Anomaly Detection in Catalog Streams

Chen Yang, Zhihui Du, *Senior Member, IEEE,* Xiaofeng Meng, *Member, IEEE,* Xukang Zhang, Xinli Hao, and David A. Bader, *Fellow, IEEE*

Abstract—Detecting anomalies with high accuracy and real time from large amounts of streaming data is a challenge for many real-world applications, such as smart city, astronomical observations, and remote sensing. This article focuses on a special kind of stream, catalog stream, whose high-level catalog structure can be used to analyze the stream effectively. We first formulate the anomaly detection in catalog streams as a constrained optimization problem based on a catalog stream matrix. Then, a novel filtering-identifying based anomaly detection algorithm (*FIAD*) is proposed, which includes two complementary strategies, true event identifying and false alarm filtering, data-oriented general method and domain-oriented specific method together, to detect truly valuable anomalies. Furthermore, different kinds of attention windows are developed to provide corresponding data for various algorithm components. A scalable and lightweight catalog stream processing framework *CSPF* is designed to support and implement the proposed method efficiently. A prototype system is developed to evaluate the proposed algorithm. Extensive experiments are conducted on the catalog stream data sets from an operational super large field-of-view high-cadence astronomy observation. The experimental results show that the proposed method can achieve a false-positive rate as low as 0.04%, reduces the false alarms by 98.6% compared with the existing methods, and the latency to handle each catalog is 2.1 seconds (much less than the required 15 seconds). Furthermore, a total of 36 transient candidates, including seven microlensing events, 27 superflares, and two dual-superflares, are detected from 21.67 million stars (involving 1.09 million catalogs) from one observation season.

Index Terms—streaming data analysis, anomaly detection, distributed stream processing, big scientific data.

1 Introduction

THANKS to the development of large-scale advanced data facilities [1], [2], [3], monitoring of massive targets (or hotspot locations) has become feasible. Such data facilities will continuously generate enormous amounts of data, hence the stream for the data. Some critical values of the data will be lost after a small time window. Through anomaly detection [4] in the monitoring data, we can identify emergency events in real time, which may have a significant impact on our daily life, national security and even human beings' safety. Figure 1 gives some examples of such applications. All such applications may generate large-scale data streams.

Due to massive monitored targets, these applications usually generate a special stream whose data are organized as many catalogs. One catalog has many targets, and all the targets have the same timestamp. It is natural to introduce a high-level catalog structure into a large data stream for two significant reasons. The first is that the data are often collected from independent physical devices, and different devices will produce different data catalogs. The second reason is that when many targets are organized into one catalog, it will be easy to manage millions or even billions of targets efficiently. Here, we name such streams as *catalog streams*. The targets could be very different in their physical or logical characteristics in different applications. As shown in Figure 1, for transient discovery in time-domain astronomy [5], targets are millions of stars; for light analysis in nighttime remote sensing, targets are cities [6]; for bacterial growth observation in microbiology, targets are many colonies [7]; for traffic flow monitoring in smart cities, targets are many user-defined small areas, such as road intersections [8]. However, all of them can be abstracted as a catalog stream anomaly detection problem.

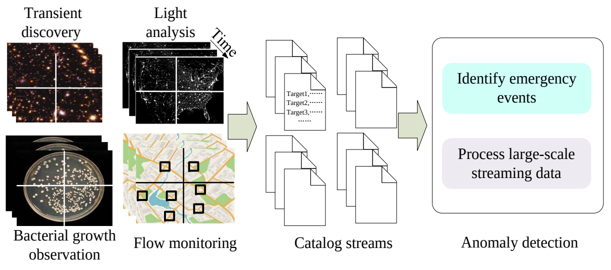


Fig. 1: Application examples of catalog streams. Here each application includes four cameras and each camera will generate an independent catalog stream.

Generally, there are two essential and challenging requirements when identifying anomalies in catalog streams and processing data from these emerging applications, *high accuracy* and *real time*.

In catalog streams, effective anomaly detection often faces three challenges. (1) High noise usually causes anomaly detection methods to contain many false-positive events. (2) Data missing, which is hard to avoid in catalog streams, may cause irregular and unpredictable data series and trigger false alarms. (3) The high-quality requirement for an anomaly event, which means detecting the continuous procedure of an anomaly event instead of a few discrete points (high coverage), detecting anomaly events at

- *Chen Yang is with China National Clearing Center and Department of Computer Science and Technology, Tsinghua University, Beijing, China.*
- *Zhihui Du and David A. Bader are with Department of Data Science, New Jersey Institute of Technology, Newark, USA.*
- *Xiaofeng Meng (Corresponding author), Xukang Zhang, and Xinli Hao are with Information School, Renmin University of China, Beijing, China.*

their early stage (high instantness), and selecting interesting anomaly events (high value) is critical to trigger follow-up actions. So, high accuracy means that our anomaly detection algorithm should address these data challenges to identify anomalies from vast streaming data. Reducing the false-positive rate is critical to achieving accuracy. However, reducing the false-positive rate is not trivial because it is hard to distinguish a special event from outliers (maybe caused by noise, data missing, or uninteresting events). For example, when we employ the existing anomaly detection method, the 3.19% false-positive rate in our experiment will cause too many false alarms due to the enormous volume of data. The high false-positive rate makes it impossible to conduct efficient follow-up action and data analysis.

Catalog streams may vary widely in size due to the difference in monitoring area and time, and they may have duplicated targets between different catalogs due to overlapping monitoring areas (a more detailed example in Figure 2). So, real time means that our stream processing framework should be able to finish all the stream analysis procedures with a given time constraint using commercial off-the-shelf computing resources at a low cost when catalog streams change dynamically. Under practical scenarios, the single-node analysis often cannot meet actual performance requirements. Also, this framework should identify duplicated targets without losing real-time performance. So, it is another challenging problem to develop a scalable parallel stream processing framework and lightweight algorithm implementation to achieve real-time performance.

We have designed a very efficient algorithm to identify anomalies with a very low false-positive rate by taking advantage of the intrinsic structure feature in catalog streams and domain-specific knowledge. Furthermore, we develop a highly efficient stream processing framework to achieve real-time anomaly detection that is critical in many applications. The major contributions are as follows.

- We formulate the problem of anomaly detection in catalog streams as a constrained optimization problem on a catalog stream matrix. The high accuracy and real-time constraints make this problem a grand challenge in many emerging applications.
- A novel catalog stream anomaly detection algorithm (*FIAD*) is proposed. The critical idea of *FIAD* is employing two complementary methods to develop efficient algorithm components on the suitable data provided by different attention windows to achieve high accuracy.
- A scalable and lightweight framework (*CSPF*) is proposed to enable a distributed and shared-nothing real-time stream processing and online duplicate detection. *CSPF* is critical to achieving real-time catalog stream analysis.
- Extensive experiments are conducted to evaluate the proposed method using a publicly available astronomy observation data set. The proposed method was implemented in a prototype system and detected 36 transient candidates that may lead to scientific discoveries from one season's observation data set.

The rest of the paper is organized as follows. Section 2 presents the detailed problem formulation. Section 3 gives an overview of our methods. Section 4 and Section 5 present the implementation of both *FIAD* and *CSPF*, respectively. Section 6 shows our experimental results. Section 7 is the related work. Section 8 discusses how to adapt our method to more application fields. Section 9 summaries our work.

2 Problem Formulation

In this section, we first define the concept of a catalog stream and then model all catalog streams of an application using a catalog stream matrix. Finally, we formulate the problem as a constrained optimization problem.

2.1 Catalog Stream

A catalog stream CS is a data element series

$$CS =< E^{t_1}, E^{t_2}, ..., E^{t_L} > . \tag{1}$$

Each data element E^{t_i} is a catalog. E^{t_L} is the newly arriving catalog. For any two Element E^{t_i} and E^{t_j}, if $i < j$, it means that the element E^{t_i} was generated earlier than element E^{t_j}.

A major difference between a catalog stream from previous streams is that each element of a CS will include a huge amount of targets instead of a single target. Compared with other streams, this fact increases the big data challenge of a catalog stream. Each element E^{t_i} can be expressed as a tuple $E^{t_i} =< TgS^{t_i}, SInfo^{t_i} >$, where $TgS^{t_i} \subseteq TgS = \{Tgt_1, Tgt_2, ..., Tgt_Z\}$ is a subset of the complete target set TgS that includes maximum Z targets in the catalog stream CS and $SInfo^{t_i}$ is the common information shared by all the targets in TgS^{t_i}.

A catalog stream CS can be divided into many target streams, corresponding to different targets. For the sake of generality, $\forall Tgt_x \in TgS$, it is often expressed as a vector that includes many different features instead of one single feature. So, the target stream TS_x for the target Tgt_x can be expressed as follows. $TS_x = \{FV^{t_1,x}, FV^{t_2,x}, ..., FV^{t_L,x}\}$, where $FV^{t_i,x}$ is the feature vector for target Tgt_x at time t_i. The number of targets collected in each catalog can be dynamically changing, not a constant. The set of all target streams in the same catalog stream CS can be expressed as $TSS = \{TS_1^{CS}, TS_2^{CS}, ..., TS_Z^{CS}\}$.

In one single catalog E^{t_i}, $\forall Tgt_x, Tgt_y \in TgS^{t_i}$, their corresponding feature vector $FV^{t_i,x}, FV^{t_i,y}$ can often be correlative with each other because some environment or system factors exist at the same time. Noise in one catalog can often affect many targets concurrently. We call such noise as intra-catalog *concurrent noise*. Concurrent noises are a major feature of a catalog stream, and they are unpredictable and usually have very irregular patterns. Generally, they do not follow a stationary distribution, so they cannot be easily filtered out through the traditional noise filtering methods.

A catalog stream may miss some data at different time points or even different intervals. If $Tgt_x \in TgS$ but $Tgt_x \notin TgS^{t_i}$, it means that the data $FV^{t_i,x}$ of target Tgt_x is missing at time t_i. Specifically, when $TgS^{t_i} = \emptyset$, the complete catalog E^{t_i} is missing at t_i. This is another difference between a catalog stream and a standard time series. Since a catalog stream sequence can be unequally spaced points in time, many existing analysis methods cannot handle the proposed catalog streams directly.

2.2 Catalog Stream Matrix

Assume totally N different devices in a set $D = \{D_1, ..., D_N\}$ can be used to collect data from millions or even billions of targets distributed in an area $A = \{A_1, ..., A_M\}$, then a catalog stream matrix $CSM = [CS_{D_n,A_m}]_{N\times M}$ can be built. Any entry CS_{D_n,A_m} represents a catalog stream generated by the device D_n from area A_m. Although N catalog streams can be generated at any specific time, the maximum number of total catalog streams during a long period is $N \times M$. Under practical scenarios, many of the entries will be empty because one device often focuses on one or a few areas instead of all the possible areas.

At the same time, the set of catalogs generated from different devices is called a *heterologous* catalog set. The heterologous catalog set defines the total workload at a given time. For any two catalogs in a heterologous catalog set, the targets in their catalogs can be completely different. However, if their catalogs are collected from the same area, the targets in their catalogs may be the same or almost the same; if their catalogs are collected from close areas, the targets in their target sets may overlap. This situation is called *inter-catalog overlapping*. Inter-catalog overlapping may cause different anomalies detected from *heterologous* catalogs to point to the same target.

2.3 Optimization Model

Given a catalog stream matrix CSM, let AS be the set of detected anomalies. Then, our problem can be formulated as the following optimization problem.

$$\textit{NumOfAnomalies} = \arg\max_{AS} |AS| \quad (2)$$

$$\text{subject to} \quad \textit{fpr}(AS, CSM) < r \quad (3)$$

$$\textit{latency}(CSM) < l \quad (4)$$

where r and l are the requirements for the false-positive rate and the latency. $\textit{fpr}(AS, CSM)$ is the false-positive rate for detecting AS from CSM and $\textit{latency}(CSM)$ is the longest time used to process any catalog for a given CSM. So, our optimization objective is to identify as many anomalies as possible, but the following constraints must be met. First, the false-positive rate should not be larger than the given value r. The second is that the longest time used to process a catalog should be no longer than the real-time requirement l. Under practical scenarios, often very small r and l are given, so these two constraints are the major challenges we face when identifying interesting anomalies. This paper proposes a novel anomaly detection algorithm and a scalable and lightweight framework to handle the two challenges.

3 Proposed Approach

In this section, we aim to give an overview of our method. The basic idea of the proposed algorithm *FIAD* is proposed first. Next, the concept of an attention window that aims to provide appropriate data for our algorithm components is presented. Then the real-time data processing framework *CSPF* is described. Finally, a typical application on catalog streams is given as the target of system development.

3.1 A Novel Catalog Stream Anomaly Detection Algorithm

The goal of *FIAD* is to identify as many true events as possible while suppressing as many false alarms as possible. The novelty and the core idea of *FIAD* are that it successfully integrates two complementary methods to achieve this goal.

The first complementary method integrates identifying methods and filtering methods. The second complementary method integrates data-oriented general methods with domain-oriented specific methods. Four different algorithm components are developed to solve different sub-problems based on the two kernel complementary ideas.

The four proposed algorithm components have been carefully arranged to achieve the best results. *Building high-quality data* is the first algorithm component and it takes advantage of the intra-catalog correlative feature to remove high noise data so the following components can work on high-quality data to achieve a better effect. *Data-oriented anomaly detection* is the second algorithm component and it picks up the true event candidates based on the features of given data. *Missing data downside removal* is the third algorithm component and it uses the statistical characteristics of streaming data to remove false positives caused by missing data. Finally, *domain-oriented anomaly detection* is the fourth algorithm component and it selects candidates with high domain values as the final alarms based on the domain-specific characteristics of the catalog streams. We will describe the four algorithm components as follows.

Building high-quality data. The level of concurrent noise that may cause outliers in the same catalog is uneven. Irregular effects of noise area and level should be isolated separately. *Building high-quality data* implements a stepwise clustering method to solve the problem. We first cluster the targets in a catalog into several large sets based on distance (spatial clustering) because closer targets are likely to be affected by the same noise. Then, we further cluster the targets in a big set into smaller clusters based on their features (feature clustering) because the noise will have a similar effect on the targets with similar features. Finally, targets in each small cluster share the same threshold to measure their concurrent noises. Setting a suitable threshold value to remove most of the high noise data is critical. Not all clusters can share the same threshold value because the same noise may have a very different effect on different clusters. We treat threshold selection as a Peaks-Over-Threshold (POT) problem [9], so the appropriate threshold can be learned automatically without the need to make strong assumptions about the data distribution. For our typical application, the corresponding algorithm component is named as **Concurrent Noise Filtering** or *Co-Filtering*.

Data oriented anomaly detection. We develop an integrated method based on machine learning to implement the *Data Oriented Anomaly Detection*. A learner is used to train a prediction model on historical data. The errors between the observed and predicted values can be approximate as a normal distribution, so we can develop a simple but very efficient online method to detect anomalies in a normal distribution. The learner does not make any assumptions about the data distribution, so it is suitable for very different target series in the catalog stream. Multiple integrated detectors are used for online detection. Different detectors are sensitive to events at different stages, so *Data-Oriented Anomaly Detection* can employ multiple detectors to achieve good instantness and coverage. For our typical application, this algorithm component is named as **Deviation-based Identifying** or *D-Identifying*.

Missing data downside removal. This algorithm component uses kurtosis to evaluate the impact of any data missing. Suppose the data distribution in a window is leptokurtic. In that case, the data missing in this window will have a low possibility to cause an outlier or a false alarm because most of the data are close to each other. However, suppose the data distribution in a window is platykurtic. In that case, the possibility of an outlier caused by data missing will be very high because most of the data are different, and missing data will significantly change the trend of a data series. However, it is hard to decide on the ambiguous interval between platykurtic and leptokurtic distribution (mesokurtic distribution). We will further split a long window into many short windows to evaluate their local kurtosis to solve this problem. The proportion of local kurtosis will decide the ambiguous interval's final distribution. Based on this idea, if the data series in a window follows the platykurtic distribution and there is data missing in the window, we will remove the alarm that appears very close to some missing data interval in this window because it can be caused by data missing with a high possibility. For our typical application, this algorithm component is named as **Kurtosis-based Filtering** or *K-Filtering*.

Domain-oriented anomaly detection. The domain-specific features can be used to identify anomalies efficiently. For our typical application, the geometry of different targets' feature curves in a short window where an anomaly occurs can be crest or trough (we ignore the straight-line situation because this situation means that no abnormalities will occur). Anomalies with different geometric curves represent different physical information. Therefore, this geometric analysis result can help us select abnormal events with different physical processes. So, we developed a **Crest-Trough-based Identifying** (*CT-Identifying*) method based on geometric analysis to achieve high-value anomaly selection. This algorithm will first calculate the crest/trough feature of a data series in a short window until the current time, then select highly valued events based on some specific geometry of the target's feature curve. If *CT-Identifying* selects an alarm, it will finally be marked as an anomaly.

3.2 Attention Window

To enable comprehensive data analysis and further improve the effectiveness of our algorithm, we propose the concept of an attention window that will provide appropriate data for different algorithm components. We define an attention window based on whether data missing is allowed in the window and the amount of data.

First, based on if missing data will be allowed in a window, a sliding window can be a **missing data forbidden window**, which means that we must pad the positions of missing data with placeholders in the window, or a **missing data tolerance window**, which means that we can ignore the missing data and do not need to pad placeholders at the missing positions. A missing data forbidden window has two states, the suspended state and the normal state. If there is any placeholder in the window or the window is not full, it will be in the suspended state. Otherwise, it will be in a normal state. Identifying or filtering operations cannot normally work on a missing data forbidden window when it is suspended, but the window can be updated with the latest data. A missing data tolerance window also has two states: the initial and normal states. At the beginning stage, a missing data tolerance window will be put in the initial state when its buffer is not full. Otherwise, it will be in the normal state when no vacant position is in its buffer. After a missing data tolerance window accepts enough data and passes its initial stage, it will never block any operations.

Second, a sliding window can be long or short based on the length or amount of data in a window. The long window may provide accurate statistical features (such as mean, deviation, and kurtosis) of a lot of data and is often tolerant of missing data. So we let a missing data tolerance window be very long to alleviate the impact of data missing. Oppositely, the short window provides the current local information to make a real-time decision, and it often cannot be tolerant of missing data. Therefore, we let a missing data forbidden window be short, so we can skip the invalid data to avoid wrong decisions and shorten the window's blocking operations.

In this work, we design three sliding windows with different properties. They are one long missing data tolerance window, historical data window $HWin = [t_H, t_L]$, and two short missing data forbidden windows, anomaly decision window $AWin = [t_A, t_L]$ and noise decision window $NWin = [t_N, t_L]$, where t_L is the latest or current time, t_H, t_A, and t_N are some previous time points and they meet $t_H << t_A < t_N < t_L$ under our configuration.

3.3 A Catalog Stream Processing Framework

The basic idea of the proposed framework (*CSPF*) is using the static partitioning scheme to split a large-scale catalog stream into many sub-catalog streams and distribute them onto different computing nodes to enable scalable parallel processing within the real-time requirement. This dispatching strategy can avoid unnecessary task migration across nodes when catalog streams change dynamically. A proposed pre-partitioning-based performance model will be trained on many historical performance data to generate such a partitioning scheme.

Because of the inter-catalog overlapping, *CSPF* also needs to identify duplicate targets across catalog streams. Interacting with different nodes to identify the duplicate targets will introduce synchronization overhead. At the same time, the node interaction topology cannot be predefined since the overlapping pattern information is unknown under practical scenarios. Considering that it is only necessary to identify duplicate anomalies when querying, we propose the *Dup-order* method that supports duplicate identification without node interaction. When launching a query, Dup-order can load duplicate targets into the same group through a distributed position mapping mechanism and keep them contiguous in order regardless of whether the same node processes them. Users will quickly find

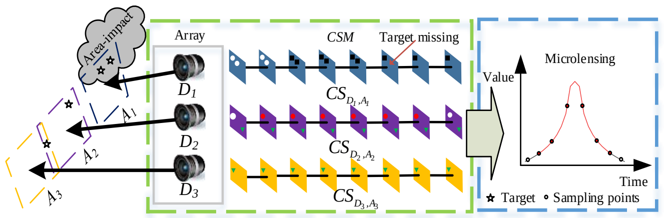


Fig. 2: A schematic diagram of a *CSM* and the detected transient events. The parallelograms of different colors in the *CSM* represent different catalog streams from different areas A_1, A_2, A_3 and device D_1, D_2, D_3. White dots are normal sampling points. Red dots illustrate the detected anomaly event, microlensing. Green triangles are duplicate targets due to the overlapping between A_2 and A_3. Black squares represent targets affected by concurrent noise, even causing target missing.

duplicate targets based on this method. Our method can maximize the parallel execution and complete the duplicate identification by sorting within the group on the query side.

3.4 Transient Detection for Scientific Discovery

In this subsection, we describe one typical application in time-domain astronomy. Real-time tracking of transient events (abnormal univariate time sub-series) by continuously monitoring a large number of space targets (i.e., stars) can lead to important scientific discoveries, such as new superflares and microlensing events [10].

To identify transients from a large number of targets, a large field-of-view observation array that can produce data with a very high cadence is necessary. For example, GWAC [11] consists of 20 devices and collects 20 catalogs every 15 seconds, including up to millions of stars. Other telescopes also employ the similar observation style, such as ASAS-SN [12] and MASTER [13]. Along the time, catalogs will form different catalog streams. It is a perfect example of Catalog Stream Matrix (*CSM*). AS shown in Figure 2, this application involves all the properties of catalog streams: concurrent noise, data missing, and overlapping. The catalog structure and specific challenges are as follows.

Catalog structure. Each catalog is a structured list, including many feature vectors describing the target sampling information. The feature vector (7 features) of given spatial target Tgt_x is in the form of $FV^{t,x} =< SInfo^t, RA, DEC, mag, ... >$, where $SInfo^t = \{D_n, A_m, t\}$ and t is the timestamp. Due to independent data collection, Tgt_x is not unique to duplicate targets. Ascension RA and declination DEC are the global positions of a target, but there will be slight errors in duplicate targets. These factors often cause exact-matching-based methods for duplicate identification to be inefficient. The mag is the main feature in the feature vector, representing the physical observation value used for transient event discovery. The stars with different mag values have different sensitivity to noise. For example, the anti-noise ability of bright stars is greater than that of dark stars.

Irregular spatio-temporal patterns. Due to data collection from an open environment, the Spatio-temporal patterns are usually irregular. For concurrent noise, on the one hand, it is not trivial for us to correctly select the affected targets because their spatial patterns are very irregular; on the other hand, the irregular noise level is another challenge to set an accurate filtering threshold. We cannot use simple strategies to eliminate the irregular temporal pattern of data missing to reduce the false-positive rate. For duplicate targets caused by data overlapping, their irregular spatio-temporal patterns make it impossible to identify duplicates by simply setting a threshold based on the fixed distance for all targets.

4 Filtering-Identifying based Anomaly Detection

In this section, we give details of *FIAD* in combination with the application in Section 3.4. Firstly, the filtering-identifying framework is presented to support the complete anomaly detection pipeline. Then, four major algorithm components are described in detail.

4.1 Filtering-Identifying Framework

Figure 3 gives a view of the complete detection pipeline. Filtering and identifying are performed alternately in the pipeline. The advantage for *FIAD* is to relax the identifying condition to achieve early and continuous detection and tighten the filtering condition only to remove false alarms, thereby gaining both goals by splitting the two contradictory requirements into different algorithm components.

As shown in Figure 3, *FIAD* first receives TSS, and the interpolator uses incalculable placeholders to replace the missing data. If there are no placeholders in $NWin$ i.e., normal state, *Co-Filtering* will check whether the latest data $TS_x(t_L)$ is valid. For invalid high noise data, *Co-Filtering* replaces $TS_x(t_L)$ in TS_x with a placeholder (Step one and Step two in Figure 4). For a suspended $NWin$, *Co-Filtering* will continuously replace the latest data $TS_x(t_L)$ in TS_x with a placeholder at the current timestamp until $NWin$ is full of valid data again. *D-Identifying* consists of many subdetectors with different lengths of $AWin$. The maximum anomaly score will be as the final output. If some of these $AWins$ are suspended (there are placeholders in it), the corresponding sub-detectors will recognize $TS_x(t_L)$ as a normal point and will not execute the anomaly identifying operations. $HWin$ ignores placeholders in a stream (Step three in Figure 4) so the data-missing recognizer is designed to check whether it is continuous. If it is not continuous, *K-Filtering* will check the validity of an alarm.

Different variants of the target stream are needed at the different phases of our filtering-identifying pipeline. So, we set up separate window buffers to support the operations of our pipeline at different stages instead of sharing the same data of a target stream by sliding pointers.

4.2 Concurrent Noise Filtering

To accurately remove data with high concurrent noise, *Co-Filtering* needs to evaluate the concurrent noise level and set different noise thresholds for different targets because neither a fixed absolute noise threshold nor a fixed relative noise threshold can be employed on all targets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2022.3161925, IEEE Transactions on Big Data

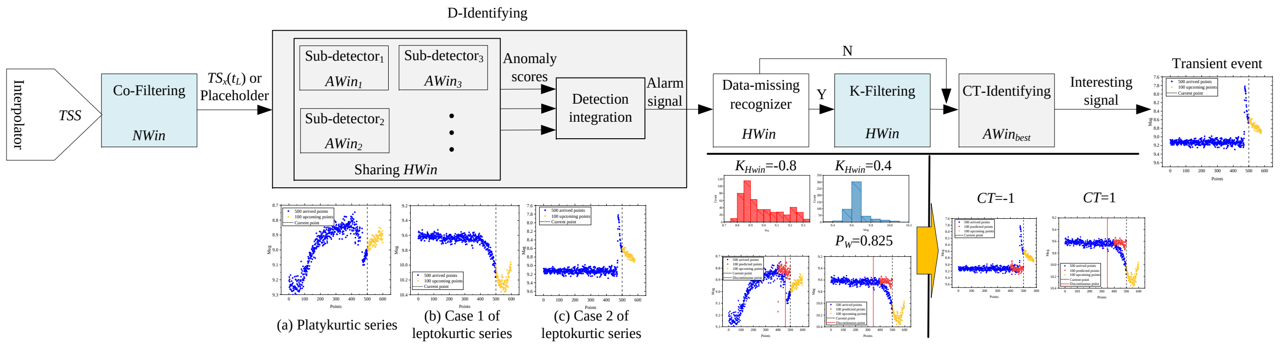


Fig. 3: The filtering-identifying framework of *FIAD*. For scatter figures, three target series are used as the input data (blue points) and $|HWin| = 500$; the red points are predicted points by *D-Identifying*, and yellow points are upcoming points; the red line is the position of data missing found by the data-missing recognizer. Figure (c) is a real transient event without missing data, showing a superflare event. Two histogram figures represent the distribution of the platykurtic series and leptokurtic series, respectively. Different sub-detectors derived from the same model have different lengths of anomaly decision windows. *D-Identifying* will issue the final alarm by a detection integration strategy.

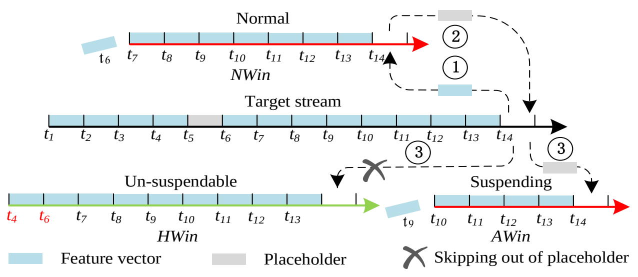

Normal
t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
NWin
Target stream
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
1
2
3
3
Un-suspendable
t_4 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13}
HWin
Suspending
t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
AWin
Feature vector
Placeholder
Skipping out of placeholder

Fig. 4: Example of different attention windows working together. The red arrow is the missing data forbidden window, and the green is the missing data tolerance window. The data at t_{14} is noise. *Co-Filtering* ingests it in $NWin$ and writes back a placeholder into the target stream. Next, this placeholder will also be ingested into $AWin$, making it suspend identifying operations on it. $HWin$ will ignore the placeholder.

4.2.1 Concurrent Noise Evaluation

Given the noise decision window $NWin$, if there are no placeholders in $NWin$, we let TS'_x be the corresponding noise-free series of TS_x, and the error $e_x(t)$ of each data in the window can be defined as $e_x(t) = TS'_x(t) - TS_x(t)$, where $t \in NWin$. Furthermore, we use the root mean square deviation (RMSD) of the error as the distortion of the latest point $TS_x(t_L)$. As a smoothing result, this value can reveal the distortion tendency and reduce the effect of random factors. The distortion $dist(Tgt_x, t_L)$ is calculated as follows.

$$dist(Tgt_x, t_L) = \sqrt{\frac{\sum_{t=t_N}^{t_L} e_x(t)^2}{|NWin|}}. \tag{5}$$

We empirically select $|NWin| = 64$ in this application. The rest problem is getting TS'_x. We can estimate it as TS^*_x by Discrete Wavelet Transform (DWT) [14]. DWT is usually used to decompose the discrete basic signal into the high-frequency part and low-frequency part through the appropriate wavelet function. The low-frequency part is usually regarded as the noise-free part, and we also use

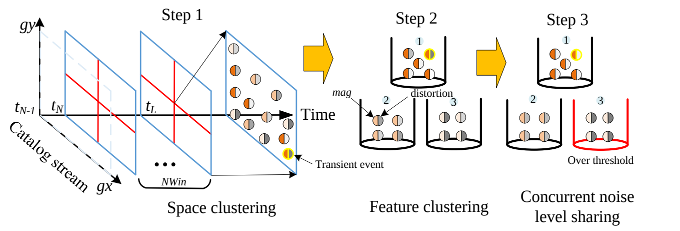


Fig. 5: Example of concurrent noise filtering. The heavy color means a brighter target or larger distortion. The concurrent noise level of the third slot is over the threshold so we remove all of the points in it.

the low-frequency part as TS^*_x, which will be employed to estimate the distortion instead of TS'_x.

Concurrent noise level sharing. We note that the distortion only represents the deviation between the real target stream and the expected tendency. Both transient events and noises may cause increasing distortions. So, we cannot simply employ the distortion to remove concurrent noise. The concurrent noise may affect multiple targets simultaneously, but it is almost impossible for multiple targets to have transient events simultaneously. Therefore, we take advantage of this property to calculate the distortion statistics of similar targets as the estimation of the concurrent noise level. For a set of highly related targets S in a *feature cluster*, we estimate their concurrent noise level at time T_L using Eq.(6). All targets in the feature cluster S will share the same concurrent noise level.

$$CNL(S, t_L) = \underset{Tgt_x \in S}{quantile}(dist(Tgt_x, t_L), \lambda_d), \tag{6}$$

where $\lambda_d \in [0, 1]$ is fractile. Especially, here we select $\lambda_d = 0.5$ meaning the median. The large λ_d will cause very large distortion to be selected as the concurrent noise level value, and vice versa. Further, we present a stepwise clustering method (first space clustering and then feature clustering) to decide feature clusters in Figure 5.

Space clustering partitions a catalog area into several subregions to isolate the impact of unknown noise area. We can employ HEALPix [15] as a partition function. HEALPix is a partition index algorithm of the hierarchical equal area of a sphere. Given the celestial coordinate system and the partition layer pl, HEALPix will return a unique partition number $pn = HEALPix(RA, DEC, pl)$ (i.e., subregion).

Feature clustering clusters the mags of one subregion into several mag slots (i.e., feature cluster) according to predefined mag value division scheme. In the same feature cluster, the mags are similar to each other. If a catalog is partitioned into four subregions and the mag value range is divided into ten subranges, finally, get 40 feature clusters.

Discussion. If a transient event occurs in one target alone (yellow point in Figure 5), its high distortion cannot improve the noise level of its feature cluster, so it does not trigger *Co-Filtering* to remove the data. However, if some concurrent noise occurs, more targets in the same feature cluster will be affected, and this will increase the concurrent noise level and cause *Co-Filtering* to remove the high noise data. A transient event with high concurrent noise is often unacceptable because most related data are untrustworthy. Overall, *Co-Filtering* does not cause trustworthy transient events to be removed.

4.2.2 Concurrent Noise Filtering Threshold

Threshold selection includes method selection and training data selection. The selected method should generate high-quality thresholds suitable for different target streams when using the training data set flooded with high-quality and low-quality data because pure, high-quality data are hard to provide under practical scenarios.

We treat the threshold selection as a Peaks-Over-Threshold (POT) problem [9]. The advantage of POT method is that it can generate highly optimized threshold value from initial value without any strong assumption on the distribution of a target stream. *We adopt POT to learn the threshold for each mag subrange* because targets in the same mag subrange have a similar antinoise level. For a given mag subrange msb including m feature clusters, if our training data have n different noise decision windows $NWin$, then we can build the concurrent noise level set $CNLS_{msb} = \{CNL(S_1, t_1), ..., CNL(S_1, t_m), ..., CNL(S_m, t_1), ..., CNL(S_m, t_n)\}$ (totally the $CNLS_{msb}$ has $m \times n$ different concurrent noise level values). The POT approach tries to fit a generalized Pareto distribution (GPD) with parameters to the portion upon the given initial threshold th'_{msb} of $CNLS_{msb}$. The final threshold th_{msb} is then computed by

$$th_{msb} \simeq th'_{msb} + \frac{\hat{\delta}}{\hat{\varepsilon}} \left(\left(\frac{q|CNLS_{msb}|}{|CNLS_{msb}|_{th'_{msb}}} \right)^{-\hat{\varepsilon}} - 1 \right), \quad (7)$$

where q is the desired probability to observe $z > th'_{msb}, z \in CNLS_{msb}$, $|CNLS_{msb}|_{th'_{msb}}$ is the number of z s.t., $z > th'_{msb}$, and $\hat{\varepsilon}$ and $\hat{\delta}$ are the estimation of shape and scale parameters of GPD through maximum likelihood estimation. We empirically set high quantile th'_{msb}, such as 0.95 and $q = 10^{-4}$. For each point of target stream, if the measured concurrent noise level is higher than th_{msb}, we will remove it from TS_x and set a placeholder at this timestamp.

Selecting suitable training data is very critical to achieving a reasonable threshold value. Here, the training data could contain some noise, so the learned threshold value will be poor if the noise level is very high. So, the domain experts can view the historical data and select a time range with little external noise to train the POT model and get the corresponding parameters. For example, the training data should be collected when no cloud, no external light, and equipment is working well. This information can be easily obtained by recording the weather and analyzing equipment logs.

4.3 Deviation-based Identifying

D-Identifying includes two parts, offline training to generate prediction models for different targets and online detection to identify the anomalies.

4.3.1 Offline Training

The standard Temporal Convolutional Network architecture (TCN) [16] is employed as the base algorithm because it is easier to train and has less resource consumption than the recurrent neural network, such as LSTM [17]. These features are essential for large-scale catalog streams. TCN will model each target stream and predicts the next point using an unsupervised one-step-ahead style.

The TCN does not require a complete series but a batch of continuous windows without missing data for training. To enable sufficient data windows, we impute them by Kalman filter [18], when the data is slightly missing. Kalman filter can estimate the hidden variables of each point (including missing data) of a univariate time series, with the mean and variance by using the past and future information. It can consider the pattern evolution of time series and normal random changes. So, we input the training target streams to the Kalman filter. If the gap of missing data is less than ML points, we can sample n values for each missing data and calculate the average as the missing value. If the gap of missing data is greater than ML points, we do not impute it because the imputation of long gaps easily introduces incorrect data patterns. We empirically set $ML = 30$ and $n = 10$. Finally, a target stream is split into many equal-length windows using the $HWin$ length and skip out of the data missing due to the long gap to ensure the continuity inside windows, and use them as training data units.

4.3.2 Online Detection

Inspired by the NFD algorithm [19], we leverage $HWin$ and $AWin$, $|HWin| >> |AWin|$ to calculate the long time statistical result and the current result. We first assume that there are no placeholders in $AWin$. When the data in the window $[t_{A-1}, t_{Q-1}]$ are ready, the predicted data in $HWin$ will be returned. The normal change of target streams is often regular and predictable so we use the prediction error as the feature $\tau(t)$ to measure the abnormal degree of the current data. Supposing the predicted value of $TS_x(t)$ is $TS^p_x(t)$, *D-Identifying* sets $\tau(t) = |TS^p_x(t) - TS_x(t)|$. Further, *D-Identifying* uses the mean μ_{HWin} and the standard deviation σ_{HWin} of $\tau(t)$ in $HWin$ to capture the long time feature of the current stream. Especially, it uses the mean μ_{AWin} of $\tau(t)$ in $AWin$ to stand for the state of the current point. The normalized variable N_{t_L} is defined in Eq.(8) to represent the deviation of the current point from most of the existing points.

$$N_{t_L} = \frac{\mu_{AWin} - \mu_{HWin}}{\sigma_{HWin}}. \tag{8}$$

Anomaly score calculation. *D-Identifying* employs Q-function[1] as the discriminant function to measure the deviation of the normalized variable. Q-function is the tail probability of the standard normal distribution. In other words, Q-function measures the probability that the observed value is larger than the mean. The smaller the Q-function is, the more deviated N_{t_L} is from the mean. Finally, $Q(N_{t_L})$ will be as *anomaly score*.

Detection integration. The length of $AWin$ has a high impact on the coverage and instantness. Generally speaking, the shorter $AWin$ is sensitive to the early stage of the transient event, and the longer $AWin$ is sensitive to the last stage. To balance coverage and instantness, we integrate three sub-detectors with different $AWin$ lengths (e.g., 5 points, 15 points, and 45 points) to calculate anomaly scores together, denoting them as Sub-detector5, Sub-detector15, Sub-detector45, and they share the same $HWin$ (empirically set $|HWin| = 500$). Further, we could set an appropriate threshold ε_Q to issue alarms. If $min(Q(N_{t_L})) < \varepsilon_Q$, *D-Identifying* will issue $TS_x(t_L)$ as an alarm signal. We prefer to set a large ε_Q (e.g., 0.375 in our paper). The larger ε_Q means the more relaxed alarm condition, leading to the early alarm but low precision, especially when there is missing data. *K-Filtering* will handle this problem.

4.4 Kurtosis-based Filtering

If there is data missing in $HWin$, the false alarms will be easily generated by *D-Identifying*, but they have a high relationship with kurtosis, one important statistical feature of $HWin$. So, we develop the *K-Filtering* algorithm to take advantage of kurtosis to remove such false alarms.

Kurtosis measures whether the data are platykurtic or leptokurtic relative to a normal distribution. The formula for kurtosis is as follows.

$$K_W = \frac{\sum_{i=l}^{l+|W|-1} (TS_x(t_i) - \mu_W)^4 / |W|}{\sigma_W^4} - 3, \tag{9}$$

where W is a window on $HWin$, the starting time point is t_l and the length is $|W|$. The σ_W is the standard deviation and the μ_W is the mean. $K_W \in [-2, |W|]$ and this definition implies that the standard normal distribution has a kurtosis of zero. The negative kurtosis indicates a platykurtic distribution. Otherwise, the positive kurtosis indicates a leptokurtic distribution.

In a platykurtic distribution, most of the data is prone to be uniform, and the standard deviation is relatively large. So, data missing may cause a rather significant deviation based on Eq. (8), and it will cause false alarms with high possibility. However, in a leptokurtic distribution, most of the data are very close to the mean value, and data missing may cause a relatively small deviation. So, we may remove an alarm in a significant platykurtic distribution window with data missing. We use global kurtosis and local kurtosis to evaluate whether the data have a significantly platykurtic distribution.

Global kurtosis. If $W = HWin$, $K_{HWin} \le -0.6$ empirically means that the global platykurtic feature is significant. *K-Filtering* removes the current alarm.

Local kurtosis. If $-0.6 < K_{HWin} < 0$, the global platykurtic feature is not very significant, then we take advantage of the local kurtosis to measure the significance of its platykurtic feature further. So, we partition $HWin$ into many small windows to evaluate their local kurtosis using Eq. (9). We slide the window W on $HWin$, where $|W| = \alpha_W |HWin|$ and the step is S_W. We empirically set $\alpha_W = 0.2$ and $S_W = 10$. If lots of local kurtosis is less than the given threshold $\epsilon_W = -0.1$, we mark its platykurtic feature as significant. We assume that the number of local kurtosis being less than ϵ_W is N_{ϵ_W}. The number of local kurtosis is $\mathrm{N}_W = \frac{|HWin| - |W|}{\mathrm{S}_W}$. When $P_W = \frac{N_{\epsilon_W}}{N_W} \ge 0.5$, *K-Filtering* removes the current alarm.

4.5 Crest-Trough-based Identifying

After *K-Filtering*, we will further check each candidate anomaly. If we take the prediction value $TS_x^p(t)$ as the baseline, the observation value $TS_x(t)$ will be fluctuating around the baseline. According to domain knowledge, the crest-trough feature is evaluated under the inverted Y-axis because the low mag usually means the bright targets. If the observation values happen above the baseline, we name the observation values having a crest shape. Otherwise, we call them having a trough shape. Under practical scenarios, users only prefer to crest transient events that mean abnormal energy fluctuations of stars. So, we developed the *CT-Identifying* algorithm component to select the anomalies that happened with some specific geometric shapes.

Let $f(t) = TS_x^p(t) - TS_x(t)$. We use the exponential average to estimate an anomaly's crest or trough shape to smooth the results.

$$CT = sgn\left(\sum_{i=0}^{|AWin_{best}|-1} (1-\alpha)^i f(t_{L-i})\right), \tag{10}$$

where $AWin_{best}$ is the anomaly decision window which can maximize $Q(N_{t_L})$; the decay factor $0 < \alpha \le 1$ is set as 0.7 which will let the current value have larger effect. sgn is a sign function. Thus, if $CT = 1$, the alarm signal is crest; if $CT = -1$, the alarm signal is trough.

4.6 Discussion

In essence, both *Co-Filtering* and *D-Identifying* are anomaly detection procedures on feature series. The difference is that the detection feature for *Co-Filtering* is noise level, but it is a prediction error for *D-Identifying*. However, the detection feature generation technique is not interchangeable. The TCN cannot replace the DWT to produce the detection feature series of *Co-Filtering*. The TCN as a learning algorithm will learn concurrent noises in prediction values, which is not conducive to noise filtering. Similarly, the prediction error produced by the DWT will contain too much noise information.

The threshold setting methods are not interchangeable either. The threshold for *Co-Filtering* is global and static,

1. https://en.wikipedia.org/wiki/Q-function

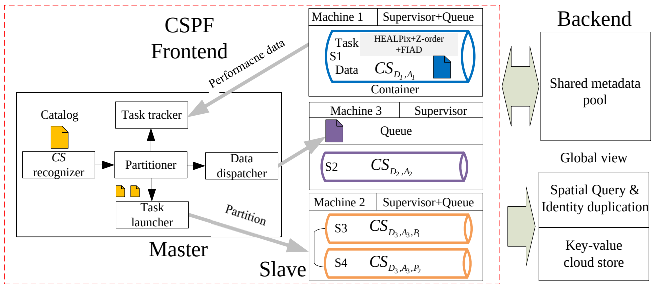


Fig. 6: Catalog stream processing framework *CSPF*. S* is the slave node. The data example comes from Figure 2. CS_{D_3,A_3} is partitioned into two parts running on different slaves for meeting the real-time requirement.

learned from historical data by the POT. For *D-Identifying*, it is a dynamic threshold with the sliding of the online windows. We can easily find the hybrid data with low and high noise from historical data for noise filtering. This scenario satisfies the hypothesis of POT. However, transient events are infrequent, so it is difficult to set the anomaly threshold by POT. Further, *D-Identifying* dynamically evaluates the feature difference between Q-function's long and short windows. It does not depend on historical transient events, and it is also easy to set up due to the clear statistical meaning. However, this dynamic setting method is unsuitable for noise filtering because it only considers the local information within a window, possibly causing it not to work normally when the window only contains the high-noise data.

We are noting that *FIAD* does not attempt to impute missing data using some calculable values in *online detection*. The imputation for univariate time series usually exploits additional information, such as seasonality, trend, or future data, to get a good result [20]. However, our scenario only gives a history window, not involving this information, because it is hard to determine them for many targets with many missing data. Adding a poor imputation component may introduce more false alarms.

5 Catalog Stream Processing Framework

This section includes the framework design, the pre-partitioning-based performance model to meet the real-time requirement, and the duplicate target identification method.

5.1 Framework Design

The *CSPF* follows master/slave mode, as shown in Figure 6. An agent-based scheduling model is designed, including master, supervisor, and slave. The master is unique to managing the cluster and dispatch catalogs. Each slave as a micro-service, e.g., using Docker [21], monopolizes a CPU logical core. It allows that different slaves can reside in the same physical machine simultaneously. The supervisor is a daemon process (including a queue to cache catalogs) on each physical machine to manage both the local computing resource and slaves. Such a two-level framework can implement the shared-nothing distributed processing and reduce the management overhead of the master.

In the framework backend, the shared metadata pool stores required data of all slaves, such as model parameters for transient event detection. Since targets in a feature cluster must be calculated together, the feature cluster will be the minimum dispatching unit. The metadata of all targets in a feature cluster will be packaged together to store. The shared metadata pool is implemented as a distributed in-memory file system (e.g., NFS+RamDisk [22]).

The CS recognizer receives the heterologous catalog set to identify which catalog stream the catalog belongs to. The partitioner uses the pre-defined partition scheme to divide the catalog into sub-catalogs. If a catalog belongs to a new catalog stream, the task launcher will select a machine based on the number of running tasks and inform its supervisor to launch a new slave. Finally, the data dispatcher sends these sub-catalogs to the corresponding supervisor queue. Task tracker is used to collecting the performance data, including the latency of each feature cluster. These data will be fitted into our performance model to decide the pre-partition scheme for each catalog stream. The catalog as the observation data usually has the clear target amount range, so *CSPF* can fix each slave on a specified machine and not dynamically change the partition scheme and divide the task across the cluster. The advantage is to avoid pipeline interruption and metadata migration.

5.2 Pre-partitioning-based Performance Model

We design an initial phase for *CSPF* to train our performance model to decide the pre-partitioning scheme for each historical catalog stream. *CSPF* uses the same pre-defined scheme to partition all catalogs in a catalog stream when processing real-time catalogs.

The key idea of giving a pre-partitioning scheme is to make the most time-consuming catalog to meet the real-time requirement. Assuming the computing cluster is heterogeneous to have K different performance CPUs denoted as $C = \{C_1, C_2, ..., C_K\}$. Given a catalog stream following Eq. (1), we process it on different CPUs of the cluster and get a performance matrix $Perf = [RT_{E^{t_i},C_k}]_{L\times K}$ where $RT_{E^{t_i},C_k} = (RT^{t_i}_{S_1,C_k}, RT^{t_i}_{S_2,C_k}, ...)$ is a vector recording the latency of all feature clusters. We assume that any $RT^{t_i}_{S_l,C_k}$ can be finished within the real-time requirement l. Otherwise, the subregion number in *Co-Filtering* needs to be increased. Further, $\sum RT_{E^{t_i},C_k}$ is the latency of Catalog E^{t_i} on C_k. We further define the worst acceptable latency as follows.

$$RT_{worst} = \underset{t_i\in[t_1,t_L],k\in[1,K]}{quantile} (\sum RT_{E^{t_i},C_k}, \beta) \tag{11}$$

where $\beta \in [0, 1]$ is the fractile as timeout tolerance factor, we set it as 0.9. If $RT_{worst} < l$, our performance model does not partition this catalog stream. Otherwise, too many catalogs will time out. For any $\sum RT_{E^{t_i},C_k} > RT_{worst}$, a partition scheme requires to be assigned to CS. This partition scheme should use the least partition number to make all the timeout catalogs meet the real-time requirement. Considering all latencies of a feature cluster S_l in $\sum RT_{E^{t_i},C_k} > RT_{worst}$ consist of a multidimensional vector $Timeout_{S_l} = (RT^{t_{i'}}_{S_l,C_{k'}}, RT^{t_{i''}}_{S_l,C_{k''}}, ...)$, our optimisation goal is to use the smallest partition number to arrange feature clusters into sub-catalogs and make the latency of these sub-catalogs be less than l. It is actually a multidimensional bin packing problem (MBPP) [23], which is a well-studied NP-hard problem only having the approximate solution. The boxes defined in MBPP could be a series of $Timeout_{S_l}$, the bin capacity per dimension is l, and MBPP will search the smallest number of bins and the corresponding arranging scheme. We employ a linear programming solver to give an approximate solution as the pre-partitioning scheme and assign it to partition the corresponding catalog stream during the online pipeline. As long as the training data is sufficient, most of the catalogs will meet the real-time requirement. Noting that the initialization phase is not necessary for the cluster launching only based on user preferences.

5.3 Duplicate Target Identification

We decouple the analysis task and duplicate identification to make them asynchronous to avoid unnecessary task synchronization. The idea of Dup-order is to calculate position features in a distributed manner for each target and sort targets through their position features in a group to make duplicate targets close when launching the real-time query. A global hash function HEALPix [15] and a space-filling curve function Z-order [24] packaged into each slave are used to calculate the position feature. HEALPix maps neighboring targets into the same slot, and Z-order transforms their 2D positions into a 1D number. HEALPix supports spatial queries, such as the disc search and polygon search, to help users quickly get the corresponding partitions. Z-order can maintain these reduced 1D features to be still spatially ordered.

The global view includes a key-value cloud store, such as Redis cluster [25] used to store the position features as a key-list structure. The key consists of the partition number returned by HEALPix. The partition number is used to make all transient events in the same partition physically cluster, even though different slaves process them. In addition, the list includes the Z-order numbers of different targets. When launching a real-time query, HEALPix returns a series of partition numbers. Further, the global view can search its key-value cloud store and get the corresponding data. In each partition, Dup-order leverage the Z-order number of each target to sort them. Since duplicate targets are close to each other, they should be adjacent in order. Users can easily distinguish duplicate targets inside a group when the global view returns the results using the grouping style (i.e., partitions).

6 Experiments

This section will introduce the data sets used in the experiments and the corresponding setup, the effect of both *FIAD* and *CSPF*, and the scientific candidate events discovered by our methods.

6.1 Datasets and Experimental Setup

6.1.1 Datasets

We conduct performance experiments on 3-day 10-device GWAC real data (called GWACD3), which consists of 71 catalog streams. We must balance the rationality of experimental results with the labor of data labeling. So, four types of data sets are produced.

GWACD3 dataset. GWACD3 will be used in part of the experiments for distributed processing evaluation.

CS* datasets. We select three catalog streams from GWACD3, which involve most transient events (six events). There are different numbers of targets involved to cover different performance profiles. We denote three data sets as CS793, CS1369 and CS1655 (collectively referred to as CS*), because they contain 793 catalogs ($19,431\pm1,537$ targets per catalog, two event), 1,369 catalogs ($53,215 \pm 11,271$ targets per catalog, three events) and 1,655 catalogs ($19,348\pm5,254$ targets per catalog, one event). CS* will be used for anomaly detection and distribution processing evaluation in the experiments.

CSL* datasets. We use HEALPix to partition CS* at $pl = 9$, select the subregion where the transient event is located, and 5% random subregions that make up the data sets CSL793, CSL1369, and CSL1655 (collectively referred to as CSL*). We manually label the concurrent noise and transient events for each target. CSL* will be used in some experiments for anomaly detection and evaluation.

2Cam dataset and 3Cam dataset. We choose catalogs that overlap completely to test duplicate recognition because they contain the most duplicate targets. In GWACD3, we can find 2-catalog overlapping and 3-catalog overlapping, denoting them as 2Cam and 3Cam, respectively. We use a cross-matching algorithm [26] to label duplicate targets and manually check the labeling results. They are used for distribution processing evaluation.

6.1.2 Experimental Setup

According to the description in Section 5, we customize *CSPF* to provide more flexibility and support some domain computing operations. The *CSPF* is implemented in C++. The *FIAD* is implemented in Golang. They are deployed on a cluster system with 18 computing nodes (24 1.6GHz logical CPUs, 96GB RAM, and 20TB SATA disk per node).

We use F1-Score (denoted as F1) and FPR (False-Positive Rate) to evaluate the performance of *FIAD*. F1 is defined as $F1 = \frac{2\times Precision\times Recall}{Precision+Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. FPR is defined as $FPR = \frac{FP}{FP+TN}$. FP means false position; TP means true position; FN means false negative; TN means true negative.

6.2 Effect of Anomaly Detection

This subsection includes a comparison of transient event detection, comparison of concurrent noise filtering, component effectiveness evaluation, and parameter impact evaluation.

6.2.1 Transient Event Detection Evaluation

Here, we provide the comparison results of the anomaly detection performance of *FIAD* and other methods, including F1, alarm coverage rate, and alarm instantness rate.

Method comparison. As shown in Figure 7, we use F1 to evaluate the performance of finding the right transient events, not all abnormal points of events. We set $CT = 1$ to keep the crest event. For the fairness of the experiment, we also introduced domain knowledge similar to *CT-Identifying*

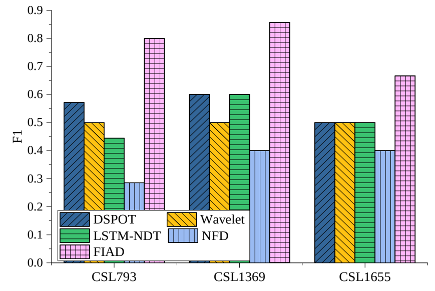


Fig. 7: Comparison of anomaly detection methods.

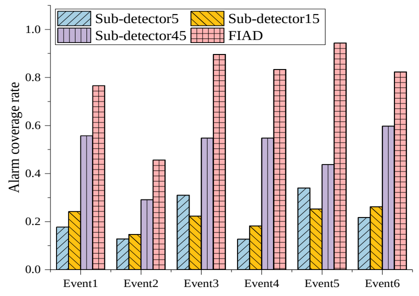


Fig. 8: Alarm coverage rate.

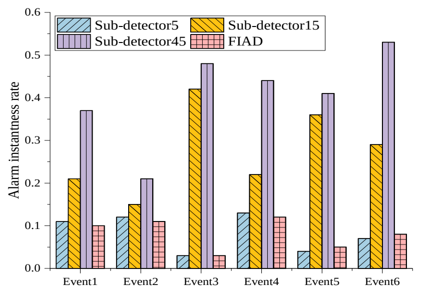


Fig. 9: Alarm instantness rate.

to eliminate the trough events of the comparison methods. What's more, these comparison methods are either classic domain algorithms or outstanding anomaly detection algorithms in recent years. They cover a variety of anomaly detection methodologies. The NFD [19] is the domain algorithm specifically used to detect transient events. Different from our sub-detectors, it uses the observation values as the feature. We add *CT-Identifying* in NFD to identify its crest events. LSTM-NDT is another typical algorithm that uses prediction errors to detect anomalies, using a nonparametric dynamic thresholding [27]. We use the sign of its prediction error to identify crest events. Wavelet uses the discrete wavelet decomposition to detect the anomaly [28]. Residual symbols are used to identify crest events. The DSPOT [9] employs the extreme value theory to detect the anomaly with the dynamic threshold. We disable its upper threshold to make trough transient events pass the detection. We use zero to impute missing data for the above methods. These comparison methods use grid search to select the best parameters. Noting that the parameter setup of NFD ($\varepsilon_Q = 0.15$) is more strict than *FIAD* ($\varepsilon_Q = 0.375$), due to the lack of false alarm filtering.

The recall rate of all methods is 1, which means they can find all transient events. However, the precision of these comparison methods is poor (0.3 on average), resulting in their F1 average being 21%$\sim$41% lower than *FIAD* (0.77). Such a low precision is due to rare transient events (only 0.14% in CSL*), data missing, and concurrent noise making it more difficult to detect anomalies in catalog streams, leading to false positives provided by other methods. Noting that when we try to tighten detection conditions of NFD, the performance has not improved significantly. The reason is that NFD cannot eliminate the false alarms caused by the change of normal temporal patterns. *FIAD* does not trigger such false alarms when using the prediction error as the feature. However, *FIAD* relies on an attention window based filtering-identifying framework to ingest the high-quality data and reject possible false alarms to improve precision.

Alarm coverage and instantness evaluation. As shown in Figure 8 and Figure 9, we have evaluated the alarm coverage rate and alarm latency rate. For a transient event TE, the alarm signal set on it is $S_a = \{s_1, s_2, ..., s_t\}, s.t., s_i = 0|1$. $s_i = 1$ means the i^{th} point is an anomaly. We define the alarm coverage rate as $\frac{|S_a(s|s=1)|}{|TE|}$, where $|*|$ is the length. The larger the coverage rate is, the better the method can cover most anomaly events. We further define the alarm instantness as $\frac{FIRST(S_a)}{|TE|}$, where $FIRST(*)$ returns the index of the first 1. The smaller the instantness value is, the earlier the anomaly is detected. We list the results of six transient events. The average alarm coverage rate of *FIAD* is 0.79, showing that our method can cover almost all of the range of transient events. However, the comparison algorithms have different instantness for different stages of transient events, resulting in poor alarm coverage. In addition, the average alarm instantness rate of *FIAD* is 0.08, which is equal to Sub-detector5 (the most sensitive detector for early phase detection). This is reasonable because *FIAD* relies on Sub-detector5 to achieve good instantness. These results show that our detection integration strategy effectively achieves good alarm coverage and alarm instantness.

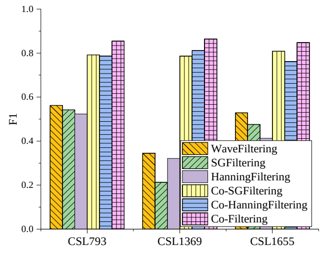


Fig. 10: Comparison of concurrent noise filtering.

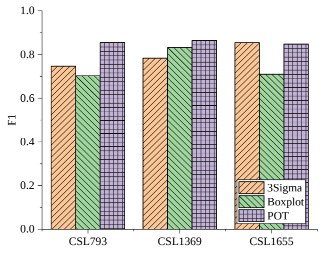


Fig. 11: Comparison of noise threshold selecting.

In short, the F1 of *FIAD* is 0.77 (21%$\sim$41% higher than comparison methods), the alarm coverage rate is 0.79, and the alarm instantness rate is 0.08. Attention windows and an filtering-identifying framework can work together to reduce false alarms significantly.

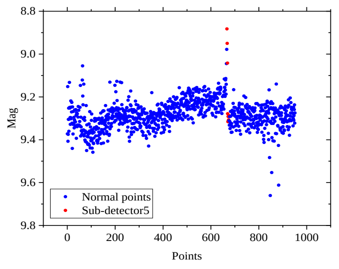
6.2.2 *Concurrent Noise Filtering Evaluation*

This subsection compares the concurrent noise filtering performance with other filtering methods and threshold selection methods. Finally, give an example to show the filtering effect.

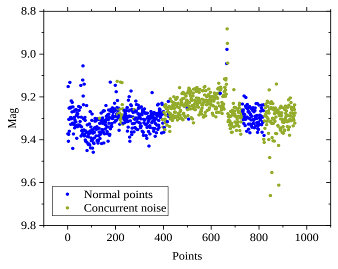
Filtering method comparison. As shown in Figure 10, we compare our method with five typical filtering methods. The value range of mag in GWACD3 is [0,17]. Combining with the necessary domain knowledge, we divide it into 7 mag subranges: [0,6], (6,7],(7,8],...,(10,11], and (11,17]. WaveFiltering only uses the high-frequency part of the discrete wavelet transform as the noise for each target stream. SGFiltering and HanningFiltering leverage the smoothing function to remove noise data. SGFiltering is to output the

TABLE 1: Comparison of transient event misfiltering

Methods	CSL793	CSL1369	CSL1655
WaveFiltering	1	3	1
SGFiltering	1	4	1
HanningFiltering	1	4	1
Co-SGFiltering	0	0	0
Co-HanningFiltering	0	0	1
Co-Filtering	0	0	0



(a) Turning off Co-Filtering



(b) Turning on Co-Filtering

Fig. 12: Example of alarm signals issued by *FIAD* when turning off/on *Co-Filtering*.

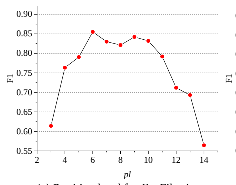
residual of Savitzky-Golay filter [29] as the noise. HanningFiltering is the residual of 1D time convolve with Hanning window[2] as the noise. The three above methods do not consider concurrent noise. Co-SGFiltering replaces the discrete wavelet transform in *Co-Filtering* with the reconstruction result of SGFiltering. Co-HanningFiltering replaces the discrete wavelet transform in *Co-Filtering* with the reconstruction result of HanningFiltering. Both Co-SGFiltering and HanningFiltering consider concurrent noise. Their sliding window length is set to 64. The λ_d is set to 0.5. The F1 of *Co-Filtering* is an average of 0.855 for the three data sets (6%$\sim$44% higher than comparison methods), which is the best result. These methods that do not leverage noise concurrency have poor precision, leading to the removal of some normal points and transient events. These methods of considering noise concurrency are better. It also shows that our method is effective. As shown in Table 1, we illustrate the number of transient events that are incorrectly removed by different methods. *Co-Filtering* and Co-SGFiltering are the best. It shows that leveraging noise concurrency does not lead to the mis-filtering, and improves the filtering performance.

Threshold selection method comparison. In Figure. 11, we compare our unsupervised threshold selection method POT with two unsupervised threshold selection methods 3Sigma and Boxplot. We do not compare the dynamic threshold selection method, because we only need a static threshold to remove low-quality points. 3Sigma sets the threshold as $\mu + 3\sigma$ and Boxplot sets the threshold as $Q3 + 1.5IQR$. μ and σ are the mean and standard deviation of the training data, respectively. Q3 is the third quarter, and IQR is the interquartile range. Overall, our method is better than the comparison methods. The F1 of POT is 6%$\sim$11% higher than both 3Sigma and Boxplot. 3Sigma and Boxplot assume that the data follow a normal distribution, but due to complex noise, $CNLS_{msb}$ does not meet this assumption,

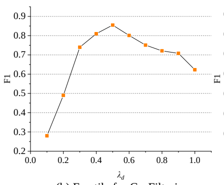
2. https://scipy-cookbook.readthedocs.io/items/SignalSmooth.html?highlight=smooth

TABLE 2: Component effectiveness

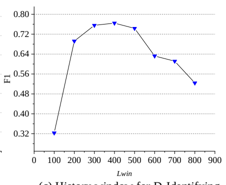
Datasets	Eval	Components				
		D	Co	K	CT	All
CS*	Events	3707+6	2571+6	257+6	707+6	52+6
	FPR%	3.19	2.21	0.22	0.61	0.04
CSL*	Events	149+6	85+6	20+6	22+6	3+6
	FPR%	2.64	1.56	0.44	0.49	0.15



(a) Partition level for Co-Filtering



(b) Fractile for Co-Filtering



(c) History window for D-Identifying

Fig. 13: Impact of different parameters.

resulting in poor results.

Example. We show an example of removing concurrent noise through *Co-Filtering*. As shown in Figure 12, some noise causes the distortion of the target stream in CSL793. Especially, this noise is similar to transient events (Figure 3c). When *Co-Filtering* (Figure 12a) is turned off, some noise points are identified as the transient event of Sub-detector5, leading to false alarms. After turning on *Co-Filtering*, the noise points are removed so we can avoid false alarms (Figure 12b).

In short, the F1 of *Co-Filtering* is 0.855 (6%$\sim$44% higher than comparison methods). The F1 of our threshold selection method is 6%$\sim$11% higher than comparison methods. *Co-Filtering* can use the concurrency of the noise in catalog streams to distinguish transient events and noise, and the threshold selection is efficient.

6.2.3 Component Effectiveness Evaluation

To evaluate the component effectiveness of our method, we first enable *D-Identifying* to detect the total number of transient events on CS* and CSL* as the baseline. We apply our proposed optimization methods one by one to see the reduction of false alarms to illustrate the effectiveness of these components. As shown in Table 2, we evaluated 5 cases. The "Events" value is "FP+TP". The "D" column means that we only enable *D-Identifying*, i.e., the baseline. The "Co" column means to enable *Co-Filtering* and *D-Identifying*. The "K" column means to enable *Co-Filtering*, *D-Identifying* and *K-Filtering*. The "CT" column means to enable *Co-Filtering*, *D-Identifying*, and *CT-Identifying*. The "All" column means to enable all modules of *FIAD*. The interpolator and the data-missing recognizer are always active.

The results show that *Co-Filtering* led to a 30.6%$\sim$42.2% reduction of false alarms. On the basis of *Co-Filtering*, *K-Filtering* and *CT-Identifying* lead to an 86.5%$\sim$93% reduction of false alarms, and an 81%$\sim$84.5% reduction of false alarms, respectively, compared with the "D" case. When opening all of these, we finally achieved a reduction of 98%$\sim$98.6% and found six transient events, and the FPR on the real catalog streams was 0.04%.

In short, each of our optimization components is efficacious in discovering transient events. The FPR reduction of 98%$\sim$98.6% is achieved, and the FPR on the real catalog streams was 0.04%.

TABLE 3: Results for *CSPF* dispatching performance

Device numbers	Catalog stream numbers	Catalog numbers	Total dispatch latency (seconds)	Dispatch through-put (cata-logs/second)
1	11	5,637	301	18.73
2	22	10,729	292	36.74
4	29	13,305	390	34.12
6	48	22,042	626	35.21
8	58	27,504	691	39.8
10	71	34,215	901	37.97

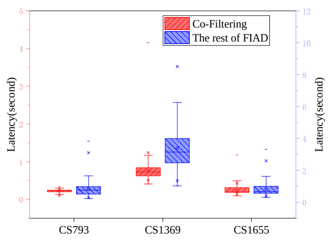


Fig. 14: Detection latency.

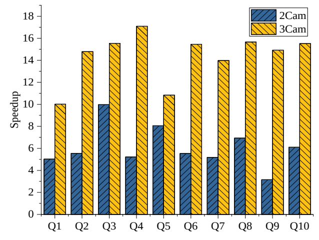


Fig. 15: Query performance comparison.

6.2.4 Parameter Impact

We evaluate the impact of 3 parameters including λ_d and pl of *Co-Filtering*, and $HWin$ of *D-Identifying*. When evaluating λ_d and pl, we only enable *Co-Filtering* and observe the performance of noise filtering. What's more, we only evaluate the online filtering phase. In the threshold setting phase, we fix $\lambda_d = 0.5$ and $pl = 6$. Figure 13 shows the average F1 on CSL* to evaluate the impact of λ_d and pl. Figure 13a shows the impact of HEALPix's partition level pl when fixing $\lambda_d = 0.5$. A small pl will cause each feature cluster to contain too many targets, resulting in increased error filtering. For example, in CSL1655, when $pl = 3$, each feature cluster includes an average of 857 targets. If the CNL for this feature cluster exceeds the threshold, 857 targets will be removed. Correspondingly, when the partition level is large, too many subregions will lead to weak statistical characteristics in each feature cluster so *Co-Filtering* will degenerate into WaveFiltering, causing a drop in filtering performance. In Figure 13b, we fix $pl = 6$ and evaluate the impact of different fractiles λ_d. The result shows that the median has a good effect. The small fractile causes the noise to be unremovable, and the large fractile causes too many normal points to be removed. For other scenarios, users need to tune the two parameters. In addition, we evaluate the impact of the length of $HWin$ because it can affect *D-Identifying* and *K-Filtering*. As shown in Figure 13c, we repeatedly run *FIAD* on CSL* with different lengths of $HWin$ and use the average F1 as a result on the three data sets. It shows that $HWin$ has a significant impact on the performance of *FIAD*. When $HWin$ is short, the Q-function cannot identify the tail of the distribution well, resulting in poor performance. When $HWin$ is lengthened, too old missing data will also activate *K-Filtering* causing it to possibly remove some right alarms. We found that 300-500 may be a good $HWin$ length in our data sets.

6.3 Distribution Processing Evaluation

This subsection evaluates the real-time processing latency and the duplicate identification method.

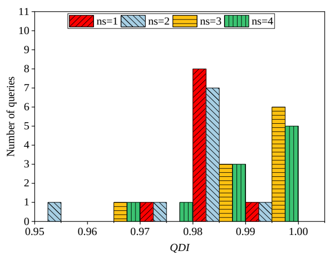
6.3.1 Effect of Real-Time Performance

***CSPF*'s performance**. To evaluate the *CSPF*'s distributed processing performance, we gradually increase the amount of data produced from one device to ten devices, as shown in Table 3. We simulate sending these catalog streams in chronological order without delay. Noting that the total dispatch latency is only the scheduling time and does not include the catalog analysis delay (i.e., *FIAD*). In detail, the container launching latency is 8 seconds, which is an acceptable time. What's more, when *CSPF* processes the data only generated by one device, it will degrade into the single-node mode, causing a reduced throughput. As the data increases, the throughput of *CSPF* is also increased. It is maintained at an average of 36.77 catalogs per second, i.e., 30 milliseconds per catalog on this cluster environment. It shows good scalability. GWAC currently consists of 20 devices and collects data within 15 seconds. This result means that *CSPF* can dispatch the data produced by 551 devices per 15 seconds. It is 27.5× faster than the required processing performance. GWAC's data collection latency (15 seconds) has represented a high time resolution compared with other sky survey projects in time-domain astronomy [11], [30] so we think this implementation of *CSPF* should be able to meet the real-time requirement of most sky survey projects. ***FIAD*'s performance**. We only evaluate the real-time detection latency, and it can also show the effect of the performance model. It does not include the offline training latency because it will not become a performance bottleneck. For many transient discovery scenarios, the observation does not always work. For example, time-domain astronomy cannot collect data in the daytime. Thus, lots of time can be used for offline training. Figure 14 shows the latency of *FIAD* when processing the CS* datasets. Since the most time-consuming modules of *FIAD* are *Co-Filtering* and *D-Identifying*, we divide the total latency into two parts: *Co-Filtering* and other. Both *K-Filtering* and *CT-Identifying* are very fast, so we no longer show their latencies separately. The latency of *Co-Filtering* is about 0.4 seconds per catalog (0.012 milliseconds per target). Except for *Co-Filtering*, the latency of the rest of *FIAD* is about 1.6 seconds per catalog (0.045 milliseconds per target). The total latency is an average of 2.1 seconds, and the maximum is 11.1 seconds. This shows that our performance model can enable *FIAD* to meet the real-time requirements of real scenarios. We also find that the number of targets significantly impacts the performance, but they show a more or less linear correlation, revealing that *FIAD* is robust.

In short, the average dispatch latency is 30 milliseconds per catalog (27.5× faster than the required processing performance), and the average detection latency is 2.1 seconds (maximum 11.1 seconds). *CSPF* has good scalability and can meet the real-time requirement of practical scenarios.

6.3.2 Effect of Duplicate Identification

In this subsection, we calculate the probability of retrieving duplicate targets within N steps in the group to evaluate the effect of Dup-order. It reveals how close the duplicate


(a) 2Cam

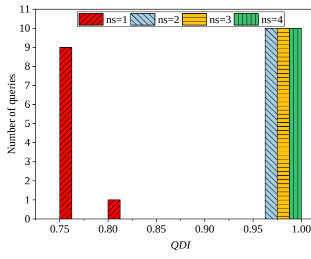

(b) 3Cam

Fig. 16: Duplicate neighborhood rate for query.

targets are in the group. In addition, we further compare the query performance with a matching-based method.

We set $pl = 9$ for Dup-order to process 2Cam and 3Cam, respectively. We first evaluate the duplicate neighborhood rate GDN for each group containing duplicate targets. For Group g, we assume that its duplicate target set is Tar_g, where $\forall Tgt_x \in Tar_g$ has duplicate targets. We further define $GDN_g = \frac{\sum_{Tgt_x \in Tar_g} NR_{ns}(Tgt_x)}{|Tar_g|}$ where $NR_{ns}(Tgt_x) = \frac{near_{ns}(Tgt_x)}{dup(Tgt_x)}$ is the number of targets that duplicate with Tgt_x within neighboring ns steps and $dup(Tgt_x)$ is the total number of targets that duplicate with Tgt_x in the data set. For example, $ns = 1$ means to search the duplicate targets forward 1 step and backward 1 step in the group. The $GDN_g \in [0, 1]$ reveals the aggregation of duplicate targets in a group. Further, we can define the duplicate neighborhood rate QDN for a query. If the result of Query q has groups containing duplicate targets, we assign them to the duplicate group set Gro_q. We define $QDN_q = \frac{\sum_{g \in Gro_q} GDN_g}{|Gro_q|} \in [0, 1]$. This definition indicates that if the query results have duplicate targets, we will use ns steps to find the duplicates in the group with the probability of QDN_q.

We launch ten spatial queries to search data from a global view and set ns from 1 to 4. They can cover 0.4%-94% groups. As shown in Figure 16, we illustrate the effect of Dup-order with a histogram. The X-axis is QDI, and the Y-axis is the number of queries. The average QDI is 0.98 for 2Cam and 0.93 for 3Cam, showing that our method is effective. For 2Cam, we can find duplicate targets with high probability within 1 step. However, for 3Cam, due to more duplicate targets, QDI is lower when $ns = 1$. When $ns = 2$, we get a good result again ($QDI = 0.97$). That shows that duplicate targets are next to each other. This feature can help users quickly eliminate the duplicate effect and find the right transient events.

To further illustrate the advantages of Dup-order, we give a comparison method Dup-match. The difference between Dup-match and Dup-order is that Dup-match removes Z-order at the front-end of *CSPF* and uses distance-based matching within each group to identify duplicates when initiating a query. The distance employs the great-circle distance on a sphere[3]. The disadvantage of Dup-match is (1) the distance threshold is difficult to determine, and (2) the query performance is poor. Our results show that the best distance threshold is 17 and 25 for 2Cam and 3Cam, respectively. They can ensure that Dup-match finds 97% and 96% of duplicate targets from 2Cam and 3Cam, respectively. If the threshold of 3Cam is set to 17, Dup-match will only find 91% duplicate targets. So, the distance threshold is different when the observation environment changes. In addition, Figure 15 shows the query speedup of Dup-order relative to Dup-match on the 2Cam and 3Cam datasets. The X-axis represents the ten spatial queries mentioned in the previous paragraph, and the Y-axis is the speedup. The results show that Dup-order is 6∼ 14× faster than Dup-match. As the data size increases, this performance advantage becomes more apparent. Distance-based matching is a time-consuming operation, and it is hard to be sped up through the computing cluster, causing poor performance. So, Dup-order is more appropriate for real-time query scenarios.

3. https://en.wikipedia.org/wiki/Great-circle_distance

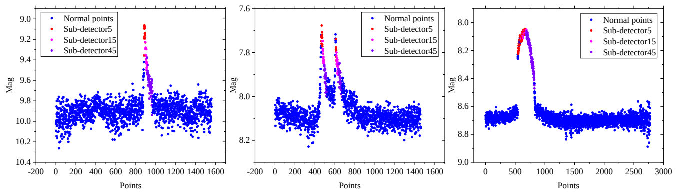


Fig. 17: Three cases of transient event candidates. The color points are alarms issued by *FIAD*.

In short, our method can retrieve all duplicate targets with a probability of 0.93∼0.97, and the performance is 6 ∼ 14× faster than the comparison method. Dup-order supports real-time spatial query, and the query results can make the duplicate targets in a close order within the group.

6.4 Case Study-Achievement in Scientific Event Discovery

We employ our methods to find 27 superflare candidates, two dual-superflare candidates, and seven microlensing candidates from 21.67 million stars, involving 1.09 million catalogs, through real-time analysis of half-year GWAC real data from December 12, 2018, to May 19, 2019. Figure 17 shows 3 cases of valuable transient event candidates.

Superflare. A superflare is a powerful explosion observed on stars. This superflare candidate in Figure 17 (Left) is found at 12:20:22 UT, January 14, 2019. The duration is 28 minutes. Its celestial coordinate system is (RA=63.4557, DEC=9.21318). Our method sends out the first alarm signal in the first 2 minutes. What's more, our alarm signal is also early enough, and these alarm signals can cover the important part of this superflare.

Dual-superflare. A dual-superflare is that two consecutive superflares happen on stars. This dual-superflare candidate in Figure 17 (Middle) was discovered at 14:44:31 UT, on December 23, 2018. The duration is 62 minutes. Its celestial coordinate system is (RA=49.4564, DEC=1.10126). Our method can send out the first alarm signal in the first 5.75 minutes. This phenomenon has a high scientific value. Generally speaking, a star has exploded most of its unconventional energy through a superflare. So, a possible research problem is where the explosion energy is from for the second superflare and its physical mechanism.

Microlensing. Microlensing is an astronomical phenomenon due to the gravitational lens effect. It can be used to detect targets that range from the mass of a planet to the mass of a star, regardless of the light they emit. This microlensing candidate in Figure 17 (Right) is found at 16:31:26 UT, January 7, 2019. The duration is 73 minutes. Its celestial coordinate system is (RA=106.353, DEC=30.9166). Out method issues the first alarm signal at the beginning of 1.75 minutes. Generally speaking, a short microlensing event often means an important scientific discovery.

According to the scientific criterion, it is worth noting that these transient event candidates might not be actual scientific events. More additional scientific analysis, evaluation, and confirmation are needed. However, these cases show that our methodology can find potential transient events in real time.

7 Related Work

The existing researches have extensively studied the analysis of streaming data in various flavors, including (1) unsupervised anomaly detection, (2) anomaly detection in time-domain astronomy, and (3) distributed stream processing. However, they do not provide the real-time analysis methodology to meet the challenges of catalog streams.

Unsupervised anomaly detection. The unsupervised anomaly detection methods for univariate time series have been extensively studied. Recently, many flavors have been developed for this task, such as neural network fitting [27], [31], extreme value recognition [9], [32], time series decomposition [28], [33] and partition-based methods [34], etc. They assume that the anomaly points are being generated by a process that is different from the process that generates the nominal points and tries to identify outliers as the anomalies [4]. However, the definition of an anomaly in catalog streams is highly dependent on the domain. The outliers may be caused by concurrent noise, data missing, and trivial transient events. These factors will lead to a high false-positive rate for these existing methods. In actual scenarios, not all outliers are valuable anomalies. *FIAD* can use the characteristics of the catalog stream to control false positives. In fact, some methods to mitigate false positives have been proposed, such as noise filtering [35], [36], online imputation [31], [37] and lag correction [27]. The popular methods of noise filtering on time-series are wavelet transform [35] and smoothing methods [36]. However, they only work well on the single time series and do not consider intra-catalog concurrent noise. Online imputation interpolates the estimation of missing data to improve performance. However, they are either too expensive or require extra information to support many missing data. Otherwise, more false alarms will be introduced due to the wrong imputation. Our attention windows and *K-Filtering* are both lightweight and do not require additional information. They do not generate additional false alarms and are more suitable for large-scale data missing in the catalog stream. Lag correction can prune the prior false alarms by using the future anomaly trend. This method may cause a false transient event to trigger the expensive follow-up action or miss the best time for the follow-up action.

Anomaly detection in time-domain astronomy. Anomaly events are valuable scientific phenomena included in the catalog stream in time-domain astronomy. They can be identified as transient events [19] and temporary events [38]. A temporary event refers to the sudden visibility of targets that cannot be normally observed below the resolution. Further, transient events are the anomalies of visible targets. These methods for detecting temporary events perform a typical "subtraction" operation to find extra targets as the temporary event through comparing the sampled catalogs with the baseline template [39], [40], [41], [42]. A real-time alarm system [38] is developed to find the temporary event signal to be rapidly cross-correlated with the huge historical catalogs through the rational data organization. However, they cannot search transient events due to the lack of time-dependent patterns of catalog streams. For the detection of transient events, some detection methods [43] and detection framework [19] are proposed for the recognition of abnormal patterns. Still, they ignore the high-level structure of the catalog stream and only treat the directory stream as many target streams. For the detection of transient events, some detection methods [43] and detection frameworks [19] are proposed for the recognition of abnormal patterns. Still, they ignore catalog streams' high-level structure, only treating them as many target streams. Thus, these methods cannot work well under practical scenarios. In addition, some works focused on the real-time challenge of catalog data management [11], [44], [45], [46], and did not analyze it. Our methods support anomaly detection in the catalog stream.

Distributed stream processing. Nowadays, the industry has proposed many distributed data stream processing engines. They can be divided into mini-batch processing mode, e.g., Spark streaming [47] and non-blocking tuple processing mode, e.g., Storm [48], Flink [49] and Samza [50]. For mini-batch processing mode, it improves the system throughput by the data delay processing, but it also increases the processing latency of each data tuple. This processing delay will make it impossible to detect transient events in time. The *CSPF* does not lead to such an issue. The non-blocking tuple processing mode can process the data in the stream without delay. However, these proposed engines are general frameworks and do not include specific optimizations. On the one hand, they cannot adapt to the dynamic changes of catalog streams and meet the real-time constraint; on the other hand, the backend of *CSPF*, i.e., Dup-order, can support the lazy duplicate identification. However, these engines do not involve a built-in operator to process the inter-catalog overlapping problem. In addition, some distributed messaging systems, such as Kafka [51], can cache streaming data, usually being the data middleware. However, our *CSPF* is a catalog stream processing framework, being as the data consumer.

8 Adaptation to Different Domains

Although we give the implementation details of *FIAD* and *CSPF* to support the application of transient event discovery, it does not mean that all algorithm components are only applicable to the proposed typical application. Our methods address anomaly detection of catalog streams in different domains. The whole method framework is still applicable for other application domains, but some algorithm components need to be adapted in combination with different and specific domain knowledge. The basic idea of *building high-quality data* is the same. However, specific clustering methods should be chosen for other applications. First, the position partition method HEALPix in both *Co-Filtering* is suitable for astronomical applications, and other partition methods should be chosen for other fields. The same should apply in *Dup-order* of *CSPF*. We suggest that the simple grid partition method is suitable for 2D plane space. Next, magnitude-related feature clustering should also be replaced by other kinds of feature values. Obviously, in the *domain-oriented anomaly detection* component, the proposed geometry features are also for the astronomical applications, so this component should be updated for other domains. The *data-oriented anomaly detection* and the *missing data downside removal* components are general algorithm components, and they can be used in different domains. Of course, the values of some specific algorithm parameters should be reset based on different application data.

9 Conclusion

Monitoring of massive targets is constantly generating large-scale data in the form of streams. This paper focuses on a special kind of stream, catalog streams, which organize many targets into a high-level data structure catalog. One catalog will be the basic unit of the generated data. Detecting anomalies in catalog streams requires high accuracy and real-time latency to solve two challenging problems.

We propose a novel algorithm *FIAD* that can integrate two complementary methods to detect the anomalies in catalog streams accurately. We propose the concept of an attention window that aims to collect appropriate data for different algorithm components to further optimize the performance of our algorithm. In addition, *FIAD* organizes the decision procedure into four different steps. The first step will filter out high level concurrent noise that may trigger false alerts (building high-quality data); based on the noise free data, we will dynamically identify the data that are far away from their prediction value as the true candidates (data-oriented detection method); the third step will further filter out the false alerts caused by data missing in the second step (data missing downside removal method); finally, we only select the events meeting the domain requirements as the output anomalies (domain-oriented detection method). We develop different attention windows based on different algorithm components to provide the corresponding algorithm component suitable data. So, different algorithm components can achieve better performance on appropriate attention windows. The four filtering-identifying procedures can help each other to include as many anomalies as possible with a very low false-positive rate. The presented algorithm is efficient in improving the true positive rate and reducing the false positive rate simultaneously without sacrificing another. Our method finally achieves a false-positive rate as low as 0.04%.

The proposed processing framework *CSPF* uses the pre-partitioning based performance model to partition catalog streams and distribute them onto different computing resources and process the data in parallel to improve the performance and meet the real-time requirement. To avoid the synchronization operation problem between different nodes to check duplicated targets caused by data overlapping, we develop a mapping method that can transform the 2D positions of different targets into a 1D position feature. Sorting the 1D position features can make the duplicated targets close to each other during the online query. In this way, we can remove the expensive synchronization operations. The experimental results show that the average dispatch latency is 30 milliseconds per catalog and the average detection latency is 2.1 seconds, meeting the 15 seconds real-time performance requirement. In addition, *CSPF* can retrieve all duplicate targets with a probability of 0.93$\sim$0.97, and the performance is $6 \sim 14\times$ faster than the comparison method.

Furthermore, the proposed methods have been employed on the half-year data generated by a telescope GWAC[4] and 36 scientific event candidates have been discovered.

Acknowledgments

This research was partially supported by the grants from the Natural Science Foundation of China (No. 62172423, 61941121, 91846204) and NSF grant number CCF-2109988 (Bader).

References

[1] K. Claffy and M. Fomenkov, "Supporting research and development of security technologies through network and security data collection," University of California San Diego LaJolla United States, Tech. Rep., 2018.

[2] P. E. Dewdney, P. J. Hall, R. T. Schilizzi, and T. J. L. Lazio, "The square kilometre array," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1482–1496, 2009.

[3] H. Zhan and J. A. Tyson, "Cosmology with the large synoptic survey telescope: an overview," *Reports on Progress in Physics*, vol. 81, no. 6, p. 066901, 2018.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[5] M. J. Graham, S. G. Djorgovski, A. Mahabal, C. Donalek, A. Drake, and G. Longo, "Data challenges of time domain astronomy," *Distributed and Parallel Databases*, vol. 30, no. 5, pp. 371–384, 2012.

[6] M. Zhao, Y. Zhou, X. Li, W. Cao, C. He, B. Yu, X. Li, C. D. Elvidge, W. Cheng, and C. Zhou, "Applications of satellite remote sensing of nighttime light observations: Advances, challenges, and perspectives," *Remote Sensing*, vol. 11, no. 17, pp. 1971–2005, 2019.

[7] E. Puchkov *et al.*, "Image analysis in microbiology: a review," *Journal of Computer and Communications*, vol. 4, no. 15, pp. 8–32, 2016.

[8] W. C. Tchuitcheu, C. Bobda, and M. J. H. Pantho, "Internet of smart-cameras for traffic lights optimization in smart cities," *Internet of Things*, vol. 11, p. 100207, 2020.

[9] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.

[10] J. R. Zaur, "From astronomy to data science," *Nature Astronomy*, vol. 2, no. 1, pp. 10–11, 2018.

[11] M. Wan, C. Wu, J. Wang, Y. Qiu, L. Xin, S. Mullender, H. Mühleisen, B. Scheers, Y. Zhang, N. Nes *et al.*, "Column store for GWAC: A high-cadence, high-density, large-scale astronomical light curve pipeline and distributed shared-nothing database," *Publications of the Astronomical Society of the Pacific*, vol. 128, no. 969, pp. 114 501–114 532, 2016.

4. GWAC observation data can be downloaded from https://tianchi.aliyun.com/dataset/dataDetail?dataId=88856.

[12] T. Jayasinghe, C. Kochanek, K. Stanek, B. Shappee, T. W. Holoien, T. A. Thompson, J. Prieto, S. Dong, M. Pawlak, J. Shields *et al.*, "The asas-sn catalogue of variable stars i: The serendipitous survey," *Monthly Notices of the Royal Astronomical Society*, vol. 477, no. 3, pp. 3145–3163, 2018.

[13] V. Lipunov, V. Kornilov, E. Gorbovskoy, N. Tiurina, A. Kuznetsov, P. Balanutsa, V. Chazov, O. Gress, D. Kuvshinov, V. Vladimirov *et al.*, "Master global robotic net: new sites and new result," *Revista Mexicana de Astronomía y Astrofísica*, vol. 48, pp. 42–47, 2016.

[14] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

[15] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, "Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere," *The Astrophysical Journal*, vol. 622, no. 2, pp. 759–771, 2005.

[16] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] G. Welch, G. Bishop *et al.*, "An introduction to the Kalman filter," pp. 127–132, 1995.

[19] J. Qiu, Y. Sun, C. Wu, Z. Du, and J. Wei, "NFD: Toward real-time mining of short-timescale gravitational microlensing events," *Publications of the Astronomical Society of the Pacific*, vol. 130, no. 992, pp. 104 504–104 516, 2018.

[20] S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork, "Comparison of different methods for univariate time series imputation in R," *arXiv preprint arXiv:1510.03924*, 2015.

[21] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[22] P. Koutoupis, "The Linux RAM disk," *Linux+ Magzine*, pp. 36–39, 2009.

[23] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Computer Science Review*, vol. 24, pp. 63–79, 2017.

[24] K. Lee, B. Zheng, H. J. Li, and W. C. Lee, "Approaching the skyline in z order," *Vldb*, pp. 279–290, 2007.

[25] T. Macedo and F. Oliveira, *Redis cookbook: Practical techniques for fast data manipulation*. " O'Reilly Media, Inc.", 2011.

[26] Y. Xu, C. Wu, M. Wan, J. Zhao, H. Tian, Y. Qiu, J. Wei, and Y. Liu, "A fast cross-identification algorithm for searching optical transient sources," *Astronomical Research and Technology*, vol. 10, no. 3, pp. 273–282, 2013.

[27] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.

[28] K. Zhang, R. Gençay, and M. E. Yazgan, "Application of wavelet decomposition in time-series forecasting," *Economics Letters*, vol. 158, pp. 41–46, 2017.

[29] W. H. Press and S. A. Teukolsky, "Savitzky-golay smoothing filters," *Computers in Physics*, vol. 4, no. 6, pp. 669–672, 1990.

[30] C. Yang, X. Meng, Z. Du, Z. Duan, and Y. Du, "Data management in time-domain astronomy: Requirements and challenges," in *International Conference on Big Scientific Data Management*. Springer, 2018, pp. 32–43.

[31] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 187–196.

[32] J. Li, S. Di, Y. Shen, and L. Chen, "Fluxev: A fast and effective unsupervised framework for time-series anomaly detection," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 824–832.

[33] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "RobustSTL: A robust seasonal-trend decomposition algorithm for long time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5409–5416.

[34] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[35] Ç. P. Dautov and M. S. Özerdem, "Wavelet transform and signal denoising using wavelet method," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2018, pp. 1–4.

[36] E. de Bézenac, S. S. Rangapuram, K. Benidis, M. Bohlke-Schneider, R. Kurle, L. Stella, H. Hasson, P. Gallinari, and T. Januschowski, "Normalizing Kalman filters for multivariate time series analysis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2995–3007, 2020.

[37] T. G. Dietterich and T. Zemicheal, "Anomaly detection in the presence of missing values," *arXiv preprint arXiv:1809.01605*, 2018.

[38] J. Becla, K.-T. Lim, S. Monkewitz, M. Nieto-Santisteban, and A. Thakar, "Organizing the lsst database for real-time astronomical processing," in *Astronomical Data Analysis Software and Systems*, 2008, pp. 114–117.

[39] C. Alard, "Image subtraction using a space-varying kernel," *Astronomy and Astrophysics Supplement Series*, vol. 144, no. 2, pp. 363–370, 2000.

[40] D. Bramich, "A new algorithm for difference image analysis," *Monthly Notices of the Royal Astronomical Society: Letters*, vol. 386, no. 1, pp. 77–81, 2008.

[41] I. Telezhinsky, D. Eckert, V. Savchenko, A. Neronov, N. Produit, and T.-L. Courvoisier, "The catalog of variable sources detected by integral-i. catalog and techniques," *Astronomy & Astrophysics*, vol. 522, no. 1, pp. 280–295, 2010.

[42] W. A. Bhatti, M. W. Richmond, H. C. Ford, and L. D. Petro, "VARIABLE POINT SOURCES IN SLOAN DIGITAL SKY SURVEY STRIPE 82," *The Astrophysical Journal Supplement Series*, vol. 186, no. 2, pp. 233–258, 2010.

[43] T. Feng, Z. Du, Y. Sun, J. Wei, J. Bi, and J. Liu, "Real-time anomaly detection of short-time-scale GWAC survey light curves," in *Proceedings of IEEE International Congress on Big Data*, 2017, pp. 224–231.

[44] C. Yang, X. Meng, and Z. Du, "Cloud based real-time and low latency scientific event analysis," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 498–507.

[45] C. Yang, X. Meng, Z. Du, J. Qiu, K. Liang, Y. Du, Z. Duan, X. Ma, and Z. Fang, "AstroServ: A distributed database for serving large-scale full life-cycle astronomical data," in *Proceedings of Big Scientific Data Management*. Springer International Publishing, 2019, pp. 44–55.

[46] Z. Duan, C. Yang, X. Meng, Y. Du, J. Qiu, X. Ma, Z. Du, X. Zhang, B. Niu, and C. Wu, "Scidetector: Scientific event discovery by tracking variable source data streaming," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 2040–2043.

[47] D. Cheng, Y. Chen, X. Zhou, D. Gmach, and D. Milojicic, "Adaptive scheduling of parallel jobs in spark streaming," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[48] R. Ranjan, "Streaming big data processing in datacenter clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 78–83, 2014.

[49] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink: Stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, pp. 28–38, 2015.

[50] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, and R. H. Campbell, "Samza: stateful scalable stream processing at LinkedIn," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1634–1645, 2017.

[51] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011, pp. 1–7.

Chen Yang received the PhD degree in 2020, at Information School, Renmin University of China. He is a postdoctoral fellow at China National Clearing Center and Department of Computer Science and Technology, Tsinghua University. His primary research focus is real-time analysis of big data, performance profiling of big data system and artificial intelligence for IT operations.

Zhihui Du received the PhD degree in 1998, at Department of Computer Science, Peking University. He has published more than 200 international journal and conference articles. His research interests include high-performance computing, cloud computing, and big data analyzing. He serves on the Editorial Boards of the ACM Transactions on Parallel Computing, and the Journal of Parallel and Distributed Computing.

Xiaofeng Meng was born in 1964. He is a professor and PhD supervisor at Information School, Renmin University of China, as well as a Fellow of the CCF. His main research interests include cloud data management, web data management, scientific data management, and privacy protection.

Xukang Zhang is a PhD candidate at Information School, Renmin University of China. His main research interests include scientific data analysis and artificial intelligence for database system.

Xinli Hao is a PhD candidate at Information School, Renmin University of China. Her main research interests include time series analysis, interpretable machine learning, intelligent scientific discovery.

David A. Bader is a Distinguished Professor and founder of the Department of Data Science in the Ying Wu College of Computing and Director of the Institute for Data Science at New Jersey Institute of Technology. He is a Fellow of the IEEE, ACM, AAAS and SIAM and recipient of the IEEE Sidney Fernbach award. He advises the White House, most recently on the National Strategic Computing Initiative (NSCI). Prof. Bader is a leading expert in solving global grand challenges in science, engineering, computing, and data science. His interests are at the intersection of high-performance computing and real-world applications, including cybersecurity, massive-scale analytics, and computational genomics, and he has co-authored over 300 articles in peer-reviewed journals and conferences. Prof. Bader currently serves as Editor-in-Chief of ACM Transactions on Parallel Computing and previously served as the Editor-in-Chief of IEEE Transactions on Parallel and Distributed Systems.