RESEARCH-ARTICLE

ISSD: Indicator Selection for Time Series State Detection

CHENGYU WANG, National University of Defense Technology China, Changsha, Hunan, China

TONGQING ZHOU, National University of Defense Technology China, Changsha, Hunan, China

LIN CHEN, National University of Defense Technology China, Changsha, Hunan, China

SHAN ZHAO, Hefei University of Technology, Hefei, Anhui, China

ZHIPING CAI, National University of Defense Technology China, Changsha, Hunan, China

ISSD: Indicator Selection for Time Series State Detection

CHENGYU WANG, National University of Defense Technology, China
TONGQING ZHOU*, National University of Defense Technology, China
LIN CHEN, National University of Defense Technology, China
SHAN ZHAO*, Hefei University of Technology, China
ZHIPING CAI*, National University of Defense Technology, China

Time series data from monitoring applications captures the behaviours of objects, which can often be split into distinguishable segments that reflect the underlying state changes. Despite the recent advances in time series state detection, the indicator selection for state detection is rarely studied, most of state detection work assumes the input indicators have been properly or manually selected. However, this assumption is disconnected from practice, on one hand, manual selection is not scalable, there can be up to thousands of indicators for the runtime monitoring of certain objects, e.g., supercomputer systems. On the other hand, performing state detection on a large amount of raw indicators is both inefficient and redundant. We argue that indicator selection should be made an upstream task for selecting a subset of indicators to facilitate state analysis. To this end, we propose ISSD (**I**ndicator **S**election for **S**tate **D**etection), an indicator selection method for time series state detection. At its core, ISSD attempts to find an indicator subset that has as much high-quality states, which is measured by the channel set completeness and quality we invent based on segment-level sampling statistics. Such an indicator selection process is transformed into a multi-objective optimization problem and an approximation algorithm is designed to solve the NP-hard searching for specific end point in the Pareto front. Experiments on 5 datasets and 4 downstream methods show that ISSD has significant selection superiority compared with 6 baselines. We also elaborate on two observations of selection resilience and channel sensitivity of existing state detection methods and appeal to further research on them.

CCS Concepts: • **Information systems → Data mining**; • **Computing methodologies → Feature selection**.

Additional Key Words and Phrases: Time series state detection, time series segmentation, indicator selection

ACM Reference Format:
Chengyu Wang, Tongqing Zhou, Lin Chen, Shan Zhao, and Zhiping Cai. 2025. ISSD: Indicator Selection for Time Series State Detection. *Proc. ACM Manag. Data* 3, 1 (SIGMOD), Article 48 (February 2025), 25 pages. https://doi.org/10.1145/3709698

1 Introduction

In recent years, Internet-based services and IoT applications have become indispensable in daily life, constantly generating a large amount of multivariate time series data, with each channel

*Corresponding authors

Authors' Contact Information: Chengyu Wang, National University of Defense Technology, Changsha, China, chengyu@nudt.edu.cn; Tongqing Zhou, National University of Defense Technology, Changsha, China, zhoutongqing@nudt.edu.cn; Lin Chen, National University of Defense Technology, Changsha, China, chenlin1080@outlook.com; Shan Zhao, Hefei University of Technology, Hefei, China, zs50910@mail.ustc.edu.cn; Zhiping Cai, National University of Defense Technology, Changsha, China, zpcai@nudt.edu.cn.

ACM 2836-6573/2025/2-ART48
https://doi.org/10.1145/3709698

corresponding to an indicator that carries rich behavior information for the evolving states of the monitored objects [16, 28, 40, 46]. To mine useful patterns and actionable rules from these data, it is critical to develop state detection methods for segmenting a time series into segments and identifying the corresponding states (e.g., run and jump in activity recognition) [18, 24, 28, 46].

1.1 Backgrounds

In time series state detection, a state is defined as a sustained behavior of a target over a period (e.g., walking, running). It is an abstract and encompassing concept that cannot be rigidly defined by explicit rules, akin to clusters in clustering analysis [25, 49]. The data (i.e., segments) generated by a target within the same state is characterized by high internal similarity, often displaying recurring patterns. The persistence and alternation of states yield distinct segments. State detection is sometimes also referred to as *time series segmentation (TSS)*, but the term "TSS" is overloaded. It may also refer to boundary detection [8, 12, 16, 40], which focuses on identifying the boundaries between segments of different states, whereas state detection is an enhanced task that involves assigning labels to each segment or time point [18]. *This paper focuses on state detection as the target downstream task, that is, infer the state sequence with state labels for given time series.* A more detailed introduction to state detection can be found in [18, 24, 46].

1.2 Motivation

Similar to feature selection [23, 42], the need of indicator selection for state detection is also universal. For instance, in action recognition datasets [1], over 60 channels may be collected, yet typically only 4 channels are used [24, 28, 46]. Likewise, in sleep staging, it is common to select about 3 signals from a variety of ECG, EOG, and EEG indicators [35]. Moreover, indicator selection is more noteworthy for time series data from large-scale systems, such as supercomputers [44] and crane vessels [22], which can include thousands of indicators, whereas existing state detection methods generally take 1∼10 channels as input [18, 24, 28, 46]. Although some domains [8, 16] can achieve accurate state detection with a single, well-chosen indicator, identifying such an indicator requires a systematic approach to channel selection. Despite the extensive study of the state detection problem, previous state detection works have largely sidestepped the indicator selection problem by relying on manual selection or using pre-selected indicators based on prior knowledge or experience [18, 24, 28, 32, 46], which often disconnects from real-world practice.

On one hand, manual selection is not scalable. As will be seen in the case study (§ 4.3.2), indicator selection is often non-intuitive, particularly for heterogeneous datasets, where each time series may contain different states, and the states may be hard to distinguish through a single channel. In this situation, operators will have to check across multiple time series to confirm whether the current channel combination can cover all the state changes, making manual indicator selection time-consuming, prohibitive, and not scalable, especially for some larege-scale systems.

On the other hand, performing state detection on a large amount of indicators is inefficient and redundant w.r.t., collection and computation overheads. 1) **For inefficiency**, as noted in [46], many methods exhibit low scalability against time series dimension (i.e., the number of indicators), with computation time increases rapidly as the number of indicators grows. For example, the computation time of an advanced state detection method TICC [18] rises from from approximately 11 seconds to nearly 3 minutes as the number of channels increases from 1 to 20 when processing a time series of length 30k [46]. Another classic method, AutoPlait [28], shows similar trends, with processing times of 11 seconds for 1 channel and 472 seconds for 10 channels [46]. Recent state detection works have emphasized deployment on embedded devices with limited computational resources (e.g., CPU frequency, RAM) [24], further highlighting the importance of reducing computational overhead through effective channel selection. 2) **For redundancy**, it is observed that simply using a small subset of all channels can segment the time series more accurately than either using all channels or a single channel, summarized as the "*goldilocks*" phenomenon [16]. A similar observation has also been made in the context of time series classification [21]. These observations underscore the critical need for effective indicator selection methods for state detection.

This paper argues that indicator selection should be made an upstream task of time series state detection for selecting a subset of indicators with good characteristics and enabling a better performance and higher efficiency for analyzing states. Note that indicator selection is also referred to as *channel selection* or *sensor selection* in different domains of the literature [11, 19]. Without causing confusion, we will interchangeably use these terms.

1.3 Challenges and Contributions

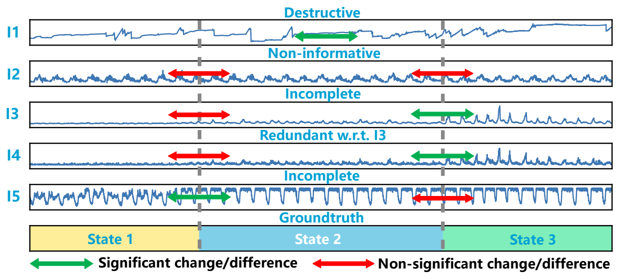

Fig. 1. An example of the indicator selection problem.

Figure 1 shows an example of indicator selection for state detection in server monitoring time series. Wherein, the bottom colored bar is the ground truth, red/green arrows indicate there is non-significant/significant difference between two adjacent segments (i.e., poor/proper indicator for state change). As shown, indicator 1 is featured as destructive for state detection as it varies constantly and behaves very differently in the same state. Indicator 2 is considered to be non-informative as it has almost no state changes. Indicators 3, 4, and 5 are not complete because each of them cannot cover all the states alone (ground truth), but their combination is complete because it can reflect the ground truth. Indicators 3, 4 are mutually redundant, as they are highly correlated and behaves similarly. Given a multivariate time series and the state sequence from out-of-band context information (e.g., system log, operation events), the indicator selection goal is to find a complete indicator/channel subset (3, 5 or 4, 5 in this example), which cover as much potential states as possible and are inherently indicative of state differences (high quality).

Attaining this goal requires more than an incremental of existing techniques. The major challenges include: 1) **Heterogeneity**: In practice, the same bunch of indicators would often capture different states (with small overlap) when monitoring different entities. For example, the movement time series (with indicators like pace, speed, and heartbeat) of person 1 may contain walk and run, while person 2 with hop and jump. Selecting on single time series would result in miss of states and poor generality. For comprehensive consideration of all time series, tuning existing indicator selection designs (e.g., those for classification) to sew channels together is unscalable by incurring drastic overheads of pair-wise segment distance calculation. 2) **Incompleteness**: Each single channel may not be sufficient to record all the state changes, thus requiring a joint exploration with other ones. For example, the jump and hop (left) may be difficult to distinguish through one indicator of the left leg, but can be easily identified through the indicator combination of both legs.

This paper proposes an effective indicator selection method ISSD. ISSD borrows the sampling difference significance in hypothesis testing to measure the differences of segments in each channel. With such segment-level measurements, it then formulates the selection goal as attaining high channel (set) completeness and quality based on setting clear boundary between inner-state and inter-state difference statistics of some selected indicators (handling challenge 2). As a channel-wise completeness and quality exploration, ISSD could naturally relieve the heterogeneity challenge (challenge 1) by efficiently scaling the above calculations to all time series. We transform the completeness- and quality-oriented selection as a multi-objective optimization with Pareto front, prove the NP-hardness for finding the favored end-point in such a front, and design an approximate solution for fast searching. The contributions are summarized as follows:

- This paper studies the rarely studied indicator selection problem in the context of time series state detection. We formally define the problem with a novel concept of channel (set) completeness and quality.
- The proposed ISSD [1] method leverages the solving of quality & completeness optimization for selecting indicators with comprehensive states. Extensive experiments show the effectiveness of ISSD, which outperforms 6 baselines by a large margin.
- We highlight valuable experimental observations on selection resilience of existing state detection designs as a overlooked metric in existing works and their channel sensitivity as an interesting further research topic.
- We also highlight the benefits of the evaluation framework built for evaluating ISSD and baselines. The framework not only supports easy integration of new selection and downstream methods to facilitate further research along this track but is also equipped with data generation scripts, visualization tools, and statistical tools to facilitate indicator selection for practitioners.

The rest of this paper is organized as follows. In § 2, we review related works. In § 3, we formulate the indicator selection problem, introduce ISSD, and provide discussions. We evaluate ISSD and discuss experimental observations in § 4. The paper is concluded in § 5.

2 Related work

2.1 Time Series channel selection

Theoretically, the channel selection methods designed for multivariate time series classification (MTSC) [10, 20] can also be applied to select indicators for state detection, as a time series can be split into segments to be classified based on its state sequence. Despite the promising results these methods have achieved in MTCS, they failed to consider the unique completeness issue in state detection, which may lead to some states being difficult to be identified. Another potential problem is that time series classification methods assume the segment boundaries are given and take the segmented subseries as input, yet the state detection methods, due to the lack of boundary information, usually take a small sliding window or time point as input, and infer the states by modeling the relationship between these windows or time points. This inherent difference in characteristics may render indicator selection methods used for classification less suitable for time series state detection. For example, the indicator selection methods for classification may focus too much on the overall differences between two segments. However, in state detection tasks, differences in patterns or motifs often play a crucial role in distinguishing states. ISSD differs from these methods in that it is particularly developed for time series state detection with dedicated design, which take into account not only the inner/inter-state distance, but also the completeness of selection results.

[1]Code is available at https://github.com/Lab-ANT/ISSD

Note that there is a significant difference between indicator selection and traditional feature selection due to the temporal relationship between time points, as analyzed in many indicator selection literature [10]. An indicator is essentially a univariate time series, the semantic of a time point is closely related to its context (i.e., the subseries around the point). Traditional feature selection methods [23, 42] treat the observations in each indicators as independent multivariate samples, ignoring the temporal relationship between them, making these methods unsatisfactory for indicator selection problem. To demonstrate this, we also used a strong traditional feature selection method for comparison, which is detailed in § 4.

2.2 Dimension reduction

Dimension reduction methods, e.g., PCA [38], LDA [17], UMAP [4, 29], and t-SNE [45], use linear or nonlinear methods to map high-dimensional data into low-dimensional space, thereby reducing the number of features. While it may seem feasible to apply these methods to reduce the number of channels by considering each time point as a multivariate observation, they are not suitable for channel selection because the channel set includes not only task-related channels, but also noise and destructive channels. Dimension reduction methods largely do not distinguish and exclude these channels, but only faithfully integrate all information into a low-dimensional representation, resulting in a large amount of noise and harmful information in the low-dimensional representation (see § 4.3.1). Furthermore, using dimension reduction methods can cause inconvenience for the online deployment of state detection methods. Unlike selection methods, newly arrived data needs to be processed by reduction methods before being input into downstream methods. ISSD differs from these methods in that it can exclude non-informative, redundant, and destructive channels, thereby trying to find a simple yet complete channel set to reduce the influence of noise/destructive channel.

3 Methods

The channel selection goal is to find a channel subset to cover all the potential states (i.e., completeness). This inherently requires assessing the quality of channels w.r.t. state, because the provided state labels are indistinguishable in some channels. For example, indicator ‘tcp_connections’ is unlikely to reflect the state changes between the I/O-intensive job and the computation-intensive job in a supercomputer system, thus should not be selected even full of states. Considering this, our method builds on sampling subseries (i.e., samples) in different segments to *statistically estimate segments’ differences for quality and completeness optimization.*

3.1 Segment-level Difference Estimation

We adopt the nearest neighbor statistics [41], commonly used for comparing the difference significance in two-sample hypothesis tests [27, 41], to measure the difference/distance of different segments by randomly sampling them.

Given a time series segment $\mathbf{s}_i$, we denote a randomly sampled subseries set from $\mathbf{s}_i$ as $A_i=\{x_k\}_{k=1}^{n_i}$ (the size of A_i is n_i). For two time series segments $\mathbf{s}_i$ and $\mathbf{s}_j$ and their corresponding subseries set A_i and A_j, let $A=A_i \cup A_j$, $NN_r(x_k, A)$ represents the r-th nearest neighbor (also a sampled subseries) to sample x_k within $A/\{x_k\}$. Then we can define an indicator function as:

$$I_r(x_k, A_i, A_j)=\begin{cases} 1, & x_k \in A_l \wedge NN_r(x_k, A) \in A_l,\ l \in \{i, j\} \\ 0, & otherwise \end{cases} \tag{1}$$

which indicates whether x_k and its r-th nearest neighbor belong to the same segment. The statistical difference between S_i and S_j, for testing whether samples from them are from the same distribution (the null hypothesis), can be calculated in the following quantity,

$$\theta_{i,j} = \frac{1}{n\mathbf{r}} \sum_{k=1}^{n} \sum_{r=1}^{\mathbf{r}} I_r(x_k, A_i, A_j) \tag{2}$$

where $n = n_i + n_j$ is the total amounts of samples in A_i and A_j and $\mathbf{r}$ is the upper bound of neighbors. The statistic $\theta_{i,j}$ is essentially a ratio that measures how many of the r nearest neighbors of all the n subseries are from the same segment. Intuitively, $\theta_{i,j}$ is small when the distributions of the two tested segments are statistically the same (i.e., the null hypothesis H_0) with well-mixed samples, while is large when the two underlying distributions (states) are different (i.e., the alternative hypothesis H_1). We refer to the above process as "nntest" in the rest of this paper.

The nntest process can be accelerated by vector retrieval algorithms [2]. Considering that the subsequences in our scenario are usually high-dimensional vectors, methods that perform poorly in these cases may not be appropriate choices, for example, KD-Tree [5] can degrade to linear search in high-dimensional spaces [31]. In this paper, we use Ball-Tree [26, 30] for nearest neighbor query because it performs well with high-dimensional spaces [31].

3.2 Formulation of State Completeness

In this section, we introduce the terminology and notations in terms of channel selection. An intuitive explanation for the definitions and concepts are shown in Figure 2.

Performing pair-wise two-sample test for all segments in a channel c yields a statistic matrix $\Theta_c = (\theta^c_{i,j})$, where $\theta^c_{i,j}$ represents the segment-level difference between segment $\mathbf{s}_i$ and $\mathbf{s}_j$ in channel c. A higher $\theta^c_{i,j}$ implies a lower likelihood that $\mathbf{s}_i$ and $\mathbf{s}_j$ belong to the same state. Simultaneously, we can obtain the true indicator matrix according to the ground truth, denoted as $\Omega = \{\omega_{i,j}\}$, wherein 0 (*false*) indicates that two segments share the same state, while 1 (*true*) indicates from different states. Note that each channel possesses its own statistic matrix Θ, while all channels share the same true indicator matrix Ω.

With Θ and Ω at hand, let $s_{inner}=\{(i,j)|\omega_{i,j}=0\}$ denote the inner-state segment indices and $s_{inter}=\{(i,j)|\omega_{i,j}=1\}$ for the inter-state. The inter-state difference can be quantified by $\frac{1}{M}\sum\theta_{i,j}$, $(i,j) \in s_{inter}$, $M = |s_{inter}|$, where a higher value indicates better quality, and the inner-state difference is $\frac{1}{N}\sum\theta_{i,j}$, $(i,j) \in s_{inner}$, $N = |s_{inner}|$, with lower values favored for selection. On this basis, we define the quality score (akin to state informativeness) of a channel:

$$\tau = \frac{1}{M} \sum_{\substack{i,j \\ (i,j)\in s_{inter}}} \theta_{i,j} - \frac{1}{N} \sum_{\substack{i,j \\ (i,j)\in s_{inner}}} \theta_{i,j} \tag{3}$$

A higher τ means higher inter-state difference and inner-state similarity. Meanwhile, to verify whether the states in a channel are distinguishable, we define two additional variables:

$$\hat{\tau} = max(\theta_{i,j}),\ (i,j) \in s_{inner} \tag{4}$$

$$\tilde{\tau} = min(\theta_{i,j}),\ (i,j) \in s_{inter} \tag{5}$$

where $\hat{\tau}$ reflects the maximum inner-state difference. A high $\hat{\tau}$ indicates the presence of segments belonging to the same state with significant differences in the channel; $\tilde{\tau}$ reflects the minimum inter-state difference. A low $\tilde{\tau}$ indicates the presence of segments belonging to different states that are difficult to distinguish in the channel. Obviously, if $\hat{\tau} < \tilde{\tau}$, there must be a threshold that can distinguish all states apart, otherwise, such a threshold does not exist.

Before defining *complete channel* (*set*), let us define a function $p(\cdot)$ to convert the statistic matrix Θ into a binary indicator matrix with boolean values, which indicates which segments can be distinguished under the given threshold δ, where 1 ($true$) for distinguishable, and 0 ($false$) for indistinguishable.

$$p(\Theta, \delta) = (\theta'_{i,j}),\ \theta'_{i,j} = \begin{cases} 1, & \theta_{i,j} > \delta \\ 0, & \theta_{i,j} \leq \delta \end{cases} \tag{6}$$

The completeness of a channel c can be defined as $cnt\big(p(\Theta_c, \hat{\tau}_c)\big)$, where $cnt(\cdot)$ is a counting function that counts how many 1 ($true$) elements there are in an indicator matrix. Using $\hat{\tau}$ as the threshold is an adaptive manner that assumes all channels are non-destructive. A destructive channel will be of a large $\hat{\tau}$, thus resulting in a low completeness.

Definition 1. *(Complete channel) A channel c is complete, iff. $\hat{\tau}_c < \tilde{\tau}_c$. At this time, $\forall \delta \in [\hat{\tau}_c, \tilde{\tau}_c]$, $p(\Theta_c, \delta) = \Omega$, and we call channel c is complete at interval $[\hat{\tau}_c, \tilde{\tau}_c]$.*

Before introducing complete channel set, let us define a binary operator $\vee$, which performs logical OR operations between the corresponding elements of two indicator matrices, i.e., $A \vee B = (a_{i,j} \vee b_{i,j})$, we will also use $\vee\{\Theta_i\}_{i=1}^N$ to denote continuous $\vee$ operations between multiple matrices, i.e., $\vee\{\Theta_i\}_{i=1}^N = \Theta_1 \vee \Theta_2 \vee \cdots \vee \Theta_N$.

Definition 2. *(Complete channel set) A channel set C, indexed by $\{c\}_{c=1}^N$, is complete, iff. $\vee\big\{p(\Theta_c, \hat{\tau}_c)\big\}_{c=1}^N = \Omega$.*

This definition can be used to check whether a channel set is complete. We refer to a complete set with the minimum number of channels as the *minimum complete set*. Obviously, the minimum complete channel set may not be unique. In practice, the minimum complete set is usually not a good choice because many time series have complete channels, thus the minimum complete set usually contains only one channel. Moreover, [16] has reported that using only one channel is not a good choice. A more reasonable choice is to specify a desired number of channels K and choose the most complete channel set with the maximum inter-state difference and inner-state similarity. The completeness of a channel set can be measured by $cnt\big(\vee\big\{p(\Theta_c, \hat{\tau}_c)\big\}_{c=1}^N\big)$, which intuitively indicates the number of inter-state segments that the channel set can distinguish.

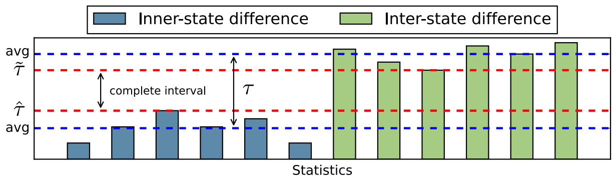

Fig. 2. An illustration of concepts.

3.3 Modeling as an Optimization Problem

With the above definitions, the channel selection problem can be formulated as follows: For a channel set at hand, given the desired number of channels K, how to determine a channel subset,

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{i=1}^{N} x_i \tau_i \quad s.t. \sum_{i=1}^{N} x_i = K, \vee\big\{p(\Theta_i, \hat{\tau}_i) | x_i = 1\big\} = \Omega \tag{7}$$

where $\mathbf{x} = [x_1 \ldots x_N]$ is the solution vector, with $x_i = 1$ indicating channel i is selected, 0, otherwise, τ_i, Θ_i are the quality score and statistic matrix of channel i, respectively.

The above situation is idealized, however, in reality, complete channel set may not exist. In this case, the completeness guarantee cannot be satisfied, the solution for eqn (7) may not exist.

Therefore, an appropriate approach is to treat completeness as an optimization objective, instead of a constrain, and formulate the channel selection problem as a multi-objective optimization problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \big(f_1(\mathbf{x}), f_2(\mathbf{x})\big), \ \ s.t. \sum_{i=1}^{N} x_i = K \tag{8}$$

$$f_1(\mathbf{x}) = \sum_{i=1}^{N} x_i \tau_i \tag{9}$$

$$f_2(\mathbf{x}) = cnt\Big(\vee \big\{ p(\Theta_i, \hat{\tau}_i) \big\} \Big), i \in \{ i | x_i = 1 \} \tag{10}$$

The solution for eqn (8) may be more than one, because the objectives f_1, f_2 may have conflict, i.e., the channel set with the best completeness may not necessarily have the largest quality score, and the channel set with the largest quality score may not necessarily have the best completeness. The set of all solutions for eqn (8) is called the Pareto front [7] in multi-objective optimization. Theoretically, every solution in the Pareto front is the optimal solution and cannot be further optimized. Achieving optimization on one objective inevitably sacrifices other objectives.

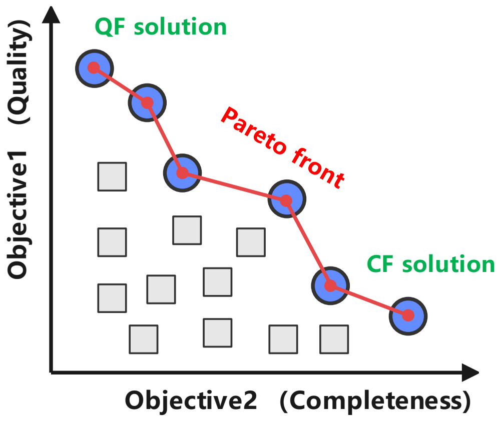

Fig. 3. An illustration of Pareto front in our problem. Blue dots represent the Pareto optimal solutions. The set of all Pareto optimal solutions forms the Pareto front.

An example of pareto front in our problem is shown in Figure 3, the solutions can be approximated by pareto front searching algorithms such as [7, 52]. The two end points of the Pareto front have special meanings. The upper end point corresponds to prioritizing the optimization of quality (f_1), which means selecting the solution with the highest completeness among all solutions that reach the highest quality. While the lower end point corresponds to prioritizing the optimization of completeness (f_2), which means selecting the solution with the highest quality among all solutions that reach the highest completeness. We call them Quality-first (QF) solution and Completeness-first (CF) solution, respectively.

Note that obtaining the QF and CF solutions does not correspond to only optimizing f_1 or f_2 separately, as there may be multiple solutions achieving the highest total quality but have different levels of completeness, and there may also be multiple solutions achieving the highest completeness but have different total quality. We will further discuss this point in § 3.4.

3.4 Fast Search for the QF and CF Solutions

There is no superiority or inferiority between the solutions in Pareto front, unless there is preferences for the completeness of some states. Therefore, generally, it is unnecessary to obtain all solutions, and the most commonly used solutions are the two end points, which represent the quality-first (QF) and completeness-first (CF) solution, respectively (see Figure 2). For example, on datasets with better completeness, most channels are complete, it is more reasonable to use the QF solution. In contrast, on datasets with poor completeness, there are almost no complete channels, it is more appropriate to use the CF solution.

3.4.1 NP-hardness Analysis. For simplicity, let us first make the following assumptions. **Assumption 1**: There is only one solution with the highest total quality, or multiple solutions with the highest total quality have the same completeness; **Assumption 2**: There is only one solution with the highest completeness, or multiple solutions with the highest completeness have the same total quality. When Assumption 1 holds, the QF solution can theoretically be found by optimizing only f_1; Correspondingly, when Assumption 2 holds, the CF solution can theoretically be found by optimizing only f_2.

QF solution: For f_1, due to its larger and continuous range, the probability of different solutions obtaining the same total quality is small. Therefore, Assumption 1 could hold to a large extent. In this case, the QF solution can be safely found by eqn. (9), which is essentially selecting the top-K channels with the highest quality score τ.

CF solution: Searching for the CF solution is not that easy, because completeness is of discrete value with a much smaller range, it is quite often that Assumption 2 does not hold. Only optimizing f_2 may not achieve the desired CF solution, there may be multiple solutions with the highest completeness, but having different quality scores. Therefore, it is necessary to optimize f_1 simultaneously as well to achieve the CF solution. Taking a step back, even if Assumption 2 holds, searching for the QF solution is NP-hard [15], because optimizing f_2 is essentially a maximum coverage problem [33]. As will be readily seen, each indicator matrix $\Lambda = p(\Theta, \hat{\tau}) = (\lambda_{i,j})$ has a corresponding set representation $S = \{(i, j)|\lambda_{i,j} = 1\}$. Obviously, the $\vee$ operation between indicator matrices is equivalent to the union ($\cup$) operation between set representations, i.e., $\Lambda_1 \vee \Lambda_2 = \Lambda_3 \iff S_1 \cup S_2 = S_3$. Hence, maximizing the completeness is essentially to select at most K channels such that the maximum elements are covered, which is NP-hard [33]. LN-approximability results show that greedy algorithm is essentially the best-possible polynomial time approximation algorithm for maximum coverage unless $P = NP$ [13]. The problem at hand is more complex than the simple maximum coverage problem, as we need to simultaneously maximize objective f_1 while searching for the maximum coverage.

3.4.2 Approximation Algorithm. Given the above consideration, we design a greedy algorithm for approximating the CF solution, which chooses channels according to two rules: 1) At each step, choose the channel with the largest number of uncovered elements; 2) If there are more than one candidate channels, choose the channel with the maximum τ. Details are described in algorithm 1 (in set representation).

3.5 Adapt ISSD to Multiple Time Series

We have explained how ISSD works on a single time series. However, selecting channels on one time series can easily lead to local optima, now we explain how to adapt ISSD to multiple time series. To achieve this goal, we do not need to modify ISSD, but only fuse the information of corresponding channels across different time series. The optimization algorithm of ISSD requires three pre-calculated statistic information, which are the quality score of each channel τ_j, the

Algorithm 1: Fast Searching for CF Solution

```
Input: $\mathcal{S} = \{S_i\},\ \mathcal{T} = \{\tau_i\},\ K$
Output: Index set of selected channels $\Gamma$
1  $\Gamma \leftarrow \emptyset$ // initialize result set
2  $C \leftarrow \emptyset$ // initialize current coverage
3  $N \leftarrow |\mathcal{S}|$ // $N = |\mathcal{S}| = |\mathcal{T}|$
4  $R \leftarrow \{1, 2, \ldots, N\}$ // initialize index set
5  while $|\Gamma| < \min(K, N)$ do
6      $\{c\} \leftarrow \arg\max_{i \in R} |S_i \setminus C|$ // find the set of candidates with maximum coverage
7      $idx \leftarrow \arg\max_{i \in \{c\}} \tau_i$ // select the channel with highest quality score $\tau$ from candidates
8      $\Gamma \leftarrow \Gamma \cup \{idx\}$ // update result set
9      $R \leftarrow R \setminus \{idx\}$ // remove the selected one from index set
10     $C \leftarrow \cup\{S_i\}, i \in \Gamma$ // update current coverage
11 end
12 return $\Gamma$ // return the indices of selected channels
```

indicator matrix of each channel Λ_j, and the true indicator matrix Ω shared by all channels. We use superscripts to indicate which time series the information comes from, e.g., τ_j^i denote the quality score of j-th channel in i-th time series. The averaged quality score of j-th channels across all time series can be calculated by $\bar{\tau}_j = \frac{1}{M}\sum_{i=1}^{M} \tau_j^i$, where M is the number of time series. For the indicator matrix of each channel, we convert them into set representation and calculation the union of corresponding channels $E_j = \cup\{S_j^i\}_{i=1}^{M}$. Then these fused $\{\bar{\tau}_j\}_{j=1}^{M}$, $\{E_j\}_{j=1}^{M}$ can be directly input into the optimization algorithm as in the case of single time series.

3.6 Handling Redundant Channels

The above CF and QF optimization in ISSD are at essence additive selection. Considering that the form of different channels may have redundancy, subtractive redundancy removal may also be necessary. To exclude redundant channels, ISSD performs linear correlated channel removal. Specifically, it calculates the Pearson correlation coefficient between each pair of channels to obtain a correlation matrix, and applies hierarchical clustering [3] to group the channels. In each cluster, only the channel with the highest quality score is retained. Positive and negative correlations are treated equally by taking the absolute values. The clustering threshold is set to 0.8 and used throughout this paper, as it is generally believed that a correlation coefficient between 0.8 and 1 indicates a high degree of correlation.

The rationale for not remove the non-linear correlated channels (e.g., Spearman and Kendall) is that such interactive information or rules among channels are generally used in the downstream tasks. There are some methods that identify states based on the interaction rules between channels, e.g., TICC [18] treats multiple channels as a correlation network for identifying states. Therefore, simply removing non-linearly correlated channels may cause negative impacts on such methods.

3.7 Analyses and Discussions

3.7.1 Complexity Analysis. ISSD contains two main processes, which are the pair-wise nntest and the optimization algorithm. ISSD uses Ball-Tree [26, 30] for nearest neighbor query when implementing the nntest. A common estimate for the time complexity of constructing a BallTree ranges from $O(n \log n)$ to $O(n^2)$, where n is the number of samples (subseries in our context). In the best case, the construction process is similar to that of a balanced binary split, which is $O(n \log n)$; In the worst case, each split is highly unbalanced, the complexity could approach $O(n^2)$. The complexity of a single query is usually $O(\log n)$, but in the worst case, performance may degrades to that of linear search, with a complexity of $O(n)$. Hence, the nntest can be implemented within a complexity ranging from $O(n \log n)$ to $O(n^2)$. Correspondingly, the complexity of pair-wise nntest ranges from $O(m^2 n \log n)$ to $O(m^2 n^2)$, where m is the number of segments.

The complexity of QF-solution searching is $O(Kc)$, where c is the number of channels, because we just need to select the top-K channels with the highest quality score. The complexity of the CF-solution searching algorithm varies with the situation, in the worst case, the complexity is $O(Kc^2)$. When $c \gg K$, which is quite often, the complexity of QF and CF solution searching can be approximated as $O(c)$ and $O(c^2)$ (worst case), respectively.

3.7.2 Parallelization. Parallelization is another important factor that affects computation time. As is mentioned above, ISSD contains two main processes, which are the pair-wise nntest for calculating the statistic and indicator matrices and the optimization algorithm for finding the favored channel combination. The calculation of statistic and indicator matrices (including nntest) is independent for each channel and time series, thus this process can be highly parallelized. Once the matrices for all channels and time series are obtained, they are feed into the optimization algorithm, which works serially. Our complexity analysis (§ 3.7.1) and empirical evaluation (§ 4.4) show that the computation time of the optimization algorithm is trivial compared to the former process, therefore, as a whole, ISSD can also be highly parallelized.

3.7.3 Parameters. ISSD requires four parameters, but only one main parameter (K) needs to be determined by users; the other three parameters are fixed or adaptive. The desired number of channels K generally depends on the user needs and the state detection methods at hand. The other three parameters, derived from nntest, are the subseries length l, the number of neighbors r, and the number of sampled subseries n, respectively.

For the subsequence length l, it is intuitive to set it close to the average length of the patterns for effectively capturing pattern difference. Using a larger or smaller l would lead to inaccurate segment-level difference estimation due to the inability to effectively capture the pattern shapes. Therefore, following [27], the l for each channel is adaptively determined by calculating the position of the first peak in the auto-correlation function, which reflects the average period of the frequent patterns in the channel.

The number of neighbors r is not intuitive, setting a small r may result in most nearest neighbors coming from the same segment, as even segments with the same state may could have slight differences. Conversely, setting a large r may lead to numerous trivial matches between segments, thus resulting in inaccurate difference estimation. In ISSD, r is adaptively determined by following the recommendation value of $\ln(n)$ in [27, 41].

Regarding the number of sampled subseries n, since nntest is a statistic-based method, a larger n will lead to a more reliable result, while would undoubtedly increase computation cost. To take balance between the computational cost and reliability, n is set to 30 and used across this paper, experimental results also show that $n = 30$ is sufficient.

3.7.4 Selecting a Solution from the Pareto Front. Theoretically, every solution in the Pareto front is the optimal solution and cannot be further optimized. Achieving optimization on one objective inevitably sacrifices other objectives. In practice, if there are no special preferences, the most common choices are the two end points of the Pareto front, i.e., the QF solution and the CF solution. The QF solution is suitable for complete datasets, while the CF solution is suitable for incomplete datasets. In general, obtaining the QF and CF solutions simultaneously does not consume much time because the time cost of ISSD mainly comes from pair-wise nntest, while the QF and CF solutions can be quickly found. When there is available validation set, we recommend directly decide which solution to use with the downstream method performance on the validation set. If unfortunately not, ISSD has a mechanism for automatically deciding whether to use the QF or CF solution, which works by selecting the one that has greater mutual information with the ground truth state sequence. The calculation of mutual information between a sequence of continuous value and a discrete sequence is hard, we use mutual information regression [39] to estimate it. In this paper, we assume the downstream methods are unknown and use the above method for automatic selection.

3.7.5 Assumption and Limitations. Recall the definition of state given in § 1, state is characterized by high internal similarity (e.g., recurring patterns), which is also the fundamental assumption underlying state detection methods. ISSD operates under the same assumption as state detection methods. In other words, ISSD is not designed for, and will likely underperform on data that does not adhere to this assumption. Consequently, the performance of ISSD may be affected by inherent noise and outliers in the data. If the noise and outliers are 'sufficiently' significant to distort the shape characteristics, they may affect the segment-level difference estimation of nntest in ISSD, leading to a performance degradation. One possible mitigation measure is to use low-pass filters and anomaly detection methods to filter out noise or outliers. Meanwhile, from another perspective, channels with higher level of noise and outliers will tend to have lower completeness and quality, allowing ISSD to effectively exclude them and prioritize the selection of high-quality channels with less noise and fewer outliers.

3.7.6 Target Downstream Task and Future Extension. ISSD focuses on state detection as its downstream task and is not intended as a one-size-fits-all tool for all time series analysis tasks. While it might be possible to adapt ISSD for other downstream tasks, such as classification, further research is required, as the difference measure could vary greatly across different tasks. For instance, classification prioritizes overall shape similarity, whereas state detection emphasizes the similarity of local patterns. Future work may involve modifying the current similarity measurement approach (nntest) implemented in ISSD. One potential direction is to incorporate representation learning encoders [50], pre-trained in a self-supervised manner, for inter/intra-class difference measurement, thereby masking the measure diversity across various tasks and exploring the applicability of ISSD in additional downstream tasks.

3.7.7 Data Sufficiency. Determining the sufficiency of data is a common concern in feature selection. Generally, the more comprehensive the data used for selection, the more reliable the results. However, in practice, the amount of historical data may be limited, which poses a cold start challenge. Fortunately, ISSD has demonstrated optimistic results by performing effectively with limited data, achieving outcomes competitive with manually selected results in [24, 46] (see § 4). For instance, it has successfully operated on heterogeneous datasets like MoCap, even with as few as 4 to 5 instances. In the context of cold starts for online deployed methods, such as E2USD, re-selection may be necessary if unseen states are observed in newly arrived data. Empirically, periodic re-selection is recommended for dynamic scenarios. Additionally, *it would be beneficial if future state detection methods could incorporate the ability to recognize unseen states* and initiate the re-selection process once a certain amount of unseen states are accumulated.

4 Evaluation

In this section, we first introduce the evaluation setup, including baselines, datasets, and downstream methods. Then we present experimental results and analyses, as well as case studies. Finally, we present valuable observations that have rarely been reported in literatures, along with corresponding analyses and insights.

Table 1. Details on the datasets

Datasets	# states	# channels	homogeneous	# segments	state duration (k)	length (k)	# time series
SynSeg	54	18	✗	10	2.0~4.0	20.0	6
MoCap [1]	26	62	✗	7~12	0.19~2.04	4.58~10.62	9
ActRecTut [6]	6	24	✓	36~39	0.06~5.10	31.12~32.40	2
PAMAP2 [36]	13	52	✓	20~29	0.05~8.60	50.57~89.40	8
USC-HAD [51]	12	6	✓	13	0.30~6.70	12.68 26.73	14

# states is the total states appear in all time series. # channels, # segments, state duration and length are for each time series.

4.1 Setup

4.1.1 Metrics. We evaluate the effectiveness of ISSD by comparing the performance changes (gain or loss) of time series state detection methods on the selection results of ISSD and typical baselines. To be specific, we employ 4 advanced state detection methods, which we refer to as *downstream methods* hereinafter, and run these methods on the selected channels of ISSD and baselines. The detection results are compared to the ground truth under the metrics of Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Purity [37].

4.1.2 Baselines. We use 2 channel selection methods, 3 dimension reduction methods, and a strong traditional feature selection method as baselines. Details are as follows:

- PCA [38]: PCA transforms raw data into a set of linearly independent representations through linear transformation, which is commonly used for dimension reduction.
- UMAP [4, 29]: UMAP is a scalable dimension reduction method built on Riemannian geometry and algebraic topology.
- ECS&ECP [10]: ECS and ECP work by representing each class with a prototype, assuming that channels with a longer distance between class prototypes are more useful.
- LDA [17]: Linear Discriminant Analysis (LDA) is a supervised dimension reduction technique, which seeks to project the data into a lower-dimensional space while preserving as much class discriminatory information as possible.
- SFM: SelectFromModel is a strong feature selection methods from the scikit-learn library [34], which uses logistic regression to measure the feature importance and selects the most important K features based on the importance score.

Most of the parameters can be automatically determined (e.g., the SVD solver of PCA and the shrinkage parameter of LDA), for those non-automatic parameters (e.g., the shrinkage parameter for ECP/ECS), we use the recommended or default values provided in their original papers or evaluation codes.

4.1.3 Datasets. We use 4 commonly used real-world datasets and a synthetic dataset for evaluation. The real-world datasets are MoCap [1], USC-HAD [51], ActRecTut [6], and PAMAP2 [36]. Details are enclosed in the original papers. We conduct 2× and 5× downsampling on ActRecTut and PAMAP2 to accelerate the experiment. The USC-HAD dataset is highly homogeneous, we use a subset collected from different subjects. Following [46], we use a subset of ActRecTut related to walking activity. The synthetic dataset, namely, SynSeg, is generated by a tool called TSAGen [47] in a similar manner as in [46], wherein each time series contains random states. During the generation process, we also generate channels with unclear states, noise channels, and irrelevant channels. Note that there are no complete channels within each time series in SynSeg, but there is complete channel set composed of incomplete channels, which poses higher challenges for channel selection. The statistics about these datasets are summarized in Table 1.

It is worth noting that there are some popular univariate time series segmentation datasets, such as UCR-SEG and TSSB [16, 40]. Despite covering a wide range of time series from different domains, they only provide a single channel for each instance. Such limitation makes these datasets not suitable for evaluating the indicator selection problem. This does not imply that these domains do not require indicator selection, on the contrary, it underscores the universal need for indicator selection. There are indeed some domains can achieve accurate state detection through a signle, well-chosen indicator, but it is important to recognize that finding such an indicator requires a systematic approach to channel selection.

4.1.4 Downstream Methods. We use 4 advanced time series state detection methods to evaluate the effectiveness of ISSD and baselines. When selecting downstream methods, we try to cover methods with different underlying principles to make them more representative.

- Time2State [46]: A state-of-the-art hybrid time series state detection methods based on representation learning and Non-parametric Bayes methods.
- TICC [18]: A Toeplitz inverse covariance-based time series segmentation method that treats each cluster as a dependency network.
- E2USD [24]: A state-of-the-art method that exploits a Fast Fourier Transform-based Time Series Compressor and a Decomposed Dual-view Embedding to learn high-quality embeddings for time series state detection.
- AutoPlait [28]: A fully automatic time series state detection method based on Hidden Markov Models (HMMs) and Minimum Description Length (MDL), which requires no parameter tuning.

The hyperparameters of downstream methods are not optimized and set with the recommended values in the original papers, because our objective is to measure the relative gain or loss in performance, instead of benchmarking downstream methods. On one hand, recent state detection works has increasingly emphasized the importance of parameter free or working adaptively [28, 46], thus their parameters are usually adaptive (or with recommended values) and have a relative small impact on the performance. On the other hand, from the perspective of reality and convention, when evaluating upstream methods, the non-adaptive downstream paremeters are usually fixed at recommended or default values to avoid overly large parameter search space, e.g., the evaluation of time series representation learning [14, 50] and indicator selection [10] tasks.

Also, TICC may not converge and refuse to output result on the selected channels of some selection methods, this phenomenon is only observed on poor selection results and raw data, indicating that the quality of the selected channels are poor, wherein the corresponding ARI, NMI, and Purity will be recorded as 0. Furthermore, the results of TICC on the raw data of some datasets is not accessible because of the low scalability against channel number. TICC takes more than one hour to segment a raw time series on MoCap dataset, and even if the final result can be obtained, applying TICC on raw data in real-world is not realistic, thus its performance on the raw data of all datasets are recorded as 0.

4.1.5 Environment. All experiments are conducted on a virtual machine equipped with an AMD EPYC 9554 CPU @ 3.1 GHz and 400GB RAM, running Ubuntu 20.04 LTS with access to 4 A800 GPUs.

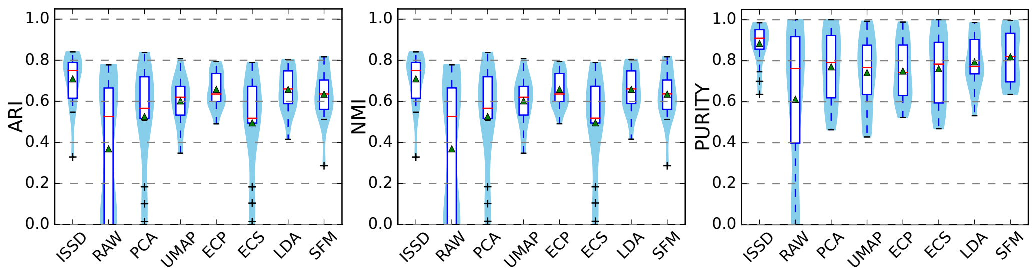

Fig. 4. The performance statistics and distribution of downstream methods on all datasets when using different selection/reduction methods. ▲ represents the mean value, 🟦 area is the probability density.

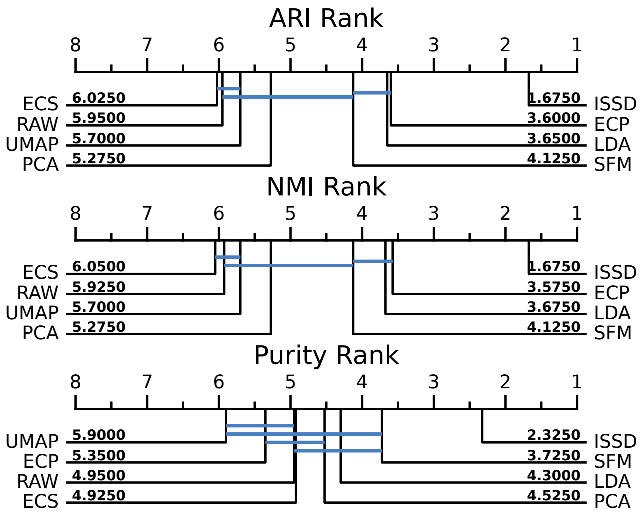

Fig. 5. Critical difference diagrams of ISSD and baselines. Methods that are not connected by a blue line are significantly different in their ranks with a confidence level of 95%.

4.2 Overall Performance

During the experiment, each dataset is divided into two parts (50%), channel selection is conducted on one part and downstream methods are run on the other part. We rotate the two parts to obtain the performance of downstream methods on all instances of the dataset. For PCA and UMAP, they are directly adopted on every instance for better performance. LDA is a supervised method and can learn rules, therefore adhering to the settings of selection methods. Since some selection/reduction methods and downstream methods are not deterministic, the experiment is independently conducted 5 times to get the averaged performance.

The number of selected channels is set to 4. On one hand, our manual and visual inspection on these datasets suggests that using 4 channels is sufficient to cover the state changes in these datasets; On the other hand, this setting is also based on practical considerations, using more channels can be redundant and may lead to the inclusion of low-quality channels, as the number of high-quality channels is limited. It is also necessary to make similar trade-offs when selecting channels for real-world deployment, this paper provides corresponding visualization tool to facilitate the inspection of selection results (see § 4.3.1).

4.2.1 Performance Distribution. Figure 4 presents the box plots with superimposed probability densities (a.k.a., violin plot). The plots not only show the statistics but also illustrate the performance distribution of all downstream methods across all datasets when using different selection/reduction methods. The box plot shows the key statistics in performance, including maximum, minimum, upper and lower quartiles, median, and mean, all of which are more favorable at higher values; The superimposed probability density provides a direct view of the performance distribution of downstream methods, with a thicker upper-end suggesting a superior performance distribution.

It can be seen that when using ISSD, the downstream method performance exhibits higher maxima, upper/lower quartiles, medians, and mean values on all metrics. Furthermore, the probability densities of ISSD are greater at higher performance levels on all metrics, indicating that ISSD overall enhances the performance distribution of the downstream methods compared to baselines. In contrast, although some methods can achieve nearly the same maximum as ISSD (e.g., PCA on ARI), their medians, quartiles, and mean values are lower, and the overall distribution is not satisfactory. It is worth noting that when using raw data, the results of TICC are recorded as 0 due to its running time exceeding the limit, while AutoPlait directly rejects to work by outputting only one segment on most datasets (except for USC-HAD, which has only 6 channels), indicating that using raw data is an not an appropriate choice.

4.2.2 Performance Rank. Figure 5 presents the Critical Difference (CD) diagrams of ISSD and baselines w.r.t. ARI, NMI, and Purity, wherein all methods are ranked by their average ranks on all downstream method and dataset combinations (20 cases in total). Methods that are not connected by a blue line are significantly different in their ranks under the Nemenyi two-tailed test [9] with a confidence level of 95%. The results validate that ISSD significantly outperforms all the baselines in average rank.

4.2.3 Average Performance. Figure 6 shows the averaged performance of all downstream methods on the same dataset with different selection or reduction methods. As shown, ISSD brings better performance for downstream methods w.r.t. ARI, NMI, and Purity. On one hand, ISSD achieves competitive results on homogeneous datasets (ActRecTut, USC-HAD, and PAMAP2); On the other hand, the advantage of ISSD becomes more pronounced on heterogeneous datasets (MoCap and SynSeg), achieving the best results across all metrics. The above results intuitively demonstrate the superiority of ISSD.

There is a noteworthy phenomenon, ISSD achieves a comprehensive advantage on the Purity metric (tying with the best baseline on USC-HAD). Purity not only measures the consistency between clustering results and class labels, but also reflects the "purity" of data points within each cluster, i.e., the extent to which data points within each cluster primarily come from a single actual class. In the context of time series state detection, Purity intuitively measures how many points in each segment are incorrectly assigned labels from other states, whereas ARI and NMI are less sensitive to this aspect due to their focus on the overall matching degree. This advantage could be important for some applications, such as constructing state transition graphs [48] or event propagation chains [43] based on state detection results. If Purity is low, it can lead to numerous false and trivial state transitions within a segment, necessitating additional effort to filter out these false transitions.

Another noteworthy phenomenon is that the performance of ISSD on PAMAP2 is not as good as on other datasets. By digging into the data, we find that one possible reason is that the shape characteristics are relatively poor in PAMAP2, and the nntest may be affected by this factor, resulting in performance degradation. It is quite often that some undersampled data can lead to distortion in the shape characteristics, and one possible solution is to increase the sampling frequency to avoid distortion.

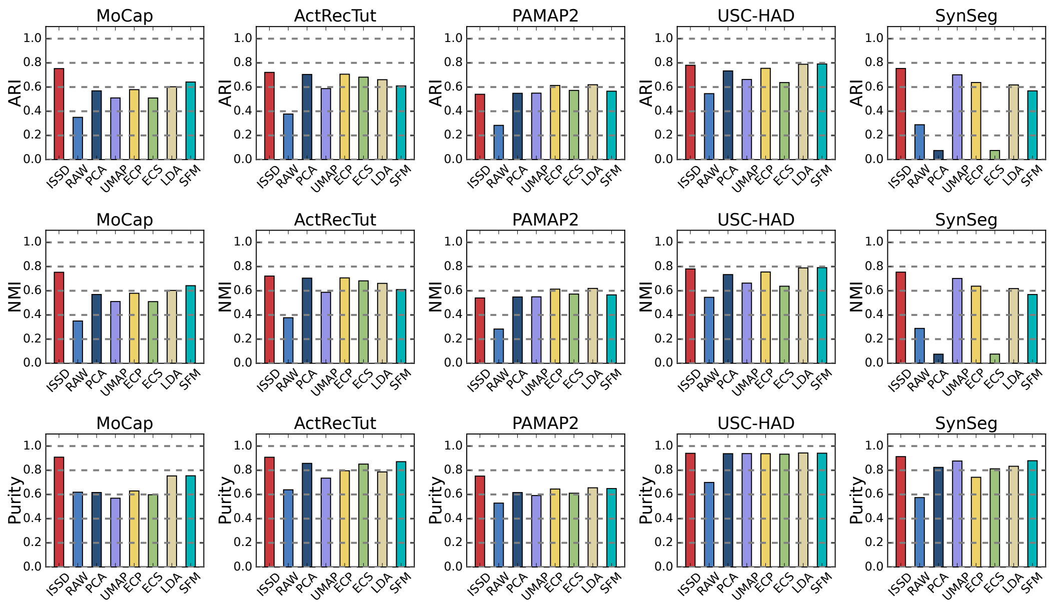

Fig. 6. The averaged performance of 4 downstream methods on the selection or reduction result of ISSD and Baselines.

4.3 A Case Study on MoCap

In this part, we 1) explore the cause of downstream performance differences on the selection results of ISSD and baselines through result visualization; 2) demonstrate the selection process of ISSD, especially the order in which each channel is added to the final selection, to show how the completeness is improved.

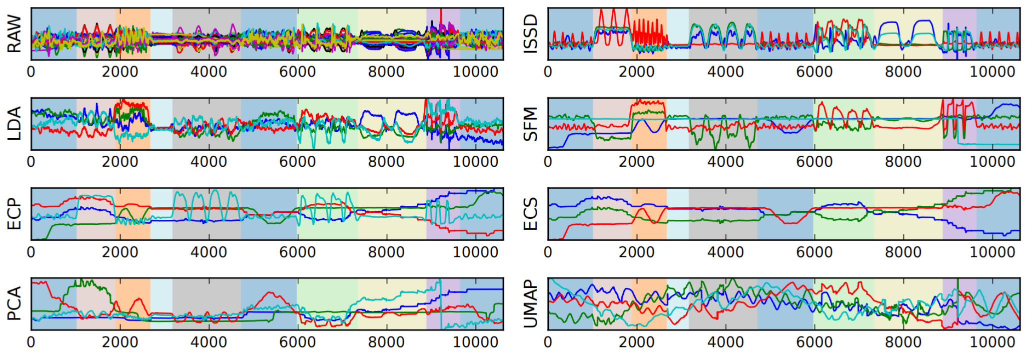

Fig. 7. Visualization of channel selection/reduction results on MoCap. Background colors label the states. The selected channels by ISSD clearly indicates the state changes.

4.3.1 Result Visualization. Figure 7 shows the selection/reduction results of all the methods on a typical time series in the MoCap dataset, wherein background colors label the states. As shown, the selected channels of ISSD clearly indicates the state changes, and it is easy to distinguish different states through visual inspection. In contrast, the results of LAD, SFM, and ECP are not as intuitive as ISSD, which contain some low-quality channels, showing ambiguous boundary and low discriminability between different states. For example, it is hard to distinguish whether the first and last segments belong to the same state through the selection results of LDA, SFM, and ECP. As for PCA, UMAP, and ECS, the results are even worse, the selected channels can hardly reflect the state changes.

As explained in § 2, dimension reduction methods largely do not distinguish and exclude destructive and non-informative channels, but faithfully integrate all information into a low-dimensional representation. As a result they are easy to suffer from the destructive/noise information in different channels, resulting in a large amount of noise in the low-dimensional representation. Traditional feature selector treats each channel as a feature and each time point as a sample, thus leading to insufficient context modeling for each time point. Although ECP/ECS does not have such problems, they are designed for time series classification task, which focus more on the overall difference/similarity of two segments. However, the states in time series task is usually identified by recurring patterns. Another problem is that there are quite a few states that only appear once, and the length of each segment is usually much longer than the classification data, which would affect the construction of class prototypes and affect their performance. Unlike the settings in time series classification tasks, where each time series has the same length, the length of each segment in state detection task is random, these settings may also cause adaptation difficulties.

We have also rendered the visualization of the selection results on all datasets other than MoCap. The script for automatic visualization of selection results are published in the project page of this paper, which we believe is beneficial to facilitate decision making through visual inspection.

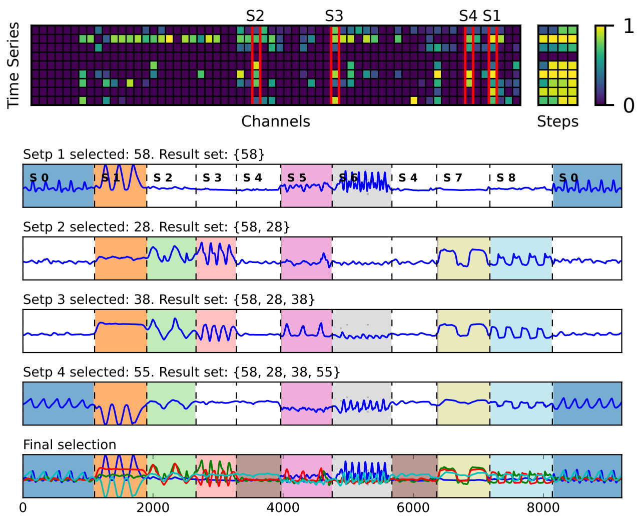

Fig. 8. Completeness improvement during optimization and channel selection order.

4.3.2 *Completeness Improvement.* Figure 8 shows the detailed CF selection process of ISSD on the entire MoCap dataset. The upper heatmap shows the channel completeness (estimated by ISSD) for each time series, wherein the rows correspond to time series, the columns correspond to channels. The small heatmap on the right shows the changes in *channel set completeness* during the process;

each column, from left to right, corresponds to the completeness of result set at each selection step, with a total of 4 channels are selected. As shown, the small heatmap becomes increasingly bright from left to right, indicating an overall increase in completeness with the addition of new channels.

The lower figure shows 4 channels of a time series, arranged in the order of selection. The text above each channel annotates the result set in each step. Uncolored segments indicate the presence of segments (from other states) that are indistinguishable from the segment in the channel. As shown, although none of these channels are individually complete, the addition of new channels continuously increases the number of distinguishable states, ultimately making the states distinguishable through the combination of channel.

4.4 Computation Time and Efficiency

4.4.1 Computation Time. Figure 10 shows the computation time of ISSD and baselines, with the y-axis on a log scale. Although the computation time of ISSD is longer than all baselines except for UMAP, it is important to note that the absolute time consumption of ISSD is not excessive. For instance, in the experimental environment (see § 4.1.5) of this paper, ISSD took the longest time to process the entire PAMAP2 dataset among all datasets, with approximately 50 seconds. In contrast, the state-of-the-art downstream method, E2USD, has an average processing time of around 6.92 seconds per time series on the PAMAP2 dataset, while TICC (using 20 CPU cores) took even longer, with an average of 37.8 seconds per time series.

It is also worth noting that the benefits of channel selection are long-lasting, because once the selection is done, there is no need for re-selection untill unseen states are observed. In conclusion, indicator selection does not incur much overhead for the entire state detection pipeline, and the selection results may be partially transformed into prior knowledge for reuse in the same or similar tasks.

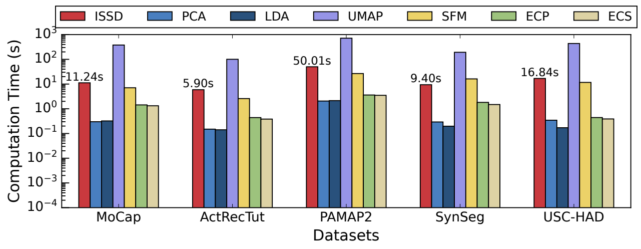

Fig. 9. Computation time of ISSD and baselines.

4.4.2 Time Breakdown. Figure 10 (left) shows the time breakdown of ISSD on all datasets, the time consumption at different stages (e.g., QF searching, CF searching) are separately accounted to illustrate their respective proportions. Note that the y-axis is log-scaled. The number of processes (CPU cores) used in the experiment is set to 20. As shown, nntest accounts for the majority of the computation time, with durations ranging from 4.1 ∼ 37.97 seconds across different datasets. The automatic decision-making process (CF & QF Evaluation) of QF or CF solution occupies the second largest time cost, with a time consumption of 0.51∼7.82 seconds across different datasets. In contrast, the time consumed by the QF and CF searching algorithms is trivial, with the maximum time spent on any dataset not exceeding 0.02 seconds. This result validates that the proposed QF and CF searching algorithms are efficient and aligns with the analysis in § 3.7.2, i.e., the non-parallelizable portion in ISSD is trivial.

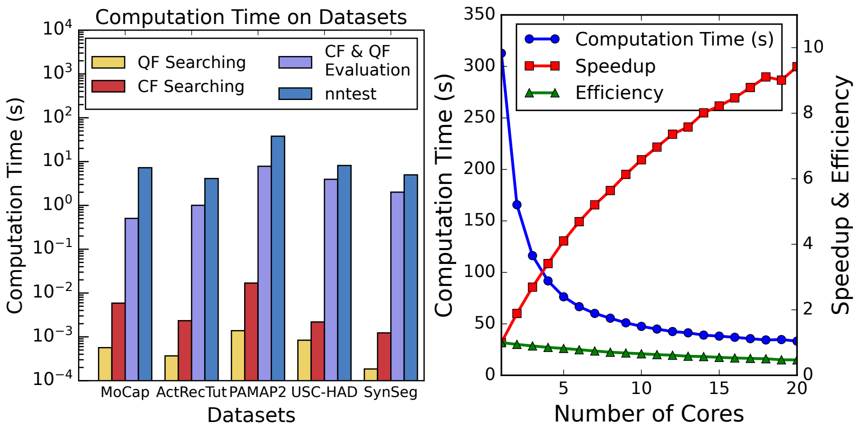

Fig. 10. Time breakdown of ISSD on all datasets (left) and the parallelization efficiency of ISSD (right).

4.4.3 Parallelization Efficiency. Figure 10 (right) shows the computation time, speedup, and efficiency of ISSD as functions of the number of processes. The experiment was conducted on a time series with 1000 channels and 10 segments, running 5 times to obtain averaged performance. The computation time in the figure corresponds to the left y-axis, the speedup and efficiency correspond to the right y-axis. Efficiency is the ratio of speedup to the number of processors, indicating the utilization efficiency of resources. ISSD is implemented in Python, with each process uses one CPU core, thus the number of processes equals the number of CPU cores. As shown, as the number of CPU cores increases, there is a fast decrease in computation time, accompanied by fast increasing speedup, demonstrating a high parallelization efficiency. Meanwhile, it can be observed that the efficiency decreases as the number of cores increases, which is a common observation because higher process counts may introduce competition among processes for shared resources (e.g., memory, I/O bandwidth, etc.), leading to resource contention and increased waiting times, thereby affecting parallel efficiency. When using 20 cores, the efficiency is about 47.1%. Therefore, we do not recommend using more cores as this will result in lower resource utilization.

4.5 ISSD-based Dataset Analysis

Having validated the effectiveness of ISSD, we further conduct a retrospective analysis on the completeness and quality of datasets based on ISSD to provide some insights into the characteristics of the datasets to a certain extent. We note that such a 'profiling' process on the datasets show that ISSD could be an auxiliary tool for understanding the datasets.

Figure 11 shows the results for completeness and quality, respectively. The completeness and quality is first computed for every channel within every time series, and then averaged across the channels of each time series. The red star (★) is the completeness of combining all channels together (also averaged). For easy comparison, the completeness values in the distribution is scaled to $0 \sim 1$ according to the theoritical maximum of each instance. As shown, compared to the distribution of quality, the completeness distribution is of a relatively wider range (especially, PAMAP2 and USC-HAD), indicating that there could be large completeness difference between channels with small quality differences. Meanwhile, from the perspective of ISSD, the channels in these datasets are not complete, but the completeness of the channel set can be improved through their combination.

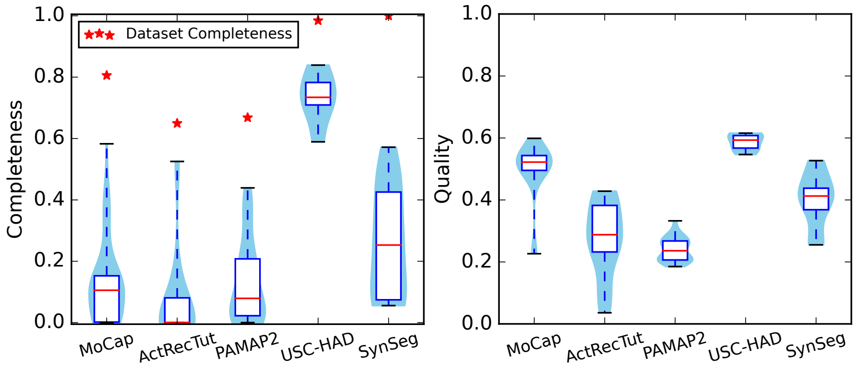

Fig. 11. Completeness and Quality estimated by ISSD

4.6 Effect of the Correlation Threshold

Figure 12 (right) shows the inter-channel correlation distributions of all datasets. Figure 12 (left) shows the downstream performance when using ISSD with different correlation thresholds for redundancy removal, wherein the curves are the averaged performance of 4 downstreawm methods and the overall results are averaged over 5 independent executions. Adjcent points without channel changes are connected by dashed lines, otherwise, solid lines. As shown, the selection results of ISSD do not change frequently when using different thresholds, and the downstream performance remains relatively stable at higher thresholds. This indicates that ISSD is relatively insensitive to the correlation threshold on these datasets, and the impact of the threshold on downstream performance may be much smaller than expected.

The correlation threshold indirectly influences the channel selection of ISSD by affecting the clustering results. However, as shown in Figure 12 (right), the correlation distribution varies greatly across datasets, and high inter-channel correlations are sparse. Due to the completeness and quality difference, ISSD may not ultimately select these channels, and the redundancy removal step may not have any impact on the selection results of ISSD. As a result, the effectiveness of this correlation-based redundancy removal step is influenced by the characteristics of the dataset itself. For instance, in our results, only the MoCap dataset is significantly affected at lower thresholds (< 0.5), while the effect is less pronounced in other datasets.

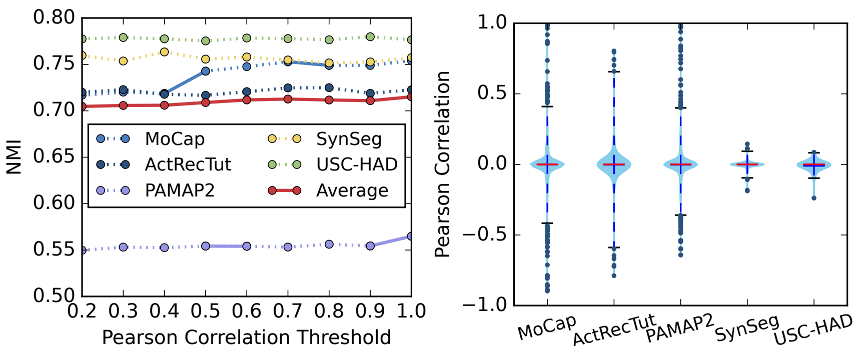

Fig. 12. The effect of clustering threshold (left) and correlation distributions of datasets (right)

4.7 Observations and Analyses

In this section, we introduce two notable experimental observations, along with the possible explanations and the insights brought by these observations.

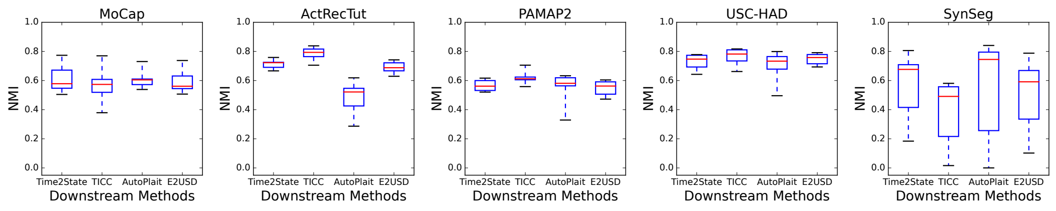

Fig. 13. Selection resilience of downstream methods. The flatter the box, the more robust the method is to selection results.

Observation 1: AutoPlait performs poorly (NMI$\simeq$ 0.14) on ActRecTut using 10 channels in previous works [24, 46], while achieving better results in this paper with just 4 channels (NMI ranging from 0.29 $\sim$ 0.62 with different selection methods). In contrast, the performance changes of Time2State and E2USD are relatively trivial. This reveals two key points: 1) *Previous studies may have underestimated the impact of channel selection, the full potential of some state detection methods remains unexplored.* 2) *State detection methods vary in their generalization capacity to indicator selection results, i.e., selection resilience.* Figure 13 shows the NMI box plots of downstream methods using different selection methods, wherein the results on raw data are excluded. As shown, AutoPlait achieves higher best NMI on USC-HAD and PAMAP2 than Time2State and E2usd, but shows wider performance variance, indicating a weaker selection resilience. This resilience difference can be observed across all datasets. TICC performs well on all dataset, but has weaker resilience on MoCap, while Time2State and E2USD demonstrate strong selection resilience on all datasets. This observation suggests that special attention should be paid to the selection quality when using methods with weaker resilience, and the results obtained on carefully selected channels may not generalize well to wild data with lower quality. We argue that carefully working on the low-quality channels is also an important path to consider in practice, because high quality channel with informative states may not be always available.

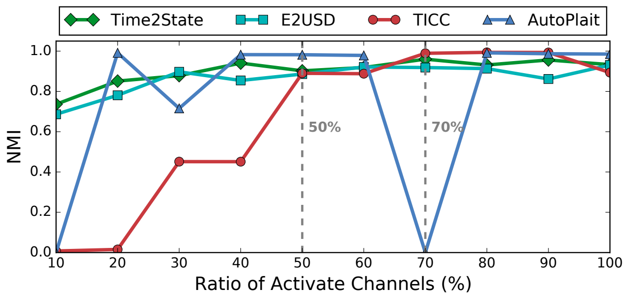

Fig. 14. Channel sensitivity of downstream methods.

Observation 2: The downstream state detection methods exhibit different "channel sensitivity" (i.e., sensitivity to state changes in single channels). As shown in Figure 13, TICC performs well across most datasets but struggles with SynSeg. As mentioned, there is no complete channel in SynSeg, and the state changes typically affect only one channel (the "active channel") in its complete channel set. This indicates that TICC has a weak sensitivity against state changes in single channel. To explore further, we generate time series with varying proportions of active channels (10% ∼ 100%) to test downstream methods (conducted 5 times). As shown in Figure 14, Time2State and E2USD exhibit high channel sentivity, work well with fewer active channels, while TICC fails when the ratio is below 50%. AutoPlait behaves irregularly, sometimes failing even with high active channel ratios (70%). Considering that TICC is clustering-based, adjusting the clustering thresholds may alleviate this issue but would undoubtedly increase manual effort. Obviously, higher channel sensitivity is more desirable and crucial for detecting subtle state changes in the real world, yet often overlooked in prior works. We have released the data generation scripts and call for the evaluation of channel sensitivity in future works.

5 Conclusion

Indicator/channel selection plays a key role in facilitating efficient and effective time series state detection, but has been rarely studied. This paper argues that channel selection should be made an upstream task before state detection and fills this gap by proposing a novel channel selection method for state detection. It invents the concept of channel (set) completeness based on segment-level difference estimation, formulates the channel selection problem into a multi-objective optimization problem, and proposes an approximation algorithm for searching the specific end point in the Pareto front. Experimental results not only demonstrate the effectiveness of the proposed ISSD method, but also bring insightful observations and implications for further research in state analysis of time series.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62472434, U22B2005, 62006243), and the Science and Technology Innovation Program of Hunan Province (2022RC3061, 2023RC3027).

References

[1] MoCap. http://mocap.cs.cmu.edu.

[2] Mohammad Reza Abbasifard, Bijan Ghahremani, and Hassan Naderi. 2014. A survey on nearest neighbor search methods. *International Journal of Computer Applications* 95, 25 (2014).

[3] Maria-Florina Balcan, Yingyu Liang, and Pramod Gupta. 2014. Robust hierarchical clustering. *J. Mach. Learn. Res.* 15, 1 (jan 2014), 3831–3871.

[4] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. 2019. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* 37, 1 (2019), 38–44.

[5] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (sep 1975), 509–517. https://doi.org/10.1145/361002.361007

[6] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.* 46, 3, Article 33 (jan 2014), 33 pages. https://doi.org/10.1145/2499621

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017

[8] Shohreh Deldari, Daniel V. Smith, Amin Sadri, and Flora Salim. 2020. ESPRESSO: Entropy and ShaPe AwaRe TimE-Series SegmentatiOn for Processing Heterogeneous Sensor Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 77 (2020), 24 pages. https://doi.org/10.1145/3411832

[9] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7, 1 (2006), 1–30. http://jmlr.org/papers/v7/demsar06a.html

[10] Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim. 2023. Scalable classifier-agnostic channel selection for multivariate time series classification. *Data Mining and Knowledge Discovery* 37, 2 (2023), 1010–1054.

[11] Bhaskar Dhariyal, Thach Le Nguyen, and Georgiana Ifrim. 2021. Fast channel selection for scalable multivariate time series classification. In *Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021, Bilbao, Spain, September 13, 2021, Revised Selected Papers 6*. Springer, 36–54.

[12] Arik Ermshaus, Patrick Schäfer, and Ulf Leser. 2024. Raising the ClaSS of Streaming Time Series Segmentation. *Proc. VLDB Endow.* 17, 8 (May 2024), 1953–1966. https://doi.org/10.14778/3659437.3659450

[13] Uriel Feige. 1998. A threshold of ln n for approximating set cover. *J. ACM* 45, 4 (jul 1998), 634–652. https://doi.org/10.1145/285055.285059

[14] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised Scalable Representation Learning for Multivariate Time Series. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., New York, NY.

[15] Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA.

[16] Shaghayegh Gharghabi, Chin-Chia Michael Yeh, Yifei Ding, Wei Ding, Paul Hibbing, Samuel LaMunion, Andrew Kaplan, Scott E Crouter, and Eamonn Keogh. 2019. Domain agnostic online semantic segmentation for multi-dimensional time series. *Data mining and knowledge discovery* 33, 1 (2019), 96–130.

[17] Benyamin Ghojogh and Mark Crowley. 2019. Linear and Quadratic Discriminant Analysis: Tutorial. arXiv:1906.02590 [stat.ML]

[18] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. 215–223.

[19] Shuchu Han and Alexandru Niculescu-Mizil. 2020. Supervised Feature Subset Selection and Feature Ranking for Multivariate Time Series without Feature Extraction. arXiv:2005.00259 [cs.LG]

[20] Shuchu Han and Alexandru Niculescu-Mizil. 2020. Supervised feature subset selection and feature ranking for multivariate time series without feature extraction. *arXiv preprint arXiv:2005.00259* (2020).

[21] Bing Hu, Yanping Chen, and Eamonn Keogh. 2016. Classification of streaming time series under more realistic assumptions. *Data mining and knowledge discovery* 30, 2 (2016), 403–437.

[22] Qi Huang, Thomas Bäck, and Niki van Stein. 2023. Application of Functional Kernel Hypothesis Testing for Channel Selection in Time Series Classification. In *2023 IEEE Conference on Artificial Intelligence (CAI)*. 245–247.

[23] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. 2014. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*. 372–378. https://doi.org/10.1109/SAI.2014.6918213

[24] Zhichen Lai, Huan Li, Dalin Zhang, Yan Zhao, Weizhu Qian, and Christian S. Jensen. 2024. E2Usd: Efficient-yet-effective Unsupervised State Detection for Multivariate Time Series. In *Proceedings of the ACM on Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 3010–3021. https://doi.org/10.1145/3589334.3645593

[25] Jiyuan Liu, Xinwang Liu, Jian Xiong, Qing Liao, Sihang Zhou, Siwei Wang, and Yuexiang Yang. 2022. Optimal Neighborhood Multiple Kernel Clustering With Adaptive Local Kernels. *IEEE Transactions on Knowledge and Data Engineering* 34, 6 (2022), 2872–2885.

[26] Ting Liu, Andrew W Moore, Alexander Gray, and Claire Cardie. 2006. New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research (JMLR)* 7, 6 (2006).

[27] Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. 2014. Correlating events with time series for incident diagnosis. In *Proc. of the 20th ACM SIGKDD*.

[28] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2014. AutoPlait: Automatic Mining of Co-Evolving Time Sequences. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD '14)*. 193–204.

[29] Leland McInnes, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426 [stat.ML]

[30] Andrew Moore, Alexander Gray, et al. 2003. New algorithms for efficient high dimensional non-parametric classification. *Advances in Neural Information Processing Systems (NeurIPS)* 16 (2003).

[31] Marius Muja and David G Lowe. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)* 2, 331-340 (2009), 2.

[32] Masatoshi Nagano, Tomoaki Nakamura, Takayuki Nagai, Daichi Mochihashi, Ichiro Kobayashi, and Wataru Takano. 2019. HVGH: Unsupervised Segmentation for High-Dimensional Time Series Using Deep Neural Compression and Statistical Generative Model. *Frontiers in Robotics and AI* 6 (2019).

[33] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14 (1978), 265–294.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[35] Mathias Perslev, Michael Jensen, Sune Darkner, Poul Jø rgen Jennum, and Christian Igel. 2019. U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., New York, NY.

[36] Attila Reiss and Didier Stricker. 2012. Introducing a New Benchmarked Dataset for Activity Monitoring. In *2012 16th International Symposium on Wearable Computers*. 108–109. https://doi.org/10.1109/ISWC.2012.13

[37] Eréndira Rendón, Itzel M Abundez, Citlalih Gutierrez, Sergio Díaz Zagal, Alejandra Arizmendi, Elvia M Quiroz, and H Elsa Arzate. 2011. A comparison of internal and external cluster validation indexes. In *Proceedings of the 2011 American Conference, San Francisco, CA, USA*, Vol. 29. 1–10.

[38] Markus Ringnér. 2008. What is principal component analysis? *Nature biotechnology* 26, 3 (2008), 303–304.

[39] Brian C Ross. 2014. Mutual information between discrete and continuous data sets. *PloS one* 9, 2 (2014), e87357.

[40] Patrick Schäfer, Arik Ermshaus, and Ulf Leser. 2021. ClaSP - Time Series Segmentation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 1578–1587.

[41] Mark F Schilling. 1986. Multivariate two-sample tests based on nearest neighbors. *J. Amer. Statist. Assoc.* 81, 395 (1986), 799–806.

[42] A H Shahana and V Preeja. 2016. Survey on feature subset selection for high dimensional data. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. 1–4. https://doi.org/10.1109/ICCPCT.2016.7530147

[43] Ya Su, Youjian Zhao, Wentao Xia, Rong Liu, Jiahao Bu, Jing Zhu, Yuanpu Cao, Haibin Li, Chenhao Niu, Yiyin Zhang, Zhaogang Wang, and Dan Pei. 2019. CoFlux: robustly correlating KPIs by fluctuations for service troubleshooting. In *Proceedings of the International Symposium on Quality of Service*.

[44] Nitin Sukhija, Elizabeth Bautista, Owen James, Daniel Gens, Siqi Deng, Yulok Lam, Tony Quan, and Basil Lalli. 2020. Event management and monitoring framework for HPC environments using ServiceNow and Prometheus. In *Proceedings of the 12th international conference on management of digital ecosystems*. 149–156.

[45] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[46] Chengyu Wang, Kui Wu, Tongqing Zhou, and Zhiping Cai. 2023. Time2State: An Unsupervised Framework for Inferring the Latent States in Time Series Data. *Proc. ACM Manag. Data (SIGMOD'23)* 1, 1, Article 17 (may 2023), 18 pages.

[47] Chengyu Wang, Kui Wu, Tongqing Zhou, Guang Yu, and Zhiping Cai. 2022. TSAGen: Synthetic Time Series Generation for KPI Anomaly Detection. *IEEE Transactions on Network and Service Management* 19, 1 (2022). https://doi.org/10.1109/TNSM.2021.3098784

[48] Peng Wang, Haixun Wang, and Wei Wang. 2011. Finding Semantics in Time Series. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. New York, NY, USA, 385–396.

[49] Siwei Wang, Xinwang Liu, Xinzhong Zhu, Pei Zhang, Yi Zhang, Feng Gao, and En Zhu. 2022. Fast Parameter-Free Multi-View Subspace Clustering With Consensus Anchor Guidance. *IEEE Transactions on Image Processing* 31 (2022), 556–568.

[50] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8980–8987.

[51] Mi Zhang and Alexander A. Sawchuk. 2012. USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (Pittsburgh, Pennsylvania) *(UbiComp '12)*. Association for Computing Machinery, New York, NY, USA, 1036–1043. https://doi.org/10.1145/2370216.2370438

[52] Qingfu Zhang and Hui Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731. https://doi.org/10.1109/TEVC.2007.892759

Received July 2024; revised September 2024; accepted November 2024