Clustering of Time Series Regarding Their Over-Time Stability

1st Gerhard Klassen
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
klassen@hhu.de

2nd Martha Tatusch
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
tatusch@hhu.de

3rd Stefan Conrad
Department of Computer Science
Heinrich Heine University
Düsseldorf, Germany
stefan.conrad@hhu.de

***Abstract*—The clustering of time series data is still a challenging task. There are different approaches which consider either multiple time series or a single one. While some interpret the whole sequence as one feature vector, others examine subsequences or extract relevant features first. Because of these various perspectives, very different statements result. In this paper, we present the clustering algorithm C(OTS)^2 for multivariate time series data sets, that delivers a clustering per time point. It not only optimizes the quality of the clusters regarding intuitive demands, such as the spatial closeness of objects to their neighborhood within a cluster, but also the stability over time. Additionally, it can easily handle missing data points. The algorithm is of benefit whenever a cohesion of groups of time series can be assumed. One advantage is, that it requires only one parameter. Our experiments on different synthetic and real world data sets show, that our method works reasonable and fulfills the intention of finding temporal stable clusters without presupposing that the exact courses of the time series resemble.**

***Index Terms*—Time Series Analysis, Clustering Methods, Unsupervised Learning.**

I. Introduction

The analysis of sequentially registered data, so called time series, has strongly grown in interest over the past years, as there are more and more data sources for temporal data, such as online shops, IoT devices or medical sensors. The research field, in which databases of multiple multivariate time series are considered, often focuses on the task of classifying these or parts of them to investigate different properties. In many cases, the desired classes are not known beforehand, so that clustering algorithms need to be used. Various approaches consider the whole time series as a vector or extract feature vectors first. Some make use of a decomposition into seasonal, trend and other components.

In this paper, we present C(OTS)^2, a clustering algorithm for multivariate time series data, that clusters the data points per timestamp without missing the temporal context. The presented idea is based on two connection factors: one which expresses the connection factor of one object to another at a certain timestamps, and the other including the temporal context. With the help of these factors and a sliding window, we are able to construct a graph which describes the distance- and time-based cohesions. Its connected components represent the resulting clusters. The basic intention of C(OTS)^2 is the detection of clusters that are stable over time. The intuitive definition of compact clusters, where cluster members have a low distance to each other, is thereby mostly maintained. The maximization of the so called *over-time stability* does not force unintuitive clusters that are spread over the feature space. Still, it has a significant impact on the resulting clusters. Since the temporal components can easily be removed from the cluster calculation, the algorithm can also be used for clustering non-temporal data. Figure 1 shows two examples of clusterings provided by C(OTS)^2.

In contrast to approaches like the detection of Moving Clusters [1], our assumption is, that time series might change their cluster members over time and that this transition would consist of important information. Therefore we build a foundation for further analysis, whereby the detection of outliers is already partly included as our clustering algorithm is able to handle distance- and time-based noise.

Besides the use cases of tracking topics in online forums or the analysis of customer's purchasing behavior, our algorithm is useful whenever it can be assumed that there are groups of time series that behave in a similar way over time. In finance for example, the identification of misstatements regarding the annual financial statements of companies is of great interest. One approach is to interpret these statements as anomalous points with regard to a *common* behavior. This might be described by the behavior of other companies' financial statements that showed a similar behavior over time. With C(OTS)^2 these groups and possibly even the outliers may be identified and a solid foundation for further analysis could be provided. Another example is the analysis of the effectiveness and tolerance of medication regarding different patients. Every human body responds different to different medications. Thus, the formation of groups of patients whose bodies react similar to the drugs, can be assumed. However, it is possible that patients change their groups over time due to different circumstances, for example simply because their body is unique and responds different to the medication than its former cluster members. The group transitions of patients can be an indicator for an anomaly or the necessity of a

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

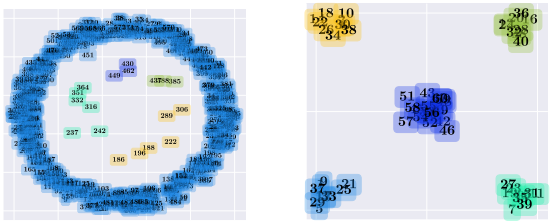

Fig. 1: Examples for resulting clusterings by C(OTS)^2, when only one timestamp is considered. Different colors indicate different cluster belongings.

change in medication, and might help in the prediction of future disease progression. With C(OTS)^2 this behavior can be discovered.

II. Fundamentals

Before we explain our method in detail in the next section, we clarify some important terms. Generally we stick to the definitions of Tatusch et al.'s paper [2], which deals with the detection of outliers with the help of a time series clustering per timestamp. We only make small adaptations and introduce one new definition. In addition to the mathematical statements, the most important definitions are illustrated in Figure 2.

Definition 1. *Time Series*
A time series $T_l = (o_{l,1}, ..., o_{l,n})$ is a tuple of n data points with $o_{l,i} \in \mathbb{R}^d$, $d \geq 1$. The data points are chronologically ordered by their time of recording.

Definition 2. *Time Series Data Set*
A time series data set $D = \{T_1, ..., T_m\}$ is a set of m time series with equivalent points in time. The set of data points of all time series at a timestamp t_i is denoted as O_{t_i}.
Different lengths and missing values are acceptable, as long as the time points can be mapped to a uniform scheme.

Definition 3. *Subsequence*
A subsequence $T_{l,i,k} = (o_{l,i}, ..., o_{l,k})$ with $i > k$ is a tuple of successive data points from time series T_l beginning at time t_i and ending at t_k.

Definition 4. *Sliding Window*
A sliding window of size s is a set of timestamps and is denoted as $w_{i,s} = \{t_{i-\lceil \frac{s-1}{2} \rceil}, ..., t_i, ..., t_{i+\lfloor \frac{s-1}{2} \rfloor}\}$. In case a timestamp is not represented by any time series of the data set, it does not occur in the set.

Definition 5. *Cluster*
A cluster $C_{t_i,j} \subseteq O_{t_i}$ at time t_i, with $|C_{t_i,j}| \geq 2$ and $j \in \{1, ..., q\}$ being a unique identifier (e.g. counter), is a set of similar data points, identified by a cluster algorithm.

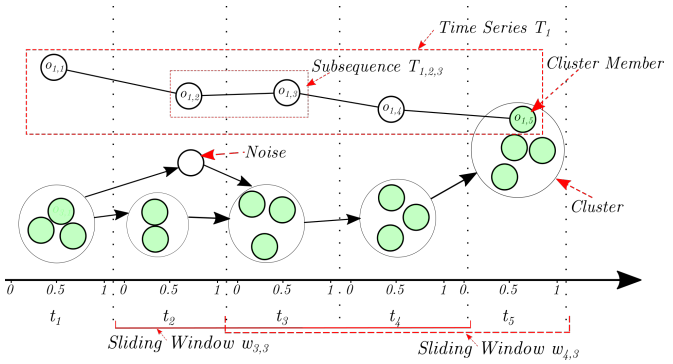

Fig. 2: Illustration of the most important definitions. Note, that a black arrow represents the development of a cluster, while a black line between objects of a time series represents the development of the sequence.

Definition 6. *Cluster Member*
A data point $o_{l,i}$ from time series T_l at time t_i, that is assigned to a cluster $C_{t_i,j}$ is called a member of cluster $C_{t_i,j}$.

Definition 7. *Noise*
A data point $o_{l,i}$ from time series T_l at time t_i is considered as noise, if it is not assigned to any cluster.

Definition 8. *Clustering*
A clustering is the overall result of a clustering algorithm for all timestamps. In concrete it is the set $\zeta = \{C_{1,1}, ..., C_{n,q}\} \cup Noise$.

III. Related Work

With the growing amount of time-dependent data in many applications, researches have presented different approaches to classify time series. The Time Series Classification Repository (TSCR) [3] established by the University of California, Riverside (UCR) and the University of East Anglia (UEA) led to a growth in the number of algorithms for time series classification problems. Besides the hosting of suitable data, the repository also offers a performance comparison on algorithms to the data. The methods presented in the TSCR target the identification of groups, so called classes, and the assignment of objects to those. Therefore, in contrast to our approach, these techniques regard the time series as a whole, so that the classification task refers to the entirety of the sequences. Referring the classification problems presented in the UCR this is reasonable, especially since Eamonn Keogh and Jessica Lin remarkably argued that clustering of time series subsequences is "meaningless" [4]. The problem we are tackling in this paper is different. Instead of identifying the class of a time series, we are interested in the behavior of time series in relation to other sequences. Especially changes in their behavior can contain significant information. This problem is related to cluster evolution over time [1], [5] with the difference, that instead of recognizing earlier clusters at later timestamps, our approach targets a clustering as a basis for the identification

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

of anomalous subsequences. Therefore we introduce the term *time series adaptability* which reflects a time series' ability to adapt to other sequences, that means its average similarity in relation to the data set. It may also be understood as the degree of team spirit of a sequence. Besides the tracking of topics in online forums, detecting outliers in financial data or fMRI scans, there are various other applications which could profit by our clustering algorithm.

The idea of using graphs in clustering algorithms is not new. In 2003, Stuetzle [6] proposes a graph-based clustering algorithm based on runt test for multimodality [7]. The clusters are identified by breaking edges in the minimal spanning tree of the regarded data set. Other graph-based clustering approaches make use of Delaunay Triangulations [8], which represent the dual graph of the Voronoi diagram for a discrete point set. There are techniques which make use of a user input as a threshold for the construction of the graph [9] and methods like AUTOCLUST [10] which do not require any user input. However, in contrary to our algorithm, these approaches do not take the temporal aspect of time series into account.

Finally classic clustering algorithms like k-Means [11] or DBSCAN [12] could be adapted to time series. Since the initial design does not handle time-based data, the modification is not simple. Regarding subsequences as vectors does not reflect the impact of time accordingly. Developing a distance function that includes the temporal aspect might be more promising. However, this is again a complex problem as a time series' neighborhood has to be considered as well. Of course, the naive approach of clustering the data at all time points independently of each other should also be taken into account. Obviously this approach lacks the temporal linkage, but in addition the clustering algorithm's hyperparameters have to be determined for every timestamp. In all cases, an analysis of cohesion post clustering has to be made. This would further influence the time complexity in a negative way. The design of our algorithm is targeted to time series, hence the cohesion analysis is done on the fly during the determination of clusters.

IV. Method

Our algorithm is designed to detect stable over-time clusters. That means, that the actual position of an object at one timestamp is not as important as its surrounding. We accept a certain deviation of an object to a cluster, if it moved with the same cluster members over a certain time period. The sliding window is optional. If not given we regard the whole time series. Our method is based on an arbitrary distance function which is normalized by the maximum distance $d_{\max}$ and minimum distance $d_{\min}$ over all timestamps. This is necessary to convert the distance measure to a similarity measure. For two sequences T_a, T_b we define the distance d at time point t_j as follows.

$$d(T_a, T_b, t_j) = dist(o_{a,j}, o_{b,j}) \tag{1}$$

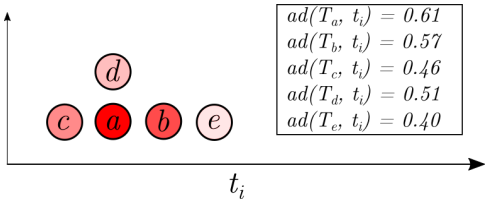

Fig. 3: A high opacity represents a high adaptability. The central location and the low distances to its neighbors lead to a high adaptability of a. In contrary, the high distance and peripheral location of e causes a low adaptability for it.

Here, $dist$ is an arbitrary distance function. Now the similarity at timestamp t_j of two time series can be calculated by

$$sim(T_a, T_b, t_j) = \left(1 - \left(\frac{d(T_a, T_b, t_j) - d_{\min}}{d_{\max} - d_{\min}}\right)\right)^2. \tag{2}$$

We square the term to get overall smaller similarities with a greater difference to each other. Later this becomes handy, when determining the only parameter in our method. The creation of clusters not only depends on the similarity of two objects, but also on the factor how adaptive an object is. We denote an object as being adaptive if it has a high similarity to many objects. It is the average similarity of the regarded object to every other object. Mathematically expressed, the adaptability ad of a time series at the timestamp t_j is defined as

$$ad(T_a, t_j) = \frac{1}{m-1} \cdot \sum_{x \in [1,m], x \neq a} sim(T_x, T_a, t_j)$$

with m describing the total number of objects at timestamp t_j.

In Figure 3 an illustration for the adaptability can be seen. It is important to understand this concept in order to rate the later influence of it to the resulting clustering.

The combination of the adaptability of an object and its similarity to another object represents the connection factor cf at time point t_j:

$$cf(T_a, T_b, t_j) = sim(T_a, T_b, t_j) \cdot ad(T_a, t_j). \tag{3}$$

Because of the adaptability of time series T_a, the connection factor between two time series at a certain timestamp is not symmetric. More precisely, in most cases it is $cf(T_a, T_b, t_j) \neq cf(T_b, T_a, t_j)$.

Finally, we introduce the temporal linkage by the average connection factor avg_cf, which then is incorporated in the temporal connection factor $temp\_cf$. The calculation of avg_cf is introduced with a sliding window, but can be adapted to the whole time series easily. This is particularly useful when short time series are considered. First have a look at the definition of avg_cf with a sliding window $w_{j,s}$:

$$avg\_cf(T_a, T_b, w_{j,s}) = \frac{1}{|w_{j,s}| - 1} \cdot \sum_{i \in w, i \neq j} cf(T_a, T_b, t_i).$$

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

In case the application on the whole time series is wanted, the size of the sliding window can be set to the number of time points of the largest time series. As noted in the previous section, points in time which are not present in the data set, do not occur within the set of the sliding window. The temporal connection factor tmp_cf is then defined as the average of cf at time point t_j and avg_cf. This leads to a higher influence of cf at the regarded point in time.

$$tmp\_cf(T_a, T_b, t_j) = \frac{cf(T_a, T_b, t_j) + avg\_cf(T_a, T_b, w_{j,s})}{2}.$$

With the help of the temporal connection factor tmp_cf and a parameter min_cf indicating the minimum connection factor to build an edge between two data points, an undirected graph $G_{t_j} = (V, E)$ can be created for every timestamp. $V = O_{t_j}$ denotes the set of nodes in the graph which are given by the data points of all time series at time t_j. The set $E \subseteq \{\{o_{a,j}, o_{b,j}\} | \forall o_{a,j}, o_{b,j} \in O_{t_j}\}$ contains all undirected edges between pairs of nodes of the graph. Using the minimum connection factor min_cf, an edge between $o_{a,j}$ and $o_{b,j}$ is added to the graph whenever the temporal connection of $o_{a,j}$ to $o_{b,j}$ or the temporal connection of $o_{b,j}$ to $o_{a,j}$ is greater or equal min_cf. So E is defined as

$$E = \{\{o_{a,j}, o_{b,j}\} | tmp\_cf(T_a, T_b, t_j) \geq min\_cf \vee tmp\_cf(T_b, T_a, t_j) \geq min\_cf\}. \quad (4)$$

After the graph is built, the clusters can be extracted by calculating the connected components[1] of it. Each component represents one cluster, whereby single-element components are marked as noise. Due to the usage of the introduced connection factors non-convex cluster shapes can be detected.

Since the connection factor cf and the adaptability ad are based on the similarity sim of the time series at a timestamp, the threshold min_cf highly depends on the closeness of objects belonging to the same cluster. The more compact the groups of data points are the higher min_cf must be set. Because of avg_cf the over-time stability of course has impact on the parameter choice as well. The more stable the time series are, the clearer the gradation of their connection factors, since cf and avg_cf are converging.

The summarized algorithm can be seen in Algorithm 1. The time complexity of the calculation of tmp_cf for all object pairs is in $O(n^2)$ as it can be done by matrix multiplication and all its components, like calculating the distance, are in $O(n^2)$. Since tmp_cf must be calculated for all m timestamps, the time complexity gets $O(m \cdot n^2)$. The graph again can be created in quadratic runtime and the connected components can even be extracted in linear time. So the overall time complexity of C(OTS)^2 is $O(m \cdot n^2)$ with m being the number of timestamps and n being the number of time series. Compared to the use of k-Means, which is in $O(n^2)$ and would have to be applied for every timestamp, which results in $O(m \cdot n^2)$, too, this time complexity is competitive.

[1]"A connected component of an undirected graph is a maximal set of nodes such that each pair of nodes is connected by a path." – https://www.sci.unich.it/~francesc/teaching/network/components.html

Algorithm 1 C(OTS)^2

procedure COTS(D, min_cf) ▷ $D = \{T_1, .., T_m\}$
  $clusters \leftarrow list\ of\ empty\ dictionaries$
  $V \leftarrow \{\}, E \leftarrow \{\}$
  for $t_i \in \{t_1, .., t_n\}$ **do**
    for all $(o_{a,i}, o_{b,i}) \in O^2_{t_i}$ **do**
      $calculate\ tmp\_cf(T_a, T_b, i)$ ▷ use the aforementioned formula tmp_cf
      if $tmp\_cf(T_a, T_b, i) \geq min\_cf \wedge (o_{a,i}, o_{b,i})\ not\ in\ E$ **then**
        $E \leftarrow E \cup \{(o_{a,i}, o_{b,i})\}$
      end if
    end for
    $G \leftarrow (V, E)$
    $components \leftarrow extract\_connected\_components(G)$
    $components \leftarrow mark\_noise(components)$ ▷ one-element sets denote noise
    $clusters \leftarrow clusters \cup components$
  end for
  return $clusters$
end procedure

As the temporal connection factor is based on the average connection factor, which is zero when considering only one timestamp, the approach can also be used for clusterings of non-temporal data using only the connection factor. Examples of resulting clusterings with C(OTS)^2 on non-temporal data are illustrated in Figure 1.

V. Experiments

Since our approach is a novelty in the field of time series analysis, unfortunately there is no appropriate quality measure which consists of the over-time stability as well as a shape-based measure for clusters. Therefore, we evaluate the accuracy of C(OTS)^2 by visual inspection. For illustration reasons, we generated three different data sets G_1, G_2, G_3 with time series containing two dimensional features, and between four and eight timestamps. Additionally, we consider two real world data sets comprising financial figures from the annual financial statements of publicly listed companies and a data set based on macroeconomic features of countries.

All experiments are explained along with figures. For reasons of illustrations the time series shown are never lasting for more than eight years and do not hold a high number of objects per timestamp. This does not indicate, that our method is not capable of handling greater data sets with more points in time. Quite in the contrary, especially the use of the sliding window, allows us the application to longer sequences. The amount of data points per timestamp changes the results, as it is expectable of a clustering algorithm but the results are still reasonable as can be seen in the experiments with the generated data sets. The shown explanatory figures always follow the same color code. Red indicates outliers, while other colors indicate a cluster.

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

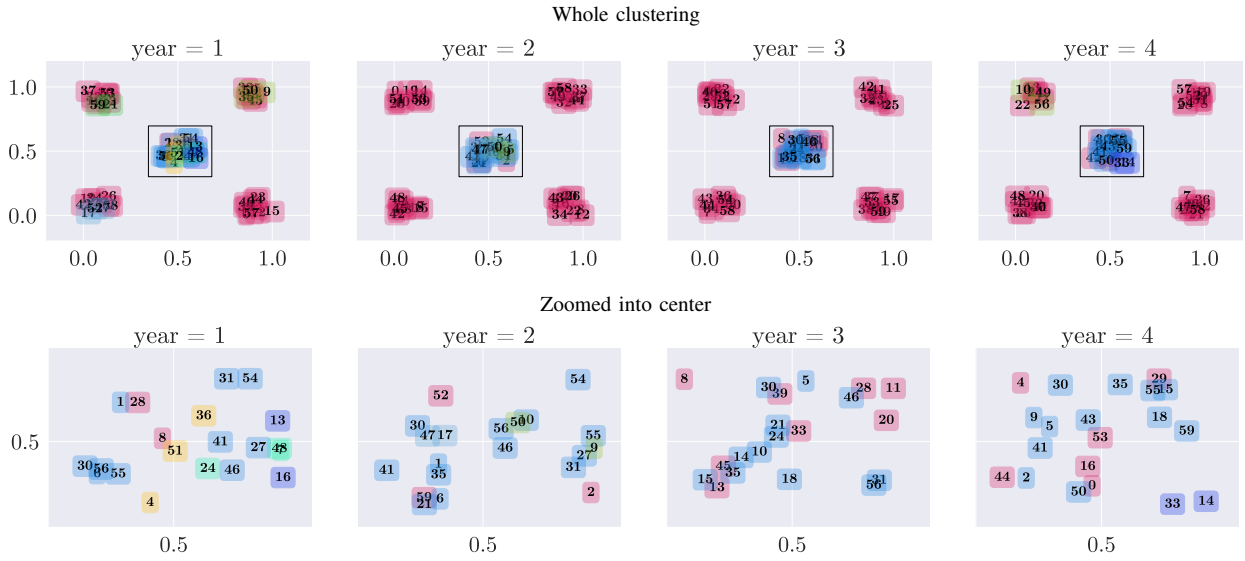

Fig. 4: Resulting clustering by C(OTS)^2 with $min\_cf = 0.35$ and $s = 3$ on the generated data set G_1 with an overall low over-time stability.

A. Synthetic Data

The first data set G_1 is supposed to represent a very *un*stable data set over time. It includes 60 time series and four timestamps. In order to point out the impact of the temporal behavior on the resulting clustering, five clusters were positioned fix in the feature space for all timestamps. For each point in time every time series was placed randomly into one of the five clusters. The data set and C(OTS)^2 clustering result can be seen in Figure 4. The first row shows the resulting clustering. The second row illustrates the excerpt from the upper clustering marked by a rectangle. Red data points indicate noise while other colors represent cluster belongings. Classic clustering algorithms which do not include a temporal aspect would have found five clusters per timestamp as there obviously are always five dense groups of data points. C(OTS)^2, however, marks most objects as noise and finds only small clusters. This can be explained by the fact, that only a few time series move with their cluster members over time. Most of them behave individually and therefore do not show a good team spirit. When considering the zoomed illustration in the second row, it is noticeable, that there are points in the center of the group, which are marked as noise or a separate cluster. This as well is caused by the over-time stability, which is aimed to be optimized in C(OTS)^2.

Note, that this experiment was executed with different parameter settings, thus never a good clustering result could be achieved, except of the case, when only one big cluster results. This is a desired behavior, since regarding the over-time stability, this data set can not be reasonably clustered. An example of the same cluster formation but perfectly stable time series can be seen in Figure 1. The result is the same for one or multiple timestamps if the time series behave stable over time.

The second data set G_2 consists of 15 stable time series and 4 timestamps, and intends to show an over-time clustering that slightly differs from a clustering per time point without temporal context. This effect can be caused by inserting time series which move between two clusters or a merge of clusters over time. In our case there is both, transitions as well as a merge. The result is shown in Figure 5. Since the data points

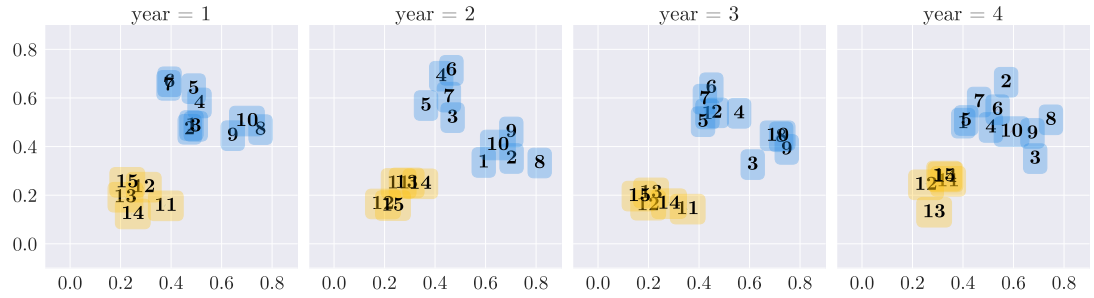

Fig. 5: Resulting clustering by C(OTS)^2 with $min\_cf = 0.15$ and $s = 3$ on the artificially generated data set G_2.

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

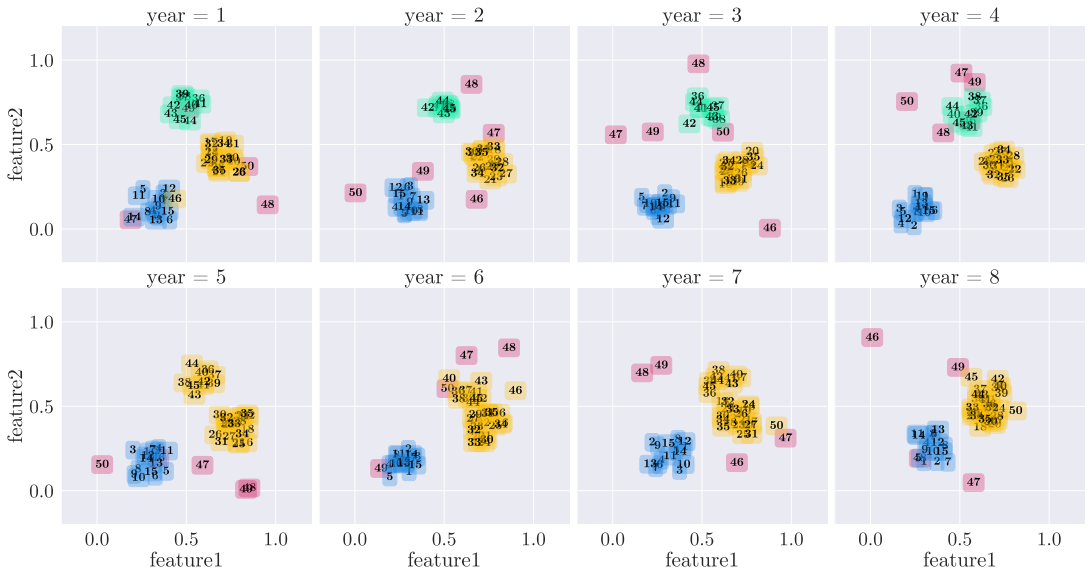

Fig. 6: Resulting clustering by C(OTS)^2 with $min\_cf = 0.32$ and $s = 3$ on the artificially generated data set G_3.

of the upper cluster are not very close to each other, min_cf has to be chosen comparatively small.

In every time point there can be seen two up to three groups of data points. C(OTS)^2 always identifies two clusters, although at least in timestamp two, a classic clustering algorithm without temporal component probably would have found three clusters. This can be explained on the one hand by the transitions of time series 1, 2 and 3 between the two upper clusters, and on the other hand by the fact that the aforementioned clusters merge at a later point in time. The result of a subsequence clustering would probably look different, too, as the exact course of the individual time series differs a lot. The time series 1, 2 and 3 might for example be recognized as noise, because their curves stick out with regard to the other time series.

In the third data set G_3, 3 clusters for 8 timestamps and a total of 50 time series were generated. Five sequences, namely 46 – 50, were inserted as outliers, by placing them randomly in the feature space for every point in time. Figure 6 illustrates the clustering result of C(OTS)^2 with $min\_cf = 0.32$ and a sliding window with size $s = 3$. Since the cluster members of each cluster lie close to each other and the time series are rather stable over time, all connection factors get higher values, so that min_cf is chosen higher than for example in Figure 5.

At first sight it is visible that C(OTS)^2 manages to detect all outlier sequences as such. As in the last two timestamps time series 50 is positioned near to the right cluster, the algorithm assigns it to it. This behavior is reasonable, because both, the similarity and the stability are given. In time point 3 on the other hand, time series 50 is not assigned to the upper cluster although it is located very near to the cluster's members. That is the effect of the considered over-time stability.

In the first four points in time C(OTS)^2 detects three clusters, which probably would also be recognized by common clustering algorithms. From time point five there are only two clusters, which is apparently a good choice especially for the timestamps six to eight. Although in timestamp five there are three obvious groups of time series, C(OTS)^2 merges the upper ones in terms of the further course. Because of the sliding window with width 3, the time points 4, 5 and 6 are considered in order to make a clustering for time point 5. Since the connection factors in timestamp 6 are generally higher than in timestamp 4, as more objects lie in small distance to each other, this timestamp has a higher impact on the clustering in timestamp 5. Therefore, the merge already happens in time point 5.

B. Real World Data

In order to test our method on real world data, we present two data sets. After presenting a financial data set, we present a macroeconomic data set and demonstrate how one could discover knowledge with the help of our algorithm.

First, we chose a financial data set which we obtained from EIKON [13], a product provided by Revinitiv (former provided by Thomson Reuters). We selected 30 arbitrary companies and two random features, namely *SoftAssets* and *Pension* over a timespan of eight years (2007 to 2014). The latter represents reserves for retirement plans of workers, while the first represent assets which have no physical nature such as patents, copyrights and trademarks. Unfortunately not every feature is available for every company at every point in time, therefore new companies may appear and other companies may disappear over time. In Figure 7 one can see the results of C(OTS)^2 applied with a sliding window of size five. Every box represents a company, the label corresponds to the stock symbol of the company. In 2009 one can observe a good example for the time aspect of this clustering algorithm.

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

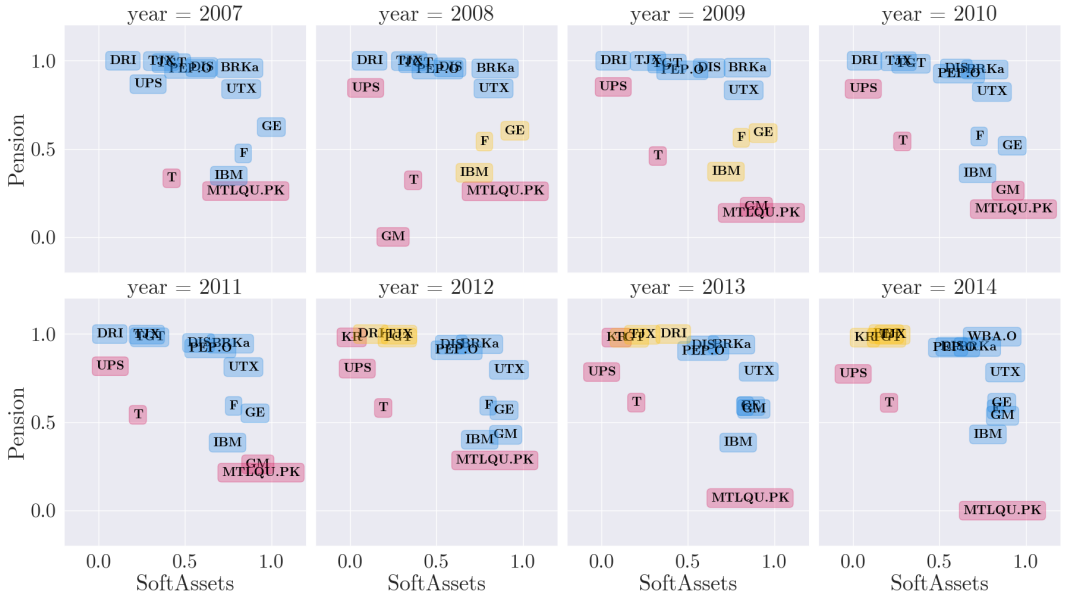

Fig. 7: Resulting clustering by C(OTS)^2 with $min\_cf = 0.26$ and $s = 5$ on the financial data set.

Although the companies *GM* and *MTLQU.PK* are very close to each other our method marks both as noise instead of creating a new cluster for them. This has two main reasons, first the *adaptability* of *GM* and *MTLQU.PK* is low in all three years from 2008 to 2010, second the *connection factor* of *MTLQU.PK* and *GM* in 2008 is very low and therefore also lowers it in the *temporal connection factor*. Comparing this to a clustering per timestamp, most cluster algorithms would assign *GM* and *MTLQU.PK* to a new cluster in 2009. It is also noticeable, that *MTLQU.PK* is detected as an outlier in the year 2012, where it is actually very close to a small agglomeration. In contrary the behavior of *GM* adapts to those of *F*, *GE* and *IBM* from 2012 on. The small splitting of *IBM* in 2013 is not punished and the stability of this cluster is preserved. On the other hand, the small splitting of *UPS* from 2007 to 2008 causes *UPS* being recognized as an outlier for the rest of the time. Since the distance of *UPS* to its peers is rising over time, this is a correct behavior of our algorithm. Another interesting aspect is the handling of overall outliers. In this excerpt AT&T, which is represented by the symbol *T* is always far away from the other companies. Therefore it is also always marked as noise. The second data set is obtained from theglobaleconomy.com [14]. It contains different features to countries over several years. We have chosen two figures to illustrate our algorithm on this data set. Additionally, we have chosen 2007 to be the beginning and 2012 to be the end of the regarded time. Because of this time span and the data basis, only 19 countries remained. The first figure we chose, is the household consumption as percent of the GDP and the second feature is the unemployment rate. The results of our method can be seen in Figure 8. We chose the given timespan, because of the financial crisis in 2008. The first observation, we made is, that the number of outliers increases from the year 2010. In conclusion that means, that the unemployment rate and the household consumption as percent of the GDP did not develop everywhere in the same way after the crisis. It can also be said, that the effect of the crisis is long-term, especially when inspecting numbers later than 2012. While the unemployment rate in some countries remained almost the same as before the crisis, some countries had a bad development. For example Estonia (*EST*) and Spain (*ESP*) had an increase of unemployment from 2009 on. While the change of Estonia is still close to the majority in 2009, Spain had a worse development and therefore is marked as an outlier. In 2010, Estonia almost has the same unemployment rate as Spain and both countries are far away from the majority in the blue cluster. Finally, Estonia somehow reacted on the crisis and could significantly lower its unemployment rate, so that it came very close to those of the majority. Spain on the contrary, had a rising unemployment rate until the last regarded year. In econometrics, this could be a helpful and quick analysis, which puts economic figures into the relation of groups of other countries.

VI. Conclusion & Future Work

The clustering of time series data is a broad field of research. Depending on the application there exist various approaches. When considering multiple multivariate time series, often whole sequences or parts of them are clustered using different preprocessing. In this paper, we presented a novel approach for clustering multivariate time series data. Our over-time clustering algorithm is named C(OTS)^2 and produces clusterings for every timestamp. One particularity of our approach is, that the exact course of (parts of) time series not necessarily has to resemble but the spatial location with regard to other time series over time. Another advantage is, that C(OTS)^2 requires

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516

Fig. 8: Resulting clustering by C(OTS)^2 with $min\_cf = 0.34$ and $s = 3$ on the globaleconomy data set.

only one parameter. The results on various data sets showed, that the resulting clusters are stable over time, while satisfying intuitive demands on clusters, like spatial closeness of objects belonging to the same cluster. Since the calculation is based on two components of which one is time-independent, our algorithm can be used on non-temporal data for connection-based clusterings, as well. Additionally, it can easily handle missing data points. Because of a sliding window, the user is furthermore able to control the temporal impact on the clustering. We are keen to see a development in this field of research. It is important to benchmark the results against other algorithms with the same objective. However, regarding our algorithm, improvements still can be done. Although, we had no real difficulties to find good values for min_cf, a determination method would be very helpful. We are also aware of the optional second parameter s, the size of the sliding window. However, we think, that this is depending on the targeted analysis and should be determined by the domain specialist. In addition, we believe that runtime optimization could make the algorithm even faster, than it already is. Finally, it would be interesting to develop a fuzzy derivative of C(OTS)^2, where data points can belong to more than one cluster, as there are many applications where a hard clustering is not possible or wanted.

VII. Acknowledgement

This work was partly supported by the Jürgen Manchot Foundation, which funds the AI research group *Decision-making with the help of Artificial Intelligence* at Heinrich Heine University Duesseldorf.

References

[1] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In Claudia Bauzer Medeiros, Max J. Egenhofer, and Elisa Bertino, editors, *Advances in Spatial and Temporal Databases*, pages 364–381, 2005.

[2] Martha Tatusch, Gerhard Klassen, Marcus Bravidor, and Stefan Conrad. Show me your friends and i'll tell you who you are. finding anomalous time series by conspicuous cluster transitions. In *Data Mining. AusDM 2019. Communications in Computer and Information Science*, volume 1127, pages 91–103, 2019.

[3] Anthony J. Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn J. Keogh. The UEA multivariate time series classification archive, 2018. *CoRR*, 2018.

[4] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177, August 2005.

[5] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 554–560, New York, NY, USA, 2006. Association for Computing Machinery.

[6] Werner Stuetzle. Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *J. Classification*, 20:025–047, 05 2003.

[7] J. Hartigan and Surya Mohanty. The runt test for multimodality. *Journal of Classification*, 9(1):63–70, 1992.

[8] Xiankun Yang and Weihong Cui. A novel spatial clustering algorithm based on delaunay triangulation. *JSEA*, 3:141–149, 01 2010.

[9] G. Papari and N. Petkov. Algorithm that mimics human perceptual grouping of dot patterns. In *Proceedings of the First International Conference on Brain, Vision, and Artificial Intelligence*, BVAI'05, page 497–506, Berlin, Heidelberg, 2005. Springer-Verlag.

[10] Vladimir Estivill-Castro and Ickjai Lee. Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets. In *In Proceedings of the 5th International Conference on Geocomputation*, pages 23–25, 2000.

[11] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[13] Thomson Reuters. Eikon financial analysis and trading software.

[14] Global economy, world economy.

The final authenticated publication is available online at https://doi.org/10.1109/SSCI47803.2020.9308516