文章编号：0253-374X(2024)03-0462-10

DOI：10.11908/j.issn.0253-374x.22219

面向周期性工业时序数据的流式清洗系统

王　耀[1]，赵　炯[1]，周奇才[1]，熊肖磊[2]，陈传林[3]，张　恒[3]

（1. 同济大学 机械与能源工程学院，上海 201804；2. 同济大学 浙江学院，浙江 嘉兴 314051；
3. 上海地铁盾构设备工程有限公司，上海 200233）

摘要：为了高效清洗具有时序性、周期性等特点的工业数据，首先利用分布式组件设计了一套流式清洗系统，系统以Mosquitto作为采集数据的汇集中心，以Flume为连接组件，以Kafka为缓冲组件，对接数据清洗组件，使系统具有高吞吐、大缓冲等优势。然后基于速度约束模型，设计了一种周期性数据清洗算法，综合工业数据的时序性、周期性、物理意义等特性，在原有速度约束算法基础上增加周期性检测和数据切片机制，以解决速度约束算法处理周期性数据的失真问题，提高可用度。最后文中以盾构掘进数据集为样本，验证了系统和算法的有效性，以及改进算法的适用性。

关键词：数据清洗；工业大数据；时序数据；速度约束；周期性

中图分类号：TP391　　**文献标志码**：A

Streaming Cleaning System for Periodic Industrial Time Series Data

WANG Yao[1], *ZHAO Jiong*[1], *ZHOU Qicai*[1], *XIONG Xiaolei*[2], *CHEN Chuanlin*[3], *ZHANG Heng*[3]

(1. College of Mechanical Engineering, Tongji University, Shanghai 201804, China; 2. Tongji Zhejiang College, Zhejiang Jiaxing 314051, China; 3. Shanghai Metro Shield Machine Equipment & Engineering Co.Ltd, Shanghai 200233, China)

Abstract: To efficiently clean industrial time series with the characteristics of periodicity, a streaming data cleaning system was first designed using distributed components. The system employs Mosquitto for data gathering, Flume for connection, and Kafka for the buffer, which provides benefits of high throughput and a large buffer. The data cleaning component serves as the core of the system. Then, a periodic time series cleaning algorithm was proposed based on a constraint model. Integrating the characteristics of temporality, periodicity, and physical meaning, the methods of periodic detection and data slicing were added to the original speed constraint algorithm, so as to solve the distortion problem of the original algorithm and improve the availability to deal with periodic data. Finally, the effectiveness of the system and the improved algorithm was verified using a tunnel boring machine data set as a case study.

Keywords: data cleaning; industrial big data; time series data; speed constraint; periodic

“工业4.0”背景下，生产制造过程、工程装备状态监测等环节产生了海量数据[1-2]。工业大数据的分析与应用为企业的生产、管理，装备的智能控制、故障诊断提供了可靠的决策依据[3-4]。工业数据的分析效果又与数据质量紧密相关[5]，然而在工业环境中，采集的数据受环境、可靠性、通信等因素影响，缺失值、异常值、冗余等[6]脏数据已成为常态。使用含有异常的数据进行数据分析、建模等应用，会直接影响决策结果，导致模型失效[2,5]，因此数据在应用前需进行必要的清洗。而工业数据的清洗又独具特点，工业大数据区别于传统互联网大数据，时序性、关联性、周期性是工业大数据的显著特点[7-8]，同时工业数据来源于实际的工程装备，采集的数据具有物理含义。工业数据的时序性、关联性已成为工业数据清洗系统设计的关键因素[9]，对实时性较高的控制系统，还要求数据可实时处理。虽然异常数据会导致模型的不准确，但工业场景的异常值往往成为状态检测和健康评估的重要依据，如故障分析应用中异常数据价值较高，数据清洗系统还应当为异常数据的保留或修复提供妥善的选择。

目前，常见的工业数据清洗模式为离线清洗。离线清洗系统对实时数据先存储、后清洗，如Kumar

收稿日期：2022-05-12
基金项目：上海申通地铁集团有限公司科研计划(JS-KY21R003-3)
第一作者：王　耀，博士生，主要研究方向为工业大数据、智能盾构系统。E-mail：1910425@tongji.edu.cn
通信作者：赵　炯，副教授，博士生导师，工学博士，主要研究方向为远程设备智能维护、计算机网络。
E-mail：Jiong.Zhao@tongji.edu.cn

论文拓展介绍

等人[10]针对数据库设计的离线清洗系统，传统的离线方法无法适用于较高实时性的工业环境。在新兴分布式框架流行后，实时流式清洗系统逐渐成为工业数据处理的主流系统，陈志云[11]等人提出了一种基于Storm的工业流水线实时分析系统，该系统采用Storm + Kafka + Flume + HBase的实时流处理框架，证实了此类系统具有较好的可靠性与并行度。Geng[12]等人也利用Flume，Kafka，Hbase等框架搭建了工业数据分析系统，并证实了主流分布式框架能满足工业现场的数据采集、分析需求。

流式数据清洗系统通常集成了高精度、低偏差的清洗算法。工业时序数据的清洗算法通常以处理缺失值、错列、异常值等为目标[13-14]。其中异常值又分为连续异常、单点异常、平移异常等类型[15]。在来源于工程装备的时序数据中，单点异常是常见的一种。所谓单点异常，是指观测值和真实值具有较大偏差，且这种大偏差在一组时序数据中不连续出现。单点异常是工业现场较普遍的类型，常用的清洗方法有①基于平滑方法的清洗。利用数据拟合去除时间序列中的异常值，如移动平均法（SMA），加权平均法（WMA）等[16]。Jeffery等人基于滑动窗口法设计了在线数据流清洗系统[17]。通常平滑算法能够在时序数据清洗中发挥良好的效果，但清洗准确率较低，清洗时会影响部分正常数据导致整体清洗偏差增大。②基于约束的清洗。通过设计某种规则，调整异常数据使其满足设定的约束。Zhang等人[9,13]提出了速度约束清洗算法，通过限制一个时间段内的两个点数值的变化率，从全局或局部识别违反约束的点，利用异常点前后时刻的值对异常点修复。Song等人[18]在速度约束研究的基础上扩展了基于加速度约束的数据清洗方法。这种基于速度约束的清洗方法处理单点异常值具有很好的效果。但该方法在处理周期性数据时或周期跳变数据时产生数据失真，无法适用强周期性工业数据的清洗需求。③基于机器学习的清洗。通过构建聚类、回归、神经网络等模型识别并修复异常值[19-20]，机器学习的清洗算法虽然高效，但数据集中异常值偏少时，类别不均衡，对标签集的修复结果受样本特征影响，含有较多异常值的样本集会降低修复质量。

本文为了高效清洗具有时序性、周期性等特点的工业数据，设计了一套面向周期性工业时序数据的流式清洗系统和清洗算法，使系统在处理周期性突变数据时，也具有良好的数据修复效果。首先，利用Flume、Kafka等组件，构建一套流式数据处理系统，克服传统离线清洗系统先存储、后清洗的实时性差问题；其次，在构建的流式清洗系统基础上，使用基于速度约束的清洗方法[9,13]，并综合数据的时序性、周期性、物理意义，对速度约束模型做周期性数据的适应性优化，解决传统算法在清洗距离、准确度方面的不足，提升速度约束算法在周期性工业数据的适应性。最后通过试验验证所提周期性数据清洗算法的先进性。

1　工业时序数据的流式清洗系统架构

为满足工业数据先清洗、再存储的实时处理需求，本文使用主流的大数据框架，搭建了一套用于工业时序数据清洗、存储的流式处理系统。整套系统包括数据汇集组件，连接组件，缓存组件，清洗组件，存储组件，系统框架如图1所示。

数据汇集组件选用Mosquitto MQTT Broker，其位于系统下层，使用MQTT协议实现底层采集装置和数据清洗系统的通信。采集的数据须包含时间戳以及至少一列的有效观测数据。Mosquitto汇集并封装采集的数据，随后数据被发往数据连接组件进行后续流转。

数据连接组件选用Flume框架，其在清洗系统中的位置和业务逻辑如图1所示。汇集的数据由MQTT Source转入Interceptor组件做简单过滤，修复明显的异常值，并将初步清洗后的数据嵌入PreCleaned标签，由Sink送入缓存组件。过滤时任何不符合格式的数据、类型错误数据被嵌入Error标签直接进入存储组件。使用Memory Channel和多路复用结构，不同标签的数据分别发往Kafka进行清洗或直接存储。

数据缓存组件由Kafka集群组成，用于待清洗数据的缓存和系统解耦，适配高速产生的数据和清洗组件较慢的处理速率。图1右侧展示了其业务逻辑，每个数据采集装置使用一个Topic，每个Topic包含一组特定的时间序列数据。Producer为Flume组件的Sink端，用于将可用数据转入到缓存组件。Consumer为数据清洗组件，其周期性地从Kafka消费数据，完成对数据的清洗工作，结果写入数据存储组件。

数据清洗组件是数据清洗系统的核心部分。清洗组件从Kafka持续获取指定Topic下的数据，完成基于速度约束的数据清洗工作。消费者每隔一定间隔从Kafka消费数据，依据数据格式进行预处理，把原始数据封装成可清洗数据集；随后经过处理算法完成异常值检测和异常数据修复工作。

数据存储组件由HDFS、Hive、HBase构成。清

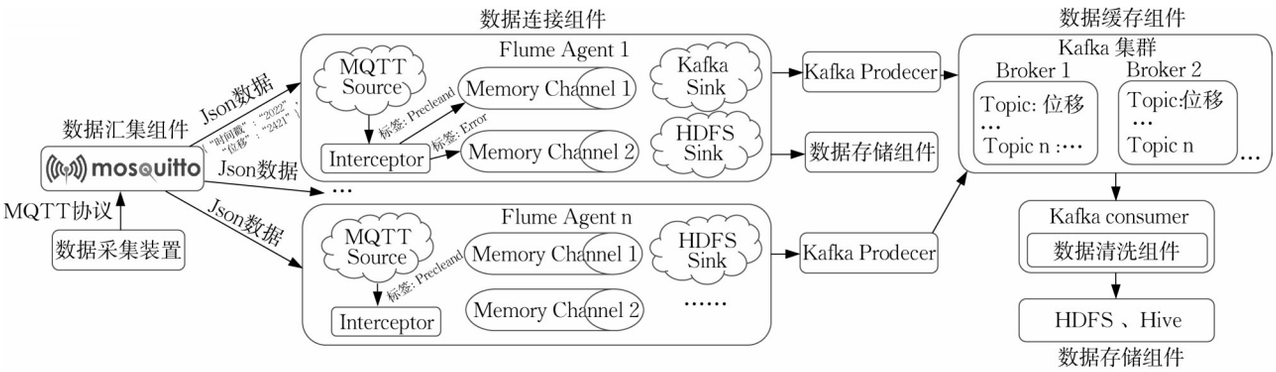


图1 清洗系统组件和架构

Fig.1 Component and architecture in the cleaning system

洗组件依据Flume连接阶段数据标签的差异,调用不同API写入存储组件。清洗系统修复所有不满足约束的异常数据,并存储修复结果。但被清洗组件检测为异常的数据以及被连接组件标记为Error的数据,会被冗余存储,方便应用于故障检测等工业应用;存储组件同时存储修复前异常值和修复后结果,为数据清洗后的数据分析工作提供保障。

清洗系统由上述组件构成,作为清洗算法的载体,能够实现海量数据的流式处理和清洗任务,相比传统清洗系统具有低耦合、可扩展、易维护、大缓冲等特点。

2 面向周期性时序数据的速度约束清洗方法

首先给出速度约束的概念:一组时序数据中,若两个时序点数值的变化率限制在一定范围内,则称两点满足速度约束。图2a展示了一组时间序列,设时间序列$X=x[t_1],x[t_2],x[t_3],\cdots\cdots$其中$t_i$为时间戳,$x[t_i]$是时间为$t_i$时的数值,简记为$x_i$。

考察图2a中左侧时间间隔为w的两个数据点,若两个数据点的值x_n,x_m之间满足式(1),则称两个数据点在窗口w内满足速度约束$s=(s_{\min},s_{\max})$。由于机械结构或运动构件具有惯性,正常工作时其物理数值通常不能剧烈跳变,变化率会被限制在某范围内,因此速度约束在工业场景中普遍存在。

$$s_{\min}\leqslant\frac{x_m-x_n}{t_m-t_n}\leqslant s_{\max} \tag{1}$$

式中:x_n,x_m为两个点的数值;t_n,t_m为数据点对应的时间戳;$s_{\min},s_{\max}$分别为两点速度约束的下限和上限。

2.1 基于速度约束的清洗方法

现有的速度约束清洗方法[9,13],使用了约束模型的中位数近似求解方法。考察图2a时间序列X中在时间窗口w内的两个点x_i和x_k,其中x_k为待修复的异常点。

根据速度约束条件,x_k应当在x_i的约束s所限制的范围内,图2a中短线表示的范围边界是数据点x_k在前时刻数据点x_i约束下的最大取值和最小取值,最大取值和最小取值构成的边界限制了x_k的取值范围。

再考查图2b中待清洗的数据点x_k,以x_k为起点的窗口w内有m个数据点。若已知$k+1$时刻,$k+2$时刻…$k+m$时刻的数据点,则可以依据速度约束回溯x_k的值。图2b中星号标记表示了x_k后面m个的数据点回溯x_k范围的上限$x_{k\min}$和下限$x_{k\max}$。

将x_k本身和x_k前时刻点约束范围端点以及后时刻点$x_{k+1},x_{k+2},\cdots,x_{k+m}$的回溯结果成候选结果集,记作$X_k$,则$X_k=$

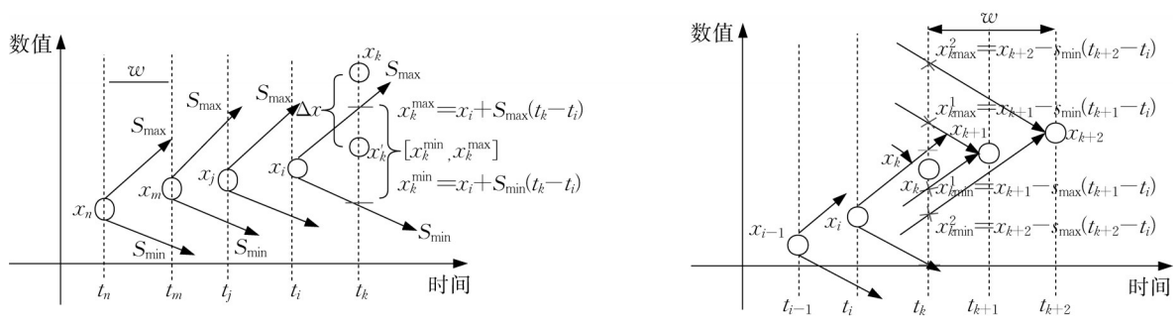
$$\{x_k,x_k^{\min},x_k^{\max},x_{k\min}^1,x_{k\max}^1,x_{k\min}^2,x_{k\max}^2,\ldots,x_{k\min}^m,x_{k\max}^m\}$$

为了满足最小修复距离原则,位于候选集X_k中间的数值通常具有较小的修复距离。把候选结果集X_k的中位数记作x_k^{mid},x_k^{mid}就作为x_k数据点的修复结果,图2b中点x_k即为修复结果,其满足速度约束。

2.2 周期性数据速度约束清洗算法

通常情况下,该方法能完成大多数清洗任务[13],但在部分工业环境中,机械装备或运动构件往往具有周期运转特性,采集的数据也呈现周期变化;往复运动的组件,其数值会有周期突变,例如图3a展示的盾构机油缸行程数据,在一个掘进周期中,油缸以设定的速度推进,完成一环任务时,油缸迅速返回,并为下一环的推进工作做好准备。

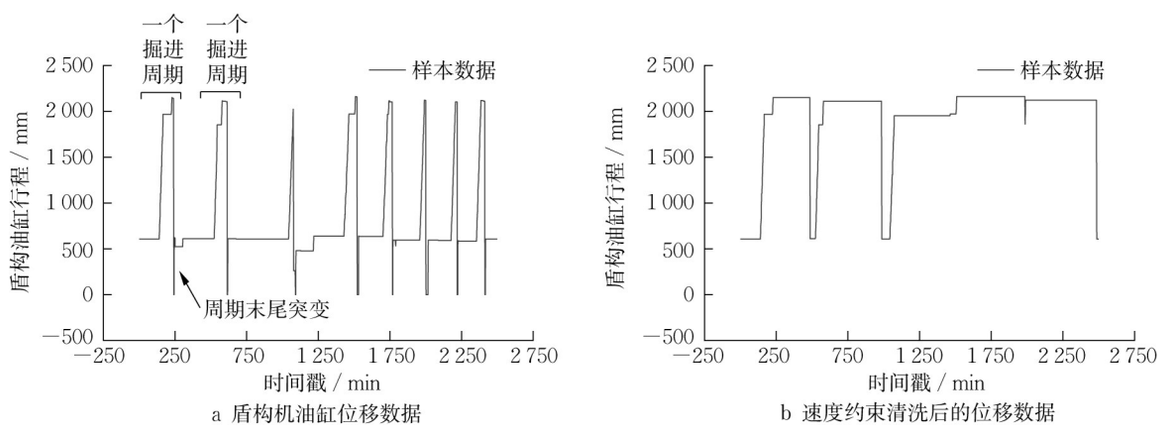
而速度约束方法用于此类设备的数据清洗时,不能获得良好的清洗结果。速度约束方法会把每个工作周期的数据按照连续数据清洗,导致两个周期间、周期末尾的数据点发生较大失真,见图3b。这是由于当清洗窗口包含两个周期的数据时,每个周期最后时刻的突变点会被视作窗口内的异常值,速度约束算法会将其修改为“合理值”,而实际上是油缸



a x_k受前时刻点约束　　b x_k受后时刻点约束

图2　时间序列和速度约束示意[9]

Fig.2　Time series and speed constraints



a 盾构机油缸位移数据　　b 速度约束清洗后的位移数据

图3　盾构机样本数据和清洗结果

Fig.3　Sample data of a tunnel boring machine and the cleaning results

的回退时造成了该数据的突变，这些数据应被作为真实数据而不能被清洗和修复。

为提高速度约束算法的适用性，对周期性数据清洗前，应先识别并划分数据的周期，限制窗口范围。以油缸行程数据为例，应仅对盾构推进时的数据进行清洗，油缸回退时的数据不应当参与速度约束求解。

算法1描述了本清洗系统中周期性数据速度约束清洗算法（Periodic Data Cleaning Based on Speed Method）的流程。数据清洗算法主要面向时序数据中的单点异常值，异常值的识别依据式(1)，不满足式(1)约束的被视作为异常值。清洗算法按照最小修复原则[6]，使得清洗后的序列X'内的所有点能够满足约束，并且清洗后的值与原始数据值尽可能接近。即在满足速度约束s条件下，为周期性时间序列X寻找修复距离最小的修复序列X'，其中修复距离指原始时间序列X中的点x和修复后的数据点x'的差别，记作$\triangle x$，如图2a所示。

算法模型以Kafka Consumer身份接入系统并作为清洗组件。清洗组件先依据Kafka配置连接到Kafka集群，再通过Kafka主题订阅待清洗数据。清洗组件每隔一段间隔调用数据拉取函数从集群中获取一批数据，记作DatasRecords。

随后，清洗组件调用预处理函数对每条数据Record做格式转换等预处理，预处理方式按照不同数据格式定制，预处理后的数据被暂存到RangofResult队列中等待清洗；当RangofResult中的数据达到一定窗口时进行数据清洗工作。数据清洗工作通过调用清洗周期时间序列函数完成，清洗后的结果被记录在CleanedDatas集合中。

算法1中的清洗周期时间序列函数会先用调用判断函数对队列RangOfResult的数据逐条判断，判断数据是否属于同一周期。周期判断的方法应依据现场数据进行特性化设计。本例中，有效工作周期和非工作周期数据间有明显的突变，对图3a所示的油缸行程数据可使用阈值判断法，大于特定值的数据属于同一周期。当检测到周期突变数据时，终止检测，并将终止前的一个切片数据记录在OneSplitPeriod队列中。本例还可通过数据跳变检测，数据突变检测等方式对数据进行周期性划分。对于周期性明确的数据，可通过限制清洗窗口来划分清洗周期。

数据被划分为周期片段后,调用算法2清洗周期片段函数清洗每一个切片数据。最后,清洗组件将算法2的返回结果写入存储系统。

算法1 数据清洗组件

Algorithm1 DataCleaningModule

输入 待处理的周期性时间序列

输出 清洗后的时间序列

主函数 数据清洗{

数据清洗消费者 = 新的Kafka Consumer(Kafka配置);

数据清洗消费者. 连接和订阅(待清洗数据主题);

循环{

DatasRecords = 数据清洗消费者. 从Kafka拉取数据(一段间隔);

对于DatasRecords中的每一个Record{

如果Record不为空{

Result = 预处理(Record);

RangofResult. 添加(Result);}}

如果RangofResult. 大小> 窗口大小{

CleanedDatas = 清洗周期时间序列(RangofResult);

清空RangofResult;}

写入数据存储组件(CleanedDatas);}}

函数 清洗周期时间序列(RangofResult){

对于RangofResult中的每一个Record{

归属的周期 = 判断(Record);

OneSplitPeriod. 添加(归属的周期, Record);}

如果(OneSplitPeriod 不为空){

清洗后的序列片段 = 清洗周期片段(OneSplitPeriod);

返回 清洗后的序列片段;}}

单个周期片段数据的速度约束清洗流程如算法2所示[9],先依据x_k的前时刻数据点x_{k-1},计算前时刻点的约束集$[x_k^{\min}, x_k^{\max}]$,再依据x_k后时刻m个点$x_{k+1}, x_{k+2}, \cdots$,计算x_k的后时刻回溯集。由前时刻点的约束集和后时刻点的回溯集组成候选集X_k,从候选集中求出中值x_k^{mid}。最后再判断中值与$[x_k^{\min}, x_k^{\max}]$的包含关系选出x_k的数据清洗结果。候选值x_k^{mid}与约束范围$[x_k^{\min}, x_k^{\max}]$只能有以下几种关系:属于关系,$x_k^{\mathrm{mid}} \in [x_k^{\min}, x_k^{\max}]$,则清洗结果为$x_k^{\mathrm{mid}}$;超过上限$x_k^{\mathrm{mid}} > x_k^{\max}$,清洗结果为$x_k^{\max}$;低于下限$x_k^{\mathrm{mid}} < x_k^{\min}$,清洗结果为$x_k^{\min}$。

算法2 清洗周期片段

Algorithm2 CleanAFragmentofTimeSeries

输入 一个时间序列片段;

输出 清洗后的时间序列片段;

清洗后的时间序列片段 = 新建 一个列表;

对于 一个时间序列片段 中的每一个 x_k{

候选集X_k = 新建 一个列表;

$x_k^{\max} = x_{k-1} + s_{\max} \times (t_{k-1} - t_k)$;

$x_k^{\min} = x_{k-1} + s_{\min} \times (t_{k-1} - t_k)$;

m个后续点 = 获取x_k后的m个数据点;

对于m个后续点 中的每一个 x_k^{next}{

$x_{k\max} = x_k^{\mathrm{next}} - s_{\max} \times (t_{\mathrm{next}} - t_k)$;

X_k. 添加($x_{k\max}$);

$x_{k\min} = x_k^{\mathrm{next}} - s_{\min} \times (t_{\mathrm{next}} - t_k)$;

X_k. 添加($x_{k\min}$);}

X_k. add(x_k); X_k. add($x_k^{\max}$); X_k. add($x_k^{\min}$);

x_k的清洗结果 = 判断包含关系(候选集X_k的中位数, $x_k^{\max}$, $x_k^{\min}$);

清洗后的时间序列片段. 添加(x_k的清洗结果);}

返回 清洗后的时间序列片段。

3 实验评估

为了测试周期性数据清洗算法用于周期时序数据的有效性,试验选取了来源于上海申通盾构集团的中铁CREC929-932系列盾构机掘进数据集进行模拟实验。原始样本集包含油缸行程、油缸推力等200维数据,选取周期性明显的A组油缸行程数据为清洗对象,再从此列数据中选取2 500个时序点组成原始序列,如图4所示。该组数据记录了盾构机的8环掘进过程,图4中8个上升阶段为每个推进过程,图4中的迅速下降阶段为油缸的回退过程。

所有实验程序用Java实现,其中数据生成程序会模拟工业现场底层采集单元的数据上传。系统中各组件和算法程序运行在Intel(R) Xeon(R) CPU E5-2603 v4 @ 1.70GHz的CPU和8GB内存的PC上,操作系统为CentOS Linux Release 7.8。5台上述规格的PC组成实验集群,编号为PC1~PC5,集群配置如表1所示,其中Zookeeper用于辅助Kafka集群管理元数据信息。

实验选用移动平均法SMA、加权平均法WMA、速度约束法[9,13]、随机森林、神经网络与周期性数据速度约束清洗算法(简记为周期性清洗算法)对比。其中机器学习模型对照组以待清洗的A组油

表 1　集群配置信息

Tab.1　Cluster configuration information

试验集群组件	组件部署节点
Hadoop－2.7.2 Namenode	PC1
Hadoop－2.7.2 Datanode	PC1、PC2、PC3、PC4、PC5
Mosquitto－1.4.14 MQTT	PC2
Flume－1.7.0	PC3
Kafka－0.11	PC2、PC3、PC4
Zookeeper－3.4.10	PC1、PC2、PC3、PC4、PC5
数据清洗组件	PC5

缸行程为标签集建模。模型评价指标包括:异常数据修复距离和,修复结果偏差和,清洗准确度,指标含义见3.2节。

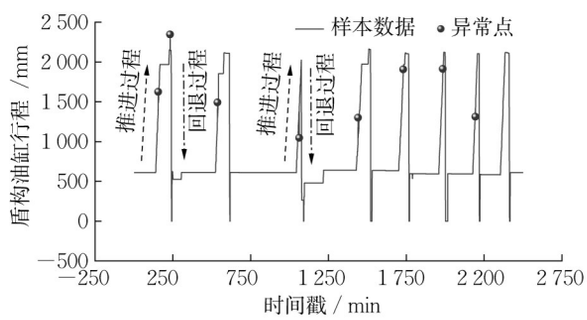
本实验中,因盾构推进中途油缸不可回缩,油缸行程的速度约束值下限选取为0。速度约束的上限依据施工组织方案中的施工计划设定,油缸推进速度最大为80mm·min^{-1}。对于无特定限制的其他系统可以采用统计方法选取合理的速度约束值。

实验使用异常点替换原始序列中的部分样本点,来模拟工业现场采集到异常数据的场景,验证系统的可用性和算法的有效性。异常点的位置会对数据的清洗效果产生影响,清洗系统面向周期性数据,系统会检测并划分一组数据的不同周期,本实验中异常数据只被添加在油缸每个工作周期的推进行程中。

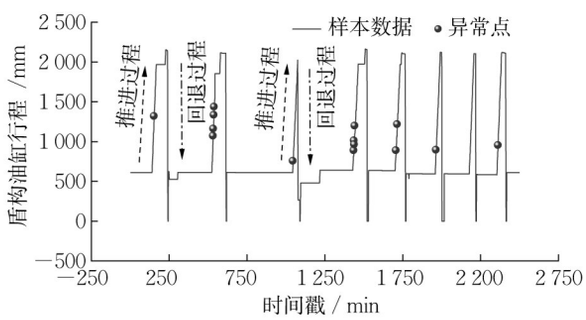
为体现算法在不同异常值分布下的泛化性,实验使用均匀分布和偏态分布模拟工程装备运转时出现的异常值。均匀分布用于模拟装备在正常工作阶段时普遍出现的异常值。异常数据的位置服从均匀分布,样本集中每个数据点等可能地出现异常;手动添加的均匀分布异常点与原本正常点数值上存在20%～30%的偏差。偏态分布用于模拟装备启动阶段的高概率异常。周期运转的大型装备,在工作循环的启动阶段,由于环境复杂,交叉作业多,易发生信号干扰,初始阶段异常值的出现概率高,正常工作阶段异常概率相对较低。在该场景下,实验使用Zipf分布模拟这种偏态分布;手动添加的偏态分布异常点与原本正常点数值上存在20%～30%的偏差。

3.1　清洗算法效果直观对比

首先直观地对比在两种分布下使用不同方法的清洗效果,分别在原始序列中添加1%的服从均匀分布的异常点和5%的服从偏态分布的异常点,图中圆点为手动添加的异常点,如图4所示。



a 油缸行程数据(含1%服从均匀分布的异常值)



b 油缸行程数据(含5%服从偏态分布的异常值)

图 4　油缸行程数据和异常值点示例

Fig.4　Cylinder stroke data and outliers

清洗系统运用不同算法清洗序列的结果如图5和图6所示。移动平均算法会将每个油缸推进周期末尾的突变数据及其临近点做平滑处理,影响了部分正常数据,如图5a、5b和图6a、6b所示。图5c和图6c为周期性数据清洗算法结果,算法通过数据切片,识别有效工作阶段的数据,在保留原始数据突变趋势的条件下较好地完成异常值的清洗。普通速度约束算法未对数据的周期性进行识别,当清洗窗口跨越不同周期时,窗口内周期末尾的突变会被判断为异常值,并受到速度约束(速度大于0)的限制,被速度约束方法修复,导致清洗结果偏离,见图5d和图6d。图5e、5f和图6e、6f展示了两种机器学习方法的清洗效果,机器学习方法能较准确地识别异常值,但对部分异常值的修复结果偏差大。

3.2　数据清洗结果指标对比

为了测试周期性数据清洗算法在不同异常数据占比时的清洗效果,对比各算法在异常点比例为1%、2%、5%…25%的8组样本条件下的异常数据修复距离和,修复结果偏差和,清洗准确度3种指标。

异常数据修复距离是指修复前数据与修复后数据的距离;修复距离越小,则系统越能反应数据原有的变化趋势,修复距离和是指所有点的修复距离加和,如图7所示。修复结果偏差是指修复后的数据点和真实数据点的距离,用于判断系统修复后的点和真实数据的偏差,偏差和距离越小,则系统清洗后的结果越能反应真实数据。清洗算法的准确度用被改动的数据比例表

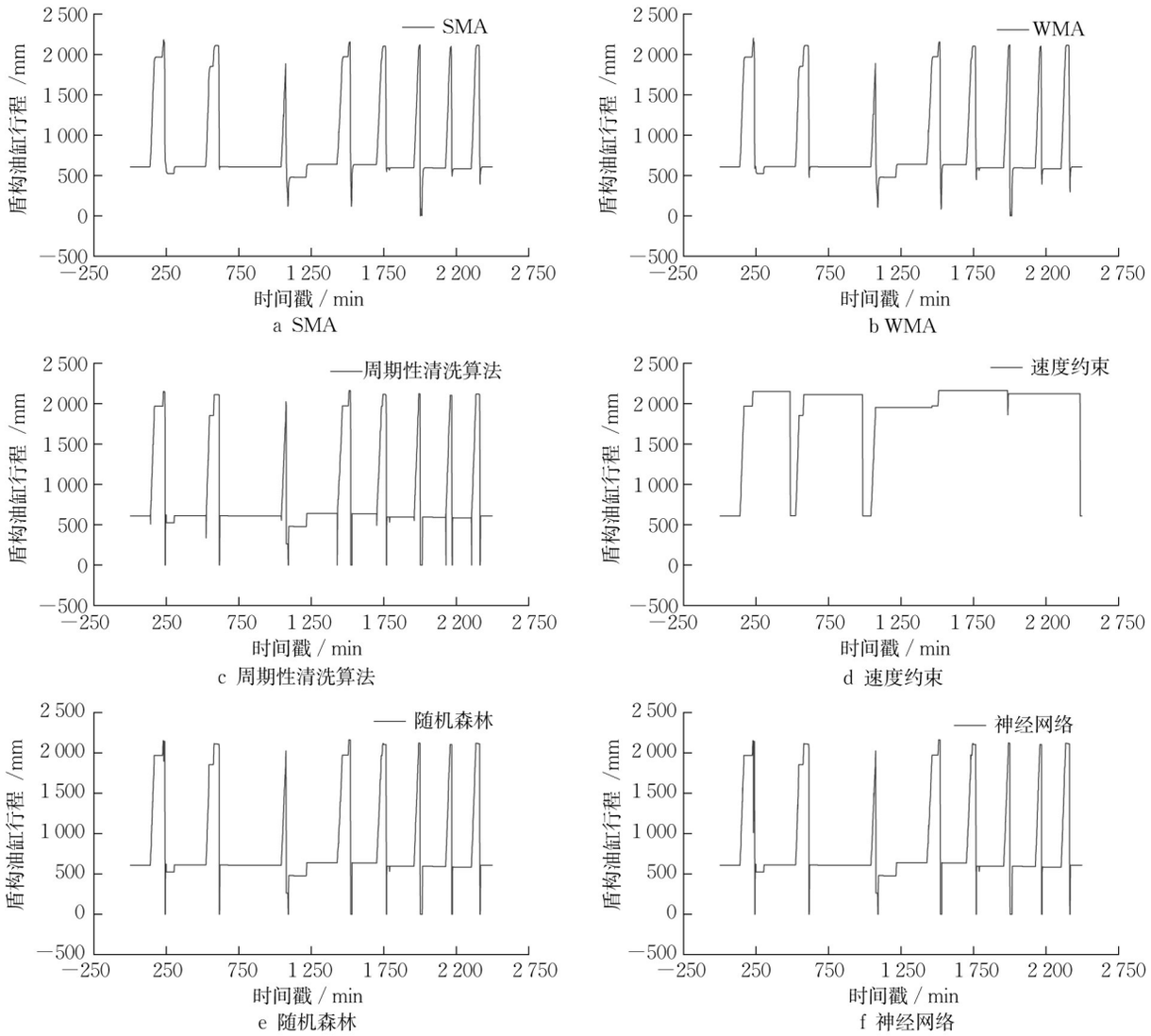


图5 系统应用不同清洗算法的直观清洗结果(含服从均匀分布的1% 异常值)

Fig.5 Cleaning results of different algorithms applied to the system (Including 1 % outliers obeying uniform distribution)

示,比例越少则说明清洗系统的修复结果对原本正常数据的影响越小,其清洗准确程度越高。

清洗系统使用不同算法的修复距离和如图8a和图8b所示。其中修复距离随着异常点的增多而增加,但周期性清洗算法比平滑算法SMA,WMA具有更小的修复距离,修复距离约为平滑算法的1/3或更低,能够更好的反应数据原有的变化趋势。普通速度约束算法的清洗结果由于偏差较大没有在图中展示。机器学习方法受异常值点分布位置的影响,当异常值服从均匀分布时,机器学习方法清洗含10%异常样本的修复结果高于含15%异常样本的结果,这是由于部分异常点位于周期突变处,产生了大偏差;当服从偏态分布时,异常值多数分布在前半周期,机器学习方法在异常值较少时具有较好的修复距离,当错误数据增加时修复距离明显增加。在两种异常值分布条件下,周期性清洗算法均比其他算法具有较小的修复距离。

所有被修复的数据点和真实数据点的偏差值如图8c和图8d所示,结果偏差随着异常点比例的增加而增加。运用平滑类方法时,周期末尾的突变数据放大了平滑方法的修复偏差,异常点的临近点会被清洗,增加了结果偏差。机器学习方法的结果偏差和受异常值分布影响大,不同异常分布时修复结果波动大。周期性清洗算法其偏差和约为平滑算法的1/3或更低,证实了通过周期切片的优化可较大程度上避免对正常数据的影响。周期性清洗算法在不同异常值分布下均具有较小的修复结果偏差,具有更好的泛化性。

不同算法导致数据变化的比例如图8e和图8f所

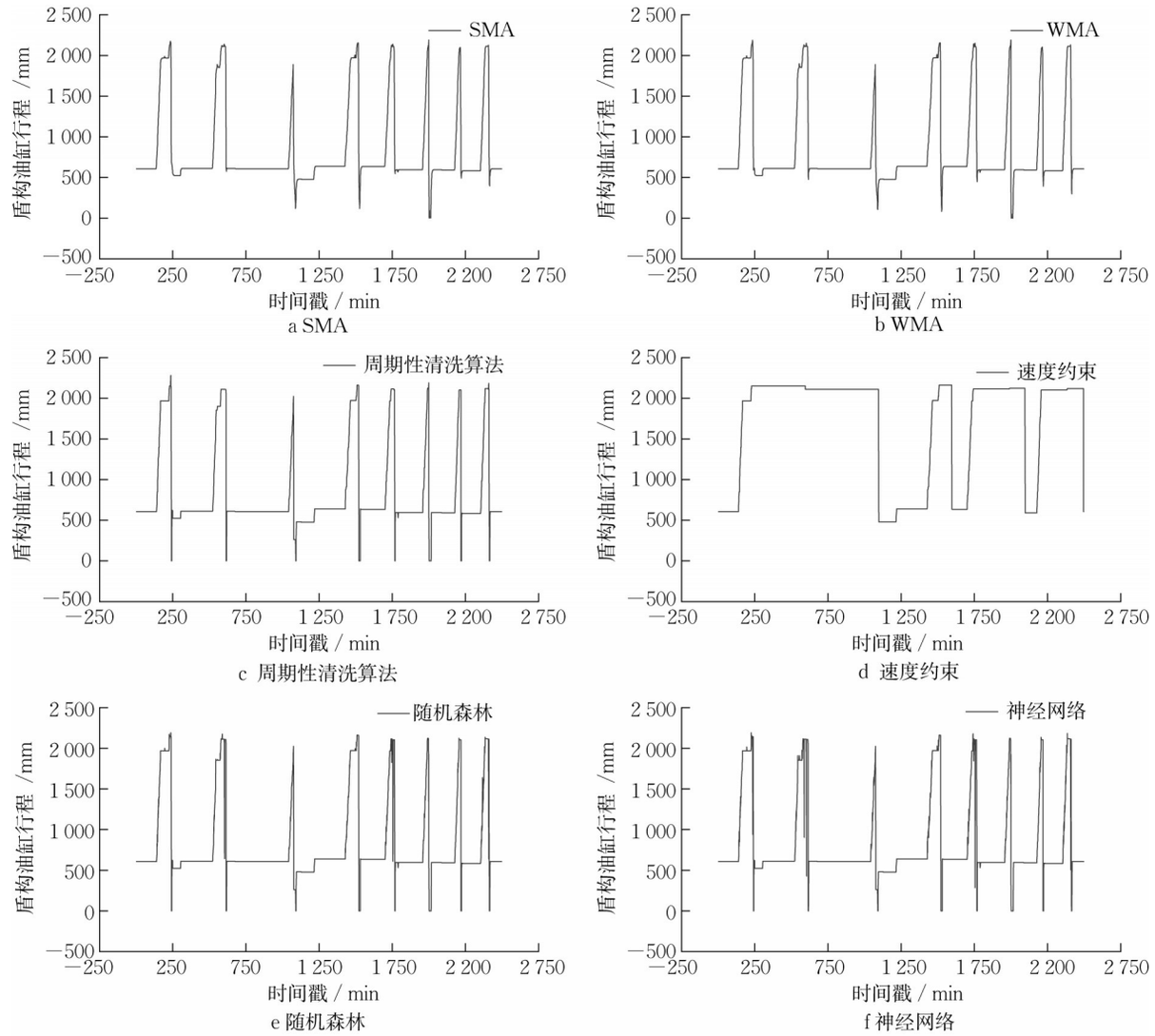


图 6　系统应用不同清洗算法的直观清洗结果（含服从偏态分布的 5% 异常值）

Fig.6　Cleaning results of different algorithms applied to the system (Including 5 % outliers obeying skewed distribution)

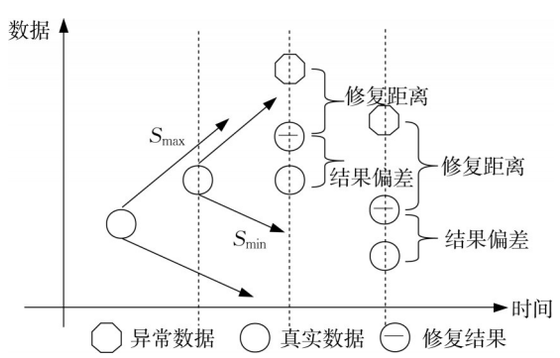


图 7　修复距离和、结果偏差和示意[9]

Fig.7　Repair distance and result distance

示。平滑方法会将异常点和突变点的临近点清洗，导致了数据清洗的准确度降低；但平滑方法使用均值逐个修复数据，在不同异常比例时均具有稳定的准确率，平滑方法对数据集的影响程度约 23%。普通的速度约束方法，会修复每个窗口内未满足约束的值，导致周期突变的原始数据被大量修改，对数据集样本点的影响率 80%。机器学习方法能较为准确的修复异常数据，但修复距离偏大。周期性数据速度约束算法克服了普通速度方法对突变值的缺陷，能较为准确的对错误数据进行清洗，当错误率增加时仍对样本数据影响较小。

综上，周期性数据清洗算法在用于周期性的工业数据修复时，且具有良好的泛化性，能用于不同类型的异常值分布。周期性数据清洗算法比平滑方法和普通速度约束方法有更小的修复距离，且修复结果能更好的反应数据变化趋势。通过周期识别与切分，提高了周期性清洗算法的准确度，对原本正确的数据影响较小。机器学习方法在异常值较少时清洗效果好，随着异常值比例增多修复效果波动大偏差增加，且受异常

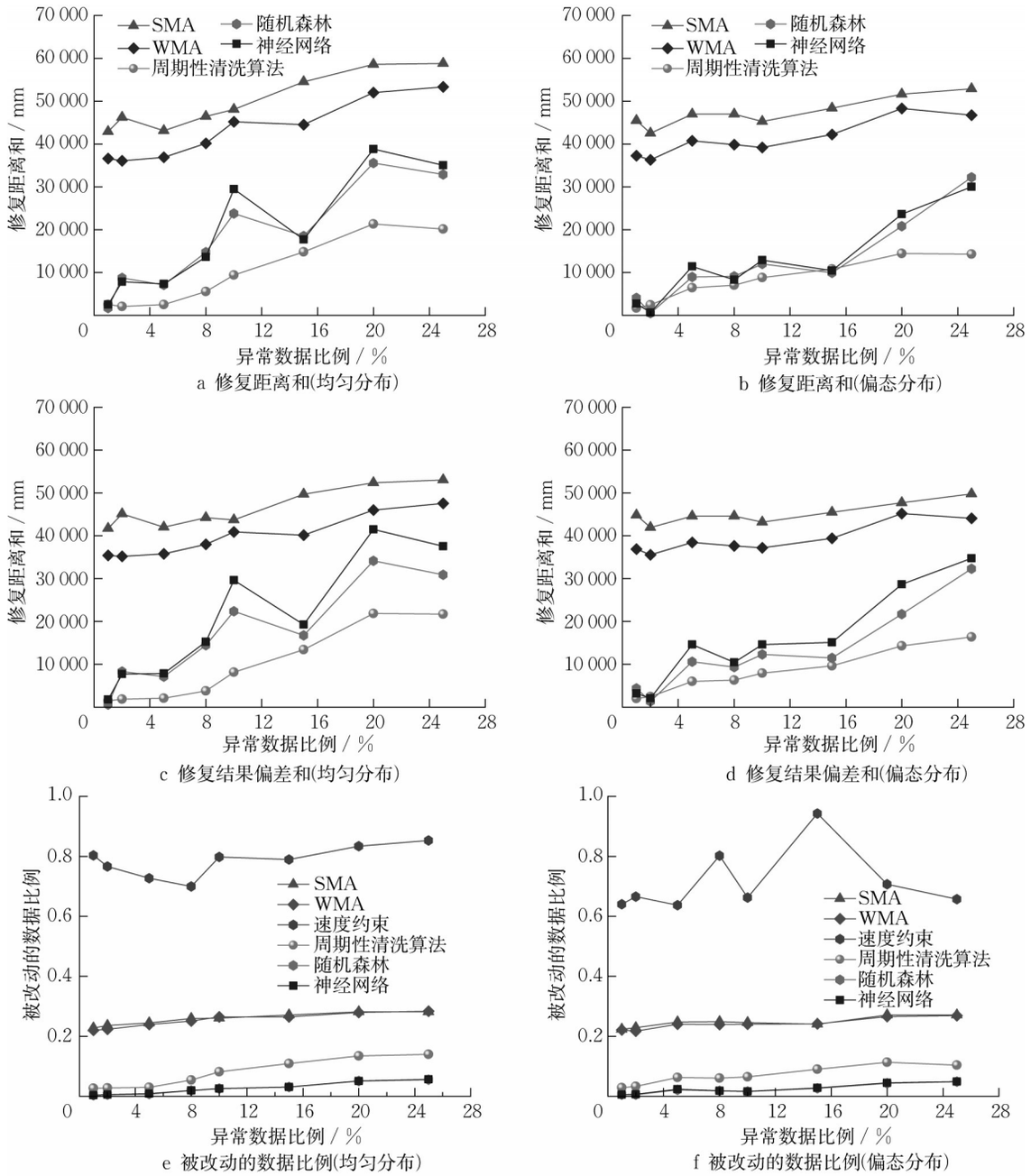


图8 清洗系统应用不同算法的修复质量指标对比

Fig.8 Comparison of repair quality indicators for different algorithms used in the cleaning system

数据分布影响,而周期性清洗算法在不同数据分布下均有良好的清洗效果。

4 结语

本文搭建了一套面向工业时序数据的流式清洗系统,系统以Mosquitto作为底层采集单元的汇集中心,通过Flume连接Mosquitto和Kafka实现整套系统的数据流转,将清洗后的数据进行存储或进分析。该系统相比传统的离线时序数据清洗系统,具有高吞吐,缓冲,解耦等优势。

本文还设计了一种基于速度约束的周期性数据清洗方法,在速度约束算法基础上将周期性数据进行周期识别和数据分片,解决速度约束算法用于周期突变数据的清洗失效问题,更好地完成具有周期性的工业数据清洗任务。最后通过实验验证了本系统在处理周期性工业数据时的可用性,验证了周期性清洗方法比原有速度约束、平滑算法有更小的修复距离和结果偏差,验证了周期性数据清洗算法对原始数据的影响比例更小,修复结果能更好的反应

真实数据和现有数据的变化趋势;在不同异常数据分布和占比条件下,周期性数据清洗方法仍具有较好的泛化性。

作者贡献声明:

王　耀:进行系统构建和算法设计,论文撰写;

赵　炯:拟定研究方向,项目规划和指导,论文修改和定稿;

周奇才:指导系统构建,指导论文修改;

熊肖磊:帮助数据分析;

陈传林、张恒:提供样本数据,分析实验场景,协助数据分析。

参考文献:

[1] 王建民. 工业大数据技术综述[J]. 大数据, 2017, 3(6): 3.
WANG Jianmin. Survey on industrial big data[J]. Big Data, 2017, 3(6):3.

[2] 丁小欧, 王宏志, 于晟健. 工业时序大数据质量管理[J]. 大数据, 2019, 5(6): 1.
DING Xiaoou, WANG hongzhi, YU Shengjian. Data quality management of industrialtemporal big data [J]. Big Data, 2019, 5(6): 1.

[3] 金晓航, 王宇, ZHANG Bin. 工业大数据驱动的故障预测与健康管理[J]. 计算机集成制造系统, 2022, 28(5): 1314.
JIN Xiaohang, WANG Yu , ZHANG Bin. Industrial big data-driven fault prognostics and health management[J]. Computer Integrated Manufacturing Systems, 2022, 28(5): 1314.

[4] GIEREJ S. Big data in the industry — overview of selected issues[J]. Management Systems in Production Engineering, 2017, 25(4): 251.

[5] CHU Xu, IHAB F I, KRISHNAN S, *et al*. Data cleaning: overview and emerging challenges [C]//Proceedings of the 2016 ACM SIGMOD. San Francisco California USA: International Conference on Management of Data, 2016: 2201-2206.

[6] IHAB F I. Effective Data cleaning with continuous evaluation [J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2016, 39(2): 38.

[7] DING Xiaoou, WANG Hongzhi, SU Jiaxuan, *et al*. Cleanits: a data cleaning system for industrial time series[J]. Proceedings of the VLDB Endowment, 2019, 12(12): 1786.

[8] 郑树泉, 覃海焕, 王倩. 工业大数据技术与架构[J]. 大数据, 2017, 3(4):14.
ZHENG Shuquan, TAN Haihuan, WANG Qin. Industrial big data technologies and architecture [J]. Big Data, 2017, 3(4): 14.

[9] SONG Shaoxu, ZHANG Aoqian, WANG Jianmin, *et al*. SCREEN: stream data cleaning under speed constraints[C]// Proceedings of the 2015 ACM SIGMOD. Melbourne Victoria [S. l.]: International Conference on Management of Data, 2015: 827-841.

[10] BHATTACHARJEE A K, PARTHA C, SHAW M P, *et al*. ETL-based cleaning on database [J]. International Journal of Computer Applications, 2014, 105(8): 34.

[11] 陈志云, 肖楚乔. 基于Storm的工业流水线实时分析系统设计与实现[J]. 计算机应用与软件, 2017, 34(11): 48.
CHEN Zhiyun, XIAO Chuqiao. Design of industrial assembly line real-time analysis system based on storm and its implementation [J]. Computer Applications and Software, 2017, 34(11): 48.

[12] GENG Daoqu , ZHANG Chengyun , XIA Chengjing, *et al*. Big data-based improved data acquisition and storage system for designing industrial data platform[J]. IEEE Access, 2019, 7: 44574.

[13] 张奥千. 时间序列数据清洗方法研究[D]. 北京: 清华大学, 2018.
ZHANG Aoqian, Research on time series data cleaning[D]. Beijing: Tsinghua University, 2018.

[14] 丁小欧. 时态数据清洗关键技术研究[D]. 哈尔滨: 哈尔滨工业大学, 2021.
DING Xiaoou. Research on key technologies of temporal data cleaning[D]. Harbin: Harbin Institute of Technology, 2021.

[15] WANG Xi, WANG Chen. Time series data cleaning: a survey [J]. IEEE Access, 2020(8): 1866.

[16] 陈翅刚. 制造物联网海量RFID感知数据智能清洗处理技术研究[D]. 广州: 广东工业大学, 2014.
CHEN Chigang. Research on clean technology of RFID sensing data of manufacturing in IoT [D]. Guangzhou: Guangdong University of Technology, 2014

[17] JEFFERY S R, ALONSO G, FRANKLIN M J, *et al*. A pipelined framework for online cleaning of sensor data streams [C]//IEEE 22nd International Conference on Data Engineering (ICDE'06). Atlanta Georgia USA: IEEE, 2006: 140-140.

[18] SONG Shaoxu, GAO Fei, ZHANG Aoqian, *et al*. Stream data cleaning under speed and acceleration constraints [J]. ACM Transactions on Database Systems (TODS), 2021, 46(3): 1.

[19] 覃华, 苏一丹, 李陶深. 基于遗传神经网络的数据清洗方法[J]. 计算机工程与应用, 2004(3): 45.
QIN Hua, SU Yidan, LI Taoshen. A data cleaning method based on genetic algorithm and neural network[J]. Computer Engineering and Applications, 2004(3): 45.

[20] 王科昊. 基于Spark系统的数据异常检测与修复方法的设计与实现[D]. 北京: 北京邮电大学, 2019.
WANG Kehao. The design and implementation of data anomaly detection and repaired method based on spark[D]. Beijing: Beijing University of Posts and Telecommunications, 2019.