this section, we perform time series cleaning under trend and seasonal constraints. Section 4.1 detects values that violate the constraints. Then, Section 5.2 generates candidate repairs. Finally, Section 5.3 introduces an iterative repair algorithm to ensure all points satisfy the constraints.

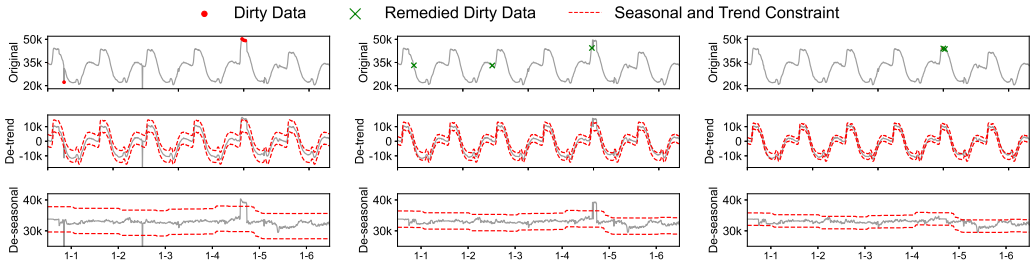
5.1 Violations to Seasonal-Trend Constraints

We introduce the method for detecting data points that violate the seasonal and trend constraints mentioned in Section 3.1, i.e. points that violate the regular features. To specify the threshold η_1 and η_2 of seasonal and trend constraint, we use the three sigma principle [19], which refers to data within three standard deviations from a mean. In particular, the threshold η in Definition 1 is computed as $\eta_1 = \eta_2 = 3\sqrt{\frac{\sum_{i=1}^{n}(x_i - s_i - t_i)^2}{n} - \left(\frac{\sum_{i=1}^{n}(x_i - s_i - t_i)}{n}\right)^2}$.

As noted in Section 3.1, data points with a large deviation between x_i and its corresponding components $s_i + t_i$ are considered violations in time series that exhibit a seasonal pattern. Therefore, the data points identified by the three sigma principle are regarded as errors and are aggregated into a set, denoted as V, i.e.,

$$V = \{i \mid 1 \leq i \leq n \ \wedge\ \eta_1 \leq s_i + t_i - x_i \leq \eta_2.\}.$$

Example 4 (Violation detection, Example 3 continued). *Consider the power consumption series with injected errors, we then treat the seasonal component s and trend component r from the decomposition result in Example 3 to detect violation. First, we calculate the constraint threshold η. We have $\eta = 1,693.23$. Then, we can identify x_i as violation where $x_i - s_i - t_i \leq -1,693.23$ or $x_i - s_i - t_i \geq 1,693.23$. Note that $x_{290} - s_{290} - t_{290}$ is -20060, and thus it can be easily detected as an violation.*



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Fig. 6. Example of iterative cleaning by refining seasonal and trend constraint.

5.2 Median Repair

As discussed in Section 1, the existing time series cleaning methods such as SCREEN [25] or IMR [39] are often based on temporally close neighbors, without considering the seasonal features in periods. To repair the aforesaid violations detected by seasonal decomposition, we propose a new method, called Seasonal Repair Generation (SRG).

Recall that, in Section 4.1, we obtain the trend components t_i and seasonal components s_i of x_i through filter. As illustrated in Figure 5, our violation-tolerant filter is robust to errors, with seasonal components s and trend components t over the dirty data almost the same as those for the clean data. In this sense, s_i and t_i of x_i have no need to be modified. In addition to regular seasonal and trend patterns, time series often exhibit random fluctuations. To account for this, we capture such variations from historical data and incorporate them into the cleaning process. Specifically, we add white noise with zero mean and a standard deviation of $\frac{\eta}{3}$. Finally, we combine the unchanged trend and seasonal repair with the white noise to obtain the repair x'_i of x_i.

$$x'_i = t_i + s_i + \epsilon, \quad \epsilon \sim \mathcal{N}(0, (\frac{\eta}{3})^2) \tag{8}$$

We now show that seasonal repair generation achieves the optimal repair under certain cases. The proof is available in [27].

Proposition 3. *Let $x^*_i \geq \max(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$ and $x^*_j \leq \min(x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n)$ be the maximum and minimum values in a clean series. For any errors $x_i > x^*_i$ and $x_j < x^*_j$ occurring on these two values, the seasonal repair x' generated by Formulas 8 is always the optimal repair with cost $\Delta(x, x') = 2$.*

Example 5 (Repair generation, Example 4 continued). *Consider the power consumption series with injected errors. In Example 4, we detect an error in the data point x_{290}. We will describe the seasonal repair generation process for x_{290} here. Firstly, we obtain the trend and seasonal repair using their original values, i.e., $t'_{290} = t_{290} = 32615.70$, $s'_{290} = s_{290} = 408.44$. Subsequently, we calculate the white noise as $\epsilon = 25.99$. Finally, we sum up all the repair components to obtain $x'_{290} = 33050.13$. Considering*

the original x^*_{290} value before injecting the error is 32652.15, our method effectively generates a suitable repair.

5.3 Iterative Repair

Note that in Section 3.2, we reduce our problem from the Set Covering Problem, whose greedy algorithm iteratively selects the subset with the minimal cost-effectiveness ratio. Inspired by this, we iteratively repair the data by cleaning the minimal number of points required to satisfy the constraint. It is necessary because the decomposed result of the cleaned series may deviate from that of the original series, which leads to other points being detected as new violations after a round of repair.

5.3.1 Iterative Algorithm. Algorithm 1 introduces the iterative algorithm procedure. Let $x^{(h)}$ denote x in the h-th iteration, where $x^{(0)}$ is the original time series. The main steps are as follows:

(1) Decomposition. Line 3 decompose $x^{(h-1)}$ into seasonal and trend components. It applies the proposed violation-tolerant filter in Section 4.1 and trend extension in Section 4.2.
(2) Detection. Line 4 detects the violations to be cleaned (in Section 5.1).
(3) Generation. Line 7 generates the repair $x_i^{(h)}$ for violations, while leaving the remaining series unchanged. We employ the proposed repair generation method in Section 5.2.

Algorithm 1: Cleaning Time Series under Seasonal and Trend Constraints

Input: time series data $x \in \mathcal{R}^n$
Output: cleaned time series data $x^{(h)} \in \mathcal{R}^n$

```
1 x ← x^(0);
2 for h ← 1 to max_iter do
3 |   s^(h), t^(h) ← Decompose(x^(h−1));
4 |   V^(h) ← Detect(r^(h));
5 |   if V^(h) = ∅ then
6 |   |   break;
7 |   x_i^(h) ← Generate(s^(h), t^(h), r^(h), V^(h));
8 return x^(h);
```

Example 6 (Iterative decomposition, Example 5 continued)**.** *The process of iterative cleaning is shown in Figure 6. Considering the power consumption series with injected errors, we show the process of iterative cleaning x_{290} and x_{600} below.*

In the first iteration, as shown in Figure 6a, we have threshold -5453.42 and 5453.42. r_{290} is -20371.98 and easily detected as an violation. However, r_{600} is 5285.96, so the violation cannot be detected. Then, we use the same method as in Example 5 to obtain the repair for r_{290}, whose value is 33050.13.

In the second iteration, as shown in Figure 6b, we have threshold -3806.94 and 3806.94. r_{290} is -47.37, thus it does not need to be repaired. However, this time x_{600} is considered an violation, since x_{600} is 5346.38. The repair $x'_{600} = 44032.43$ is obtained using the same seasonal repair generation method.

In the third iteration, as shown in Figure6c, the threshold is -2901.66 and 2901.66. r290 and r291 are -47.37 and 1.2, respectively. The other data points are also not detected as violations, so the series is completely cleaned. Note that the original value of x_{290} is 32652.15 and x_{600} is 43413.56, our method works well.

5.3.2 Convergence Analysis. We show below that the iteration of cleaning guarantees to converge in certain cases. The proof is available in [27].

Proposition 4. *When the violations occur only in one cycle, Algorithm 1 can always generate a repair x' of x that satisfies the seasonal and trend constraint, i.e., converged.*

Note that the proposed iterative cleaning scheme converges not only when violations occur only in one cycle. Indeed, the algorithm converges rapidly in real-world scenarios. Section 7.4.2 shows that the algorithm converges in 4 iterations under 5‰ error rate and converges in 8 iterations under 50‰ error rate. In practice, errors are often rare, e.g., typically below 50‰[23]. For general cases in practice, we use a parameter, max_iter, to restrict the maximum number of iterations. It avoids the situation that the repaired time series continuously detects violations.

5.3.3 Complexity Analysis. As shown in Algorithm 1, the algorithm is divided into three parts: decomposition, detection, and generation. (1) For decomposition, both the trend and seasonality filter use the median filter. Thanks to the algorithm proposed in [1], the calculation of median filter takes $O(n)$ time. (2) For detection, we only need to calculate the standard deviation, which has a time complexity of $O(n)$. (3) For generation, we only need to add the components with an time complexity of $O(1)$. In summary, the time complexity of each repair round is $O(n \log m)$. Note that we use these steps iteratively, the total time complexity of the algorithm is calculated by multiplying the number of iterations with $O(n \log m)$.

6 SYSTEM DEPLOYMENT

We deploy our proposed method for cleaning seasonal time series in an open-source time series database system, Apache IoTDB [14, 30]. The document is available on the product website [6] and the code is included in the GitHub repository of the system [5].

6.1 SQL Statement

We use SRD to refer to our time series cleaning method under seasonal and trend constraints. For the time series stored in Apache IoTDB, we implement SRD as a built-in function. Leveraging the query write-back mechanism of Apache IoTDB, the cleaned data can be directly computed and stored using SQL queries. For example, by executing the following SQL statement, users can obtain the SRD cleaning results for the time series root.Power with a period of 12.

```
select SRD(value, 'period'=12)
from root.Power
where time>=20 and time<=100
```

The parameter 'period' represents the period of the time series x, which can be estimated by period detection methods [9, 28, 33]. At this point, the cleaned time series x' with timestamps in the range of [20, 100] are returned.

6.2 Query Processing

Figure 7 presents the query process of SRD in IoTDB. Given that SRD is implemented as a function, as mentioned in Section 6.1, IoTDB first retrieves data points from TsFile that satisfy the filtering criteria through the FileSeriesReader interface, as shown in Figure 7. Based on the specified query path and the target file, all data points in the file for the specified time series are retrieved in ascending order of timestamps. For the iterative cleaning process, we detect constraint violations on the data $x^{(h)}$ in the h-th iteration. If violations are still present, the process proceeds to the $(h+1)$-th iteration, where $x^{(h+1)}$ overwrites $x^{(h)}$, thus incurring no additional space overhead. Figure 7 illustrates this process.

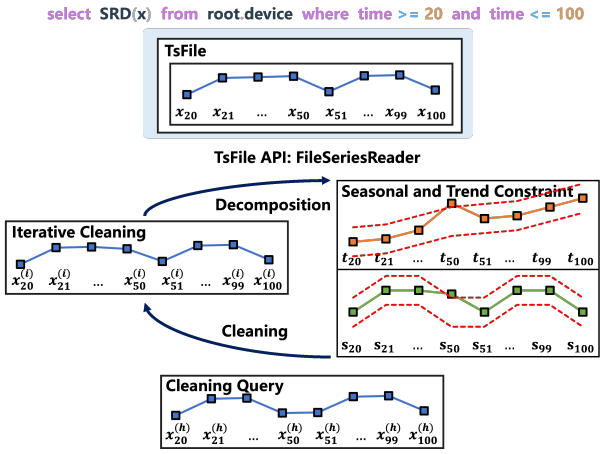


Fig. 7. Query processing of SRD in Apache IoTDB.

Table 2. Features of datasets

	Time series size	Dirty data	Number of periods
Grid	21,647	Embedded	16
Insurance	82,545	Embedded	287
Airline	37,508	Embedded	131
Telecom	49,389	Embedded	172
GPS	1,754,400	Embedded	10,442
Power	5,241,600	Clean	36,400
Voltage	22,825,440	Clean	15,851
Traffic	1,747,200	Clean	10,400
Weather	5,256,000	Clean	36,500

7 EXPERIMENT

In this section, we perform experiments to evaluate (1) the repair performance and the improvement of downstream task performance on real datasets with real violations, (2) the scalability and the repair performance on different violations on real datasets with injected errors, (3) the investigation of the proposed methods, including cleaning methods and iterations by comparing with other existing methods as baselines. The code and data of the experiments are available online for reproducibility [10].

7.1 Experimental Setup

All experiments are carried out on a computer with an Intel Core i7-9750H CPU and 16 GB of memory, running Apache IoTDB 0.9.3.

7.1.1 Datasets. We use five datasets with real-world errors naturally embedded and four real datasets with synthetic errors following the same line of [39]. We consider only time series with seasonal features. Like any other constraint-based cleaning methods [3, 25], SRD is applicable only to the data where the constraints such as functional dependencies, denial constraints [3] or speed

constraints [25] hold, instead of "all" (time series) data. Thereby, the constraint-based methods are best effort cleaning. The datasets and their properties are briefly listed in Table 2.

Grid, **Insurance**, **Airline**, and **Telecom** datasets are collected from IoT devices across different domains and contain real-world errors. However, to the best of our knowledge, no public seasonal dataset with real-world violations and ground truth exists. Therefore, we collect and annotate the **GPS** dataset around our campus buildings with real-world errors and ground truth under a bridge. **Power** dataset contains records of power consumption from three different distribution networks in Tetouan city, located in northern Morocco [22]. **Voltage** dataset contains records of electricity consumption in a house located in Sceaux [12]. Both datasets are sourced from the UCI Machine Learning Repository [29]. **Traffic** and **Weather** are widely used public datasets from [41]. We expand Power, Voltage, Traffic and Weather by 100/10 times using the method of splicing the beginning and end. The selected source datasets represent general real-world scenarios. Due to space limitations, results for the Voltage and Weather datasets are provided in the appendix [27].

7.1.2 Metrics. As shown in Table 2, some datasets employed for the evaluation contain embedded errors with positions. However, we do not know their ground truth. Therefore, we solicit the participation of domain experts to provide reference repair values.

The repair methods remove errors from dirty data x and generate repair data x'. We compare the repair data x' with clean data x^* to evaluate the performance. (1) For experiments with real errors, the original data is regarded as dirty data x due to the presence of errors, whereas the data after applied reference value substitution is considered as clean data x^*. (2) For experiments with synthetic errors, errors are intentionally injected into the original data to create dirty data x, whereas the original data without such noise is deemed as clean data x^*. Since our method makes no assumptions about error types, we inject diverse errors into the data: Gaussian noise with a few extreme values added as outliers.

Root mean squared error (RMSE) measures the difference between two series. To evaluate the performance of repair methods, we calculate the RMSE as $RMSE(x^*, x')$ where x^* denotes the clean data, and x' denotes the repair data.

7.1.3 Methods. We compare various time series cleaning methods, following the categorization of related works in Section 2. We select only one representative method from each category.

SRD is our proposed algorithm (Algorithm 1), which performs seasonal repair by capturing seasonal and trend constraints as described in Section 4.1. **SCREEN** [25] repairs time series data using speed constraints. **LsGreedy** [38] maximizes the probability of speed changes based on a modeled distribution. **IMR** [39] iteratively estimates the parameters of a time series prediction model to guide repair. **ASAP** [21] is a smoothing-based method that adaptively balances noise reduction and trend preservation. **HoloClean** [20] is a relational data cleaning framework that uses user-specified constraints, including seasonal and trend constraints, to detect errors. We also design two hybrid methods: **MP_Linear**, which combines Matrix Profile [36] for detection, based on distance spikes between subsequences, and linear interpolation for repair; and **SARIMA_Linear**, which uses SARIMA [24] to detect anomalies via thresholding the distance between predicted and observed values.

We use the default parameter settings from the original papers for all baseline methods. For example, for the pruning threshold τ in Holoclean, we set it to 0.1 according to the default value in the Holoclean source code.

7.2 Comparison on Real Violations

The experiments on real errors consist of two parts: (1) comparing the effectiveness of error repair, and (2) comparing the effectiveness of repair results on downstream data science tasks.

Table 3. RMSE of various cleaning methods over datasets with real-world dirty data

	Grid	Insurance	Airline	Telecom	GPS
SRD	**0.07**	**0.06**	**0.05**	**0.03**	**0.09**
SCREEN	0.44	0.3	0.63	0.75	0.44
LsGreedy	0.3	0.23	0.59	0.2	0.24
IMR	0.24	0.31	0.57	0.14	0.23
ASAP	0.37	0.17	0.23	0.22	0.19
HoloClean	0.2	0.12	0.17	0.16	0.18
MP_Linear	0.26	0.26	0.36	0.42	0.31
SARIMA_Linear	0.18	0.23	0.28	0.31	0.2

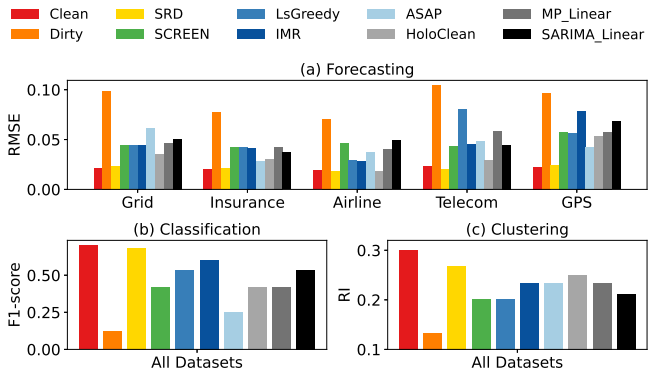


Fig. 8. Performances of applications over clean, dirty and repaired data by various approaches.

7.2.1 Repair Performance Comparison. Table 3 presents the results of various models on datasets containing real errors. Since the IMR method requires partial labeling of data, 20% of the data points are labeled, which creates an unbalanced comparison. Despite this, our method SRD demonstrates superior effectiveness. For example, in the Grid dataset with significant seasonal features, SRD achieved a more than 50% reduction in RMSE over other cleaning methods. This is because the SRD method utilizes the overall seasonal features of the data to conduct repairs, rather than restricting repairs to local data characteristics as other methods. SRD also outperforms hybrid methods such as MP_Linear, as matrix profile is sensitive to trend shifts and linear interpolation may introduce new errors. Note that SRD also achieves the best decomposition on GPS, confirming its effectiveness on real-world data with ground truth. We also find that SRD is not limited to a single data type such as IoT, and applies to any dataset with seasonal patterns.

7.2.2 Case Study on Data Science Applications. We evaluate the impact of cleaning on state-of-the-art downstream analysis. Specifically, we evaluate the performance of repaired data by domain experts as "Clean", original data with real errors as "Dirty", and repair data using different methods. Through our analysis, we provide insights into the efficacy of different data cleaning methods for improving the accuracy and reliability of data science applications.

Time Series Prediction. Time series prediction is a commonly utilized task in time series analysis. For instance, our partner factories frequently require predicting future grid loads based on the

Grid dataset. Figure 8(a) displays the prediction results by using N-Linear model [37], with RMSE serving as the evaluation metric, measuring the variance between predicted and actual values.

As expected, the performance of the Dirty dataset is poor, as errors can interfere with model training. It highlights the importance of cleaning errors before data science tasks. Compared to other methods, our proposed SRD method demonstrates significantly higher prediction accuracy. Notably, the performance of the SRD outperforms Clean on the Telecom dataset. This performance improvement indicates that the effectiveness of the SRD in cleaning errors sometimes even exceeds that of domain experts. Furthermore, SRD improves prediction performance on the GPS dataset, demonstrating its practical potential on real-world data.

Time Series Classification. According to suggestions for datasets partitioning provided by our industry partner, the Grid dataset was divided into 2 parts, while the insurance, airline, bank, and telecom datasets were divided into 11, 8, 13, and 6 parts, respectively, resulting in a total of 40 parts. Moreover, we use the dataset names as labels to perform time series classification tasks. We employed 70% of the data for training and 30% for evaluation, with the F1-score serving as the evaluation metric. Figure 8(b) presents the classification results from applying GRU-D [8], demonstrating our proposed SRD method's superior performance compared to other methods. After cleaning dirty data, it is easier to capture the similarity between time series. Furthermore, the SRD method is the most effective in making the data conform to the original seasonal features, hence it performs the best.

Time Series Clustering. We performed a clustering task on the identical 40 subsets as those utilized for classification in Section 7.2.2. We employed the Random Index (RI) as the evaluation metric and dataset names as labels. We use CRLI [17] as our clustering model. As illustrated in Figure 8(c), our method outperforms all other repair methods in enhancing clustering performance. This is due to the easier measurement of similarities between different time series as a result of error removal.

7.3 Comparison on Synthetic Violations

The experiments on synthetic violations compare (1) the scalability of the violation repair under different data sizes, and (2) the effectiveness of the repair under different violation types.

7.3.1 Scalability. Figure 9 examines scalability by varying the size of time series. MP_Linear and HoloClean require significant feature learning time, resulting in runtimes exceeding 12 hours in this experiments, significantly longer than other methods. Due to their excessive time overhead, MP_Linear and HoloClean are not suitable for large-scale data cleaning. Therefore, we exclude them from the scalability experiments. As the data size increases, the accuracy of each method generally remains stable. Not surprisingly, the proposed SRD have much lower RMSE than SCREEN, LsGreedy, and ASAP under complex error types because they ignore the seasonal patterns of the time series. SRD also outperforms seasonal methods such as SARIMA_Linear, which relies on forecasts sensitive to violations and uses linear interpolation that fails to capture seasonal shifts. In contrast, SRD adopts violation-tolerant decomposition and generates repairs from seasonal decomposition results.

The time costs of all methods are positively correlated with data size. Compared to other methods, SRD has a moderate time complexity and consumes less time than LsGreedy and IMR.

7.3.2 Period. Figure 10 examines scalability by varying the period of the time series. We adjust the period using downsampling. Overall, all methods show consistent performance across different period settings. Since larger downsampling may reduce data variability, methods such as SCREEN exhibit slightly lower time cost when the period is small.

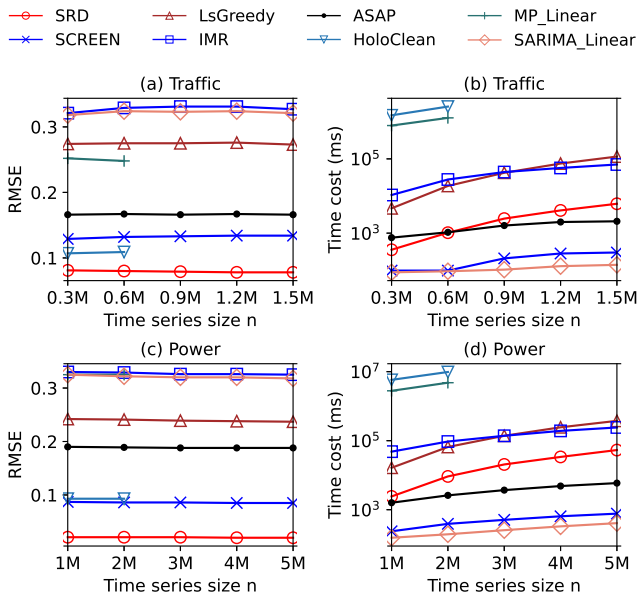


Fig. 9. Varying time series size n.

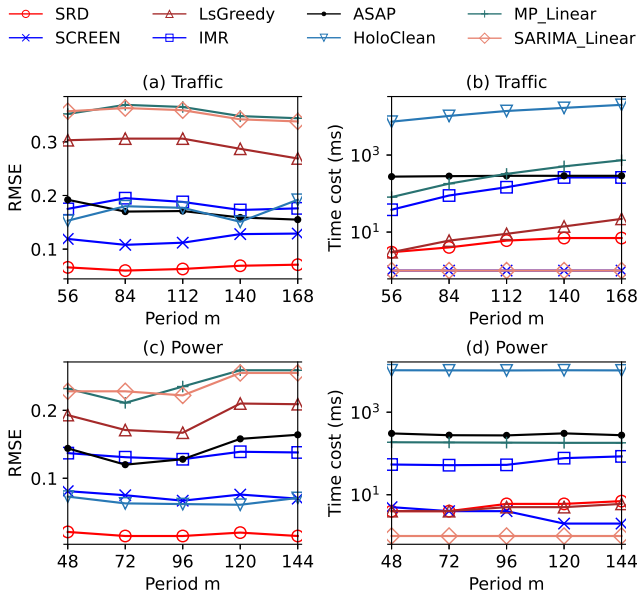


Fig. 10. Varying period m.

7.3.3 Window Size. Figure 11 explore the impact of window size w in trend filter. As the window size w increases, the performance of SRD first improves and then degrades. SRD performs best

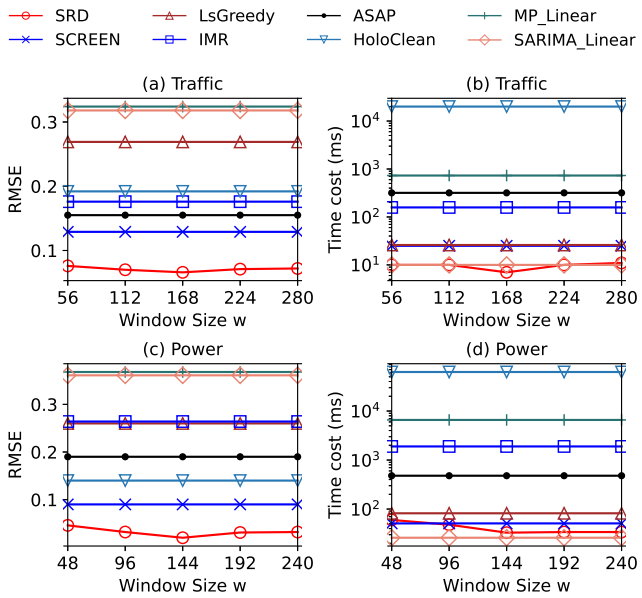


Fig. 11. Varying window size w (of SRD).

when window size w equals the period m, i.e., $w = 168$ for Traffic dataset in Figure 11(a) and $w = 144$ for Power dataset in Figure 11(c). while other methods are unaffected since they do not use this parameter.

7.3.4 Varying Error Rate. Figure 12 shows the results at different error rates. As the error rate increases, the RMSE of almost all methods increases. However, SRD exhibits low sensitivity to error rates because it effectively utilizes the median, reducing the impact of Gaussian noise with zero mean. Hence, an increase in the error rate has a minimal effect on the SRD method.

7.3.5 Varying Error Range. Error range refers to the absolute size of errors injected into the time series. In contrast to other repair methods, SRD remains almost unchanged or even decreases when the error range increases as illustrated in Figure 13. This is due to its error-tolerant design and use of means that reduce the impact of extreme values. The larger the error range, the greater the probability that the series exceeds the constraint threshold, and the more accurate the violation detection method. In contrast, existing methods such as LsGreedy train directly on original data and are sensitive to large-range errors. Furthermore, the time cost of SRD does not increase significantly with increasing error range. This is because although a broader error range may result in identifying more violations, generating repairs for them is not excessively time-consuming.

7.3.6 Varying Consecutive Errors. As for cases of consecutive violation detection, our methods still perform very well. In particular, when the length of consecutive error equals 1, this is the case of single point violation detection. Figure 14 illustrates the results on various lengths of consecutive error. In single point violation detection, LsGreedy and SCREEN output similar RMSE as SRD do. However, as the length of consecutive error increases, RMSE computed by LsGreedy and SCREEN increases significantly while RMSE computed by SRD almost keeps stable. Moreover, the time cost of SRD is not significantly impacted by longer consecutive errors, since these types of errors do not affect SRD's overall repair.

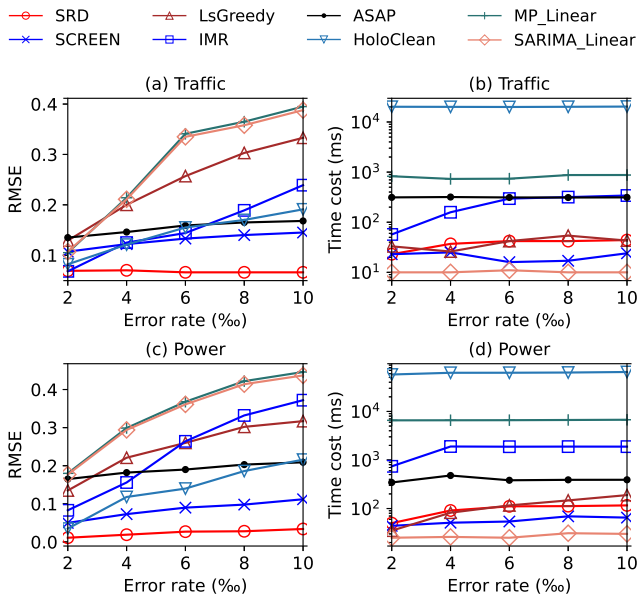


Fig. 12. Varying error rate.

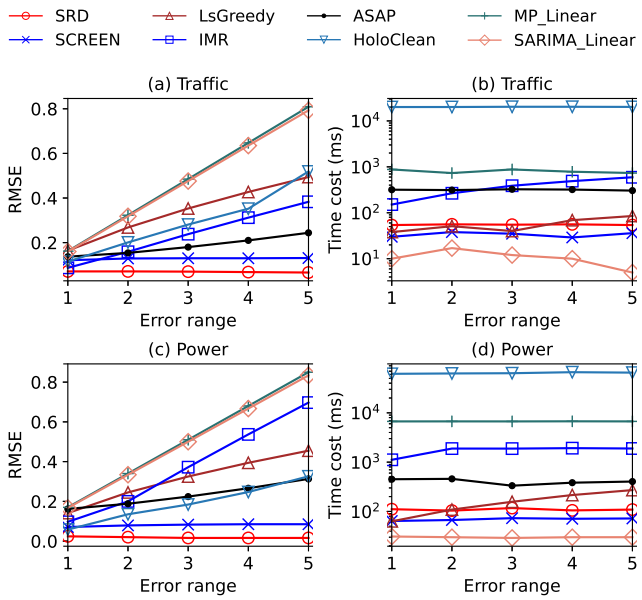


Fig. 13. Varying error range.

7.4 Evaluation of Proposed Techniques

In this section, we assess how different design choices, i.e., decomposition method, iteration strategy, and constraint thresholds, synergize to produce the observed improvements. We (1) replace

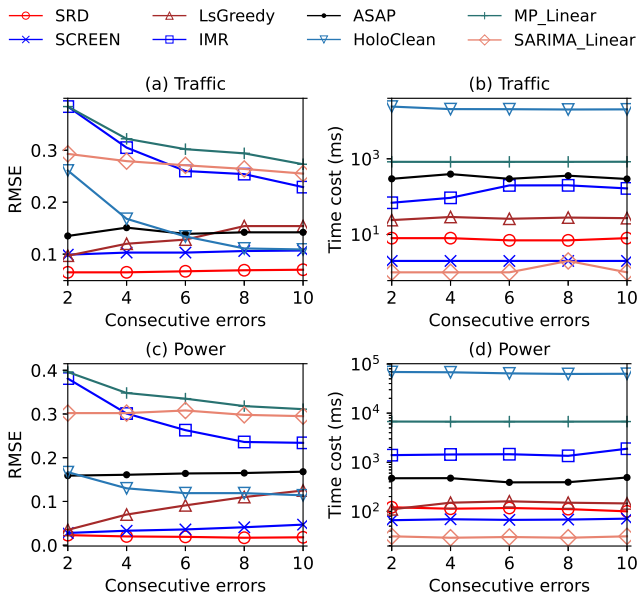


Fig. 14. Varying length of consecutive errors.

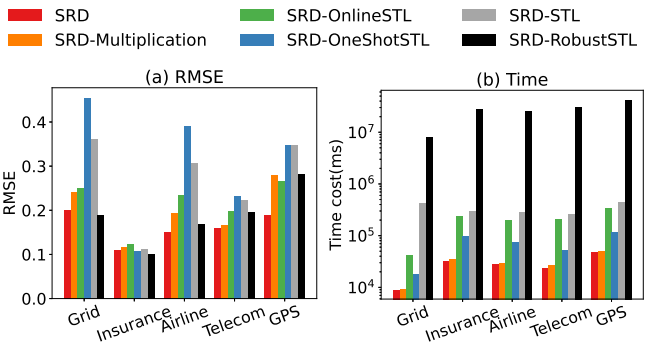


Fig. 15. SRD with various decomposition methods.

the decomposition module in SRD with different design choices in Section 7.4.1, (2) evaluate different numbers of iterations in Section 7.4.2, (3) evaluate different constraint thresholds in Section 7.4.3.

7.4.1 Impact of Decomposition Method. We evaluate the impact of different decomposition methods on the repair effect. We aim to demonstrates the contributions of different SRD components, compared to the other baseline methods, inclusing online methods OnlineSTL [18], OneShotSTL [11] and batch methods STL [4], RobustSTL [32]. Moreover, our method can be extended to support multiplicative seasonal-trend patterns by modifying the constraint extraction step (Section 4). The detection, repair, and iteration procedures remain unchanged. We compute the trend as in Equation 4, detrend via $d_i = \frac{x_i}{\tau_i}$, and estimate seasonality using Formula 5. The component extension combines trend and seasonal components multiplicatively, e.g., replacing $t_{1+\lfloor \frac{m-1}{2} \rfloor} + s_j$

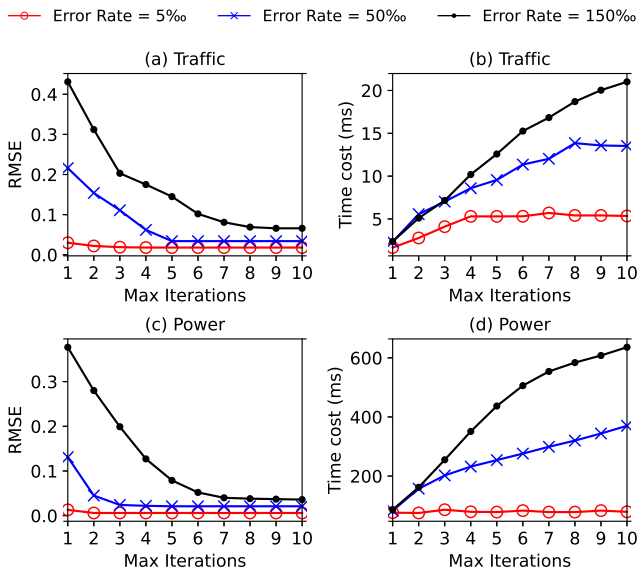


Fig. 16. SRD by varying maximum number of iterations.

with $t_{1+\lfloor \frac{m-1}{2} \rfloor} \times s_j$. We validate this method (SRD-Multiplication) below. Figure 15 shows the results, where SRD refers to the repair using violation-tolerant decomposition, SRD-OnlineSTL refers to the repair using OnlineSTL, and so on.

Figure 15(a) shows that SRD method outperforms SRD-OnlineSTL, SRD-OneShotSTL and SRD-STL in most datasets. This reflects the violation-tolerant median filter in SRD outperforms filters used in other decomposition methods, such as the tri-cube filter in OnlineSTL. Notably, real-world data is often additive, making SRD-multiplication perform worse than SRD. Note that RobustSTL, which is the method with the highest decomposition performance, can mitigate the impact of violations. Nevertheless, SRD is comparable to SRD-RobustSTL on Grid and Insurance. Figure 15(b) shows that the efficiency of SRD method is much higher than other methods on all datasets. Although SRD-RobustSTL achieved the effect only second to SRD in Figure 15(a), its decomposition efficiency is too low to be directly applied.

7.4.2 Impact of Iterations. In this section, we explore the impact of iterations in SRD by adjusting *max_iter*, which means the maximum number of iterations. Figure 16(a) and 16(c) shows that, as *max_iter* increases, the RMSE of SRD gradually decreases, because more violations are detected and repaired. However, RMSE stabilizes beyond a certain *max_iter*, indicating SRD convergence as iterations no longer increase. Figure 16(b) and 16(d) show that more iterations increase time cost, as each iteration involves full decomposition, detection, and repair. The time cost stabilizes once *max_iter* exceeds a certain point. In Figure 16(b), SRD converges in 4 iterations for the voltage dataset at a 5‰error rate, and 8 for 10‰. It validates Proposition 4, as violations are more likely to occur within fewer periods when the violation rate is low.

7.4.3 Impact of Constraint Threshold η. In this section, we explore the impact of threshold η_1 and η_2 in seasonal and trend constraint. Figure 17 shows the results. We set η_1 and η_2 as integer multiples of the standard deviation of the original data. When η_1 and η_2 are too small, even normal points may be repaired, causing over-repair. Conversely, if they are too large, real errors may be

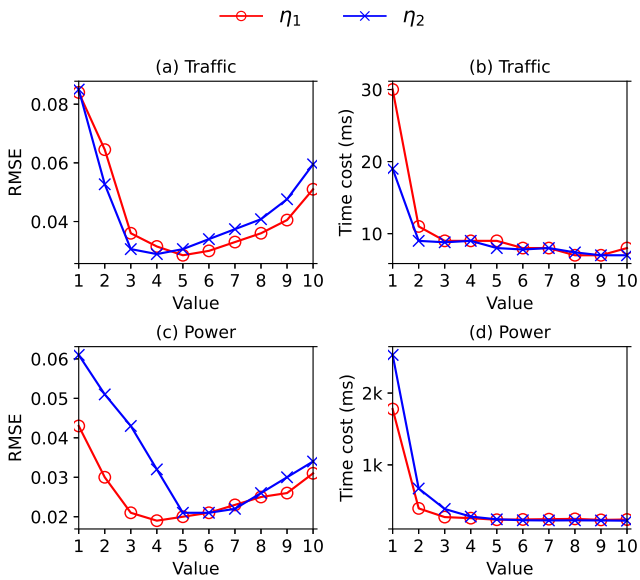


Fig. 17. SRD by varying threshold η_1 and η_2.

missed, resulting in under-repair. Proper tuning is thus essential. Note that the optimal values of η_1 and η_2 vary across datasets; for example, $\eta_1 = 4$ is optimal on Traffic, while $\eta_1 = 5$ is optimal on Power, indicating that different datasets vary in their tolerance to over- and under-estimation.

8 CONCLUSION

In this paper, we propose a novel data cleaning method SRD with seasonal and trend constraint for the time series having seasonal features. First, to capture seasonal and trend constraint, we introduce an violation-tolerant median filter that robustly extracts the long-term trend and the same-phase seasonal components. Moreover, we extend the incomplete trend components. With the remedied decomposition, violations to the seasonal and trend constraints are detected. After recognizing NP-completeness of the cleaning problem (Theorem 1), we heuristically repair the violations by referencing the seasonal and trend components of corresponding points in the same phase across cycles. Since the captured seasonal and trend constraint may vary after a round of data cleaning, we design an iterative algorithm. While the iteration of cleaning guarantees to converge in certain cases (Proposition 4), in practice, we may set a maximum number of iterations to terminate early. The proposal has now become a built-in function in Apache IoTDB, a product system of time series database. Experiments on real-world datasets show that our method outperforms existing methods in cleaning seasonal time series and improving downstream applications.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (62232005, 62021002, 92267203, 62572051), the National Key Research and Development Plan (2025ZD1601701, 2024YFB3311901, 2021YFB3300500), Beijing National Research Center for Information Science and Technology (BNR2025RC01011), Beijing Key Laboratory of Industrial Big Data System and Application, Hebei Natural Science Foundation (25130203B) and Liaoning Revitalization Talents Program (XLYC2204005). Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

REFERENCES

[1] Jaakko Astola and T. George Campbell. On computation of the running median. *IEEE Trans. Acoust. Speech Signal Process.*, 37(4):572–574, 1989.

[2] Chris Chatfield. *The analysis of time series: an introduction*. Chapman and hall/CRC, 2003.

[3] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou, editors, *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 458–469. IEEE Computer Society, 2013.

[4] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *J. Off. Stat*, 6(1):3–73, 1990.

[5] Code. https://github.com/czjthss/iotdb/tree/research/seasonal-repair, 2025.

[6] Document. https://iotdb.apache.org/UserGuide/V1.1.x/Operators-Functions/Data-Repairing.html#seasonalrepair, 2025.

[7] Alexander Dokumentov and Rob J Hyndman. Str: Seasonal-trend decomposition using regression. *arXiv preprint arXiv:2009.05894*, 2020.

[8] Wenjie Du. Pypots: A python toolbox for data mining on partially-observed time series. *CoRR*, abs/2305.18811, 2023.

[9] Mohamed G. Elfeky, Walid G. Aref, and Ahmed K. Elmagarmid. Periodicity detection in time series databases. *IEEE Trans. Knowl. Data Eng.*, 17(7):875–887, 2005.

[10] Experiment. https://github.com/czjthss/SeasonalClean, 2025.

[11] Xiao He, Ye Li, Jian Tan, Bin Wu, and Feifei Li. Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting. *Proc. VLDB Endow.*, 16(6):1399–1412, 2023.

[12] Bezzar Nour El Houda, Laimeche Lakhdar, and Abdallah Meraoumia. Time series analysis of household electric consumption with xgboost model. In *4th International Conference on Pattern Analysis and Intelligent Systems, PAIS 2022, Oum El Bouaghi, Algeria, October 12-13, 2022*, pages 1–6. IEEE, 2022.

[13] J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986.

[14] Apache IoTDB. https://iotdb.apache.org, 2025.

[15] Song Jiang, Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, and Yizhou Sun. Bridging self-attention and time series decomposition for periodic forecasting. In Mohammad Al Hasan and Li Xiong, editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 3202–3211. ACM, 2022.

[16] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[17] Qianli Ma, Chuxin Chen, Sen Li, and Garrison W. Cottrell. Learning representations for incomplete time series clustering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 8837–8846. AAAI Press, 2021.

[18] Abhinav Mishra, Ram Sriharsha, and Sichen Zhong. Onlinestl: Scaling time series decomposition by 100x. *Proc. VLDB Endow.*, 15(7):1417–1425, 2022.

[19] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[20] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, 2017.

[21] Kexin Rong and Peter Bailis. ASAP: prioritizing attention via time series smoothing. *Proc. VLDB Endow.*, 10(11):1358–1369, 2017.

[22] Abdulwahed Salam and Abdelaaziz El Hibaoui. Comparison of machine learning algorithms for the power consumption prediction:-case study of tetouan city–. In *2018 6th International Renewable and Sustainable Energy Conference (IRSEC)*, pages 1–5. IEEE, 2018.

[23] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, 2022.

[24] Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2006.

[25] Shaoxu Song, Aoqian Zhang, Jianmin Wang, and Philip S. Yu. SCREEN: stream data cleaning under speed constraints. In Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 827–841. ACM, 2015.

[26] William E. Strawderman. Statistical analysis with missing data (roderick j. a. little and donald b. rubin). *SIAM Rev.*, 31(2):348–349, 1989.

[27] Supplementary. https://github.com/czjthss/SeasonalClean/blob/main/supplementary.pdf, 2025.

[28] Maximilian Toller, Tiago Santos, and Roman Kern. SAZED: parameter-free domain-agnostic season length estimation in time series data. *Data Min. Knowl. Discov.*, 33(6):1775–1798, 2019.
[29] UCI. https://archive.ics.uci.edu/, 2025.
[30] Chen Wang, Jialin Qiao, Xiangdong Huang, Shaoxu Song, Haonan Hou, Tian Jiang, Lei Rui, Jianmin Wang, and Jiaguang Sun. Apache iotdb: A time series database for iot applications. *Proc. ACM Manag. Data*, 1(2):195:1–195:27, 2023.
[31] Xi Wang and Chen Wang. Time series data cleaning: A survey. *IEEE Access*, 8:1866–1881, 2020.
[32] Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5409–5416. AAAI Press, 2019.
[33] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. Robustperiod: Robust time-frequency mining for multiple periodicity detection. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2328–2337. ACM, 2021.
[34] Qingsong Wen, Zhe Zhang, Yan Li, and Liang Sun. Fast robuststl: Efficient and robust seasonal-trend decomposition for time series with complex patterns. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 2203–2213. ACM, 2020.
[35] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22419–22430, 2021.
[36] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1317–1322. IEEE Computer Society, 2016.
[37] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 11121–11128. AAAI Press, 2023.
[38] Aoqian Zhang, Shaoxu Song, and Jianmin Wang. Sequential data cleaning: A statistical approach. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 909–924. ACM, 2016.
[39] Aoqian Zhang, Shaoxu Song, Jianmin Wang, and Philip S. Yu. Time series data cleaning: From anomaly detection to anomaly repairing. *Proc. VLDB Endow.*, 10(10):1046–1057, 2017.
[40] Xinyan Zhang, Dan Feng, Zhipeng Tan, Yanwen Xie, Shaofeng Zhao, and Ya-Yuan Wei. LWCM: A lookahead-window constrained model for disk failure prediction in large data centers. *J. Comput. Sci. Technol.*, 40(3):748–765, 2025.
[41] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 11106–11115. AAAI Press, 2021.
[42] Jingyu Zhu, Xintong Zhao, Yu Sun, Shaoxu Song, and Xiaojie Yuan. Relational data cleaning meets artificial intelligence: A survey. *Data Sci. Eng.*, 10(2):147–174, 2025.

Received April 2025; revised July 2025; accepted August 2025