



APPENDIX

Multivariate Time Series Retrieval with Symbolic Aggregate Approximation, Regular Expression, and Query Expansion

Y. Yu^{1,2}, T. Becker^{1,2} and M. Behrisch¹

¹Utrecht University, Netherlands

²IAV GmbH Ingenieurgesellschaft Auto und Verkehr, Germany

Abstract

We present *SAXRegEx*, a method for pattern search in multivariate time series in the presence of various distortions, such as duration variation, warping, and time delay between signals. For example, in the automotive industry, calibration engineers spontaneously search for event-induced patterns in fresh measurements under time pressure. Current methods do not sufficiently address duration (horizontal along the time axis) scaling and inter-track time delay. One reason is that it can be overwhelmingly complex to consider scaling and warping jointly and analyze temporal dynamics and attribute interrelation simultaneously. *SAXRegEx* meets this challenge with a novel symbolic representation modeling adapted to handle time series with multiple tracks. We employ methods from text retrieval, i.e., regular expression matching, to perform a pattern retrieval and develop a novel query expansion algorithm to deal flexibly with pattern distortions. Experiments show the effectiveness of our approach, especially in the presence of such distortions, and its efficiency surpassing the state-of-the-art methods. While we design the method primarily for automotive data, it is well transferable to other domains.

CCS Concepts

• **Mathematics of computing** → Time series analysis; • **Information systems** → Query representation;

Contents

[A Query-Aware Symbolic Aggregate approXimation](#)

[B Datasets](#)

[C Experiment Setup](#)

[Hardware and Operating System](#)

[Method Implementation](#)

[Evaluation Setting](#)

[D Visual Inspection of the Datasets and Predictions](#)

[E Complete Accuracy Benchmark](#)

[F Complete Speed Benchmark](#)

[References](#)

Appendix A: Query-Aware Symbolic Aggregate approxImation

Symbolic Aggregate approxImation (SAX) is introduced in [LKLC03] and extensively explained in [LKWL07].

Motivation of the chosen methods: We have chosen a symbolic encoding for time series to enable text retrieval techniques, in this case regex. As a digression of this section, we choose regex, because it naturally handles length-invariant search, among many possible tricks, that distortion-invariant pattern search in time series may benefit from. Furthermore, it is potentially very fast, because it stops similarity matching immediately when the pattern partially mismatches. On the other hand, regex is a well established technique for simple and robust implementation. Finally, SAXRegEx may benefit from its future independent development. Back to SAX, we have chosen this technique, not only because it is one of the state-of-the-art symbolic representation for time series, but also because of its two properties. On one hand, the symbols in the alphabet used by SAX have an order, allowing a tolerance band for the subsequent regex search. On the other hand, SAX's numerocity reduction property inspired us to the query expansion for horizontally invariant search.

SAX's pipeline: The original SAX contains two major steps. In the first step, it performs Piecewise Aggregate Approximation (PAA), merging temporal consecutive time steps into one by calculating their average. In the second step, it quantifies the values with quantiles as breakpoints and map the values within a range bounded by the breakpoints to a symbol.

Uncommon practice in our implementation: While it's common practice to conduct SAX on the whole time series dataset, we fit SAX (bin size horizontally along the time axis and breakpoints along the value axis) with the query, and then conduct SAX on the time series dataset with the fitted parameters. We make such an alternation, because we have no clue about the appropriate bin size and breakpoints. Especially when the patterns in the time series are quite small either in terms of length or value range. This is exactly the case in our APST dataset. In fact, similar practice is proposed for Locality-Sensitive Hashing (LSH) [HFZ*15, CLL*19], where the hash tables are based on the query, not the dataset. They call the new version query-aware LSH and the previous one query-oblivious LSH. Though it requires repeating hashing every time the query changes, this technique significantly improves accuracy.

Bin size for aggregation along the time axis: For aggregation horizontally along the time axis, we do not set the bin size directly. Instead, we set the length of the SAX-encoded query to 20, which is enough for all our datasets. The bin size is inferred from the original query length and the desired query length. This arrangement aims to have some control over the regex length, because we find empirically that our regex does not scale well with its length (this drawback also somehow limits the number of tracks). A scalability test is desired as future work.

Breakpoints for discretization: The original SAX assumes normal distribution of the values in the tracks. Based on this assumption, it calculates quantiles to quantify the values and categorize them into symbols, so that each symbol has approximately the same number of values / time steps / data points. This helps zooming in on the value ranges where the values concentrate. The values in the tracks in our datasets are clearly not normally distributed. Therefore, we calculate the quantiles based on the true value distribution in each track. Again, we do not calculate the value distribution based on the whole time series dataset but on the query. Because we want sufficient number of symbols to distinguish values in the desired patterns.

Efficiency concern regarding our uncommon practice: The query-aware setting may raise concerns on its efficiency. In the speed benchmark, the execution time is end-to-end, including SAX's execution time. Actually, SAX's execution time is negligible compared with regex's search time. Nonetheless, we need to separate preprocessing time and search time in future work.

Appendix B: Datasets

Dataset	Files	Tracks	Length (time stamps)	Sampling rate (Hz)	HDF5 volume (MB)	Pattern length range (s)	Pattern length ratio	Domain
APST	6	1	1000 - 40000	10	0.04 - 1.6	3.17 - 12.21	3.8	automotive
Cable Cutter	1	1	100000	12	1.6	1.25 - 10.08	8.1	manufacture
Deep Valve	1	1	100000	100	1.6	0.5 - 4	8	automotive
EEG Eye State	1	4*	14980	128	1.7	0.4 - 20	50	medicine
Filling Prediction	1	1	61226	10	1	5 - 8	1.6	automotive
Variable Displacement	8	2	9119-108457	~ 2000	0.2 - 2.5	0.05 - 0.12	2.4	automotive
CAN 1	1	1	5100	10^7	0.08	5×10^{-5} - 9×10^{-5}	1.8	automotive
CAN 2	1	2	2700	10^7	0.07	$\sim 3 \times 10^{-5}$	~ 1	automotive

Table 1: Metadata of all Datasets

APST dataset: The APST dataset focuses on the relative air fuel ratio in an Otto motor from the engine control unit. Unfortunately, the dataset is proprietary and cannot be published. Moreover, the full name and detailed background are unknown to us. Engineers search for a “w”-shaped pattern in the data. The ground truth labels comes from a flag signal, also from the engine control unit. The patterns are very tiny in terms of both time span and value range.

Cable cutter dataset: The cable cutter dataset contains the simulated power consumption of a cable cutting machine. The engineers want to find time intervals corresponding to cutting a cable. The cutting process undergoes seven phases. They form a wave of four plateaus and three valleys. Because the cables length, thickness and the manual operation change, the lengths of the four plateaus and the space between them are inconstant and the height of the plateaus also varies from labels to labels. Because the data are simulated, it is easy to generate ground truth labels.

Deep valve dataset: The deep valve dataset records the current through a solenoid valve. The query captures a complete operating cycle of the valve, which again consists of four phases. The pattern starts with a peak when the valve begins to move; It follows a “J”-like jump corresponding to the period when the valve moves until the opening reaches maximum; A subsequent linear phase indicates the constant operating point with the maximum opening. The length of the operating cycle varies greatly.

EEG eye state dataset: EEG stands for electroencephalogram and measures brain waves with sensors mounted around the head. The EEG eye state dataset comes from one continuous EEG measurement with the Emotiv EEG Neuroheadset. During measurement, participants’ eye state (closed or open) was detected via a camera. The eye state at each video frame is labelled. Please refer to <https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State> for more information. The patterns are very fuzzy and horizontally extremely scaled. The typical task for this dataset is classification for each time step. As far as we know, this is the first time that this dataset is used for time series retrieval. The original data have 14 tracks plus a track for ground truth labels. We have selected four tracks manually, where the patterns are more distinguishable than in the other tracks, which significantly improve the performance compared to search directly in the 14 tracks.

Filling prediction dataset: Similar as the APST dataset, the filling prediction dataset tracks the relative air fuel ratio predicted by the engine control unit in a test auto. It is also proprietary, cannot be published and has backgrounds unknown to us. The given query features abrupt change near the boundaries of the pattern. The major part between the boundaries remains stationary. The patterns hide well in the signal and there are some confusing segments. The data is manually labelled by a domain expert.

Variable displacement dataset: The variable displacement dataset is derived from the rotational speed of the engine recorded by the engine control unit. There are two tracks in the data. They corresponds to a specific component of the rotational speed after decomposition with a dedicated method from signal processing. The query corresponds to the transition from the operation with half of the cylinders to operation with all cylinders in the engine. The patterns accompany large vibration and the engineer wants to suppress the vibration to improve comfort. The data look like sound waves fluctuating all the time with a changing amplitude. Unfortunately, the dataset is proprietary and cannot be published.

CAN 1 and CAN 2 dataset: The CAN 1 and CAN 2 datasets are synthesized CAN bus data. CAN bus data keep track of the communication of the components in an automobile. Our engineers continuously want to find certain events spontaneously. We are not allowed to publish the original data, but tries to reconstruct some cases that we are interested in. Because the two files have different characteristics and queries, we treat them as two separate datasets.

The plots of all publishable datasets can be found in [Appendix D](#).

Appendix C: Experiment Setup

Hardware and Operating System

We conduct all experiments locally on the same laptop HP EliteBook 850 G5 with

- Processor: Intel® Core™ i7-8650U CPU @ 1.90GHz 2.11GHz
- Memory: 16GB
- Storage: 1TB HDD
- Operating system: 64-bit Windows 10 Enterprise

Method Implementation

The code is proprietary and cannot be open-sourced.

- Correlation: our own implementation in python `float(numpy.dot(*normalize(np.vstack((seq_1, seq_2)) + bias)))`, where `seq_1` and `seq_2` are two time series segments and `bias` is a small term to tackle constant segments.
- Dynamic Time Warping (DTW): `dtw.dtw(seq_1, seq_2, dist_method="sqeuclidean", step_pattern="symmetric1", distance_only=True).distance` with `dtw-python` from <https://dynamictimewarping.github.io/python/>.
- Mueen's Algorithm for Similarity Search (MASS): from <https://github.com/matrix-profile-foundation/mass-ts>. We have used `mass_ts.mass2(ts, query)` to calculate the distance profile. GPU support is deactivated.
- Symbolic Aggregate approXimation (SAX): our own implementation. The authentic SAX assumes that the values in the time series are normally distributed. Based on this assumption, it calculates quantiles to discretize the values. Our data are not normally distributed. Therefore, we calculate the genuine value distribution then the quantiles accordingly. Furthermore, we fit the SAX (estimate distribution and calculate quantiles) with the query instead of the time series in the database, which significantly improves the accuracy, because sometimes, the query has a much smaller value range as the time series in the database, as in APST. The cardinality is always set to 10, also for SAXRegex.
- Regex: `regex.compile(query).finditer(time_series)` from <https://github.com/mrabarnett/mrab-regex>. The parameter `overlapped` can be set to `True` to find overlapping patterns.

Evaluation Setting

IoU thresholds and similarity thresholds: This metric mean Average Precision (mAP) requires a threshold for Intersection over Union (IoU) between a predicted interval and the closest ground truth label, in order to judge whether the prediction is a true positive or a false positive. We chose 30% and 50% as the IoU-threshold, denoted as mAP30 and mAP50, respectively. This is more lenient than in computer vision. For example, the Pascal competition uses 50% as the IoU-threshold. The COCO competition uses a range of [0.5, 0.55, 0.60, 0.65, 0.70, 0.85, 0.90, 0.95] and calculates the mean over all the eight thresholds. However, we found a looser IoU-threshold better suits the time series cases, because the degree of overlapping between predictions and ground truth labels in time series cases are generally smaller than in the computer vision cases. As for the other metrics, which formulate the problem as binary classification of each time step (inside or outside a target pattern), we calculate them with the (similarity) threshold that produces the best F1 score.

Distance profile to similarity profile: Instead of similarity, DTW, Euclidean Distance (ED) and SAX' distance measure calculates distance. We invert the distance profile and normalize it. For evaluation, it is not the absolute value of the similarity that matters, rather the ranking of similarities. Hence, any manipulation on the similarity profile that does not change the ranking of similarity for each time step does not change the metrics. We even scale the similarity profile heterogeneously so that the top hits stand out.

Degeneration of the predicted intervals with SAXRegex: As shown in Appendix D, SAXRegex can locate varied intervals in different tracks within a predicted pattern. Namely, rather than starting and ending at the same time, the shape in each track can have different starting and ending time. For a consistent accuracy benchmark with other methods, we use the earliest starting time and the latest ending time within a pattern as the predicted interval for SAXRegex. To find horizontally scaled patterns, we use a set of eight sliding windows of exponentially increasing window lengths to deal with the scaling problem for the benchmark methods.

SAXRegex's pseudo-similarity: SAXRegex does not calculate similarity. However, mAP requires a confidence for each prediction. To solve this problem, we calculate pseudo-similarity for the predicted intervals. The idea is, we run SAXRegex with a range of tolerance band widths (the allowed number of symbol deviations). Each prediction is found with a tolerance band. The narrower the band, the higher the similarity. We inverted and normalized the band width as the pseudo-similarity. As mentioned above, the absolute similarity value is not important for the metrics, as long as the similarity ranking between the predictions retains.

Resolution for fair comparison: SAX's first step PAA aggregates several time stamps, calculates their average and merges them into one, which will be transformed into one symbol in subsequent steps. For SAXRegex, the query of a certain length is down-scaled by SAX to 20 symbols (which is sufficient for all our datasets), because regex does not scale well with the pattern length. Accordingly, the time series is down-scaled with the same bin size horizontally along the time axis with SAX. This reduces the data volume and accelerates the subsequent

processing. To ensure a fair speed comparison, we conduct PAA with the same resolution for other benchmark methods to accelerate them. This only affects the speed benchmark. Whereas in the accuracy comparison, we do not conduct PAA for the benchmark methods. Rather, we use the finest possible resolution to achieve the highest possible accuracy for the benchmark methods.

Appendix D: Visual Inspection of the Datasets and Predictions

We keep the visual inspection of all methods for all publishable datasets in this appendix.

Each figure corresponds to a dataset, containing the result of all methods. For each method, every track in the query is plotted on the left and every track in the time series, where to search for the query, on the right. The blue curves stands for the original data.

The benchmark methods calculate similarity profiles and they are delineated with the gray curves. The similarity profile at a time step for a track records the similarity of a potential predicted interval starting at the time step. The length of the interval corresponds to the sliding window size. A range of sliding windows with increasing window sizes is used. The similarity profile keeps records of the window size with the highest similarity for a time step. The similarity profiles are averaged over all tracks and the merged similarity is shown in a separate sub-figure under all time series tracks.

The user can tune a threshold marked as red lines in the plots. If a similarity peak reaches the threshold, a predicted interval denoted as a green area is found. We conduct non-max-suppression to remove intervals with large overlapping. SAXRegEx works differently as the benchmarks. It does not have similarity profiles, accept time-shifted patterns as the query and can find such patterns in the time series.

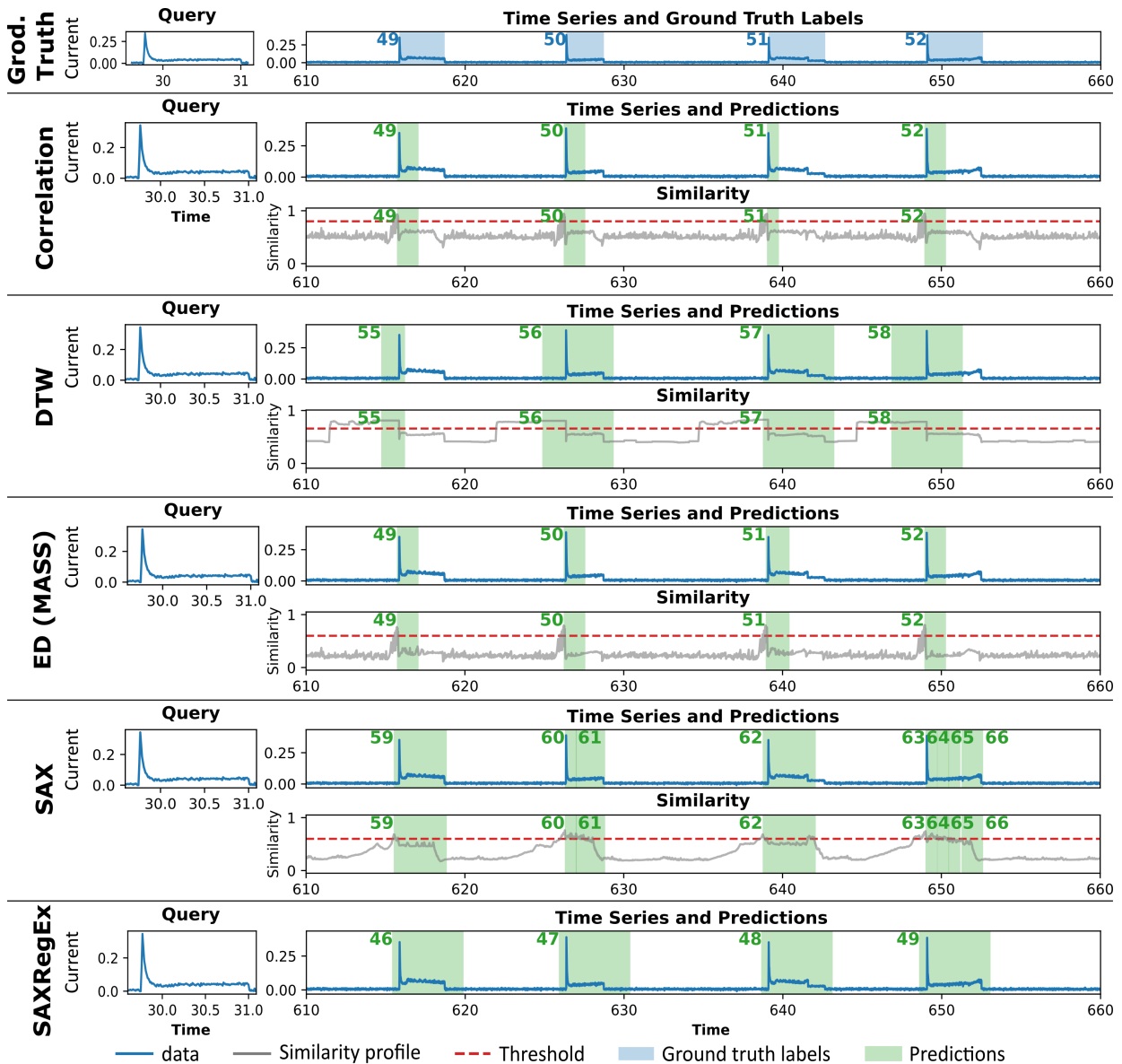


Figure D.1: Visual inspection of all methods on the deep valve dataset (excerpt).

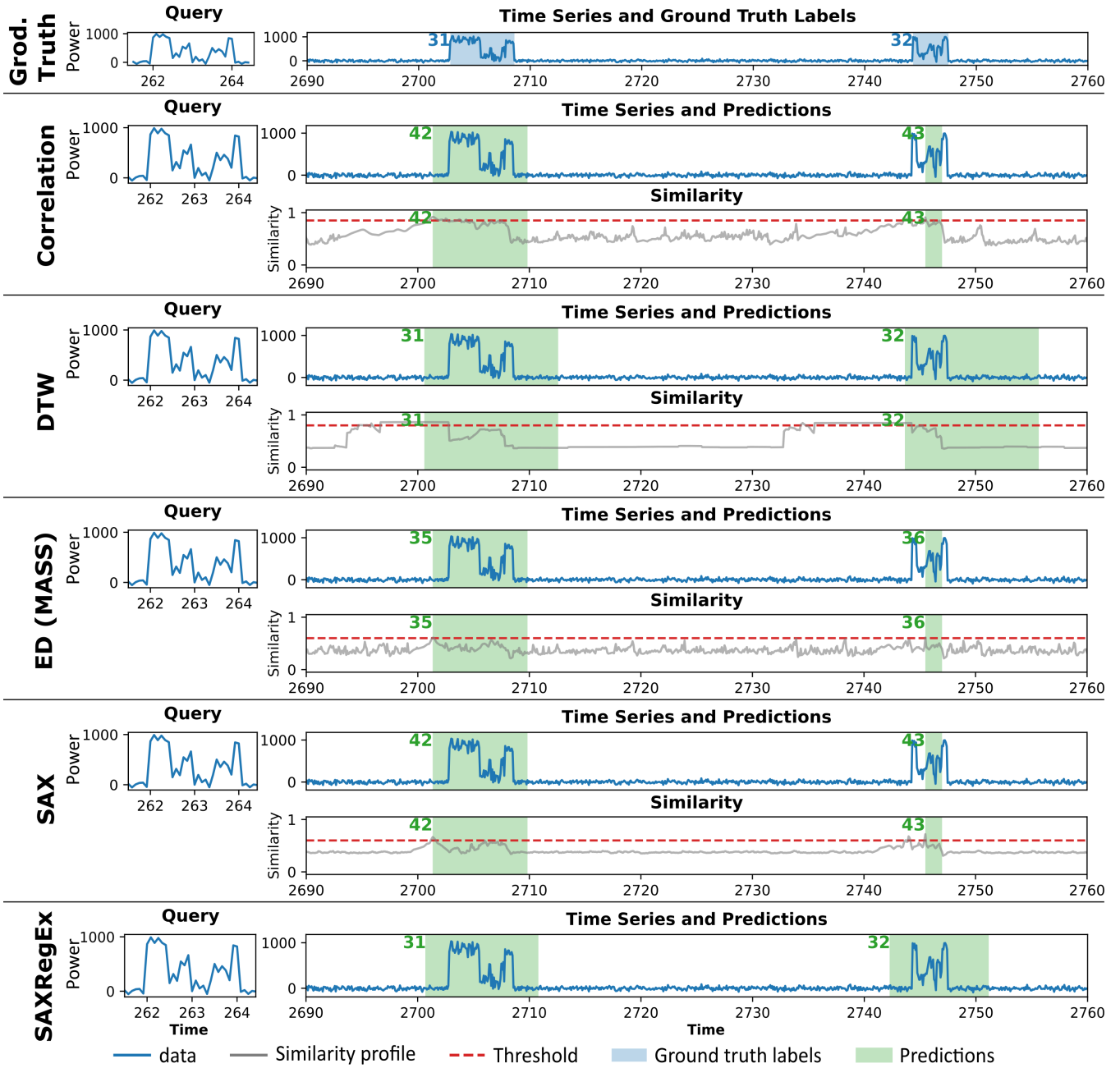


Figure D.2: Visual inspection of all methods on the cable cutter dataset (excerpt).

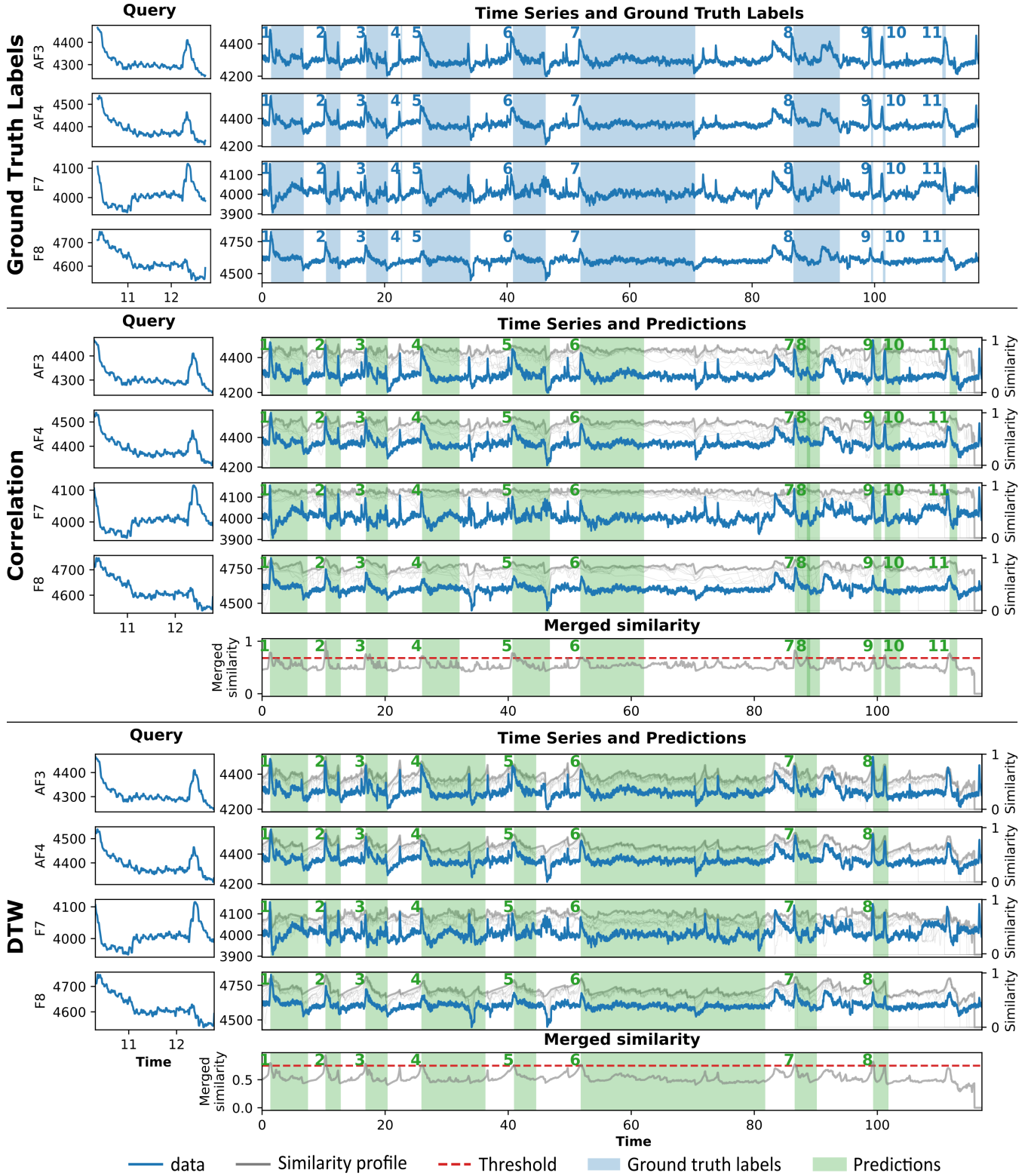


Figure D.3: Visual inspection of all methods on the EEG eye State dataset (part 1).

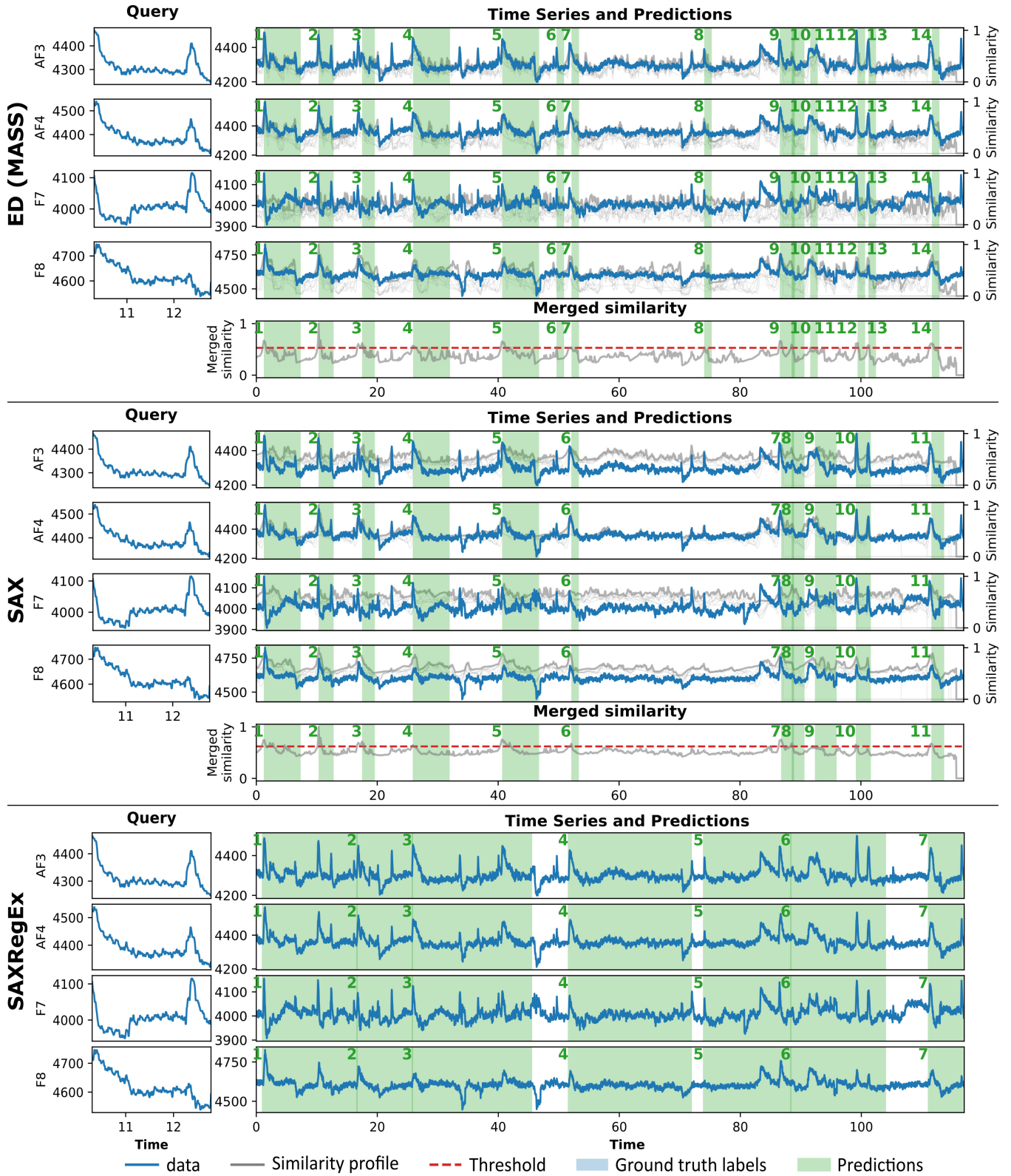


Figure D.4: Visual inspection of all methods on the EEG eye State dataset (part 2).

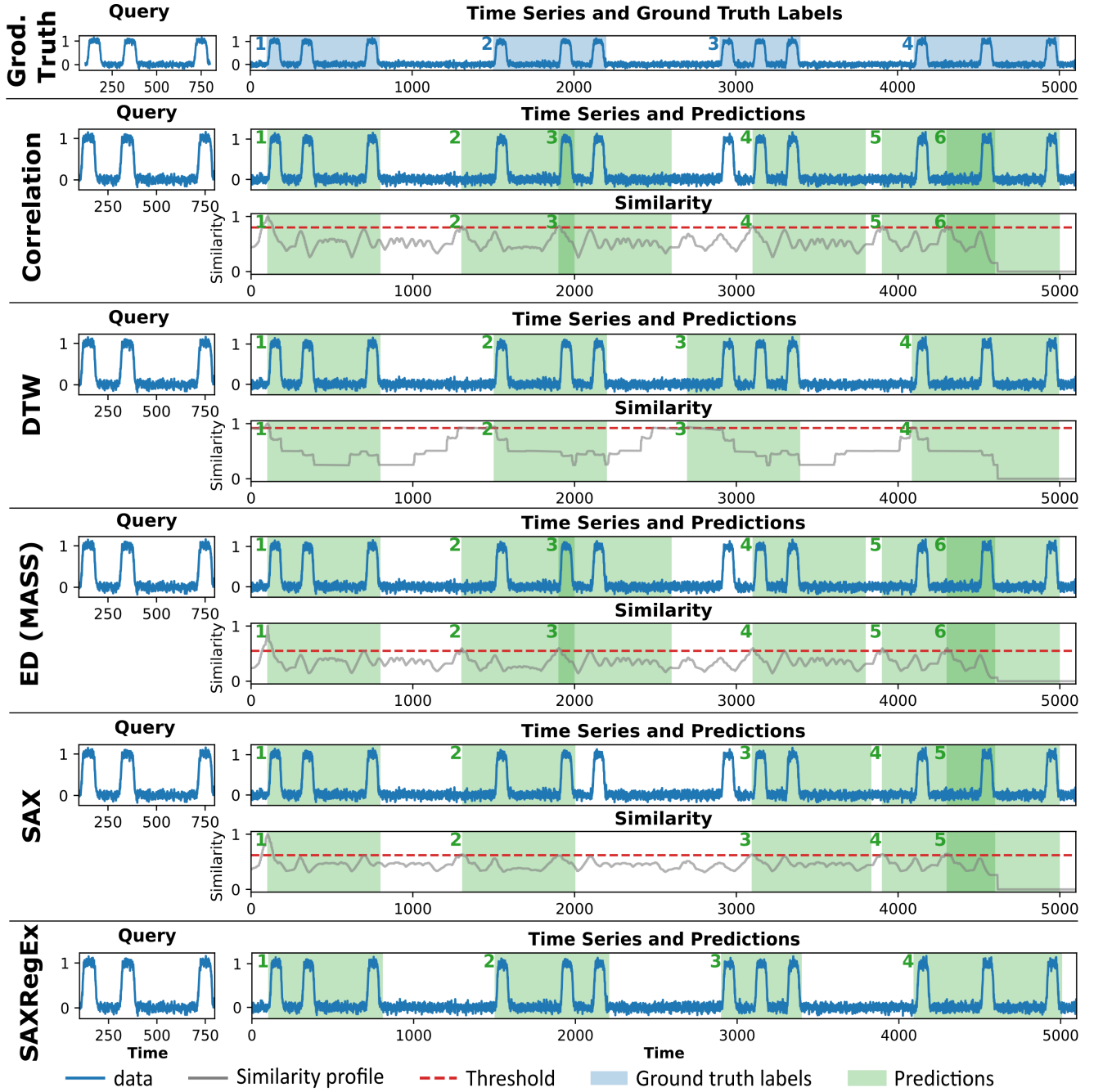


Figure D.5: Visual inspection of all methods on the CAN 1 dataset.

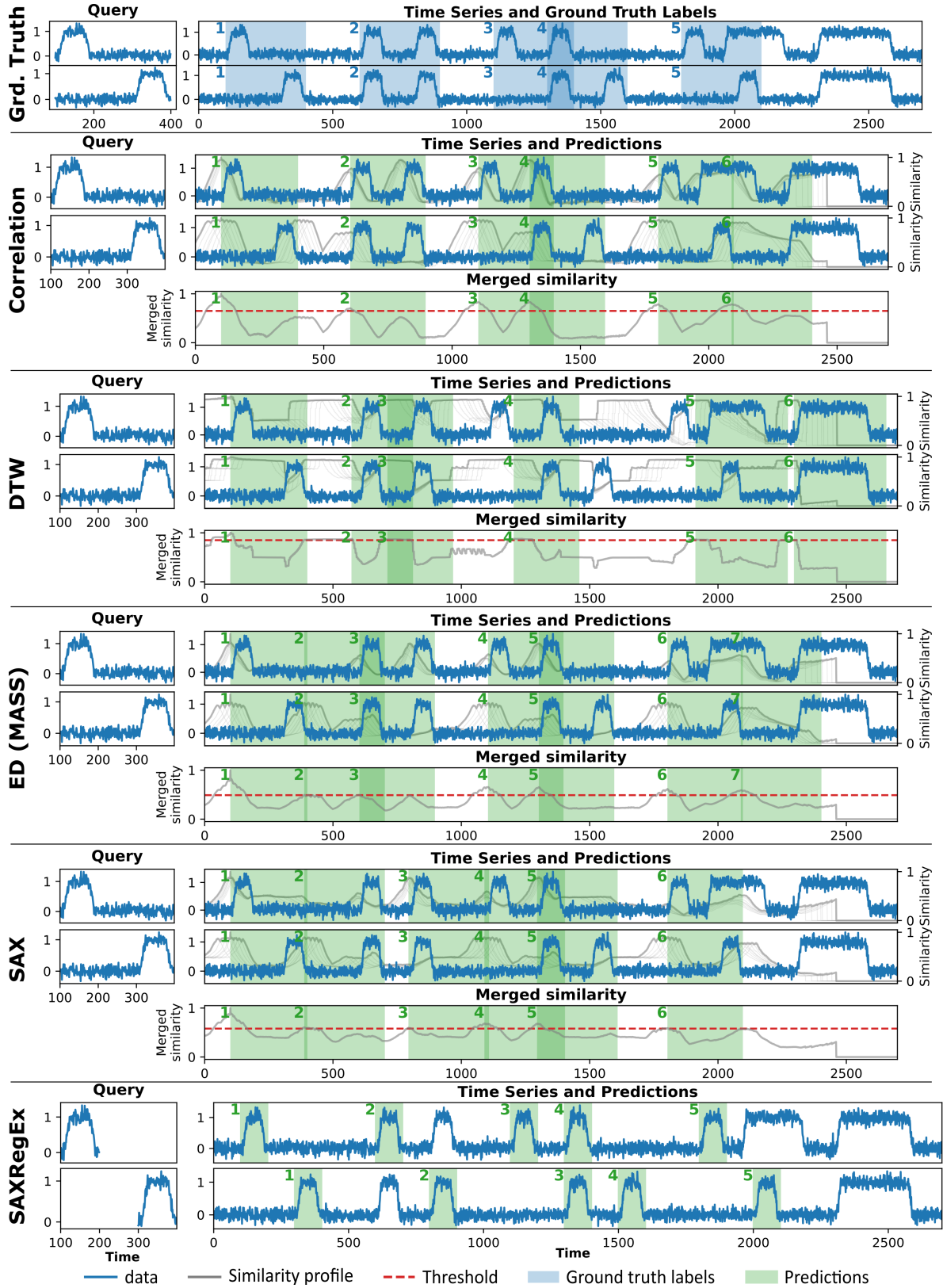


Figure D.6: Visual inspection of all methods on the CAN 2 dataset.

Appendix E: Complete Accuracy Benchmark

		Accuracy	Balanced accuracy	Precision	Recall	F1 score	mAP30	mAP50
APST	Correlation	0.99 ± 0.01	0.98 ± 0.03	0.80 ± 0.16	0.97 ± 0.07	0.88 ± 0.12	0.89 ± 0.25	0.89 ± 0.25
	DTW	0.98 ± 0.02	0.98 ± 0.03	0.65 ± 0.22	0.98 ± 0.04	0.76 ± 0.19	0.87 ± 0.30	0.87 ± 0.30
	ED (MASS)	1.00 ± 0.01	0.98 ± 0.03	0.87 ± 0.07	0.97 ± 0.07	0.92 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
	SAX	0.99 ± 0.02	0.97 ± 0.03	0.57 ± 0.28	0.96 ± 0.07	0.69 ± 0.23	0.57 ± 0.33	0.57 ± 0.33
	SAXRegEx	0.88 ± 0.27	0.92 ± 0.14	0.47 ± 0.37	0.95 ± 0.10	0.55 ± 0.36	0.64 ± 0.39	0.64 ± 0.39
Cable Cutter	Correlation	0.98	0.93	0.79	0.86	0.82	0.89	0.56
	DTW	0.94	0.96	0.48	0.98	0.64	0.84	0.19
	ED (MASS)	0.93	0.74	0.36	0.52	0.43	0.27	0.18
	SAX	0.97	0.85	0.74	0.71	0.72	0.69	0.31
	SAXRegEx	0.96	0.92	0.55	0.87	0.67	0.89	0.55
Deep Valve	Correlation	0.89	0.73	0.88	0.47	0.61	0.82	0.29
	DTW	0.79	0.85	0.46	0.94	0.62	0.46	0.19
	ED (MASS)	0.90	0.76	0.88	0.53	0.67	0.85	0.28
	SAX	0.96	0.95	0.87	0.93	0.90	0.73	0.66
	SAXRegEx	0.88	0.86	0.62	0.84	0.71	0.83	0.50
EEG Eye State	Correlation	0.79	0.78	0.76	0.77	0.76	0.57	0.40
	DTW	0.80	0.80	0.73	0.87	0.79	0.57	0.43
	ED (MASS)	0.69	0.70	0.63	0.77	0.69	0.58	0.36
	SAX	0.65	0.66	0.58	0.80	0.67	0.40	0.37
	SAXRegEx	0.68	0.70	0.59	0.92	0.72	0.62	0.62
Filling Prediction	Correlation	0.97	0.80	0.91	0.61	0.73	0.88	0.88
	DTW	0.95	0.89	0.57	0.82	0.67	0.75	0.75
	ED (MASS)	0.97	0.80	0.91	0.61	0.73	0.84	0.84
	SAX	0.97	0.92	0.69	0.87	0.77	0.95	0.95
	SAXRegEx	0.90	0.81	0.37	0.72	0.49	0.58	0.47
Variable Displacement	Correlation	1.00 ± 0.00	0.96 ± 0.02	0.98 ± 0.01	0.92 ± 0.03	0.95 ± 0.01	0.88 ± 0.33	0.88 ± 0.33
	DTW	1.00 ± 0.00	0.91 ± 0.05	0.95 ± 0.05	0.82 ± 0.10	0.88 ± 0.07	0.83 ± 0.32	0.81 ± 0.33
	ED (MASS)	1.00 ± 0.00	0.96 ± 0.01	0.97 ± 0.04	0.92 ± 0.03	0.94 ± 0.02	0.87 ± 0.33	0.87 ± 0.33
	SAX	1.00 ± 0.00	0.95 ± 0.03	0.96 ± 0.03	0.90 ± 0.06	0.93 ± 0.04	0.84 ± 0.33	0.84 ± 0.33
	SAXRegEx	1.00 ± 0.00	0.95 ± 0.03	0.96 ± 0.06	0.89 ± 0.07	0.92 ± 0.05	0.83 ± 0.32	0.83 ± 0.32
CAN 1 (with heterogeneous horizontal scaling)	Correlation	0.73	0.70	0.68	0.93	0.79	0.79	0.75
	DTW	0.95	0.95	0.93	1.00	0.96	1.00	1.00
	ED (MASS)	0.73	0.70	0.68	0.93	0.79	0.90	0.78
	SAX	0.76	0.75	0.74	0.86	0.80	0.90	0.81
	SAXRegEx	0.99	0.99	0.99	0.99	0.99	1.00	1.00
CAN 2 (with inter-track time shifts)	Correlation	0.88	0.88	0.82	0.99	0.90	0.97	0.97
	DTW	0.77	0.77	0.80	0.75	0.77	0.89	0.70
	ED (MASS)	0.88	0.89	1.00	0.78	0.87	0.94	0.94
	SAX	0.81	0.81	0.76	0.93	0.84	0.84	0.84
	SAXRegEx	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Figure E.7: Complete accuracy benchmark: best performance of all methods on all datasets. The APST dataset and the variable displacement dataset have multiple files. The evaluation metrics are calculated individually for each file. Then, we average them and calculate their standard deviation. Best F1-score, mAP30 and mAP50 among five methods are highlighted bold and red. The results show that each method suits different datasets, or different datasets favor different methods. No method outperforms the other consistently. The proposed method SAXRegEx outperforms the other, when the patterns are strongly heterogeneously horizontally scaled or have inter-track time shifts, as in the last two cases.

Appendix F: Complete Speed Benchmark

	APST	Cable Cutter	Deep Valve	EEG Eye State	Filling Prediction	Variable Displ.	CAN 1	CAN 2
DTW	24.54±2.29	43.56±3.39	9.98±0.37	2.00±0.57	1.36±0.07	94.66±5.25	0.320 ± 0.031	0.410 ± 0.028
Correlation	27.90±3.25	55.09±4.11	12.35±1.24	3.61±0.38	1.41±0.03	107.44±7.57	0.644 ± 0.636	0.542 ± 0.288
SAX	2.59±0.21	7.89±1.83	1.26±0.04	0.20±0.01	0.23±0.01	8.28±0.83	0.222 ± 0.016	0.282 ± 0.006
ED (MASS)	15.05±0.87	27.18±1.48	6.19±0.29	1.23±0.30	0.83±0.06	48.36±2.36	0.062 ± 0.003	0.074 ± 0.005
SAXRegEx	0.65±0.03	4.39±0.55	0.22±0.01	0.05±0.00	0.04±0.01	1.36±0.16	0.022 ± 0.003	0.022 ± 0.007

Figure F.8: Complete speed benchmark: unit sec. We repeat the same measurement five times and recorded the their mean and standard deviation. SAXRegEx outperforms the other methods for all datasets significantly.

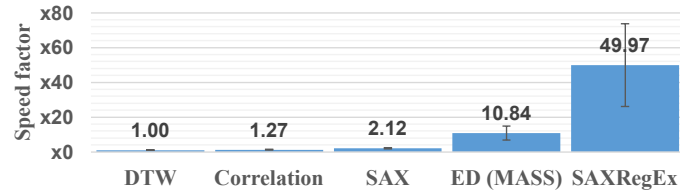


Figure F.9: Speed benchmark in factor: The mean execution time of each method is divided from that of DTW. SAXRegEx is on average $x50$ faster than DTW in our experiments. This figure is derived from [Figure F.8](#)

We use SAX to down-sample the query to 20 symbols and down-sample the time series with the same bin size. To ensure a fair comparison, we conduct PAA with the same resolution for other methods during the speed benchmark.

The speed ranking of correlation and DTW is inconsistent. Because the synthesized datasets CAN 1 and CAN 2 are fairly small. Therefore, the searching time is not dominant during the processing. The used DTW library seems to be better implemented and outperforms the correlation implementation. However, when the query length grows, its relatively inferior scalability ($O(dq^2lk)$) compared with that of correlation ($O(dqk)$) starts to reveal itself, where d stands for the number of tracks (dimensions), q the query length, l the length of the time series, where to search for the query and k the number of sliding window kernels of different size for longitudinally scaled patterns. Due to this reason, we exclude the datasets CAN 1 and CAN 2 for speed benchmark, where the result on all other datasets listed in [Table 1](#) are averaged.

By the way, SAX and ED also have the complexity $O(dqk)$ as correlation. The scalability of SAXRegEx is dependent on the regex search engine. We cannot express it simply but empirically, it does not scale well. Thus, we always down-sample the query to 20 symbols horizontally along the time axis with SAX, which is more than enough for all our datasets. As mentioned above, the effect is compensated with PAA.

References

- [CLL*19] CHENYUN YU, LINTONG LUO, LEANNE LAI-HANG CHAN, THANAWIN RAKTHANMANON, SARANA NUTANONG: A fast lsh-based similarity search method for multivariate time series. *Information Sciences* 476 (2019), 337–356. doi:10.1016/j.ins.2018.10.026. 2
- [HFZ*15] HUANG Q., FENG J., ZHANG Y., FANG Q., NG W.: Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proc. VLDB Endow.* 9, 1 (2015), 1–12. doi:10.14778/2850469.2850470. 2
- [LKLC03] LIN J., KEOGH E., LONARDI S., CHIU B.: A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (New York, NY, USA, 2003), DMKD '03, Association for Computing Machinery, pp. 2–11. doi:10.1145/882082.882086. 2
- [LKWL07] LIN J., KEOGH E., WEI L., LONARDI S.: Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15, 2 (2007), 107–144. doi:10.1007/s10618-007-0064-z. 2