SHoTClean: Bridging Soft and Hard Constraints for Multivariate Time Series Cleaning

ZIQUAN FANG, School of Software, Zhejiang University, China
WEI SHAO, School of Software, Zhejiang University, China
ZHEQI LU, School of Software, Zhejiang University, China
LU CHEN, College of Computer Science and Technology, Zhejiang University, China
YUNJUN GAO, College of Computer Science and Technology, Zhejiang University, China

Time series data frequently suffer from data quality problems during collection and transmission, such as small-jump dirty points, which existing cleaning methods often fail to detect. Since existing methods primarily address univariate series, their multivariate extensions often fail to capture complex inter-variable dependencies, significantly limiting their effectiveness. To this end, we propose SHoTClean, a family of four algorithms that bridges hard constraints (i.e., physical limits) and soft constraints (i.e., statistical patterns) within a constrained-optimization framework for effective and efficient multivariate time series cleaning.

Specifically, we formulate the cleaning task as minimizing soft-constraint violations while respecting hard-constraint bounds. Then, we propose SHoTClean that introduces: (1) SHoTClean-B for offline batch processing using pruned dynamic programming to achieve global optimality; (2) SHoTClean-S and SHoTClean-P for online streaming scenarios by employing incremental dynamic programming, where SHoTClean-P accelerates SHoTClean-S via CDQ divide-and-conquer and Fenwick tree to attain near-linear complexity; and (3) SHoTClean-C, incorporating causal discovery into soft constraints to capture multivariate dependencies. Extensive experiments across 12 real-world datasets demonstrate that our approaches achieve i) 6.8%–90.0% and 7.8%–82.1% improvements in accuracy (RMSE metric) over 10 state-of-the-art baselines in offline and online settings, respectively; ii) an average two-order-of-magnitude runtime speed-up on large-scale datasets; and iii) superior robustness, with consistent high performance under extreme 80% contamination level and high-dimensional datasets. The code is available at https://github.com/ZJU-DAILY/SHoTClean.

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

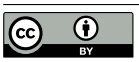
Additional Key Words and Phrases: multivariate time series, data cleaning, data quality

ACM Reference Format:
Ziquan Fang, Wei Shao, Zheqi Lu, Lu Chen, and Yunjun Gao. 2026. SHoTClean: Bridging Soft and Hard Constraints for Multivariate Time Series Cleaning . *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 84 (February 2026), 27 pages. https://doi.org/10.1145/3786698

1 Introduction

The proliferation of IoT devices and dense sensor networks has generated large-scale and multivariate time series data [41, 56], which supports a wide range of real-life applications, including financial forecasting [61], traffic safety [49], industrial equipment monitoring [42], as well as health and wellness tracking [33]. However, multivariate time series are inherently vulnerable to various

Authors' Contact Information: Ziquan Fang, zqfang@zju.edu.cn, School of Software, Zhejiang University, Ningbo, Zhejiang, China; Wei Shao, wei.shao@zju.edu.cn, School of Software, Zhejiang University, Ningbo, Zhejiang, China; Zheqi Lu, geogelu@zju.edu.cn, School of Software, Zhejiang University, Ningbo, Zhejiang, China; Lu Chen, luchen@zju.edu.cn, College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, China; Yunjun Gao, gaoyj@zju.edu.cn, College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, China.



ACM 2836-6573/2026/2-ART84
https://doi.org/10.1145/3786698

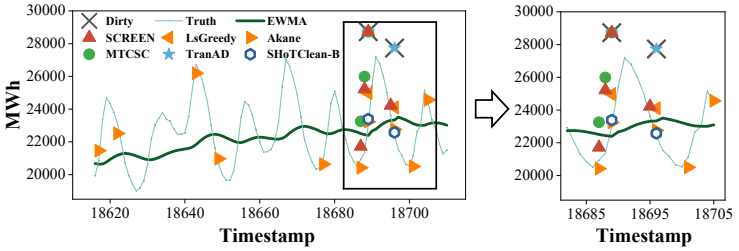


Fig. 1. Existing Multivariate Time Series Cleaning

quality issues [55] during acquisition and transmission, such as sensor noise, transient outliers, network latency, etc. These imperfections can accumulate, compromising data quality and weakening the reliability and robustness of downstream analyses and machine learning models, underscoring the urgent need for effective and robust multivariate time series cleaning solutions.

We review existing time series cleaning methodologies, which can be systematically categorized into five paradigms, each with inherent limitations. (i) *Smoothing-based methods* such as Moving Average (MA) [13], AutoRegressive (AR) [57], Kalman filters [34], and Exponentially Weighted Moving Averages (EWMA) [29], suppress high-frequency dirty points through global processing. However, they often "over-clean" the series and distort its original distribution. (ii) *Constraint-based methods* introduce prior rules like speed limits [28, 51] and acceleration bounds [50] to clean data points that violate predefined constraints. However, their reliance on fixed thresholds makes them insensitive to subtle dirty points, such as small-dirty jumps [27], whose deviations remain within permitted bounds. (iii) *Statistical-based methods* including LSGreedy [59], DP/DPC [54], and Akane [31] employ probabilistic modeling of feature distributions, but their univariate formulations fundamentally cannot capture multivariate dependencies, and their repair strategies often struggle with large spikes. (iv) *Hybrid methods* such as MTCSC [60] and Clean4MTS [20] integrate constraint rules and statistical information to achieve fine-grained cleaning, but they typically address only one aspect—either accurate detection or precise repair—rather than achieving both capabilities within a unified framework. (v) *Anomaly–detection–based methods*, such as TranAD [52] and ImDiffusion [18], utilize deep learning to detect deviations from learned patterns. Yet, they often exhibit low sensitivity to isolated dirty points and offer limited interpretability regarding the detection. We demonstrate these fundamental limitations through the following examples.

Example 1. *Fig. 1 plots real hourly energy consumption data from California [14] and the cleaning results under different time-series cleaning methods. Based on this, we yield the following observations.*

i) From a global perspective, the ***smoothing-based*** *EWMA [29] enforces overall smoothness by uniformly adjusting every data point, regardless of whether it is actually corrupted. Similarly, Akane [31] fits the entire distribution and modifies even valid samples. Both approaches violate the minimum change principle [9], which advocates repairing time series data with the fewest possible modifications, thereby significantly increasing repair costs.*

ii) From a local perspective, ***constraint-based*** *SCREEN [51] and* ***hybrid*** *MTCSC [60] fail to detect small jumps (e.g., dirty point 1) that fall within acceptable speed limits. Instead, they overcompensate by stretching adjacent fluctuations, introducing additional error. Besides,* ***statistical-based*** *LsGreedy [59] relies heavily on local distributions and is too conservative to adequately repair large spikes (e.g., dirty point 2), resulting in substantial residual error. Similarly,* ***anomaly-detection-based*** *TranAD [52] struggles to detect such isolated dirty points and therefore perform no cleaning action.*

Goals. As smoothing methods tend to over-clean and anomaly-detection methods struggle to clean, there is an urgent need to bridge hard constraints with soft statistical modeling for multivariate time series cleaning to address the limitations illustrated in Fig. 1. **G1:** We target leveraging prior knowledge and pre-defined rules (e.g., speed limits) to enable precise and efficient

repair of large spikes, while utilizing statistical models to accurately detect small jumps. The method should conform to the minimum change principle [9], thereby safeguarding data quality and preserving the intrinsic fluctuations of the original distribution. **G2:** We target supporting both batch processing of historical data and low-latency processing of streaming data, to meet the dual (offline and online) demands of real-world applications. **G3:** We target modeling co-variations and uncovering latent causal relationships in the multivariate space to enhance robustness and accuracy, rather than simply mining linear relationships. This is because, in multi-sensor, parallel-acquisition scenarios, time series signals exhibit complex interdependencies and couplings.

However, achieving these three goals requires overcoming two following major challenges.

Challenge 1: Unified Integration of Hard Constraints and Soft Statistical Modeling. The fundamental tension between hard constraints and statistical approaches presents three key challenges that existing methods fail to adequately address. First, current methods tend to adopt a unidirectional strategy—using one paradigm to enhance the strengths of the other without directly addressing its limitations. For example, MTCSC [60] employs clustering merely to refine constraint-based repairs rather than developing true bidirectional integration. As demonstrated in **Example 1**, such approaches remain fundamentally limited by their constraint-driven detection mechanisms, failing to detect small jumps. Second, hard constraints induce non-convex feasible spaces that often exclude statistically optimal solutions, while purely statistical methods frequently propose repairs that violate basic physical plausibility. This creates an optimization dilemma where satisfying one requirement may necessitate violating the other. Third, these two paradigms operate at different temporal scales: hard constraints typically govern local point-to-point relationships (e.g., maximum speed between adjacent samples), while statistical models capture global distributional properties and long-range dependencies. Overall, bridging hard constraints and soft statistical modeling requires new frameworks that can address the above three challenges.

Challenge 2: Effective and Efficient Multivariate Dependency Modeling. Most existing methods [28, 31, 50, 51, 59] target univariate time series processing. Even the limited set of multivariate approaches typically leverages dimensional information only superficially, failing to capture the complex dependencies among variables. For instance, Cleanits [23] employs Pearson correlation and similar metrics to assess inter-variable relations, but these methods merely reflect co-movement trends and overlook latent factors. Clean4MTS [20] and MTSClean [21] incorporate row-level constraints based on the predefined functions, which can capture only linear correlations and fall short of modeling nonlinear or higher-order interactions. However, variable dependencies are often multi-stage, conditionally dependent, or even causal. For example, an anomaly in one channel may stem from a delayed response in another—a relation that static metrics fail to characterize. On the other hand, as both dataset size and dimensionality grow, many existing methods suffer from the curse of dimensionality [5, 40], leading to substantial computational overhead. Our evaluation (Fig. 5) indicates that current approaches exhibit at least quadratic complexity growth with dimensionality, rendering them impractical for efficient processing of large-scale high-dimensional datasets. Overall, how to fully exploit inter-variable relations in multivariate time series while maintaining high computational efficiency is another unaddressed challenge.

To address the above challenges, we propose **SHoTClean**, a unified framework that bridges **s**oft and **h**ard c**o**nstraints for effective and efficient multivariate **t**ime series **clean**ing. To overcome **Challenge 1**, SHoTClean formulates statistical deviations as *soft constraints* and incorporates physical rules as *hard constraints*, casting the problem as one of ***minimizing soft-constraint penalties within the feasible region defined by hard constraints***. Based on this, we design a dynamic programming-based path search with exponential time decay to detect both large spikes and small jumps. Then, the detected dirty points are repaired via linear interpolation, in adherence to the minimum change principle [9], which ensures preservation of genuine signal

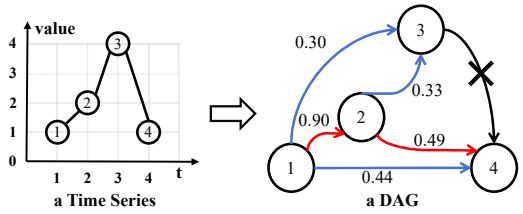


Fig. 2. An Example of Converting a Time Series into a DAG

fluctuations. We further introduce **SHoTClean-B** for offline batch processing and **SHoTClean-S** for online streaming, respectively. The latter is accelerated using a divide-and-conquer strategy [16] and a binary-indexed tree [26], resulting in **SHoTClean-P** for more efficient cleaning. To tackle **Challenge 2**, we develop **SHoTClean-C**, which captures complex and potentially asynchronous causal dependencies using the PCMCI algorithm [46]. By training a lightweight causal model offline to predict residuals and integrating these predictions into the soft-constraint framework, SHoTClean-C models higher-order inter-variable interactions with only near-linear computational overhead. Overall, this paper makes contributions below.

- We **formulate time series cleaning** under both soft and hard constraints as **constrained optimization**, and design a suite of four algorithms for multivariate time series—tailored respectively for batch processing and streaming scenarios (Section 2).
- For **batch processing**, we propose **SHoTClean-B**, a dynamic programming-based algorithm that significantly reduces the time complexity of dirty-point detection—from exponential to linear—rather than relying on off-the-shelf solvers (Section 3).
- For **streaming processing**, we develop **SHoTClean-S**, an online and incremental dynamic programming method that supports real-time cleaning. To further enhance efficiency, we introduce **SHoTClean-P**, which combines a divide-and-conquer strategy with a binary-indexed tree, reducing the overall complexity from quadratic to near-linear (Section 4).
- To **capture multivariate dependencies**, we further propose **SHoTClean-C**, which embeds causal consistency into soft constraints by predicting causal residuals. This enables the modeling of higher-order interactions among variables. With offline training, SHoTClean-C achieves near-linear inference complexity during online deployment (Section 5).
- Extensive experiments on **12 real-world datasets** show that SHoTClean consistently outperforms **10 state-of-the-art methods** in accuracy, efficiency, and robustness (Section 6).

2 Problem Statement

2.1 Preliminaries

Definition 1 (D-Dimensional Time Series). A time series is an ordered sequence of observations indexed by discrete timestamps, denoted $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(D)})$ is a D-dimensional vector, and $x_i^{(d)}$ represents the value of the d-th attribute at timestamp t_i.

Definition 2 (Hard Constraint). Hard constraints (HC) are feasibility conditions that must be strictly satisfied. A solution violating any hard constraint is invalid. In this paper, we define hard constraints as **speed bounds** between pairs of observations:

$$\forall 1 \le j < i \le n: \quad V_{\min} \le \frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{t_i - t_j} \le V_{\max}, \tag{1}$$

where $V_{\min}$ and $V_{\max}$ are user-defined minimum and maximum speeds, respectively. $||\mathbf{x}_i - \mathbf{x}_j||_2$ is the Euclidean distance between D-dimensional vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ (per Definition 1).

Note that, we denote $\mathbf{x}_i \models \text{HC}$ to indicate that the point $\mathbf{x}_i$ satisfies the hard constraint. Throughout this paper, HC specifically refers to these speed bounds. All algorithms, analyses, and experiments are instantiated under this setting.

Definition 3 (Soft Constraint). Soft constraints are probabilistic measures of data quality that tolerate violations but penalize deviations from expected patterns. Each observation $\mathbf{x}_i$ is assigned a score $s_i \in (0, 1]$ reflecting its conformance to statistical norms:

$$s_i = \exp\left(-\alpha \cdot \left\| \frac{\mathbf{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right\|_2\right), \tag{2}$$

where $\boldsymbol{\mu} = (\mu^{(1)}, \ldots, \mu^{(D)})$ and $\boldsymbol{\sigma} = (\sigma^{(1)}, \ldots, \sigma^{(D)})$ are mean and standard deviation vectors estimated from available (potentially dirty) data. $\alpha > 0$ is a sensitivity parameter controlling penalty steepness, and L_2-norm $||\cdot||_2$ operates on normalized deviations across all dimensions.

To account for temporal decay in dependency strength, we extend s_i with a time-weighted factor $\gamma(\text{gap})$ as defined below:

$$\gamma(\text{gap}) = \exp(-\beta \cdot \text{gap}), \tag{3}$$

where $\text{gap} = t_i - t_j$ is the time interval between i) current timestamp t_i and ii) timestamp t_j of its nearest valid predecessor $\mathbf{x}_j$ satisfying $\mathbf{x}_j \models \text{HC}$. Since $t_i > t_j$, we have $\text{gap} > 0$. The parameter $\beta > 0$ is the decay rate controlling how rapidly influence diminishes over time.

Definition 4 (Data Repair). Given a dirty point $\mathbf{x}_i$, its repaired version $\mathbf{x}'_i$ is obtained via temporal linear interpolation between its nearest valid neighbors:

$$\mathbf{x}'_i = \mathbf{x}_j + \frac{t_i - t_j}{t_k - t_j} \cdot (\mathbf{x}_k - \mathbf{x}_j), \tag{4}$$

where $\mathbf{x}_j$ (preceding) and $\mathbf{x}_k$ (succeeding) are the nearest points satisfying $\mathbf{x}_j, \mathbf{x}_k \models \text{HC}$ (Definition 2). Besides, the repair preserves hard constraints: $\mathbf{x}'_i \models \text{HC}$ w.r.t. adjacent points.

2.2 Problem Definition

Minimum Change Principle. Referring to previous studies [20, 21, 31], we adopt the widely used minimum change principle [9], which seeks the smallest possible modifications between the original series $\mathbf{X}$ and the cleaned series $\mathbf{X}'$. Then, we measure the total repair magnitude:

$$\Delta(\mathbf{X} - \mathbf{X}') = ||\mathbf{X} - \mathbf{X}'||_2 = \sqrt{\sum_{i=1}^{n} \sum_{d=1}^{D} \left(x_i^{(d)} - x_i'^{(d)}\right)^2}. \tag{5}$$

Problem Formulation. Given an original time series $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, we seek a cleaned version $\mathbf{X}' = (\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_n)$ that satisfies the following optimization criteria: i) **Hard Constraint Satisfaction:** Every data point $\mathbf{x}'_i$ in $\mathbf{X}'$ must strictly adhere to predefined domain-specific constraints (Definition 2); ii) **Soft Constraint Optimization:** Among all feasible solutions that satisfy HC, $\mathbf{X}'$ should maximize a weighted soft constraint score (Definition 3); and iii) **Minimal Deviation:** The cleaned series $\mathbf{X}'$ should remain as close as possible to the original $\mathbf{X}$ (Eq. 5).

$$\begin{aligned} \min_{\mathbf{X}'} \quad & \Delta(\mathbf{X}, \mathbf{X}') \\ \text{subject to} \quad & \mathbf{x}'_i \models \text{HC}, \quad \forall i \in \{1, 2, \ldots, n\}, \\ & \mathbf{X}' = \arg\max_{\mathbf{X}'} \sum_{i=1}^{n} \Big[s_i \cdot \gamma(\text{gap}) \Big]. \end{aligned} \tag{6}$$

2.3 Problem Solution

To solve the above problem (Eq. 6), it requires selecting a subsequence of valid points that simultaneously satisfies all hard constraints while maximizing the aggregated soft score, with remaining points repaired through interpolation to minimize total error. While a brute-force approach enables

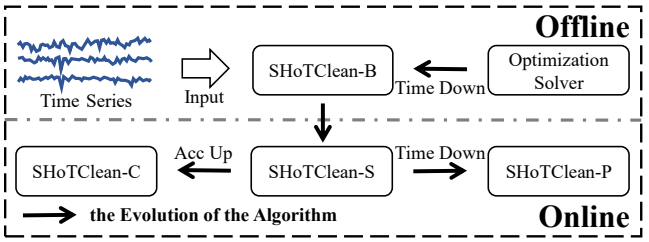


Fig. 3. The Proposed SHoTClean Algorithm Series

evaluating all possible subsequences, it would incur prohibitive exponential complexity. In contrast, we yield two key properties that enable an efficient solution:

- **Binary selection**: Each time step's inclusion represents a discrete decision point.
- **Optimal substructure**: Solutions decompose into independent subproblems [7].

Inspired by that, we formulate a directed acyclic graph (DAG) problem where: i) nodes i correspond to time indices t_i where $\mathbf{x}_i \models \text{HC}$ and ii) edges $j \to i$ connect valid transitions weighted by $w(j,i) = s_i \cdot \gamma(t_i - t_j)$. Consequently, the optimization (Eq. 6) thus reduces to ***finding the maximum-weight path in the DAG.*** We next formally demonstrate the equivalence between these two problems through the following proof.

Theorem 2.1. *(Optimality of DAG Solution). Let $\mathcal{P}^*$ be the maximum-weight path in the constructed DAG. Then the corresponding repaired time series $\mathbf{X}'_{\mathcal{P}^*}$ is exactly the optimal solution of* $\arg\min_{\mathbf{X}'} \Delta(\mathbf{X}, \mathbf{X}')$.

Proof. We prove the theorem through three steps below.

i) Bijection between Paths and Cleaned Sequences. Let $\mathcal{P} = (i_0, i_1, \ldots, i_m)$ be any path in the DAG, give the cleaned sequence $\mathbf{X}'_{\mathcal{P}}: \ \forall j \in (i_{k-1}, i_k),\ \mathbf{x}'_j = \mathbf{x}_{i_{k-1}} + \frac{j - i_{k-1}}{i_k - i_{k-1}}(\mathbf{x}_{i_k} - \mathbf{x}_{i_{k-1}})$. Conversely, any $\mathbf{X}'$ satisfying the hard constraint determines a unique path $\mathcal{P}_{\mathbf{X}'}$. Thus $\{\text{feasiable } \mathbf{X}'\} \longleftrightarrow \{paths\ \mathcal{P}\}$.

ii) Monotonicity of Cost and Weight. Define the interpolation error for any segment (i_{k-1}, i_k) is $\Delta(i_{k-1}, i_k) = \sum_{j=i_{k-1}+1}^{i_k - 1} ||\mathbf{x}_j - \mathbf{x}'_j||_2$, so the cost of $\mathcal{P}$ is $\Delta(\mathbf{X}, \mathbf{X}'_{\mathcal{P}}) = \sum_{k=1}^{m} \Delta(i_{k-1}, i_k)$. Similarly, the weight of $\mathcal{P}$ is $\mathcal{W}(\mathcal{P}) = \sum_{k=1}^{m} s_{i_k} \cdot \gamma(i_k - i_{k-1})$, with $s_{i_k} > 0$ and $\gamma(\cdot)$ strictly decreasing.

For any $a < b < c$, $\Delta(a,c) \geq \Delta(a,b) + \Delta(b,c)$, and $\Delta\mathcal{W} = s_b \cdot \gamma(b-a) + s_c \cdot \gamma(c-b) - s_c \cdot \gamma(c-a) > 0$. Hence inserting an anchor b decreases total cost and strictly increases total weight, e.g., $\Delta(\mathbf{X}, \mathbf{X}'_{\mathcal{P}_{+b}}) < \Delta(\mathbf{X}, \mathbf{X}'_{\mathcal{P}})$ and $\mathcal{W}(\mathcal{P}_{+b}) > \mathcal{W}(\mathcal{P})$.

iii) Equivalence of Objectives. Let $\mathcal{P}^*$ be a maximum-weight path. If there were $\hat{\mathcal{P}}$ with $\Delta(\mathbf{X}, \mathbf{X}'_{\hat{\mathcal{P}}}) < \Delta(\mathbf{X}, \mathbf{X}'_{\mathcal{P}^*})$, the step (2) would force $\mathcal{W}(\hat{\mathcal{P}}) > \mathcal{W}(\mathcal{P}^*)$, contradicting the maximality of $\mathcal{P}^*$. Similarly, any strictly higher-weight path than $\mathcal{P}^*$ would necessarily yield a strictly lower Δ, which is likewise impossible. Therefore, $\arg\min_{\mathbf{X}'} \Delta(\mathbf{X}, \mathbf{X}') = \{\mathbf{X}'_{\mathcal{P}} \mid \mathcal{P} \in \arg\max_{\mathcal{P}'} \mathcal{W}(\mathcal{P}')\}$. □

Example 2. *Consider the time series $\{1, 2, 4, 1\}$ observed at time indices $t = \{1, 2, 3, 4\}$, under speed bounds $V_{\min} = -2$ and $V_{\max} = 2$, a deviation steepness parameter $\alpha = 1$, and a temporal-decay factor $\beta = 0.1$. Fig. 2 illustrates the conversion of the time series into a DAG. The valid paths with both endpoints satisfying the hard constraints are highlighted in blue, while the maximum-weight path is highlighted in red. The optimal path traverses nodes $\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_4$, excluding node $\mathbf{x}_3$, which is thus identified as a dirty point requiring cleaning.*

To formally address this problem, we propose the **SHoTClean** series in Fig. 3. As observed, there are four specialized algorithms for multivariate time-series cleaning—each targeting distinct computational scenarios based on our constrained optimization formulation.

- **SHoTClean-B**: An offline batch algorithm employing pruned dynamic programming to achieve global optimality (Section 3).

- **SHoTClean-S/P**: Online streaming variants utilizing incremental processing with near-linear complexity (Section 4).
- **SHoTClean-C**: Online streaming variant employing causal modeling, designed **solely** for multivariate datasets (Section 5).

3 Offline Setting

3.1 Optimization Solver

The problem admits a natural formulation as a binary mixed-integer linear program (MILP), where all fundamental decisions–including node selection, edge activation, and path initialization–are represented through binary variables. The objective function and structural constraints can be expressed exactly using linear expressions, maintaining the problem's intrinsic combinatorial nature. As demonstrated in Algorithm 1, we solve this formulation to global optimality using off-the-shelf MILP solvers.

The algorithm begins by computing soft constraint scores for each data point using the statistical parameters (line 1). Following this initialization, we systematically examine all temporal point pairs (j, i) where $j < i$ (lines 2–5). For each pair, we verify the hard constraint condition. Valid transitions are recorded in adjacency lists, storing predecessors and successors. The optimization model employs three binary decision variables b_i, r_i, and $e_{j,i}$ (line 6), representing node selection, root indicator, and edge activation, respectively. Then, the objective function and four types of constraints are constructed (line 7) to ensure feasibility and consistency.

- Constraint (i): If node $\mathbf{x}_i$ is selected ($b_i = 1$), it must either be the root node ($r_i = 1$) or have exactly one incoming edge be activated ($\sum_j e_{j,i} = 1$).

Algorithm 1: Data Cleaning with Optimization Solver

Input: time series $\mathbf{X}$, speed limits $V_{\min}, V_{\max}$, deviation steepness parameter α, time-decay factor β.
Output: cleaned time series $\mathbf{X}'$.

1 $s[i] \leftarrow \exp(-\alpha \cdot \left\| \frac{\mathbf{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right\|_2)$;
2 **for** $i \leftarrow 1$ **to** n **do**
3 **for** $j \leftarrow i-1$ **to** 1 **do**
4 **if** $\frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{t_i - t_j} \in [V_{\min}, V_{\max}]$ **then**
5 $preds[i]$.append(j), $succs[j]$.append(i);
6 Define binary decision variables: $b_i, r_i, e_{j,i}$;
7 Maximize

$$\sum_i (s[i] \cdot r_i) + \sum_{(j,i)} (s[i] \cdot \gamma(i-j) \cdot e_{j,i})$$

subject to

$$\text{(i)} \sum_{j \in preds[i]} e_{j,i} + r_i = b_i, \quad \text{(ii)} \sum_i r_i = 1, \quad \text{(iii)} \ e_{j,i} \le b_j, e_{j,i} \le b_i, \quad \text{(iv)} \sum_{k \in succs[i]} e_{i,k} \le 1.$$

8 CBC.solve();
9 **for** $i \leftarrow 1$ **to** n **do**
10 **if** $b_i = 0$ **then**
11 $p_0 \leftarrow$ nearest preceding normal point;
12 $p_1 \leftarrow$ nearest succeeding normal point;
13 $\mathbf{x}'_i \leftarrow \mathbf{x}_{p_0} + \frac{t_i - t_{p_0}}{t_{p_1} - t_{p_0}} \cdot (\mathbf{x}_{p_1} - \mathbf{x}_{p_0})$;
14 **return** $\mathbf{X}' = \{< t_i, \mathbf{x}'_i > \mid i = 1, \ldots, n\}$

- Constraint (ii): Ensure that there is only one root node in the entire path.
- Constraint (iii): If an edge $(j \leftarrow i)$ is activated ($e_{j,i} = 1$), both nodes $\mathbf{x}_j$ and $\mathbf{x}_i$ must be selected to be part of the path.
- Constraint (iv): Each node activates at most one outgoing edge to prevent the path from branching.

Finally, once the model is constructed, we solve the Binary MILP model using the CBC solver (line 8). After the solution is obtained (lines 9–13), for all dirty points that are not selected (i.e., $b_i = 0$), we retrieve the nearest forward and backward normal points and perform linear interpolation based on the time ratio to repair them. This results in the final cleaned time series $\mathbf{X}'$.

Complexity Analysis. The CBC solver's branch-and-bound approach exhibits exponential time complexity in the worst case ($O(2^n)$), as each branching operation generates two new subproblems to explore. While cutting-plane methods provide some computational relief by pruning infeasible regions of the solution space, their effectiveness remains fundamentally constrained by problem-specific characteristics. As a result, the inherent limitations of this approach render it computationally prohibitive for large-scale time series applications, motivating our development of more efficient alternatives in subsequent sections.

3.2 SHoTClean-B Algorithm

To overcome the computational limitations of MILP solvers, we propose **SHoTClean-B**, which replaces the global integer programming formulation with an efficient pruned dynamic programming approach. This method significantly reduces computational costs while maintaining equivalent accuracy. Algorithm 2 outlines the core procedure for finding maximum-weight path in a DAG.

Specifically, for each point $\mathbf{x}_i$, we construct a predecessor set $preds[i]$ by retaining only the $\mathcal{K}$ most recent valid predecessors under the hard constraint, and aborting the search early (lines

Algorithm 2: Max-Weight Path Search in SHoTClean-B

Input: time series $\mathbf{X}$, speed limits $V_{\min}, V_{\max}$, pruning length $\mathcal{K}$, deviation steepness parameter α.
Output: dirty points.

1 $s[i] \leftarrow \exp(-\alpha \cdot \left\| \frac{\mathbf{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right\|_2)$;
2 **for** $i \leftarrow 1$ **to** n **do**
3 **for** $j \leftarrow i-1$ **to** 1 **do**
4 **if** $\frac{||\mathbf{x}_i - \mathbf{x}_j||_2}{t_i - t_j} \in [V_{\min}, V_{\max}]$ **then**
5 $preds[i]$.append(j);
6 **if** *preds[i].length()* $> \mathcal{K}$ **then**
7 break;
8 **for** $i \leftarrow 1$ *to* n **do**
9 $dp[i] \leftarrow s[i], \quad prev[i] \leftarrow i$;
10 **foreach** $j \in preds[i]$ **do**
11 $cand \leftarrow dp[j] + s[i] \cdot \gamma(i-j)$;
12 **if** $cand > dp[i]$ **then**
13 $dp[i] \leftarrow cand, \quad prev[i] \leftarrow j$;
14 **if** $dp[i] > best\_score$ **then**
15 $best\_score \leftarrow dp[i], \quad e\_id \leftarrow i$;
16 $Route \leftarrow$ `backtrack_path`$(prev, e\_id)$;
17 **return** dirty points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \setminus Route$

Table 1. A Running Example

idx	*value*	*scores*	*preds*	*dp (add with γ)*	*path*
1	*0.35*	*3.00*	*[]*	*3.00*	*1*
2	*0.40*	*2.44*	*[1]*	*5.21*	*1*
3	*0.45*	*1.98*	*[2,1]*	*7.01*	*2*
4	*0.50*	*1.61*	*[3,2,1]*	*8.47*	*3*
5	*0.90*	*0.31*	*[3,2,1]*	*7.26*	*3*
6	*1.15*	*0.11*	*[5,3,2,1]*	*7.36*	*5*
7	*0.65*	*0.87*	*[6,5,4,3,2,1]*	*9.12*	*4*
8	*0.60*	*1.07*	*[7,6,5,4,3,2,1]*	*10.09*	*7*
9	*0.55*	*1.31*	*[8,7,6,5,4,3,2,1]*	*11.28*	*8*

2–7). Because temporal decay exponentially downweights distant predecessors, and Theorem 2.1(ii) shows that anchor points increase the total weight (i.e., shorter jumps are preferred over long ones). Therefore, the optimal path naturally favors neighboring nodes. Experiments (Fig. 11(b)) further demonstrate that when $\mathcal{K} \geq 3$, no accuracy degradation is observed compared to the MILP solver.

This $\mathcal{K}$-pruned method reduces the potential $O(N^2)$ predecessor checks to $O(N \cdot \mathcal{K})$, slashing the number of DP state transitions. The core DP recurrence (lines 8–15) is then defined as $dp[i] = \max_{j \in preds[i]}\{dp[j] + s_i \cdot \gamma(i-j)\}$, where at each step we maintain the global best score $dp[i]$ among all feasible candidate paths, and a backpointer $prev[i]$ that records the predecessor index achieving $dp[i]$ for path reconstruction. After filling the DP table, we reconstruct the maximum-weight path by following the backpointers from e_id to start (line 16). Specifically, function `backtrack_path`$(prev, e\_id)$ returns the path $Route = \{\cdots \rightarrow prev[prev[e\_id]] \rightarrow prev[e\_id] \rightarrow e\_id\}$ by repeatedly chasing $prev[\cdot]$ until a self-loop is met. All remaining points not included in $Route$ are then labeled as dirty (line 17).

Example 3. *Consider the time series* $\{0.35, 0.40, 0.45, 0.50, \underline{0.90}, \underline{1.15}, 0.65, 0.60, 0.55\}$ *observed at time indices* $t = 1, \ldots, 9$. *Suppose that the observations 0.90 and 1.15 are dirty, whose true values are 0.55 and 0.60, respectively. Under speed bounds* $V_{\min} = -0.5$ *and* $V_{\max} = 0.3$, *deviation steepness* $\alpha = 1$, *and temporal-decay* $\beta = 0.1$, *Table 1 shows the process via dynamic programming.*

By selecting the dynamic-programming path with the highest score at point $\mathbf{x}_9$, *we recover the valid trajectory* $\{\mathbf{x}_9, \mathbf{x}_8, \mathbf{x}_7, \mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1\}$, *and hence identify the dirty points* $\mathbf{x}_5$ *and* $\mathbf{x}_6$. *Point* $\mathbf{x}_5$ *is ruled out by the hard constraint: its speed relative to* $\mathbf{x}_4$ *exceeds* $V_{\max} = 0.3$. *In contrast,* $\mathbf{x}_6$, *despite satisfying the hard constraint, excluded on the basis of its lower soft-constraint score (*$7.36 < 9.12$*). In the subsequent repair step, each dirty point is interpolated using its nearest valid neighbor (e.g.,* $\mathbf{x}_4$ *and* $\mathbf{x}_7$*) via linear interpolation, yielding:* $\mathbf{x}'_5 = \mathbf{x}_4 + \frac{t_5 - t_4}{t_7 - t_4} \cdot (\mathbf{x}_7 - \mathbf{x}_4) = 0.50 + \frac{5-4}{7-4} \cdot (0.65 - 0.50) = 0.55$, $\mathbf{x}'_6 = \mathbf{x}_4 + \frac{t_6 - t_4}{t_7 - t_4} \cdot (\mathbf{x}_7 - \mathbf{x}_4) = 0.50 + \frac{6-4}{7-4} \cdot (0.65 - 0.50) = 0.60$.

In contrast, if only hard constraints are enforced, the algorithm will select the longest feasible path instead of the one with highest score. This excludes $\mathbf{x}_4$ *and yields* $\mathbf{x}'_4 = 0.675$. *Yet, while all other points differ by at most 0.05, the values 0.675, 0.90 and 1.15 exhibit substantial deviations—clearly showing that a hard-constraint-only approach fails to preserve the underlying statistical structure.*

Complexity Analysis. By employing a pruned DP algorithm, SHoTClean-B reduces the time complexity of the original MILP-based global optimizer from $O(2^N)$ to $O(D \cdot N \cdot \mathcal{K})$.

4 Online Setting

In online streaming scenarios, time series data evolves continuously, making global optimization methods that require complete batch processing inapplicable. To address this fundamental challenge, we propose an online adaptation of our framework that maintains **temporal locality** through localized processing and **optimality guarantees** via incremental optimization. Specifically, for

Algorithm 3: Online Cleaning in SHoTClean-S

Input: window series $\mathbf{X} = (\mathbf{x}_{og}, \ldots, \mathbf{x}_i)$, previous cleaned point $\mathbf{x}_{og-1}$, speed limits $V_{\min}, V_{\max}$, deviation steepness parameter α.

Output: cleaned time point $\mathbf{x}'_{og}$

1 $\boldsymbol{\mu} \leftarrow local\_avg, \quad \boldsymbol{\sigma} \leftarrow local\_std$;

2 $s[k] \leftarrow \exp(-\alpha \cdot \left\| \frac{\mathbf{x}_k - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \right\|_2)$ // $k \in [o, i]$;

3 $e\_id \leftarrow$ `find_best_idx()` // as in Alg. 2, lines 2-15

4 $lb \leftarrow \max\{\mathbf{x}_{og-1} + V_{\min}(t_{og} - t_{og-1}), \mathbf{x}_{e\_id} + V_{\max}(t_{og} - t_{e\_id})\}$;

5 $ub \leftarrow \min\{\mathbf{x}_{og-1} + V_{\max}(t_{og} - t_{og-1}), \mathbf{x}_{e\_id} + V_{\min}(t_{og} - t_{e\_id})\}$;

6 **if** $\mathbf{x}_{og} \notin [lb, ub]$ **then**

7 $\quad \mathbf{x}'_{og} \leftarrow \mathbf{x}_{og-1} + \frac{t_{og} - t_{og-1}}{t_{e\_id} - t_{og-1}}(\mathbf{x}_{e\_id} - \mathbf{x}_{og-1})$;

8 **return** $\mathbf{x}'_{og}$

each incoming point $\mathbf{x}_i$, we construct a dynamic DAG_i that captures local temporal dependencies. The online cleaning task then reduces to ***finding the maximum-weight path within*** DAG_i**.**

4.1 SHoTClean-S Algorithm

In online environments, data cleaning rarely achieves the global optimum attainable offline. To bridge this gap, we adopt an **egress-based strategy**: whenever a new point $\mathbf{x}_i$ arrives, we clean the outgoing point $\mathbf{x}_{og}$, where $og = i - W$ denotes the egress index. By maintaining a buffer of W, this strategy leverages both past and future context to mitigate out-of-order arrivals.

As shown in Algorithm 3, it computes the soft score using statistics within the current window (line 1), rather than relying on global statistics. This design enables effective handling of concept drift in streaming data by prioritizing recent changes over historical values. The search for the maximum-weight path (line 3) follows the same method as in the offline scenario. Specifically, function `find_best_idx`$(\cdot)$ returns the index e_id of the terminal node of the maximum-weight path within the current window DAG_i by running the same dynamic-programming recurrence as Alg. 2 (lines 2–15) over $[og, i]$. For the repair strategy, we perform linear interpolation between the last cleaned point $\mathbf{x}_{og-1}$ and the optimal point $\mathbf{x}_{max}$ on the optimal path in DAG_i (lines 4-7). This avoids reliance on potentially uncleaned data and anchors the interpolation at the node with the highest soft-constraint score, providing a more reliable estimate of the true value.

Although the online algorithm achieves only a locally optimal solution, it can still closely approximate the offline method's global optimum in terms of cumulative performance, with a controllable upper bound on the performance gap. Formally, we quantify this gap—referred to as regret R_n—by comparing the total score achieved by the online algorithm over a finite window to the maximum score obtained by the offline optimal solution using full global information.

Theorem 4.1. *The regret R_n of the online algorithm has a theoretical upper bound.*

Proof. Let S_n^{off} and S_n^{on} denote the total scores of the offline and online algorithms. The regret is defined as $R_n = S_n^{\text{off}} - S_n^{\text{on}}$. Suppose the offline algorithm connects node $\mathbf{x}_{k^*}$ to node $\mathbf{x}_{og}$. If $k^* - og \le W$, the online algorithm has access to $\mathbf{x}_{k^*}$ and incurs no regret. However, if $k^* - og > W$, the online algorithm cannot access $\mathbf{x}_{k^*}$ and must instead choose some $\hat{k} \in (og, i]$. The one-step regret is therefore $r_{og} = s_{og} \cdot [\gamma(k^* - og) - \gamma(\hat{k} - og)] \le s_{og} \cdot \gamma(W + 1) \le e^{-\beta(W+1)}$. Thus, the total regret R_n is the sum of individual regrets:

$$R_n = \sum_{i=1}^{n} r \le \sum_{i=1}^{n} e^{-\beta(W+1)} = n \cdot e^{-\beta(W+1)}. \tag{7}$$

Overall, the average per-step regret is bounded by $\frac{R_n}{n} \leq e^{-\beta(W+1)}$, indicating that the online algorithm's performance converges exponentially to that of the offline optimum as W increases. □

Complexity Analysis. In the worst case, finding the maximum-weight path within a single window requires $O(W^2)$ time, which yields an overall time complexity $O(N \cdot W^2)$ for SHoTClean-S.

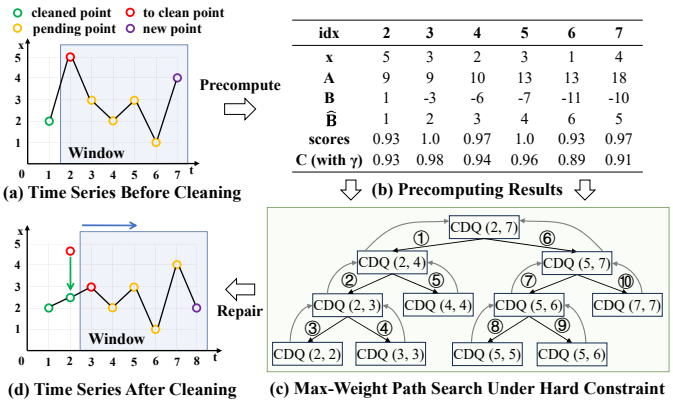
4.2 SHoTClean-P Algorithm

Referring to Eq. 7, we observe that increasing the window size W reduces the performance gap between the online algorithm and the offline global optimum. However, since SHoTClean-S has a runtime of $O(N \cdot W^2)$, enlarging W results in quadratic growth in computational cost, thereby limiting its scalability on large-window scenarios. Recall that, in SHoTClean-B, every pair of points satisfying the hard constraint is explicitly connected, resulting in up to $\frac{1}{2}W^2$ edges and a corresponding quadratic time cost. In the online setting, however, we can avoid both storing and traversing this large adjacency list by replacing edge enumeration with an on-the-fly two-dimensional range maximum query, yielding the near-linear SHoTClean-P shown in Algorithm 4.

Algorithm 4: Max-Weight Path Search in SHoTClean-P

Input: window series $\mathbf{X} = (\mathbf{x}_{og}, \ldots, \mathbf{x}_i)$, speed limits $V_{\min}, V_{\max}$.
Output: the index of max weight path e_id.

foreach $k \in [og, i]$ **do**
  $A_k \leftarrow \max(x_k^{(d)} - V_{\min} t_k)$, $B_k \leftarrow \min(x_k^{(d)} - V_{\max} t_k)$;
  $dp[k] \leftarrow s[k]$; // path weight at k (Eq. 2)
  $C_k \leftarrow s[k] \cdot \gamma(k-og)$; // time-decayed soft score
  $pts[k] \leftarrow \{idx = k, A = A_k, B_{hat} = \hat{B}_k, C = C_k, t = k-og\}$; // record structure with index, projection, discretized index, time-decayed soft score, relative time
`FenwickTree` fenw;
Function `CDQSolve`(l, r)**:**
  if $l = r$ **then**
    return
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  `CDQSolve` (l, m);
  left $\leftarrow$ sort$(pts[1..m], key = (A, idx))$;
  right $\leftarrow$ sort$(pts[m + 1..r], key = (A, idx))$;
  $iL \leftarrow 0$; // pointer in left part
  foreach $PR \in right$ **do**
    while $iL < left.length \wedge left[iL].A \leq PR.A$ **do**
      $k \leftarrow$ left$[iL].idx$;
      fenw.$update$(left$[iL].B_{hat}, dp[k]$);
      $iL \leftarrow iL + 1$;
    max_val $\leftarrow$ fenw.$query(PR.B_{hat})$;
    if $max\_val > 0 \wedge ||PR.\mathbf{x} - \mathbf{x}_k||_2 \geq V_{min}$ **then**
      $k \leftarrow PR.idx$;
      $dp[k] \leftarrow \max\{dp[k], max\_val \cdot \gamma(PR.t) + PR.C\}$;
  fenw.$clear()$;
  `CDQSolve` $(m + 1, r)$;
`CDQSolve` $(1, N)$;
return $e\_id \leftarrow \arg\max_{og} dp[og]$



idx	2	3	4	5	6	7
x	5	3	2	3	1	4
A	9	9	10	13	13	18
B	1	-3	-6	-7	-11	-10
$\hat{B}$	1	2	3	4	6	5
scores	0.93	1.0	0.97	1.0	0.93	0.97
C (with γ)	0.93	0.98	0.94	0.96	0.89	0.91

Fig. 4. An Illustration of SHoTClean-P Algorithm

Specifically, we maintain a 2D Fenwick tree [26]. As each new point arrives, we update its transformed coordinate values in the tree. To facilitate this, we define two auxiliary functions (line 2): $A(x,t) = \max(x^{(d)} - V_{\min} t)$ and $B(x,t) = \min(x^{(d)} - V_{\max} t)$. These scalar mappings serve as conservative approximations of the original multidimensional hard constraint. For any two points $\mathbf{x}_i$ and $\mathbf{x}_j$, if $A_j \le A_i$ and $B_j \ge B_i$, then gives $||\mathbf{x}_i - \mathbf{x}_j||_2 \le V_{\max}(t_i - t_j)$; similarly, if $A_j \le A_i - \frac{V_{\min}}{\sqrt{D}}(t_i - t_j)$, then $||\mathbf{x}_i - \mathbf{x}_j||_2 \ge V_{\min}(t_i - t_j)$.

By converting the hard constraint into scalar surrogates, we enable fast filtering with the Fenwick tree. We then apply an exact hard constraint check to ensure the validity of each candidate before updating DP table. This filter-and-verify scheme significantly accelerates computation while maintaining full accuracy.

Specifically, we first map A-values as the outer indices of Fenwick tree and reverse-map B-values (e.g., $\hat{B}$ in line 5) as the inner indices. This transformation converts the original adjacency-list-based constraint check into a single range-maximum query. For each point $\mathbf{x}_{og}$, we directly query $\max\{dp[k] \,|\, k > og, A_k \le A_{og}, B_k \ge B_{og}\}$ on the Fenwick tree. Once we find the successor point $\mathbf{x}_k$ that maximizes $dp[k]$ and also satisfies the hard constraint with $\mathbf{x}_{og}$, we add the current increment and immediately update $dp[og]$. Since each update or query on a Fenwick tree requires $O(\log^2 W)$ time, the dynamic program for one window runs in $O(W \cdot \log^2 W)$ time.

Building on the Fenwick tree, we incorporate the CDQ divide-and-conquer strategy [16] to enforce the temporal dependencies while evaluating the DP transition only across compatible pairs. The merge clearly separates left-side insertions from right-side queries. Specifically, the function `CDQSolve`(l, r) is the recursive subsolver that implements CDQ on the index range $[l, r]$. It first bisects the range (line 10), solves the left half (line 11), and then performs the merge: left-side indices are batch-inserted into a Fenwick tree keyed by B_{hat} (lines 16—19), then for each right record PR issuing a suffix-maximum query at $PR.B_{hat}$ (line 20) to obtain $\max_{k \in [l,mid]}\{dp[k]\}$ among candidates satisfying $A_k \le A_{og}$ and $B_k \ge B_{og}$. If the geometric constraint check passes, we update dp (lines 21–23). After completing the right-side queries, we clear the Fenwick tree (line 24) and recurse on the right half (line 25). Sorting by list A and querying by list B enforce the hard constraints, while the CDQ recursion guarantees the temporal condition $k < og$.

Example 4. *Consider a streaming time series* $(2, 5, 3, 2, 3, 1, 4, \cdots)$ *observed at time* $t = (1, 2, \cdots)$, *with window size* $W = 6$, *speed bounds* $V_{\min} = -2$ *and* $V_{\max} = 2$, *deviation steepness* $\alpha = 0.01$, *and temporal decay factor* $\beta = 0.1$, *as shown in Fig. 4. When* $\mathbf{x}_7$ *arrives, the following procedure is executed:*

We begin by precomputing values A, B, $\hat{B}$, *soft scores over the current window, and time-decayed soft score* $C_k = s_k \cdot \gamma(t_k - t_2)$ *(Alg. 4, lines 1–5). Using the CDQ strategy and Fenwick tree, we then recursively find the maximum-weight path that satisfies the hard constraint, which yields the optimal*

path $\mathbf{x}_3 \to \mathbf{x}_4 \to \mathbf{x}_5 \to \mathbf{x}_6$ *and optimal successor* $\mathbf{x}_3$. *Since* $\mathbf{x}_2$ *is excluded from this path and to be evicted, we clean it by linear interpolation, resulting in* $\mathbf{x}'_2 = \mathbf{x}_1 + \frac{t_2 - t_1}{t_3 - t_1}(\mathbf{x}_3 - \mathbf{x}_1) = 2.5$. *Finally, the cleaned point* $\mathbf{x}'_2$ *is removed from the window as the next data point arrives for subsequent cleaning.*

Complexity Analysis. By combining the CDQ strategy with the Fenwick tree, SHoTClean-P reduces the per-window complexity to $O(W \cdot logW)$, resulting in an overall time complexity of $O(N \cdot W \cdot logW)$, thus enhancing efficiency in large-window scenarios.

5 Multivariate Correlation Modeling

In the SHoTClean-S and SHoTClean-P methods, we aggregate Euclidean-distance deviations across all dimensions and incorporate them into the soft constraint. Although this strategy achieves reasonable results in practice (see Table 4), it neglects the latent causal relationships among variables and thus underperforms in scenarios dominated by strong causal effects. To overcome this limitation, we further propose SHoTClean-C, presented in Algorithm 5, which integrates causal modeling directly into the soft constraint function to jointly clean multivariate time series.

To reduce online computation, we pretrain the causal model offline on the first 1% of the data (line 1), denoted as $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{T_0})$ with $T_0 = 0.01n$. Within this subset, we both accelerate the training process and capture the primary features of the data distribution. Below, we detail the steps for causal discovery and model fitting:

i) Causal Discovery. We employ the PCMCI algorithm [46] to identify causal relations in multivariate time series (line 2). It detects lagged causal links through two phases: (a) a PC-style condition-selection selects a small set of candidate parents for each variable; (b) an MCI test that evaluates each candidate lag while conditioning on those selected parents to reduce false positives from autocorrelation. Specifically, with a maximum lag $\tau_{\max}$, for each ordered variable pair $(i, j) \in \{1, \ldots, D\}^2$, and each lag $\tau \in [0, \tau_{\max}]$, function `PCMCI` (line 2) performs a condition-selection step and then an MCI test based on partial correlation $\rho_{i \to j}(\tau)$. Let $\mathcal{S}_{i,j,\tau} = \{\mathbf{x}^{d'}_{t-\tau'} \mid (d', \tau') \neq (i, \tau), d' \in [1, D], \tau' \in [0, \tau_{\max}]\}$ denote the conditioning set from first phase, we compute:

$$\rho_{i \to j}(\tau) = \texttt{ParCorr}(\mathbf{x}^i_{t-\tau}, \mathbf{x}^j_t \mid \mathcal{S}_{i,j,\tau}), \tag{8}$$

where `ParCorr` denotes the sample partial correlation coefficient. We then obtain the two-sided raw p-values $P_{i,j,\tau}$ from the corresponding partial-correlation significance test:

$$P_{i,j,\tau} = \texttt{Pr}(|\rho_{i \to j}(\tau)| \neq |\rho^{\text{obs}}_{i \to j}(\tau)|), \tag{9}$$

Algorithm 5: Offline Causal Training of SHoTClean-C

Input: time series $\mathbf{X}$, maximum lag $\tau_{\max}$
Output: weight matrix Θ, bias vector $\mathbf{b}$

1 $\mathcal{X} \leftarrow (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_0})$;
2 $Q \leftarrow$ `FDR_correct(PCMCI(`$\mathcal{X}, \tau_{\max}$`))` ;
3 **foreach** $j \leftarrow 1$ ***to*** D **do**
4 $\quad Par_j \leftarrow \{i \mid \exists \tau \in [0, \tau_{\max}], Q[i, j, \tau] < \alpha_{\text{PC}}\}$;
5 $\quad$ **if** $Par_j = \emptyset$ **then**
6 $\quad\quad$ **continue**
7 $\quad F \leftarrow \mathcal{X}[2 : T_0, Par_j], \quad y \leftarrow \mathcal{X}[2 : T_0, j]$;
8 $\quad y \approx F\theta^j + b^j$; // fit linear regression
9 $\quad$ **foreach** $k \leftarrow 1$ ***to*** $|Par_j|$ **do**
10 $\quad\quad \Theta[j, Par_{j,k}] \leftarrow \theta^j_k$;
11 $\quad b[j] \leftarrow b^j$;
12 **return** weight matrix Θ, bias vector $\mathbf{b}$

where $\texttt{Pr}(\cdot)$ denotes the probability under the null of conditional independence, and $\rho^{\text{obs}}_{i\to j}(\tau)$ denotes the observed partial correlation computed from the actual dataset.

ii) Multiple-Testing Control. Let $M = D^2(\tau_{\max}+1)$ be the total number of tests and $P_1 \le \cdots \le P_M$ be the sorted p-values. Due to PCMCI performs M parallel tests, we use Benjamini-Hochberg (BH) correction [8] to control the expected false discovery rate across these tests (line 2). The function $\texttt{FDR\_correct}(\cdot)$ implements the BH procedure and returns the BH-adjusted q-values:

$$Q_k = \min_{m\ge k} \frac{P_m M}{m}, \quad k = 1,\dots,M \qquad \text{(line 2)}, \tag{10}$$

followed by the monotonicity enforcement $Q_{(k)} \leftarrow \min\{Q_{(k)}, Q_{(k+1)}\}$ from tail to head. We then map these values back to their original (i, j, τ) indices as $Q_{i,j,\tau}$. Given a significance threshold α_{PC} of BH correction, the parent set Par_j of each variable j is defined as:

$$\text{Par}_j = \{i \in [1, D] \mid \exists \tau \in [0, \tau_{\max}] : Q_{i,j,\tau} < \alpha_{\text{PC}}\} \quad \text{(line 4)}, \tag{11}$$

with size $p_j = |\text{Par}_j|$.

iii) Linear Fitting. For each variable j with non-empty parent set Par_j (lines 5–6), we build a feature matrix $F^j \in \mathbb{R}^{(T_0-1)\times p_j}$ from the lagged parent series (line 7). Subsequently, the regression coefficient vector $\boldsymbol{\theta}^j$ and the intercept b^j are estimated via the ordinary least-squares model $y \approx F\theta^j + b^j$ (line 8). Finally, we assemble all $\boldsymbol{\theta}^j$ into coefficient matrix $\Theta \in \mathbb{R}^{D\times D}$ (lines 9–10) and collect intercept terms into vector $\mathbf{b}$ (line 11) to form the online linear predictor.

In the online phase, we apply the coefficient matrix Θ and bias vector $\mathbf{b}$ learned from the offline to generate each forecast via the linear model $\hat{\mathbf{x}}_i = \Theta\mathbf{x}_{i-1} + \mathbf{b}$. To quantify how well each new observation adheres to the learned causal structure, we define the causal consistency score $s_{causal} = \exp(-||\mathbf{x}_i - \hat{\mathbf{x}}_i||_2^2)$, which penalizes large residuals. A low causal score implies that the current observation deviates significantly from the expected behavior implied by the historical causal structure, suggesting potential dirty points.

Concurrently, we compute a statistical score s_{stats} based on Eq. 2, and fuse it with the causal consistency score s_{causal} to obtain a statistical-causal soft constraint:

$$s_{stats-causal} = w_{stats} \cdot s_{stats} + w_{causal} \cdot s_{causal}, \tag{12}$$

where $w_{stats} + w_{causal} = 1$, allowing for a trade-off between statistical regularity and causal consistency. In SHoTClean-C, we replace the origin soft score with $s_{stats-causal}$ to uncover causal relationships among variables.

Please note that the causal component is intentionally designed to be lightweight. The ordinary least-squares predictor achieves nearly linear computational complexity with respect to data size, which is critical for streaming scenarios. This represents a deliberate trade-off between accuracy (see Table 4) and efficiency (see Fig. 5). In contrast, while nonlinear models yield modest performance improvements in certain cases, the gains are limited relative to their substantial computational overhead, making them unsuitable for online streaming settings (see Section 6.8).

Complexity Analysis. The offline training has a worst-case time complexity of $O(T_0 \cdot D^3 + D^4)$. However, since $T_0 = 0.01n$ and D is typically constant in practice, this cost becomes negligible for large n. For online inference, computing the causal consistency score $s_{\text{stats-causal}}$ requires only $O(W \cdot D^2)$ time per window, demonstrating significant computational efficiency. This theoretical analysis aligns with our experimental results shown in Fig. 5.

6 Experiment

We evaluate the SHoTClean framework (comprising four constituent algorithms) via the following Research Questions (RQs), assessing the cleaning accuracy, computational efficiency, robustness, parameter sensitivity, downstream task, and scalability:

Table 2. Dataset Statistics

Dimension	Dataset	Length	# dim	Source	Field
Univariate	TOTALSA [53]	593	1	FRED	Trade
	STOCK [51]	12,824	1	SCREEN	Finance
	COVID-19 [24]	14,001	1	JHU CSSE	Health
	CA [14]	43,824	1	California ISO	Energy
	ID_a40b [44]	137,898	1	AIOps	KPI
Multivariate	Porto [39]	16,749	2	Kaggle	Trajectory
	ECG [38]	650,000	2	MIT-BIH	Health
	Exchange [35]	7,588	8	LSTNet	Finance
	AEP [15]	19,735	21	UCI	Energy
	PSM [1]	132,481	25	eBay	Server
	SWaT [37]	14,996	26	iTrust	Industrial
	WADI [3]	784,537	73	iTrust	Industrial

RQ1: *How does SHoTClean perform compared with the state-of-the-art univariate and multivariate time-series cleaning baselines?*

RQ2: *How does the running efficiency of SHoTClean?*

RQ3: *How robust is SHoTClean under different dirty rates, dataset scales, noise schemes, and naturally occurring noise?*

RQ4: *How do key components affect SHoTClean's performance?*

RQ5: *How sensitive is SHoTClean to hyperparameter settings?*

RQ6: *How does SHoTClean affect downstream task performance?*

RQ7*: How does SHoTClean scale with hard constraints and nonlinear predictors?*

6.1 Experimental Settings

6.1.1 ***Datasets.*** We evaluate SHoTClean on 12 real-world datasets, with statistics summarized in Table 2. These datasets are widely used in time series research [18, 31, 52, 60] and exhibit diverse characteristics across five key dimensions: (1) they cover eight distinct application domains (energy, finance, etc.); (2) sampling frequencies range from seconds (WADI) to months (TOTALSA), including irregular patterns (PSM); (3) data quality varies from near-pristine to heavily contaminated (SWaT); (4) inter-variable correlations range from strong (Exchange) to negligible (Porto); and (5) scale varies from small univariate sets to large multidimensional collections (more than 100K records).

Through comprehensive experiments on these diverse datasets, we rigorously evaluate the robustness and cleaning accuracy across multiple challenging scenarios: (1) different dirty corruption levels, (2) varying temporal dependencies (short- and long-term), (3) extremely high dirty rates, (4) different noise schemes, and (5) naturally occurring errors. We primarily present the robustness results on two representative datasets: the univariate CA dataset and the multivariate SWaT dataset. The CA dataset covers a wide range of value magnitudes, which poses challenges to data cleaning algorithms in handling both large spikes and small jumps. In contrast, SWaT typifies high-dimensional industrial control system data characterized by strong inter-variable couplings and severe contamination, making it ideal for stress-testing multivariate modeling, latency, and robustness. In addition, the COVID-19 dataset is used to investigate the scalability of hard constraints, while the ECG dataset is used to evaluate the adaptability to naturally occurring errors.

6.1.2 ***Noise Injection.*** To ensure fair and direct comparability with state-of-the-art methods [21, 31, 51, 55], we implement a standardized evaluation protocol using Gaussian noise injection at randomly selected points. Specifically, for each selected timestamp t and dimension d, we perturb the value with i.i.d. Gaussian noise:

$$\hat{\mathbf{x}}_t^{(d)} = \mathbf{x}_t^{(d)} + \varepsilon_t^{(d)}, \quad \varepsilon_t^{(d)} \sim \mathcal{N}(0, s_\varepsilon^2). \tag{13}$$

Here, s_ε is set to 0.3 times the mean of the ground-truth series. The noise is injected in a point-wise manner; for multivariate datasets, it is applied consistently across all dimensions to maintain cross-channel consistency, which is typical in multi-sensor systems.

Table 3. The Evaluated Baselines

Dimension	Algorithm	Scenario	Type
Univariate	EWMA	online	smoothing
	SCREEN	online	constraint
	SpeedAcc	online	constraint
	LsGreedy	online	statistical
	Akane	offline	statistical
Multivariate	ARIMA	offline	smoothing
	Clean4MTS	offline	constraint+statistical
	MTCSC	online	constraint+statistical
	TranAD	online	anomaly detection
	ImDiffusion	online	anomaly detection
	SHoTClean-B	offline	constraint+statistical
	SHoTClean-S	online	constraint+statistical
	SHoTClean-P	online	constraint+statistical
	SHoTClean-C	online	constraint+statistical

We keep all native anomalies that already exist in datasets such as SWaT and WADI without any preprocessing removal, and directly inject noise into these two datasets to emulate realistic deployment scenarios where sensor faults and stochastic noise coexist, allowing the evaluation to capture cleaning performance under naturally contaminated conditions. This setup increases the challenge of data cleaning while more faithfully highlighting the robustness, stability, and end-to-end effectiveness of different methods in environments where noise and anomalies coexist.

6.1.3 ***Baselines.*** As shown in Table 3, we compare the SHoTClean series against 10 representative algorithms across five categories:

i) Smoothing-based methods: **EWMA** [29] suppresses dirty points using weighted moving averages, while **ARIMA** [10] captures trends via autoregression, differencing, and moving averages.

ii) Constraint-based methods: **SCREEN** [51] flags dirty points using speed constraints, while **SpeedAcc** [50] enforces both speed and acceleration bounds.

iii) Statistical-based methods: **LsGreedy** [59] segments the series greedily based on least-squares residuals, while **Akane** [31] models perplexity distributions to flag outliers.

iv) Hybrid methods: **Clean4MTS** [20] jointly applies physical constraints and linear relationships among variables, while **MTCSC** [60] combines clustering with multivariate speed constraints.

v) Anomaly-detection-based methods: **TranAD** [52] reconstructs sequences via a Transformer and flags anomalies by reconstruction error, while **ImDiffusion** [18] imputes values via a diffusion model and scores anomalies by imputation error.

For **univariate-only** algorithms, we decompose the input multivariate series into individual dimensions. Then, we apply the univariate-only method separately to each univariate component. Finally, we aggregate dimension-level results into comprehensive metrics. Note that, for the fairness of comparison, we evaluate all approaches in their native operational environments: i) **Batch methods**: offline setting with full data access and ii) **Streaming methods**: online scenario with sequential data arrival.

6.1.4 ***Evaluation Metrics.*** We evaluate our method using three metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Relative Repair Accuracy (RRA) [21, 51].

RRA measures how effectively the cleaned data converges toward the true values compared to the original dirty observations. Its value ranges from 0 to 1, with values closer to 1 indicating better repair performance. Formally, it is defined as follows:

$$\mathrm{RRA} = 1 - \frac{\sum_{i=1}^{n}\sum_{d=1}^{D}|\mathbf{x}_i'^{(d)} - \mathbf{x}_i^{*(d)}|}{\sum_{i=1}^{n}\sum_{d=1}^{D}|\mathbf{x}_i^{(d)} - \mathbf{x}_i'^{(d)}| + \sum_{i=1}^{n}\sum_{d=1}^{D}|\mathbf{x}_i^{(d)} - \mathbf{x}_i^{*(d)}|}. \tag{14}$$

Here, $\mathbf{x}_i^{(d)}$ denotes the data after noise injection, $\mathbf{x}_i^{'(d)}$ denotes the cleaned data, and $\mathbf{x}_i^{*(d)}$ denotes the pre-noise (ground truth) data. For SWaT and WADI, which inherently contain anomalies, we do not apply preprocessing to obtain ground truth. Instead, following the common practice in prior work [21], we treat the original datasets as the ground truth to compute the metrics.

The rationale is that data cleaning typically aims to suppress measurement noise while preserving genuine anomalies, which are crucial for tasks such as anomaly detection and root-cause analysis. Since anomalies capture atypical but meaningful behavior, removing them would harm downstream tasks. Therefore, following prior work [21], our methods and all baselines only clean the dirty points (or the dirty points within anomalous segments), without altering the anomalies.

6.1.5 ***Implementations.*** The parameters of SHoTClean are set as follows: $\alpha = 0.01$, $\beta = 0.1$, $\mathcal{K} = 5$, $W = 10$, $w_{stats} = 0.5$, $w_{causal} = 0.5$, $\tau_{max} = 2$. All experiments were conducted on a CentOS 7 server equipped with two Intel(R) Xeon(R) E5-2650 v4 12-core 2.20 GHz CPUs, 128 GB of RAM, and an NVIDIA RTX 3090 GPU. More implementation details can be found in the code.

6.2 Overall Performance Evaluation (*RQ1*)

Table 4 presents an overall performance evaluation across all datasets. It distinguishes between offline (gray-shaded) and online (unshaded) scenarios, with best results highlighted in bold. We evaluate all methods under a fixed dirty rate of 20%, artificially inject dirty points. Missing entries ("−") indicate method-dataset incompatibilities, particularly when methods designed solely for multivariate data are applied to either univariate datasets or two-dimensional dataset (e.g., Porto).

In the offline setting, SHoTClean-B significantly outperforms all baselines on both univariate and multivariate datasets, demonstrating **58.5%** average RMSE reduction, **69.0%** MAE improvement, and **20.0%** average RRA increase versus the second-best methods. This consistent superiority validates our constrained optimization framework's effectiveness for batch processing.

In the online setting, all three SHoTClean variants achieve the top scores on both univariate and multivariate datasets. Notably, SHoTClean-C delivers dramatically superior cleaning performance on multivariate data: reducing RMSE by an average of **49.39%**, MAE by **70.88%**, and improving RRA by **15.84%** compared to the second-best methods. As observed, the existing methods' inability to capture cross-variable causal relations limits their cleaning accuracy. In contrast, our causal modeling provides statistically significant improvements while maintaining robustness.

We also yield the following observations: (i) smoothing-based methods are effective for handling gradual distribution shifts but suffer from over-smoothing artifacts in heterogeneous data environments, often obscuring important features; (ii) constraint-based methods, which rely on physical bounds such as speed and acceleration, effectively suppress outliers, achieving moderate overall performance; (iii) statistical-based methods, while mathematically rigorous, exhibit inconsistent results and incur significantly higher computational costs when dealing with diverse distributions and high-dimensional nonlinear coupling (as shown in Fig. 5); (iv) hybrid methods integrate multiple strategies to achieve robust and balanced cleaning across most datasets, with MTCSC performing particularly well in varied environments; (v) anomaly-detection-based methods primarily identify outliers rather than performing full end-to-end cleaning, resulting in inferior performance in comprehensive data cleaning tasks. Overall, the SHoTClean series, through fusion of soft and hard constraints and incorporation of cross-dimensional causal modeling, effectively suppresses diverse dirty types while preserving fine-grained features, demonstrating superior robustness and precision across a wide range of conditions.

Beyond the main results, comparisons among SHoTClean variants yield several insights. For example, while offline methods generally outperform online ones intuitively, SHoTClean-B performs worse than online variants on the WADI dataset. The reason is that SHoTClean-B estimates global

Table 4. The Overall Performance Comparison of All Methods on All Datasets (20% Dirty Rate)

Algorithm	TOTALSA			STOCK			CA			ID_a40b			Porto		
	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA
Arima	1.25	0.42	43.19	4.96	1.78	58.88	2277.39	777.43	39.29	151.99	49.19	55.21	0.096	0.0280	48.86
Akane	0.99	0.34	68.97	1.16	0.31	90.74	705.60	234.05	89.40	196.58	60.07	42.55	0.024	0.0410	93.78
Clean4MTS	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SHoTClean-B	**0.75**	**0.24**	**80.75**	**0.57**	**0.15**	**95.63**	**657.61**	**196.08**	**90.83**	**77.41**	**22.71**	**86.03**	**0.007**	**0.0010**	**94.78**
EWMA	1.65	1.22	46.07	2.30	1.63	66.50	3355.91	2674.43	40.05	143.80	108.48	57.92	0.042	0.0305	69.49
SCREEN	1.24	0.63	52.06	2.54	1.55	62.01	2024.51	1103.74	54.34	161.45	82.73	53.42	0.049	0.0297	54.48
SpeedAcc	1.27	0.65	50.75	2.62	1.64	60.40	2784.79	2068.73	38.24	165.69	85.98	51.98	0.046	0.0285	55.24
LsGreedy	1.03	0.41	53.65	1.63	0.64	77.96	1917.8	760.51	47.77	191.92	72.89	24.31	0.016	0.0041	94.57
MTCSC	0.81	0.26	79.02	0.57	0.15	95.61	820.63	264.47	87.82	76.21	22.46	86.26	0.011	0.0020	91.18
TranAD	1.01	0.30	72.00	3.05	0.95	64.06	2367.07	899.86	62.10	287.57	98.69	49.26	0.124	0.0401	92.31
ImDiffusion	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SHoTClean-S	0.83	0.28	78.24	0.56	0.15	95.57	783.06	264.34	87.89	75.84	22.38	86.37	0.010	0.0017	91.58
SHoTClean-P	**0.72**	**0.23**	**81.82**	**0.51**	**0.14**	**96.10**	**703.60**	**207.84**	**90.28**	**70.29**	**19.44**	**88.08**	0.012	0.0019	91.54
SHoTClean-C	-	-	-	-	-	-	-	-	-	-	-	-	**0.008**	**0.0012**	**94.59**

Algorithm	Exchange			AEP			PSM			SWaT			WADI		
	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA	RMSE	MAE	RRA
Arima	0.050	0.0161	58.39	2.94	0.82	65.11	0.051	0.016	56.20	10.84	3.09	64.89	17.95	1.08	54.06
Akane	0.020	0.0043	91.68	0.82	0.16	93.22	0.038	0.010	75.62	7.50	1.52	86.62	17.76	1.17	63.06
Clean4MTS	0.074	0.0218	42.33	3.74	1.07	25.13	0.051	0.017	18.35	19.17	5.53	20.83	30.52	2.27	4.17
SHoTClean-B	**0.002**	**0.0004**	**99.03**	**0.09**	**0.01**	**99.53**	**0.006**	**0.001**	**96.37**	**0.80**	**0.01**	**99.80**	**13.20**	**0.27**	**92.24**
EWMA	0.022	0.0141	76.82	2.37	1.23	63.32	0.019	0.017	73.74	5.72	2.03	76.01	12.29	1.19	72.53
SCREEN	0.052	0.0236	52.23	3.03	1.13	33.88	0.047	0.018	32.86	7.39	2.47	67.24	12.59	0.69	81.34
SpeedAcc	0.053	0.0241	51.46	3.05	1.13	33.71	0.047	0.018	32.83	7.07	2.44	67.46	12.70	0.69	81.21
LsGreedy	0.019	0.0060	86.32	1.96	0.60	65.31	0.047	0.016	25.02	8.26	1.41	74.32	26.94	1.59	10.02
MTCSC	0.017	0.0048	89.65	1.84	0.61	65.57	0.029	0.009	68.39	1.88	0.29	95.59	5.79	0.13	96.06
TranAD	0.088	0.0295	25.69	3.75	1.15	25.40	0.053	0.021	32.16	19.19	5.58	18.15	30.14	2.36	23.22
ImDiffusion	0.090	0.0291	23.95	3.90	1.16	22.13	0.055	0.018	18.39	17.46	4.69	19.13	30.20	2.22	9.29
SHoTClean-S	0.017	0.0049	89.34	1.86	0.63	65.09	0.029	0.009	68.55	1.66	0.29	95.61	5.32	0.12	96.35
SHoTClean-P	0.016	0.0044	90.45	1.77	0.59	67.01	0.027	0.008	71.10	1.57	0.27	95.83	5.64	0.12	96.37
SHoTClean-C	**0.006**	**0.0010**	**98.01**	**0.33**	**0.04**	**98.13**	**0.006**	**0.001**	**96.33**	**1.03**	**0.02**	**99.70**	**5.29**	**0.09**	**97.12**

statistics from the available (potentially dirty) data (Def. 3), which is unreliable in contaminated datasets like WADI. In contrast, online variants compute window-local statistics and use the optimal-scoring node as the interpolation anchor, making them more robust and occasionally superior to the offline one on contaminated data.

Moreover, we can find that SHoTClean-P is both faster and slightly more accurate than SHoTClean-S across most datasets. Algorithmically, SHoTClean-P augments SHoTClean-S with the CDQ strategy and Fenwick tree, replacing exhaustive edge enumeration with range maximum queries and reducing the per-window DP from $O(W^2)$ to $O(W \cdot logW)$. Additionally, its filter-and-verify scheme eliminates redundant enumeration and updates within each window, reducing approximation bias and numerical overhead. Consequently, SHoTClean-P achieves a moderate gain over SHoTClean-S.

From these observations, we can derive practical guidance: for offline batch, SHoTClean-B is recommended; for online streaming, SHoTClean-C is preferable. Since SHoTClean-C is designed only for multivariate data, it should be avoided in univariate settings.

6.3 Efficiency Evaluation (*RQ2*)

We proceed to evaluate the efficiency of the ShoTClean series. Unlike most existing studies [31, 50, 51, 59, 60] that evaluate cleaning efficiency on small datasets, we assess time-cost performance under more demanding conditions by applying the methods to the large-scale high-dimensional SWaT and WADI datasets, validating both efficiency and practical applicability. To ensure a fair comparison, we replace the standard multivariate SHoTClean-P with its univariate variant SHoTClean-P(Uni), aligning its processing capabilities with other single-dimensional methods while retaining the multivariate implementations for the remaining SHoTClean series. The results are shown in Fig. 5.

The experiments reveal that most existing methods cannot process large-scale high-dimensional datasets efficiently. Specifically: (i) Among smoothing-based methods, EWMA is the fastest as it computes weighted moving averages without any model fitting, while ARIMA incurs substantial

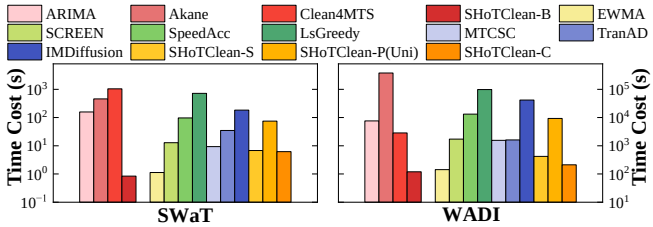


Fig. 5. Efficiency Comparison on SWaT and WADI Datasets

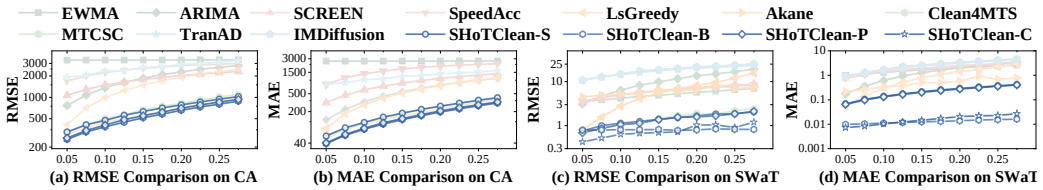


Fig. 6. Performance Comparison vs. Varying Dirty Rates

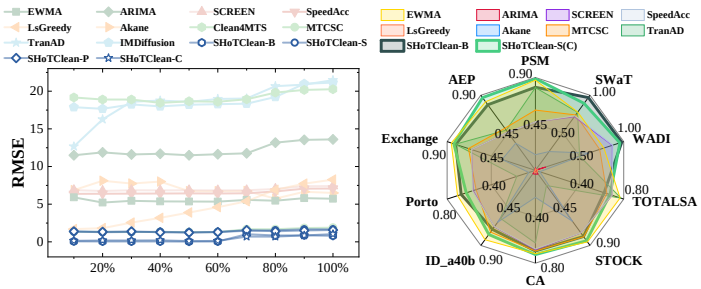
overhead from iterative estimation of autoregressive and moving-average coefficients. (ii) For constraint-based methods, SCREEN's single-pass speed-constrained scan outperforms univariate methods, whereas SpeedAcc's two-stage dynamic programming approach introduces significantly higher latency. (iii) Statistical-based methods exhibit particularly poor scalability on the WADI dataset: LsGreedy repeatedly builds and updates probabilistic speed-change models for each point, and Akane performs symbolic clustering before estimating conditional probabilities through a Markov chain. Both methods require over 20 hours of computation, making them impractical for real-world applications. (iv) Among hybrid methods, MTCSC balances speed and accuracy effectively through its combination of speed constraints and windowed clustering, while Clean4MTS suffers from slower model construction due to its four-stage framework. (v) Anomaly-detection-based methods also incur substantial delays because of offline pretraining and fine-tuning requirements. (vi) Overall, the multidimensional SHoTClean series combines the low latency of hard-constraint techniques with robust dirty suppression and fine-grained signal preservation, achieving up to two orders of magnitude greater efficiency than SOTA methods. These advantages make it particularly suitable for real-time streaming applications.

6.4 Robustness Evaluation (*RQ3*)

We then assess the robustness of SHoTClean via a series of key experiments: i) an analysis under varying dirty rates (Fig. 6); ii) an analysis across varying dataset sizes (Fig. 7 (a)); iii) an evaluation under high dirty-rate conditions (Fig. 7 (b)); iv) an assessment with different noise-injection strategies (Fig. 8); and v) a validation on dataset with natural noise (Fig. 9).

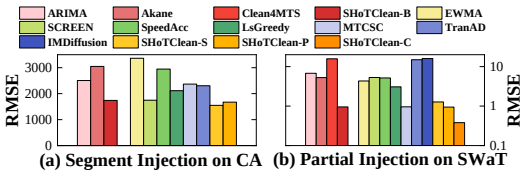
i) Varying Dirty Rates. Using the representative univariate CA dataset and multivariate SWaT dataset, the results are shown in Fig. 6. As the proportion of dirty points increases, the SHoTClean series exhibit only a modest rise in RMSE on the CA dataset, indicating strong resistance. In terms of the SWaT dataset, SHoTClean-C exhibits minor fluctuations yet still achieves the best cleaning accuracy. These results confirm that by integrating soft and hard constraints with cross-dimensional repair and causal-dependency modeling, the SHoTClean series effectively mitigates performance degradation and demonstrates consistent robustness across varying dirty rates.

ii) Varying Dataset Sizes. Using SWaT dataset with a fixed 20% dirty rate, we progressively scale the dataset size from 10% to 100%, the results are shown in Fig. 7 (a). As the size grows, statistical-based and anomaly-detection-based methods exhibit pronounced performance variability, whereas other methods remain relatively stable. Particularly, SHoTClean series consistently achieves the lowest RMSE with minimal fluctuation, demonstrating both robustness and scalability.



(a) Varying Dataset Sizes on SWaT (b) RRA Metrics under High Dirty Rate

Fig. 7. Robustness Study: Large Dataset and High Dirty Rates



(a) Segment Injection on CA (b) Partial Injection on SWaT

Fig. 8. Performance vs. Alternative Noise Injection

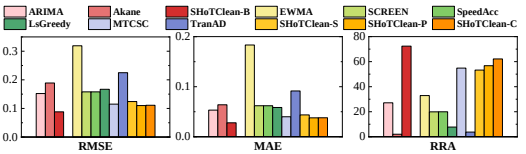


Fig. 9. Evaluation on ECG Dataset with Natural Noise

iii) High Dirty Rates. We evaluate the RRA metric under extreme 80% dirty-rate conditions, the results are shown in Fig. 7 (b). Since SHoTClean-C's exclusive design for multivariate data, we employ SHoTClean-S for univariate datasets and implement a combined SHoTClean-S(C) approach for online evaluation. Clean4MTS and ImDiffusion—being inapplicable to one-dimensional data—are omitted from this comparison. As observed, ARIMA and LsGreedy fail completely while other baseline methods exhibit substantial performance degradation. Notably, EWMA maintains competitive RRA performance due to its inherent smoothing capability, and TranAD performs even better at high dirty rates than at low ones. In both offline and online scenarios, the SHoTClean series demonstrates high robustness, particularly on large-scale high-dimensional datasets.

iv) Varying Noise Schemes. To further evaluate robustness under various noise-injection schemes, we extend the default point-wise injection by introducing two additional schemes: contiguous-segment injection (corrupting a continuous time window) and partial-dimension injection (corrupting a subset of features), with results in Fig. 8. Across both settings, SHoTClean consistently outperforms the baselines thanks to soft-constraint design: for contiguous-segment noise, it assigns low soft scores to the disturbed segments; for partial-dimension noise, SHoTClean-C uses causal residuals to penalize cross-dimensional consistency. Overall, the SHoTClean series not only achieves state-of-the-art performance under point-wise noise but also excels across diverse noise schemes, demonstrating strong robustness and broad applicability to real-world scenarios.

v) Natural Noise. To further assess external robustness to naturally occurring errors, we evaluate SHoTClean on an ECG dataset derived from the MIT-BIH Arrhythmia Database. Unlike synthetic corruptions such as Gaussian noise injections or random missingness, this dataset contains real-world electrophysiological disturbances, offering a more realistic testbed. As shown in Fig. 9, the SHoTClean series outperforms baselines across every metric, benefiting from soft and hard constraints that underpin strong cross-distribution generalization, which reduce over-smoothing on small jumps and enhance sensitivity to large spikes. These findings are consistent with our results under synthetic noise (Table 4), indicating that SHoTClean is not limited to synthetic noise injection and extends effectively to data with naturally occurring errors.

6.5 Ablation Study (*RQ4*)

We evaluate the contributions of the soft constraint mechanism in SHoTClean-B and of the causal modeling in SHoTClean-C. For causal modeling, we use the SWaT dataset to evaluate causal

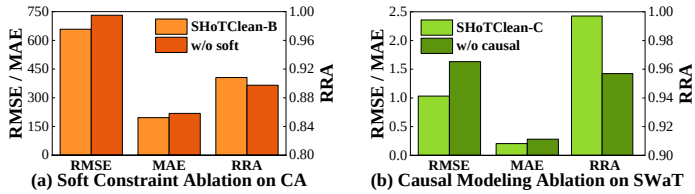


Fig. 10. Ablation Study

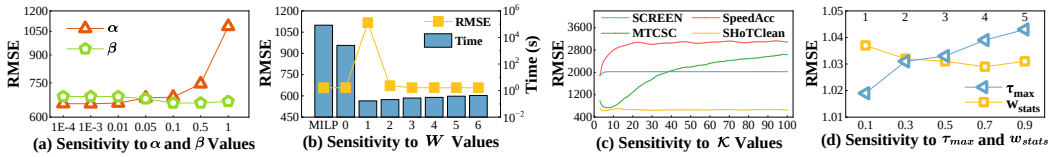


Fig. 11. Parameter Sensitivity Study

inference capabilities. As shown in Fig. 10, both the soft constraint and causal modeling significantly improve cleaning performance. These enhancements arise from their complementary strengths: the soft constraint enables fine-grained detection by identifying small jumps without excessively penalizing boundary values, while causal modeling effectively captures inherent inter-variable dependencies, enabling more accurate detection and precise repair.

6.6 Parameter Study (*RQ5*)

Fig. 11 presents the sensitivity analysis of six key parameters: soft-constraint deviation steepness (α), temporal-decay factor (β), pruning length ($\mathcal{K}$), window size (W), maximum lag (τ_{max}), and statistical soft-score weight (w_{stats}). We evaluated τ_{max} and w_{stats} on the SWaT dataset as they are specific to multivariate data analysis, while testing the remaining parameters on the CA dataset.

Overall, the following observations demonstrate the robustness of our method under parameter variations. First, Fig. 11(a) reveals that optimal performance around $\alpha \approx 0.01$ and $\beta \approx 0.1$, with smaller α values yield smoother constraint transitions while larger β values strengthen temporal decay effects. Second, Fig. 11(b) shows that, when $\mathcal{K} \geq 3$, pruning achieves runtime improvements of three orders of magnitude without compromising cleaning accuracy. Third, comparative analysis in Fig. 11(c) indicates that the SHoTClean is more robust to window size and maintains stable RMSE ($\Delta \approx 7\%$) compared to SpeedAcc ($\Delta \approx 65\%$) and MTCSC ($\Delta \approx 260\%$). Finally, Fig. 11(d) confirms that adjustments to both τ_{max} and w_{stats} have only a negligible effect on overall performance.

6.7 Case Study: Downstream Task (*RQ6*)

To assess the fidelity of data distribution preservation, we fix dirty rate at 20%, and employ both qualitative and quantitative measures: i) t-SNE visualizations [36] to inspect structural preservation, and ii) Dynamic Time Warping (DTW) distance [47] to quantify distributional similarity.

i) As shown in Fig. 12, the t-SNE visualization reveals distinct patterns of cleaning effectiveness. When cleaned data points (blue) closely overlap with ground-truth points (orange), this indicates optimal cleaning while preserving the higher-order structure. Conversely, visible separation between the colored clusters suggests either excessive smoothing of meaningful features or incomplete dirty elimination. Notably, the SHoTClean series demonstrates superior performance with near-perfect alignment between cleaned and ground-truth points, reflecting its exceptional capability to maintain both local neighborhood relations and global structure in the high-dimensional manifold.

ii) As shown in Fig. 13, the DTW analysis provides complementary quantitative evidence. The gray reference line represents the DTW distance between raw (uncleaned) data and ground truth. Most comparative methods perform at this baseline level, indicating substantial distortion of the original data distribution during cleaning. In striking contrast, SHoTClean achieves DTW values

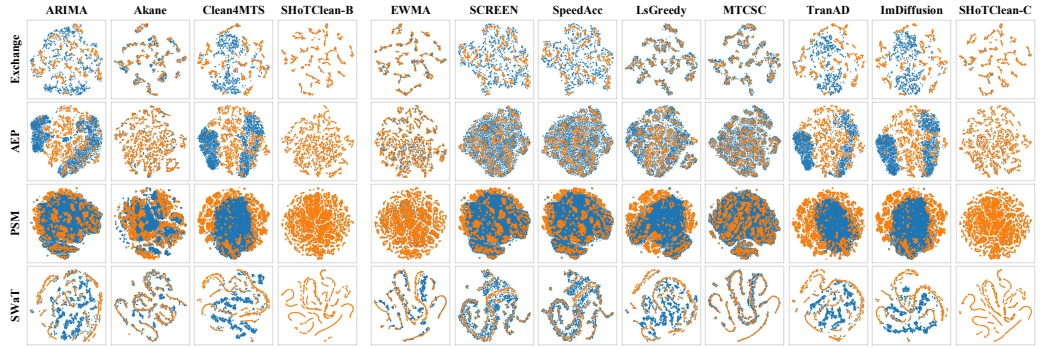


Fig. 12. Visualization by t-SNE on Multivariate Datasets (blue as clean, orange as truth)

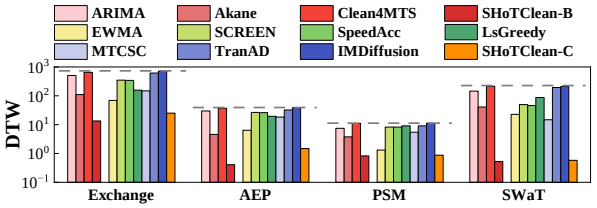


Fig. 13. Distribution Similarity via DTW Distance

consistently two orders of magnitude lower than competing methods, demonstrating its unique ability to simultaneously clean while preserving the essential features of the true data distribution.

6.8 Scalability Study (*RQ7*)

To assess the scalability of SHoTClean, we extend i) the hard constraint to enforce monotonicity and ii) the linear OLS predictor to nonlinear variants.

i) Hard Constraints Extension. To verify the generality, we extend the instantiation from speed to monotonicity constraints. Specifically, we assess SHoTClean-B on a monotonicity test derived from COVID-19 datasets [24], which record cumulative counts of confirmed cases, deaths, and recoveries. Since these data are strictly non-decreasing, we set $V_{\min} = 0$ to enforce monotonicity. As shown in Table 5, our method maintains strong RRA performance, confirming that it generalizes beyond speed constraints and can be adapted to other realistic forms of physical consistency. However, for higher-order constraints (e.g., acceleration constraint involving triplets of points), an extension would be required by augmenting the DP state to include two previous nodes and redefining feasible hyper-edges. We consider this an interesting direction for future work.

ii) Nonlinear Predictor Extension. In SHoTClean-C, we use a linear predictor (OLS) to compute residuals and strengthen the soft scores. To examine whether nonlinear predictors better capture inter-dimensional dependencies, we replace OLS with tree-based (Random Forest [11]), kernel-based (RBF [32]), neural network (MLP [45]), and information-theoretic (Transfer Entropy [48]) predictors. As shown in Fig. 14, these nonlinear predictors yield notable gains on strongly causal datasets (e.g., Exchange, UCI), but offer limited or even degraded performance on weakly or non-causal datasets (e.g., PSM, Porto). Moreover, nonlinear models incur substantial computational overhead, reducing scalability on large-scale, high-dimensional datasets. However, both linear and nonlinear predictors achieve state-of-the-art performance compared with other baselines (Table 4), validating the effectiveness of soft–hard constraints and causal discovery in data cleaning. In practice, we therefore recommend using OLS as the default predictor and introducing nonlinear predictors only when computational budget permits or when strong causal dependencies are expected.

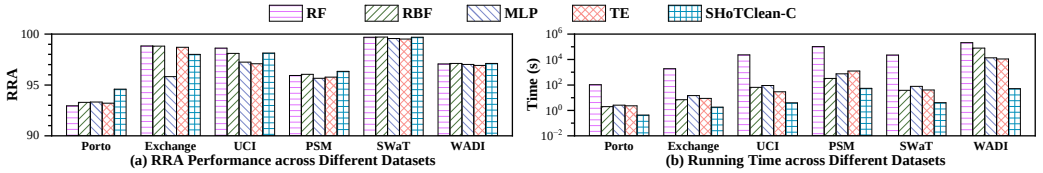


Fig. 14. Performance vs. Nonlinear Predictors

Table 5. RRA Performance Under Monotonicity Constraint

Country	Confirmed	Recovered	Deaths	Average
China	99.02	99.79	98.86	99.22
France	99.09	98.95	99.84	99.29
Russia	99.85	99.79	99.94	99.86
UK	99.78	98.04	99.99	99.27
US	99.82	98.3	99.85	99.92

7 Related Work

Smoothing-based methods constitute a fundamental class of techniques for time series data cleaning, primarily addressing numerical noise suppression in raw observations. The simplest implementation, the moving average (MA) [13], replaces each data point with the mean of values within a fixed sliding window. A more sophisticated variant, the exponentially weighted moving average (EWMA) [29], enhances this approach by applying exponentially decaying weights to historical observations to better capture recent temporal trends. Beyond these filtering techniques, autoregressive (AR) [57] models utilize historical data to predict current values, with dirty points detection performed through residual analysis. The combination of AR and MA components forms the ARMA model [4], while incorporating differencing yields the ARIMA framework [10], both widely adopted in time series cleaning applications [17, 19]. *While effective for dirty reduction, MA-based filters inherently violate two fundamental cleaning principles: the minimal fix principle [2] by uniformly modifying all points, and the minimal change principle [9] through indiscriminate smoothing. AR-based models rely on linearity assumptions rendering them inadequate for capturing nonlinear transitions or complex higher-order dependencies, leading to either under-detection of dirty points or over-smoothing of legitimate variations in highly dynamic real-world environments.*

Constraint-based methods (i.e., hard constraints) formulate time series cleaning as either a constraint-satisfaction or optimization problem by explicitly incorporating data-integrity rules and physical constraints derived from domain knowledge. The foundational work by Golab et al. [30] proposes the Sequence Dependency algorithm, which identifies and corrects consecutive value pairs violating predefined attribute dependencies. Following this direction, SCREEN [51] enforces precise speed constraints, and is extended to multi-speed constraints scenarios [28], while SpeedAcc [50] further incorporates acceleration constraints into it. *However, these methods struggle to detect small jumps and uncover latent causal or higher-order interactions among coupled variables. Most significantly, their computational complexity of the repair space search grows prohibitively as the number and dimensionality of constraints increase, severely limiting their applicability to real-time streaming scenarios where low-latency processing is essential.*

Statistical-based methods (i.e., soft constraints) address data cleaning by leveraging probabilistic frameworks to identify and correct dirty points while preserving underlying data distributions. For instance, IR-MHMM [6] utilizes a multivariate hidden Markov model to handle uncertainty in RFID time series by inferring the most probable hidden states. Similarly, LsGreedy [59] optimizes data cleaning by maximizing sequence likelihood to greedily select the best repairs—a principle subsequently validated and extended by multiple studies [43, 58]. Additional approaches include Wang et al.'s [54] probabilistic modeling of speed changes to distinguish legitimate variations from

anomalous outliers, and Akane's [31] adaptation of perplexity metrics to formalize time-series cleaning as a perplexity minimization problem. *Despite their effectiveness, statistical-based methods face practical limitations. First, these methods are largely confined to univariate data and linear relations, limiting their ability to capture complex nonlinear interactions or multivariate dependencies. Second, the computational complexity of techniques like hidden Markov models and perplexity optimization grows exponentially with data dimensionality or sequence length, making them unsuitable for real-time cleaning of large-scale high-dimensional time series data.*

Hybrid methods integrate both hard constraints and statistical modeling to achieve more precise detection and repair of dirty points at finer granularity. For example, MTCSC [60] extends speed constraint across multiple variables and integrates clustering to improve cleaning accuracy; MTSClean [21] and Clean4MTS [20] apply both row-level constraints (inter-variable linear relationships) and column-level constraints (speed, acceleration) for comprehensive data cleaning. Cleanits [23] augments sequence constraints with statistical correlations to achieve more precise missing-value imputation, while Clean4TSDB [22] introduces time-series dependency constraints to capture broader contextual relationships. *While demonstrating improved cleaning performance, current hybrid methods typically specialize in either high-precision dirty points detection or sophisticated repair mechanisms, rarely achieving both capabilities within a unified framework. This specialization creates practical challenges when handling complex real-world datasets requiring both sensitive detection and nuanced repair strategies.*

Anomaly-detection-based methods have proven effective for data cleaning tasks in recent studies [31, 55, 60]. TranAD [52] employs transformer-based encoder–decoder networks trained adversarially to reconstruct input sequence windows; ImDiffusion [18] masks portions of the raw data and then uses a diffusion model to predict the missing values. Both of these reconstruction- or prediction-driven methods can both detect anomalies and directly use the reconstructed or predicted sequences as cleaned output. Beyond these, distance-based [12] or clustering-based [25] anomaly detectors can offer cleaning suggestions by replacing outliers with the nearest clean point or the cluster centroid. *While effective at identifying contiguous dirty segments, they often demonstrate limited sensitivity to isolated dirty points. Second, their batch-based detection approach frequently results in over-cleaning by flagging entire sequences as dirty. Most significantly, their performance is particularly vulnerable to noise in training samples, which can compromise the reliability of both detection and cleaning outcomes.*

8 Conclusion

This study tackles fundamental challenges in multivariate time-series data cleaning: (1) accurate and efficient detection of subtle dirty points and (2) effective utilization of inter-variable dependencies. Specifically, we propose the SHoTClean series, through a novel constraint-optimization framework that harmoniously integrates soft and hard constraints. The experiments demonstrate that the SHoTClean series consistently outperforms baselines, including both offline and online applications, as well as univariate and multivariate datasets. In the future, we will incorporate probabilistic reasoning to increase the flexibility of soft constraints. Also, we plan to extend hard constraints such as acceleration to capture richer feature patterns and better adapt to real-world scenarios.

9 Acknowledgments

This work was supported in part by the NSFC under Grants No. (62402422, 62025206, U23A20296, and 62472377), Yongjiang Talent Introduction Programme (2024A-162-G), Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ25F020001, and Zhejiang Province's "Lingyan" R&D Project under Grant No. 2024C01259. Yunjun Gao is the corresponding author.

References

[1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. 2021. Practical Approach to Asynchronous Multivariate Time Series Anomaly Detection and Localization. In *SIGKDD*. 2485–2494.

[2] Foto N. Afrati and Phokion G. Kolaitis. 2009. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, Vol. 361. 31–41.

[3] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *CySWATER*. 25–28.

[4] Gérard Alengrin and Gérard Favier. 1978. New stochastic realization algorithms for identification of ARMA models. In *ICASSP*. 208–213.

[5] Naomi Altman and Martin Krzywinski. 2018. The curse (s) of dimensionality. *Nature Methods* 15, 6 (2018), 399–400.

[6] Asif Iqbal Baba, Manfred Jaeger, Hua Lu, Torben Bach Pedersen, Wei-Shinn Ku, and Xike Xie. 2016. Learning-Based Cleansing for Indoor RFID Data. In *SIGMOD*. 925–936.

[7] Richard Bellman. 1966. Dynamic programming. *science* 153, 3731 (1966), 34–37.

[8] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[9] Philip Bohannon, Michael Flaster, Wenfei Fan, and Rajeev Rastogi. 2005. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *SIGMOD*, Fatma Özcan (Ed.). 143–154.

[10] George EP Box and David A Pierce. 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association* 65, 332 (1970), 1509–1526.

[11] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[12] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *SIGMOD*. 93–104.

[13] David R. Brillinger. 2001. *Time series - data analysis and theory*. Classics in applied mathematics, Vol. 36. SIAM.

[14] California Independent System Operator. 2025. Open Access Same-time Information System (OASIS): System Demand [CA]. https://www.caiso.com/library/historical-ems-hourly-load. Accessed June 29, 2025.

[15] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. 2017. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings* 140 (2017), 81–97.

[16] Danqi Chen. 2008. On a Class of Divide-and-Conquer Techniques. https://oi-wiki.org/misc/cdq-divide/.

[17] Peiyuan Chen, Troels Pedersen, Birgitte Bak-Jensen, and Zhe Chen. 2009. ARIMA-based time series model of stochastic wind power generation. *IEEE transactions on power systems* 25, 2 (2009), 667–676.

[18] Yuhang Chen, Chaoyun Zhang, Minghua Ma, Yudong Liu, Ruomeng Ding, Bowen Li, Shilin He, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. ImDiffusion: Imputed Diffusion Models for Multivariate Time Series Anomaly Detection. *Proc. VLDB Endow.* 17, 3 (2023), 359–372.

[19] S Dilling and BJ MacVicar. 2017. Cleaning high-frequency velocity profile data with autoregressive moving average (ARMA) models. *Flow Measurement and Instrumentation* 54 (2017), 68–81.

[20] Xiaoou Ding, Genglong Li, Hongzhi Wang, Chen Wang, and Yichen Song. 2024. Time Series Data Cleaning Under Expressive Constraints on Both Rows and Columns. In *ICDE*. 3682–3695.

[21] Xiaoou Ding, Yichen Song, Hongzhi Wang, Chen Wang, and Donghua Yang. 2024. MTSClean: Efficient Constraint-based Cleaning for Multi-Dimensional Time Series Data. *Proc. VLDB Endow.* 17, 13 (2024), 4840–4852.

[22] Xiaoou Ding, Yichen Song, Hongzhi Wang, Donghua Yang, Chen Wang, and Jianmin Wang. 2024. Clean4TSDB: A Data Cleaning Tool for Time Series Databases. *Proc. VLDB Endow.* 17, 12 (2024), 4377–4380.

[23] Xiaoou Ding, Hongzhi Wang, Jiaxuan Su, Zijue Li, Jianzhong Li, and Hong Gao. 2019. Cleanits: A Data Cleaning System for Industrial Time Series. *Proc. VLDB Endow.* 12, 12 (2019), 1786–1789.

[24] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet infectious diseases* 20, 5 (2020), 533–534.

[25] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD-96*. 226–231.

[26] Peter M. Fenwick. 1994. A New Data Structure for Cumulative Frequency Tables. *Softw. Pract. Exp.* 24, 3 (1994), 327–336.

[27] José E Figueroa-López, Yankeng Luo, and Cheng Ouyang. 2014. Small-time expansions for local jump-diffusion models with infinite jump activity. (2014).

[28] Fei Gao, Shaoxu Song, and Jianmin Wang. 2021. Time Series Data Cleaning under Multi-Speed Constraints. *Int. J. Softw. Informatics* 11, 1 (2021), 29–54.

[29] Everette S Gardner Jr. 2006. Exponential smoothing: The state of the art—Part II. *International journal of forecasting* 22, 4 (2006), 637–666.

[30] Lukasz Golab, Howard J. Karloff, Flip Korn, Avishek Saha, and Divesh Srivastava. 2009. Sequential Dependencies. *Proc. VLDB Endow.* 2, 1 (2009), 574–585.

[31] Xiaoyu Han, Haoran Xiong, Zhenying He, Peng Wang, Chen Wang, and X. Sean Wang. 2024. Akane: Perplexity-Guided Time Series Data Cleaning. *Proc. ACM Manag. Data* 2, 3 (2024), 121.
[32] Rolland L Hardy. 1971. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research* 76, 8 (1971), 1905–1915.
[33] Lavender Yao Jiang, Xujin Chris Liu, Nima Pour Nejatian, Mustafa Nasir-Moin, Duo Wang, Anas Z. Abidin, Kevin Eaton, Howard Antony Riina, Ilya Laufer, Paawan Punjabi, et al. 2023. Health system-scale language models are all-purpose prediction engines. *Nature* 619, 7969 (2023), 357–362.
[34] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).
[35] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *SIGIR*. 95–104.
[36] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.
[37] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *CySWater*. 31–36.
[38] George B Moody, WE Muldrow, and Roger G Mark. 1984. A noise stress test for arrhythmia detectors. *Computers in cardiology* 11, 3 (1984), 381–384.
[39] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luís Damas. [n. d.]. Predicting Taxi-Passenger Demand Using Streaming Data. *IEEE Trans. Intell. Transp. Syst.* 14, 3 ([n. d.]), 1393–1402.
[40] Warren Buckler Powell. 2007. *Approximate Dynamic Programming - Solving the Curses of Dimensionality*. Wiley.
[41] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. In *Proc. VLDB Endow.* 2363–2377.
[42] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. 2025. DUET: Dual Clustering Enhanced Multivariate Time Series Forecasting. In *SIGKDD*. 1185–1196.
[43] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201.
[44] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *SIGKDD*. 3009–3017.
[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.
[46] Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. 2019. Inferring causation from time series in Earth system sciences. *Nature communications* 10, 1 (2019), 2553.
[47] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.
[48] Thomas Schreiber. 2000. Measuring information transfer. *Physical review letters* 85, 2 (2000), 461.
[49] Wei Shao, Ziquan Fang, Lu Chen, and Yunjun Gao. 2025. Towards Trajectory Anomaly Detection: a Fine-Grained and Noise-Resilient Framework. In *SIGKDD*. 2490–2501.
[50] Shaoxu Song, Fei Gao, Aoqian Zhang, Jianmin Wang, and Philip S. Yu. 2021. Stream Data Cleaning under Speed and Acceleration Constraints. *ACM Trans. Database Syst.* 46, 3 (2021), 10:1–10:44.
[51] Shaoxu Song, Aoqian Zhang, Jianmin Wang, and Philip S. Yu. 2015. SCREEN: Stream Data Cleaning under Speed Constraints. In *SIGMOD*. 827–841.
[52] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* 15, 6 (2022), 1201–1214.
[53] U.S. Bureau of Economic Analysis. 2025. Total Vehicle Sales [TOTALSA]. https://fred.stlouisfed.org/series/TOTALSA. FRED, Federal Reserve Bank of St. Louis. Accessed June 29, 2025.
[54] Haoyu Wang, Aoqian Zhang, Shaoxu Song, and Jianmin Wang. 2024. Streaming data cleaning based on speed change. *VLDB J.* 33, 1 (2024), 1–24.
[55] Xi Wang and Chen Wang. 2020. Time Series Data Cleaning: A Survey. *IEEE Access* 8 (2020), 1866–1881.
[56] Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. 2025. Enhancing Time Series Forecasting through Selective Representation Spaces: A Patch Perspective. In *NeurIPS*.
[57] Kenji Yamanishi and Jun'ichi Takeuchi. 2002. A unifying framework for detecting outliers and change points from non-stationary time series data. In *SIGKDD*. ACM, 676–681.
[58] Zhuoran Yu and Xu Chu. 2019. PIClean: A Probabilistic and Interactive Data Cleaning System. In *SIGMOD*. 2021–2024.
[59] Aoqian Zhang, Shaoxu Song, and Jianmin Wang. 2016. Sequential Data Cleaning: A Statistical Approach. In *SIGMOD*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). 909–924.

[60] Aoqian Zhang, Zexue Wu, Yifeng Gong, Ye Yuan, and Guoren Wang. 2024. Multivariate Time Series Cleaning under Speed Constraints. *Proc. ACM Manag. Data* 2, 6 (2024), 245:1–245:26.
[61] Xu Zhang, Zhengang Huang, Yunzhi Wu, Xun Lu, Erpeng Qi, Yunkai Chen, Zhongya Xue, Qitong Wang, Peng Wang, and Wei Wang. 2025. Multi-period Learning for Financial Time Series Forecasting. In *SIGKDD*. ACM, 2848–2859.

Received July 2025; revised October 2025; accepted November 2025