



# Assessment and Adaption of Pattern Discovery Approaches for Time Series Under the Requirement of Time Warping

Fabian Kai-Dietrich Noering<sup>1</sup>(✉), Konstantin Jonas<sup>1</sup>,  
and Frank Klawonn<sup>2</sup>

<sup>1</sup> Volkswagen Group Research, 38436 Wolfsburg, Germany  
{fabian.kai-dietrich.noering,  
konstantin.jonas}@volkswagen.de

<sup>2</sup> Ostfalia University of Applied Sciences,  
Salzdahlumer Str. 46/48, 38302 Wolfenbüttel, Germany  
f.klawonn@ostfalia.de

**Abstract.** In the automotive industry, the cars themselves as well as the production lines, produce a high amount of data day by day. To get information out of these data there is a need for high performance data mining tools. One of these tools is the pattern discovery. This paper addresses the assessment of different approaches to discover frequent pattern in time series. Our special requirement is the detection of time warped pattern with variable length. The comparison includes approaches based on dynamic time warping (DTW), discretization as well as Keogh's Matrix Profile. Every approach is exemplarily implemented in MATLAB and (if necessary) adapted to face our use cases. The focus of the assessment will be the quality of the results, the runtime and the effort of parametrization. For evaluation, time series test datasets are generated with predefined patterns based on random walks. The output patterns, identified by the different pattern discovery algorithms, are compared with the initial patterns and evaluated with respect to the Jaccard index. This leads to a quality score for every algorithm and every parametrization and the possibility to compare different algorithms as well as approaches.

**Keywords:** Pattern discovery · Motif discovery · Pattern enumeration  
Grammar induction · Pattern evaluation · Time series · Time warping  
Unsupervised

## 1 Motivation

During the operation of complex mechatronic systems various time-resolved data is accumulated. The analysis of such time series data provides information on usage, condition or even misbehavior of the systems or components. Focusing on fault diagnosis, usually only short error logs are analyzed with regard to a possible abnormal signal behavior. However, to obtain long-term information on usage, wear-off, behavior or system's condition an analysis of the entire data record is required. In this context, change detection methods are needed to identify slow or abrupt changes in the

signals or signal groups behavior. In the context of wear-off for systems having recurring stress patterns, e.g. manufacturing robots, it is advantageous to compare only such patterns. In our use case, we cannot expect experts to specify suitable patterns and therefore we need unsupervised methods for automatic definition of such recurring stress patterns without input of expert knowledge.

## 2 Introduction

Although there are a lot of different approaches and algorithms for the problem of (unsupervised) pattern discovery in time series data, the number of useful approaches limits with the requirements of our special use cases. We need reliable methods to detect recurring patterns, which are usually time warped and noisy, in datasets with far more than one million data points at a reasonable runtime. Beyond that we also would like to detect patterns in multidimensional time series. However, in this paper we limit our analysis for simplicity to a one dimensional case. Following the conditions described above we will compare three different approaches. There may be the need of adaption to fulfill the requirements, e.g. because some approaches aim to discover motifs<sup>1</sup>, while our goal is the pattern<sup>2</sup> discovery.

At first there are algorithms based on the principle of *dynamic time warping* (DTW). DTW is a distance or similarity measurement for time series with different length with the possibility to stretch or compress one of the time series to fit the other one [1]. Beyond that, the DTW can be adjusted and extended to the problem of discovering patterns. The authors of [2] proposed a DTW-based algorithm called *CrossMatch* to identify similar subsequences in different sequences without the need of sequences. A similar technique is described in [3] where the approach is extended by a hierarchical clustering algorithm to find patterns.

The second approach is based on the symbolic representation of the time series, which is interesting in terms of reduction of complexity and therefor also for the reduction of the runtime. Furthermore these approaches can be more robust against noise than those based on the DTW. To get a symbolic representation, there are many methods for time series discretization. The *Symbolic Aggregate Approximation* (SAX) [4] algorithm and other symbolic time series methods use intervals with equal probability. Besides, there is the possibility to use intervals of equal size. The relevance of symbolic approaches is evidenced by the variety of existing pattern discovery algorithms from various research fields like bioinformatics or text mining. For example in bioinformatics the *Smith Waterman* algorithm is used for sequence alignment, hence the pattern discovery in DNA sequences [5]. As DTW, Smith Waterman is based on dynamical programming and is able to find time warped patterns. The *Sequitur* algorithm, proposed in [6], extracts the hierarchical structure of a symbolic sequence by replacing recurring phrases with grammatical rules while forming a dictionary of these rules, which is formally known as grammar induction. Sequitur identifies only identical

---

<sup>1</sup> Definition "motif": A motif is a pair of similar subsequence in a time series.

<sup>2</sup> Definition "pattern": A pattern is a group of at least two similar subsequences in a time series.

patterns and ignores possible similarities in rules/patterns. Furthermore it is not guaranteed to find all the patterns in the sequence, because of its replacing procedure. Pattern enumeration is often referenced to cope with last mentioned problem. To perform time warped pattern discovery with those algorithms there have been proposed *numerosity reduction* techniques, as e.g. employed in [7]. Especially for long sequences the number of enumerated patterns can increase exponentially. That is why there is also research in constraint programming techniques like in [8] or [9].

The last approach we want to focus on is the principle of *Matrix Profile* introduced by Keogh et al. [10]. Basically, it is a brute force approach to find motifs or patterns of a fixed length by calculation the distances between every possible query sequences and every other sequence of the chosen length. Nevertheless it is said to be fast because of the *MASS* algorithm (Mueen’s ultra-fast Algorithm for Similarity Search) also proposed in [10]. As it uses the z-normalized Euclidian distance, i.e. a fixed length, it is not suitable to find time warped patterns. A possibility to solve this issue is to replace the Euclidian distance with the DTW. However, this makes it even more computational expensive and doesn’t solve the problem to be only able to compare sequences of equal length. Furthermore, because of the z-normalization, it ignores the original values of the compared sequences. That is why we implemented additionally a version of the standard Euclidean distance.

In this paper we focus on the comparison of the aforementioned approaches including necessary adaptations. Our target is an evaluation concerning runtime, parameterization effort and pattern quality. Therefor we apply the Jaccard Index to calculate a quality score called overlap, which gives us the opportunity to evaluate and compare different algorithms as well as approaches.

The following section gives detailed information about the applied algorithms and our exemplary implementations to fulfill our requirements. Section 4 explains our approach for the comparison of pattern discovery algorithms. Section 5 evaluates our results and gives statements to the assessment of the different approaches. The final section gives a conclusion and an outlook to future research.

### 3 Approaches for Pattern Discovery

#### 3.1 DTW-Based

The tested implementation is based on the approaches from [2] and [3]. We are using the scoring function of the CrossMatch algorithm, which is necessary to compute the matrix equivalent to the DTW algorithm. In the case of CrossMatch the matrix contains values of similarity in contrast to the DTW which computes a distance matrix. The scoring function for every cell  $(i, j)$  of the similarity matrix  $v$  of CrossMatch using the time series  $x$  and  $y$  is shown in Eq. (1).

$$v(i, j) = \max \begin{cases} \varepsilon b_d - ||x_i - y_j|| + v(i - 1, j - 1) \\ \varepsilon b_v - ||x_i - y_j|| + v(i, j - 1) \\ \varepsilon b_h - ||x_i - y_j|| + v(i - 1, j) \\ 0 \end{cases} \quad (1)$$

A major difference between the scoring functions of the classic DTW and CrossMatch is the usage of weighting factors  $\epsilon b_d$ ,  $\epsilon b_v$  and  $\epsilon b_h$ . It enables us to penalize time warping, i.e. horizontal or vertical steps in the similarity matrix. While DTW is only able to calculate a distance between two sequences, CrossMatch enables the detection of motifs as well. The end of a motif is marked by a local maximum in the similarity matrix  $v$ . Furthermore the CrossMatch algorithm calculates a position matrix, which contains the starting points of the motifs. For further explanations of DTW see [11] and CrossMatch see [2].

For the tests that we are going to perform, we limit the weighting factors to  $\epsilon b_d = 1$ , as a reward for similarity, and  $-1 \leq \epsilon b_v = \epsilon b_h \leq 0$ , as a possible penalization for time warping. Furthermore we need to make an adaption to the CrossMatch scoring function, see Eq. (2). In order to search for motifs in one and the same time series, we have to exclude the trivial matches. Therefor we implement an offset region around the diagonal of the similarity matrix  $v_{adapt}(i,j)$ . These matrix cells are set to zero by default. Due to symmetry only half of the matrix has to be calculated. These adaptations also apply to the position matrix.

$$v_{adapt}(i,j) = \begin{cases} v(i,j), & |i - j| > offset \\ 0, & |i - j| \leq offset \end{cases} \quad (2)$$

In order to form patterns, additional to the CrossMatch, we extract and cluster the motifs like in [3]. To extract possible pattern candidates from the matrix, we use a minimum length and a minimum similarity. Every candidate's score higher than the product of *min\_length* and *min\_similarity* is going to be clustered. The clustering algorithm needs another parameter specifying the sensitivity of the extracted motifs to be clustered to patterns, which is later referenced to *max\_similarity\_motif*. To ensure, that every subsequence in the time series similar to the pattern is discovered, a representative of every pattern is extracted and searched in the whole time series by recalculating the CrossMatch similarity matrix with the representative query sequence.

### 3.2 Discretization-Based

Concerning discretization-based approaches for pattern discovery we focus on an adapted Sequitur algorithm and an algorithm for pattern enumeration. Both are based on an equal probability discretization technique. However, in contrast to the technique described in [4], we don't assume a Gaussian distribution of the data values, which is necessary in terms of various signal types. We formed a simple algorithm to form classes of equal count of data points with respect to the constraints *uniqueness* and *allocation*. Uniqueness means that there must not be multiple classes with the same data value. According to the allocation constraint, every data value has to be allocated to a class. The algorithms we will describe are only dependent on one parameter, the number of discretization steps.

Because of the need of time warping we apply an optional numerosity reduction technique like in [7], which we call *symbolic reduction*. Therefore every identical consecutive symbol in the time series is reduced to a unique symbol. Hence in every

step of the time series is a change in the symbolic value. We will later discuss the benefit of this technique.

### Pattern Enumeration Algorithm

To evaluate the performance of our adapted Sequitur algorithm, we also want to apply a pattern enumeration algorithm. The algorithm repeatedly scans the whole time series with different query pattern length by using a matrix representation of the symbolic time series and a sort algorithm. In Table 1 the pseudocode of the algorithm is shown.

**Table 1.** Pseudocode for pattern enumeration

```

1  l_pattern = 1
2  empty = false
3  while ~empty
4    l_pattern = l_pattern + 1
5    ts_matrix = form_matrix(sts, l_pattern)
6    [ts_matrix_sorted, index] = sort(ts_matrix)
7    [symbolic_repr, num, loc] = get_symbolic_repr(ts_matrix_sorted)
8    for p = 1:length(symbolic_repr)
9      if num(p) > 1
10       add_to_dictionary(symbolic_repr(p), location(p))
11     end
12   end
13   if isempty(find(num > 1))
14     empty = true
15   end
16 end

```

The core of the algorithm is in lines 5 to 7. In line 5 the time series array is transformed to a matrix of the size  $(length(sts) - l\_pattern) \times l\_pattern$ .  $l\_pattern$  describes the length of the patterns that are searched in the current iteration of the while loop. In the function *form\_matrix* the symbolic time series *sts* gets multiplied  $l\_pattern$  times and shifted by a number of entries dependent on the column. This leads to matrix rows with consecutive time series values. In line 6 the matrix is sorted along the first dimension, which leads to a matrix where equal row values are located in consecutive rows. By detecting the differences of the sorted matrix within the function *get\_symbolic\_repr*, the contained symbolic combinations or representations can be extracted. Furthermore the number and the location of the occurrences can be identified. In the last step every detected pattern with a count greater than one, is documented in a dictionary. The dictionary contains every detected pattern and its location.

### Adapted Sequitur Algorithm

The Sequitur algorithm is based on two constraints. The first constraint is named *diagram uniqueness*, which means that every diagram (a pair of two adjacent symbols) is allowed to occur only once in the symbolic time series. If a diagram appears more than once, then a rule has to be formed. Every rule has to fulfill the second constraint, the *rule utility*. It says that every rule has to occur at least twice. For detailed information about the execution of the Sequitur algorithm see [6].

Our adaption of the algorithm is shown in Table 2. The first three steps within the WHILE loop are similar to the pattern enumeration algorithm, except for a fixed matrix column size of 2. To fulfill both, the diagram uniqueness and the rule utility, the algorithm adds the diagram that occurs the most to the dictionary and replaces the diagram in the symbolic time series by a new symbol. It stops when the diagram uniqueness is fulfilled.

**Table 2.** Pseudocode for adapted Sequitur

```

1  empt = false
2  while ~empt
3      ts_matrix = form_matrix(sts, 2)
4      [ts_matrix_sorted, index] = sort(ts_matrix)
5      [symbolic_repr, num, loc] = get_symbolic_repr(ts_matrix_sorted)
6      if max(num) > 1
7          [v, i] = max(num);
8          [Dict, numofrule] = addtodict(Dict, symbolic_repr(i), loc(i));
9          sts = replace_bynewrule(sts, loc, numofrule);
10     else
11         empt = true;
12     end
13 end

```

### 3.3 Matrix Profile

The Matrix Profile introduced by Keogh et al. is an approach able to be executed on raw time series without any preprocessing. Matrix Profile is based on the calculations of distance profiles. A distance profile visualizes the distances between a query sequence and every other possible sequence in a time series (with  $length(time-series) \gg length(query)$ ) while the query sequence is part of the time series and all the compared sequences have the same length. Thus the minimum of the distance profile, excluding the trivial match, is the best match with the query sequence. To extend the distance profile to a Matrix Profile, a distance profile for every possible query is calculated. The minima of every distance profile are visualized in the Matrix Profile. For further details see [10].

To extract motifs of a fixed length we need a maximum distance that is not allowed to be exceeded between two sequences of a motif. Every data point in the matrix profile that fulfills this constraint can be seen as the starting point of a motif. The corresponding distance profile gives information about other sequences that are similar to the motif. Again every data point that falls below the maximum distance leads to a similar sequence. The motif with its similar sequences can then be called a pattern. To discover patterns without a fixed query length we have to apply the algorithm multiple times for every possible query length [10].

Because of the naïve structure of the approach, the MASS algorithm was proposed in [10], which calculates the z-normalized Euclidian distance profile by convolution. In comparison to the naïve approach of calculating the Euclidian distance between the query and every other sequence, this technique reduces the time complexity for a distance profile from  $O(n * m)$  to  $O(n * \log(n))$  with  $m$  being the length of the query

and  $n$  the length of the whole time series. However, for time warped pattern discovery the Euclidian distance is not suitable. An alternative solution is to replace it with DTW, which theoretically increases the complexity to  $O(n * m^2)$ . For the calculation of a Matrix Profile we have to calculate approximately  $n$  distance profiles, which leads to a complexity of  $O(n^2 * m^2)$ . As we want to find patterns without input of a query length the Matrix Profile has to be calculated multiple times for different query lengths [10].

In a second implementation we tested Matrix Profile with the Euclidean distance but reduced complexity. It is based on the fact that the distance profiles of the query at starting point  $i$  and the one at starting point  $i + l$  have  $query\_length - l$  identical Euclidean parts. While calculating the distance profiles in serial, it is possible to keep these identical parts and calculate with just two additional computation steps (per data point in the distance profile) the next distance profile.

## 4 How to Compare Different Approaches/Algorithms?

As we want to compare different approaches to discover patterns, we need a test standard. One possibility is to use benchmark time series, for example from financial stock markets or from seismology, which are commonly used to test different time series exploration tools. However, generalized results regarding every possible kind of pattern can be obtained only by use of synthetically created time series. Our test data is composited by predefined patterns, which can be labeled automatically. The advantage of this approach is the independency of a certain test case. Furthermore we can create an infinite number of test sequences without the effort of labeling it.

The evaluation is divided in three steps. In the first step the test data is created automatically from patterns that are generated randomly. In the second step, the pattern discovery, the predefined patterns are rediscovered using different algorithms and parameter sets. By using the index information of every predefined and located pattern, in the third step a quality score for the results is calculated.

### Create Time Series Test Data

To find generalized evaluation for different use cases, the test patterns should cover every possible kind of shape. That is why we chose the random walk as the basis for every pattern. In our case we use a Gaussian random walk, which changes the distribution of the randomly chosen step size from equal to normal. For the creation of a primal pattern by a Gaussian random walk the following parameters are also chosen randomly once: Value of the first data point; number of steps; maximum step size.

After creating different primal patterns, each of them is multiplied and distorted to form a set of similar members for each of the primal pattern. The number of members within a pattern is randomly chosen. The distortion is done by adding white Gaussian noise to the members, given a fixed signal to noise ratio, and by randomly stretching or compressing the length of the members, given a maximum ratio between the length of the primal pattern and the distorted member.

To form a sequential time series based on the randomly created patterns and its members, the members are concatenated in a random order. Note that every member can only occur once in the time series. To overcome value jumps between the linked

patterns, we insert a smooth crossing. Start and end indices of every member and every pattern are saved.

### Pattern Validation

Output of the pattern algorithms are indices of starting and ending points of the located pattern. To evaluate the rediscovered patterns we calculate an overlap comparing these indices  $I_{loc}$  with the indices of the predefined pattern  $I_{pre}$  by using the Jaccard Index:

$$overlap(pre, loc) = Jaccard(pre, loc) = \frac{|I_{pre} \cap I_{loc}|}{|I_{pre} \cup I_{loc}|} \quad (3)$$

Hence an overlap score of 0 describes a mismatch and a score of 1 describes a perfect match between predefined and discovered pattern. Note that the overlap also decreases from a perfect match if the detected pattern is longer than the predefined. The calculation is performed for every combination of predefined and detected pattern. Afterwards a best match for every predefined pattern can be chosen. To express a one-score quality criterion for the algorithms, we chose the mean overlap value of the best matches.

Furthermore we perform the whole routine multiple times, to get a reliable statement for the quality of the algorithms. This gives us also the possibility to make a statement concerning the variance of the quality. Besides, in every iteration of the routine, the runtime of the different algorithms is recorded.

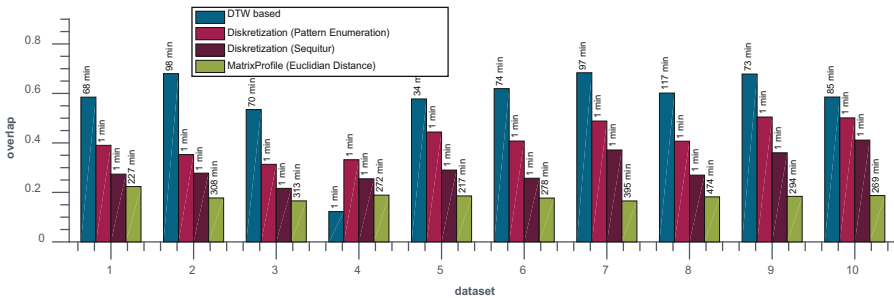
## 5 Experimental Evaluation

After explaining the different approaches for the pattern discovery and the methodology of calculating the quality score overlap, we now compare the different approaches. In our evaluation we created 10 random datasets, ran every algorithm with different parametrizations, and calculated the overlap. For each algorithm we figure out the parametrization leading to the best overlap, see Fig. 1. The overlap is plotted for every approach and for every random dataset. Next to the bars the corresponding runtime of the pattern discovery algorithm is indicated.

It is evident that the DTW-based approach outperforms every other tested approach concerning overlap. Discretization-based approaches, in turn, advance greatly concerning the runtime. The algorithm for pattern enumeration provides constantly better results than the Sequitur algorithm. As expected, the Matrix Profile with the use of the Euclidean distance is not able to compete with the other approaches because of its complexity and the inability of time warping. We expect better results when using the Matrix Profile in combination with DTW instead of the Euclidean distance, though substantially increasing runtime. As the DTW-based approach outperforms the Matrix Profile in runtime and overlap, Matrix Profile is not suitable for our use cases. However, as we calculated an entire Matrix Profile for every possible query length, there is still optimization potential for reducing the runtime based on early abandoning. Nonetheless this reduces the runtime, but doesn't improve the overlap.



In general all algorithms output a higher count of patterns than the number of predefined patterns, i.e. 16. While the DTW-based and the adapted Sequitur algorithm locates a reasonable count, the pattern enumeration algorithm and the Matrix Profile produce an extremely high amount of patterns (>20.000, >50.000, resp.). This shows the need of additional postprocessing techniques.

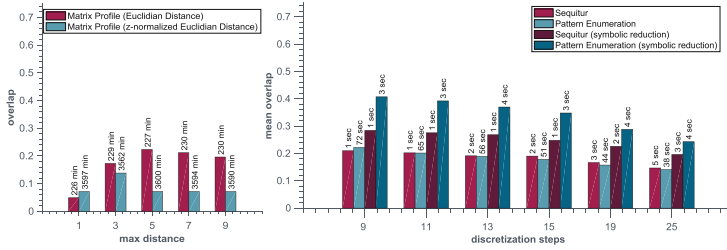


**Fig. 1.** Overview of different approaches concerning overlap and runtime; length of datasets approximately 50.000 data points; average count of located patterns: DTW ~800; Pattern Enumeration >20.000; adapted Sequitur ~200; Matrix Profile >50.000;

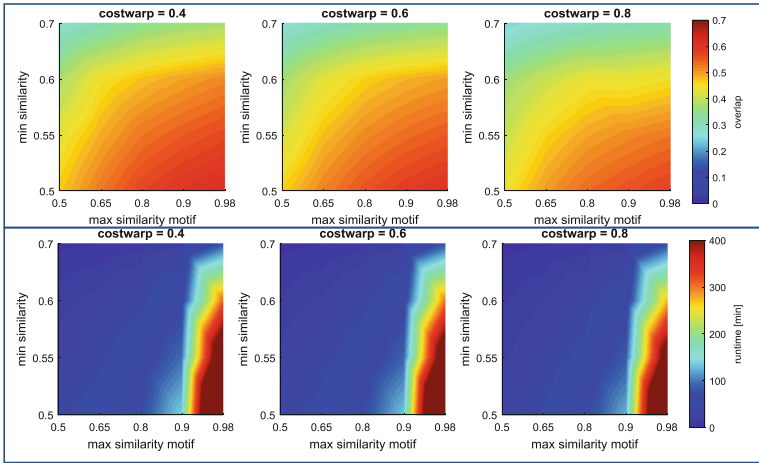
After giving a brief overview of the best case performances, we now have a closer look at the approaches including the process of parametrization. In Fig. 2 (left) the performance of Matrix Profile with variation of the maximum allowable distance within a pattern and with use of Euclidean and z-normalized Euclidean distance is shown. The Euclidean Matrix Profile is calculated with the runtime reduction technique while the z-normalized Euclidean Matrix Profile is calculated by the MASS algorithm. As shown in Fig. 2 the Euclidean Matrix Profile leads to a better overlap for most of the parametrizations, while requiring less than 10% of the runtime.

In Fig. 2 (right) the performance of the discretization-based approaches are evaluated with the variation of the number of discretization steps. Furthermore the benefit of the symbolic reduction is shown. It can be seen that the symbolic reduction has a higher impact on the results of the pattern enumeration algorithm than on the results of the Sequitur algorithm concerning overlap and runtime. However, in both cases the overlap as well as the runtime gets better with the use of the symbolic reduction, due to the time warping. Nevertheless, compared to the DTW-based approach, it is a slight disadvantage because the time warping is not controllable. i.e., after symbolic reduction, a symbol can represent two or considerably more identical consecutive symbols. A problem for both algorithms is the dependency of the discretization method, despite the dependency to only one parameter. Furthermore, we experienced a weakness in terms of multiple symbolic value toggling because of noise in value regions near the discretization borders. This effect could be mitigated by applying a frequency filter as a preprocessing step.

Figure 1 shows the best case results of the approaches concerning the overlap. For the DTW-based approach, for acceptable runtimes we chose a compromise between



**Fig. 2.** Left: results (overlap and runtime) of Matrix Profile with variation of the maximum distance; Right: results (mean overlap and mean runtime of all datasets) of the discretization approach (Sequitur, Pattern Enumeration) with and without symbolic reduction and with variation of the discretization steps

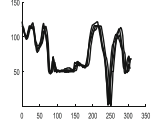
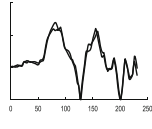
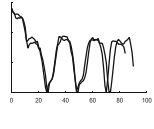
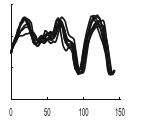


**Fig. 3.** Results of DTW-based algorithm (10 random datasets), for every *costwarp* parameter value a separate plot; top: mean overlap of random datasets with different parameters; bottom: mean runtime in minutes

acceptable runtime and high overlap. This tradeoff is shown in Fig. 3. With values of *max\_similarity\_motif* > 0.9, the runtime increases dramatically. This is caused by the clustering of motifs located by CrossMatch. In case of high values of *max\_similarity\_motif* there are only few motifs clustered together to form pattern. This leads to a high count of patterns, which are then again searched in the whole time series. For further research it has to be evaluated if this last step of the DTW-based approach can be reduced or replaced by a powerful clustering of the candidates. It is also noticeable that the effort of parametrization is high in comparison to the other approaches. At least three parameters have to be chosen carefully to get the expected results. As shown in Fig. 3 the highest overlap values result at values of *min\_similarity* = 0.5. We experienced an increase of the need of RAM in cases of values lower than 0.5, which is not practical for our use cases.

For a final validation, for each approach, we applied the best case parameter set to a real life dataset from a typical use case in the automotive industry. Table 3 shows exemplary results for each algorithm, which were produced with the formally determined best case parameters.

**Table 3.** Exemplary results of a real data set

Approach	DTW	Matrix Profile	Pattern Enum.	Sequitur
Parameters	min_similarity = 0.5 costwarp = 0.4 max_sim_motif=0.8	max_distance = 3	Discr. steps = 9	Discr.steps = 9
Runtime	142 min	31 hours	18 seconds	1 second
$\Sigma$ pattern	1.041	99.718	46.813	386
Exemplary pattern				

## 6 Conclusion and Outlook

In this paper we presented different approaches for pattern discovery in time series under the requirement of time warping. Furthermore we applied well known algorithms and adapted them for our purposes. We developed methods for a Jaccard-index-based comparison of these algorithms as well as for a creation of random time series. The comparison helped us to identify advantages and weaknesses of the different approaches. In general discretization-based approaches have a high performance concerning runtime while the DTW-based approaches perform best regarding the quality of the results. The Matrix Profile can be classified as not suitable for our use cases due to the inability of time warping. Because of our goal to mine time series with far more than one million data points our future focus will be on the discretization-based approaches. Therefore it is interesting to evaluate the influence of further preprocessing steps as well as the development of new discretization techniques with robustness regarding noise and the possibility of controllable time warping. Besides the pattern evaluation of unlabeled data and the extension of the pattern discovery algorithms to multidimensional cases are fields of interest.

## 7 Acknowledgments

We thank Thomas Lehmann ([thomas.lehmann@ivi.fraunhofer.de](mailto:thomas.lehmann@ivi.fraunhofer.de), Fraunhofer IVI, Dresden, Germany) and Dr. Ralf Bartholomäus ([ralf.bartholomäus@ivi.fraunhofer.de](mailto:ralf.bartholomäus@ivi.fraunhofer.de), Fraunhofer IVI, Dresden, Germany) for the cooperation concerning the adaption and implementation of the DTW-based approach (Sect. 3.1). Furthermore we thank Jan Piewek ([jan.piewek@volkswagen.de](mailto:jan.piewek@volkswagen.de), Volkswagen AG, Wolfsburg, Germany) for helpful ideas and discussions concerning the discretization-based approaches (Sect. 3.2).

## References

1. Höppner, F.: Improving time series similarity measures by integrating preprocessing steps. *Data Min. Knowl. Discov.* **31**, 851–878 (2017)
2. Toyoda, M., Sakurai, Y., Ishikawa, Y.: Pattern discovery in data streams under the time warping distance. *Very Large Data Bases* **22**, 295–318 (2013)
3. Jancovic, P., Köküer, M., Zakeri, M., Russel, M.: Unsupervised discovery of acoustic patterns in bird vocalisations employing DTW and clustering. In: *European Signal Processing Conference* (2013)
4. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, CA, USA (2003)
5. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197 (1981)
6. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: a linear-time algorithm. *J. Artif. Intell. Res.* **7**, 67–82 (1997)
7. Li, Y., Lin, J., Oates, T.: Visualizing variable-length time series motifs. In: *SIAM International Conference on Data Mining*, Philadelphia, USA (2012)
8. Coquery, E., Jabbour, S., Sais, L.: A constraint programming approach for enumerating motifs in a sequence. In: *International Workshop on Declarative Pattern Mining*, Vancouver, Canada (2011)
9. Rajeb, A., Loukil, Z., Hamadou, A.B.: On the enumeration of frequent patterns in sequences. In: *The International Conference on Artificial Intelligence and Pattern Recognition*, Kuala Lumpur, Malaysia (2014)
10. Yeh, C.-C., et al.: Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: *16th International Conference on Data Mining*, Barcelona, Spain (2016)
11. Salvador, S., Chan, P.: FastDTW: toward accurate dynamic time warping in linear time and space. In: *KDD Workshop on Mining Temporal and Sequential Data* (2004)